Módszerek és eszközök az informatikaoktatás hatékonyságának növelésére

Doktori (PhD) értekezés

Kátai Zoltán

Debreceni Egyetem
Természettudományi Kar
Debrecen, 2006

Ezen értekezést a Debreceni Egyetem TTK Matematika- és Számítástudományok Doktori Iskola Didaktika programja keretében készítettem a Debreceni Egyetem TTK doktori (PhD) fokozatának elnyerése céljából.

Debrecen, 2006. október 27.

_____

Tanúsítom, hogy Kátai Zoltán doktorjelölt 2003-2006 között a fent megnevezett Doktori Iskola Didaktika programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javasolom.

Debrecen, 2006. október 27.

_____

Tartalomjegyzék

1 Bevezetés

A XX. század második felétől az oktatás bizonyos didaktikai vonatkozásai egyre nagyobb hangsúlyt kaptak, és számos didaktikai szempont új megvilágításba került. Ehhez elsősorban a digitális számítógép feltalálása, az internet létrehozása, valamint az utóbbi években az agykutatás területén elért eredmények járultak hozzá. Mindhármat nyugodtan nevezhetjük forradalminak, ami az oktatásra gyakorolt hatásukat illeti.

Egyesek a számítógép megjelenéséről úgy beszélnek, mint a negyedik forradalmi lépésről az oktatásban. (Az első három: az oktatás a szülők feladatköréből átkerül a tanárokéba; az oktatás színhelye otthonról átkerül az iskolába; az írás az oktatás eszközévé válik.) Az internet kialakítása nyomán a figyelem középpontjába új tanítási, tanulási és értékelési formák kerültek: távoktatás, e-learning, stb.

Ami az emberi agyat illeti, a kutatások feltárták, hogy az egy rendkívül rugalmas biológiai szerkezet. Aszerint tud változni, ahogyan használják, vagy ahogyan visszaélnek a használatával. Úgy tűnik, két fő tényező határozza meg, hogy agyunk miként fejlődik az életünk folyamán: amit érzékszerveinkkel agyunkba beengedünk, és amit gondolkodásunk témájául választunk. Mindkét tényező kitüntetett módon van jelen a tanítás-tanulás folyamatában. E kutatási eredmények forradalmi jellegét Kotulak [1], Pulitzer-díjas szerző a következőképpen fogalmazza meg: „Senki sem sejtette, hogy az agy annyira képes a változásokra, mint ahogy azt ma már a tudomány ismeri.” Miután több mint háromszáz kutatót kérdezett meg, ezt a következtetést vonta le: „Az agy nem sztatikus szerv, a sejtkapcsolatok állandóan változó tömege, mely kapcsolatokat nagymértékben befolyásolja a tapasztalás”. Az agykutatások közvetlen hatása a tanuláselméletekre abból is nyilvánvaló, ahogy Rita L. Atkinson és Richard C. Atkinson[2] a mai pszichológia fő mondanivalóját megfogalmazzák: „... a pszichológiai jelenségek pszichológiai és biológiai szinten egyaránt magyarázhatók, s a biológiai elemzés azt mutatja meg, hogyan valósulnak meg a pszichológiai funkciók az agyban.”

Az agykutatásnak és a számítógépnek az oktatás folyamatára kifejtett hatása úgy is megnyilvánul, hogy hangsúlyt kapnak régóta ismert didaktikai alapelvek, vagy új lehetőségek tárulnak fel az alkalmazásukra. Comenius [6] például, akit egyesek a modern oktatás megalapítójának tartanak, a következő megállapítást tette: „A tanulás legyen teljesen gyakorlatias, teljesen szórakoztató, ... olyan, hogy általa az iskola valóban a játék helyévé, vagyis az egész élet előjátékává váljon.” Másik jól ismert, és a modern kutatások által erőteljesen alátámasztott didaktikai alapelv, hogy a tanulás annál hatékonyabb, minél több érzékszervet vonunk be. A számítógépek megjelenése, illetve a számítástechnika fejlődése új távlatokat nyitott e több száz éve megfogalmazott alapelvek hatékony alkalmazásához.

A fejlett társadalmak jelenlegi és a közeljövőben bekövetkező szellemi, gazdasági és politikai átalakulásai egyértelmű változásokat idéznek elő az iskolarendszerben is. A tudás korszaka - perceptív forradalomban [3,4] a súlypont a technológiáról az információra helyeződik át. Az igazi fordulópontot az információ tudássá alakításának és alkotó alkalmazásának a képessége jelenti. E kompetencia pedig úgy nyilvánul meg, hogy birtokosa a megszerzett, megtanult, elsajátított információt adaptív módon kezeli, felhasználja, alkalmazza; ily módon maga is új információt állít elő a társadalom részére [5].

E változások szempontjából határkőnek számít a Bolognai Konferencia. A bolognai oktatási reform alapvető építőkövei a tanulási eredmények: azon ismeretek, készségek, képességek és kompetenciák, amelyeket az egyén a tanulási folyamat végére elsajátít. Az elvárt tanulási eredmények meghatározzák, hogy a tanuló milyen tudással, ismeretekkel rendelkezik, hogyan tudja a megszerzett tudást alkalmazni, illetve milyen, egy szakterület sikeres műveléséhez szükséges általános kompetenciákra tett szert a képesítés megszerzéséig.

E forradalmi előrelépésekkel, illetve társadalmi színtű változásokkal egyidejűleg egy paradigmaváltásnak vagyunk a szemtanúi: a tanítástól a tanulás felé, a tanárközpontú oktatástól a diákközpontú felé. Ezt szemlélteti az is, ahogy az utóbbi években a súlypont áthelyeződött a távoktatásról a távtanulásra. (Érdemes összehasonlítani a találatok számát, amelyeket úgy kapunk, hogy az e-learning, illetve a távoktatás kulcsszavakat beírjuk egy internetes kereső keresőmaszkjába.) E paradigmaváltással megváltozott a tanár szerepe az oktatás folyamatában. A modern tanár már nem az ismeretelméleti tudás központja, hanem a tanulási folyamat segítője és előmozdítója.

Ahhoz, hogy az imént felvázolt új irányzatok kifejthessék jótékony hatásukat a tanulókra, többre van szükség a tanuláselméletek átértékelésénél. A tanároknak égetően szükségük van olyan konkrét didaktikai módszerekre és eszközökre, amelyek átviszik a gyakorlatba a legújabb kutatások eredményeit, és kihasználják a számítógépek által nyújtott lehetőségeket.

Bár az irányelvek egyre világosabbak, és a lehetőségek egyre elérhetőbbek, gyakran továbbra is hiányoznak a konkrét didaktikai módszerek és a hatékony didaktikai eszközök. A jelen dolgozat az informatikaoktatás e területein igyekszik előrelépést tenni. A dolgozat anyagának kidolgozásában 15 éves informatikaoktatási tapasztalatokra, és az e területen végzett kutatásokra támaszkodtunk.

1.1 Informatikadidaktika

Belépve a XXI. századba, az informatikaoktatás feladata, hogy minden diákban kialakítsa a számítógép-felhasználói készségeket, valamint kiképezze a következő számítógép-programozó generációt. A számítógép mindkét esetben az oktatás tárgyát képezi. Egy másik lehetőség, ahogy a számítógép kapcsolódhat az oktatáshoz, oktatási eszközként való használata bármely tantárgy tanítása során. Ezt nevezzük számítógéppel támogatott oktatásnak. Tehát a számítógép az oktatásban egy időben jelen lehet mint az oktatás tárgya is, és mint oktatási eszköz is. Ez a dolgozat elsősorban a számítógépek programozásának a tanítására helyezi a hangsúlyt, illetve a számítógépnek mint didaktikai eszköznek az ebben való felhasználására.

1.1.1 Az informatikadidaktika mint tudomány

Az informatika mint tudomány a matematikához áll a legközelebb, így az informatikadidaktika is sokat meríthet a matematikadidaktikából. Ha a matematikadidaktika fiatal, akkor az informatikadidaktika nagyon fiatal interdiszciplináris tudomány. Interdiszciplináris, mert kölcsönhatásban áll az informatikával, a számítástechnikával és a matematikával, valamint ezek fejlődésével, a tanulás és fejlődés pszichológiájával, a szociológiával, a pedagógiával, az általános

didaktikával, a tantervelmélettel, az informatikatanítás gyakorlatával, az informatikatanítás tanulásával, az iskolai élettel és a valóságos világgal, amelyben a számítástechnikai eszközök a mindennapi élet részévé váltak. E tudományág fiatalságát - más didaktikai tudományokkal szemben is - magának a számítástechnikának és informatikának mint tudománynak a fiatalsága is megmagyarázza. Örvendetes azonban, hogy amíg a matematikadidaktikának évszázadokat kellett várnia, hogy elkezdődjön a tudományos megalapozása, addig az informatikadidaktika esetében csupán néhány évtizedes az eltolódás, ami egy természetes különbség. Mindezek indokolttá teszik, hogy komolyan vegyük Krygowska [7] matematika-módszertani kutató alábbi figyelmeztetését (bár a matematikadidaktikára fogalmazta meg, az informatikadidaktikára még inkább érvényes):

„A matematikadidaktika mint tudomány fejlődésének kezdeti szakaszában van, lassan, fokozatosan dolgozza ki saját módszertanát és nyelvét. Annak ellenére, hogy sok publikáció vállalkozik e terület elméleti és gyakorlati eredményeinek bemutatására, még messze vagyunk a tudományosan megalapozott általánosításoktól, a mélyebb elméleti felfogástól, értelmezéstől, nem haladtuk még meg a csak lokális rendezés és strukturálás fázisát a matematika tanulására és tanítására vonatkozólag. A matematikadidaktika születőben levő diszciplína (in statu nascendi), ezt harag és részrehajlás nélkül (sine ira et studio) köteles elismerni az is, aki a diszciplína tudományos voltát tagadja, de téved az is, aki benne egy teljességben kifejlődött tudományt akar látni. A matematikadidaktikai kutatások úttörő jellegének tudata szükséges a matematikadidaktikusok számára is, ez megvédi őket azon tételeik idő előtti abszolutizálásától, melyek nem rendelkeznek szilárd elméleti és tapasztalati megalapozottsággal."

Az informatikadidaktikai módszerek tudományos vizsgálatánál nagy szerep jut a pszichológiai, pedagógiai, illetve általános didaktikai alapelvekkel történő elméleti megalapozásnak. Mivel ezen alapelvek is még csak körvonalazódnak a kutatók előtt, ez további ok, hogy óvatosak legyünk kijelentéseinkben. Egy másik módja annak, ahogy mérlegre tehetjük a didaktikai módszereket, ha a tanulókon mérjük le, hogy milyen eredményekhez vezetnek. Ez jelenti a tapasztalati megalapozást. A bemutatandó módszerek és eszközök elemzésénél mindkét eljárásra támaszkodunk.

Az informatikadidaktika helyzetét az is egyedülállóvá teszi, hogy hihetetlen gyorsasággal fejlődik a számítástechnika és az informatika, azok a tudományok, amelyek a tartalmat biztosítják az informatikatanításhoz. Ezzel a fejlődéssel párhuzamosan folyamatosan változik e tudományoknak a társadalomban betöltött szerepük is. Sőt, amikor „információs társadalomról" beszélünk, akkor arra utalunk, hogyan változtatta meg a számítástechnika a világot. Ez viszont megköveteli az informatikadidaktika célkitűzéseinek a változásokhoz való állandó igazítását. A céltudatosság fontosságát emeli ki Braun [8] következő kijelentése: „Nem az a feladatunk, hogy megsejtsük, mit hoz a jövő, hanem az, hogy mai munkánkkal mi alakítsuk." A fejlődés dinamikája, valamint a jövőformálás felelősségének súlya, ugyancsak a megfontoltság szükségességét hangsúlyozza.

1.1.2 Tanítás, tanulás és értékelés

A bolognai reformprogramban is megfogalmazódó új irányzatok a tanítás-tanulás-értékelés közötti szoros kapcsolatot hangsúlyozzák, mint amelyek egy visszacsatolásos folyamat szorosan összefüggő elemei. A diákközpontú oktatás előtérbe kerülésével a folyamat súlypontja a tanításról a tanulásra helyeződött át [9].

A tanárközpontú oktatás jelmondata, hogy *„maradjatok csendben, és írjatok le mindent, amit mondok”*. Néhány további jellemzője:

- A tanár az ismeret és a tudás központja, irányítja a tanulási folyamatot és ellenőrzi a diákok hozzáférését az információhoz.
- A diákokat „üres edénynek”, a tanulást pedig egyszerű összegző folyamatnak tekinti.
- A tanítást az „átlagos” diákhoz méri, és mindenkinek ugyanolyan ütemben kell haladnia.

Ezzel szemben a diákközpontú oktatást a következőképpen lehetne körülírni:

- A diákok nem passzívak, saját érzékelésük alkotta keretekkel érkeznek.
- A diákok különböző módon tanulnak.
- A tanulás aktív, dinamikus folyamat.

Ma már minden hatékony oktatási gyakorlatnak egyik alapvető tényezője egyértelműen, hogy a tanulók optimális tanulási folyamataihoz szükséges feltételekre összpontosít. Az aktív tanulás jelentősége azon a felismerésen alapszik, hogy az intelligencia elválaszthatatlanul összekapcsolódik a cselekvéssel (pszichológiai aspektus), és hogy a szervezet aktivitása az idegsejtek növekedésének és összekapcsolódásának alapvető feltétele (neurofiziológiai aspektus). Ha a tanulási tartalmakkal való aktív foglalkozásnak biztosítunk feltétlen előnyt, akkor ennek hatása van az oktatási folyamat tervezésére, a teljesítmények ellenőrzésére és a teljesítmény értékelésére is. Mindennek a tanuló személyiségéhez szükséges igazodni [10].

Az informatikadidaktikának ugyanazok az alapkérdései, mint az általános didaktikának. A modern oktatás fentebb bemutatott új irányelvei (tanulás-, illetve tanulóközpontú oktatás; a tanítás-tanulás-értékelés egységes egésze) szükségessé teszik ezen alapkérdések átfogalmazását is. Például, a „mit, miért, hogyan, mivel tanítsunk?” tanárközpontú kérdések helyett:

1. Mit tanuljon a diák? Milyen szerepet vállaljon abban a tanár? Mit kérjünk számon?
2. Miért azt tanulja a diák? Miért azt kérjük számon?
3. Hogyan tanítsunk? Hogyan tanuljon a diák? Hogyan értékeljünk?
4. Mivel tanítsunk? Mivel tanuljon a diák? Mivel értékeljünk?
5. Miért úgy tanuljon a diák? Miért úgy tanítsunk? Miért úgy értékeljünk?

Az első kérdésre a válasz adja a tartalmat. A második kérdés a célok megfogalmazásához vezet, ami - többek között - indokolja a tartalmat. A harmadik és negyedik kérdés vezet el a módszerekhez és az eszközökhöz. És végül az ötödik kérdésre adott válasz pszichológiailag, pedagógiailag, didaktikailag és kísérletileg alapozza meg a

módszereket. Ennek is a célok, valamint a hatékonyság szem előtt tartásával kell történnie.

Amíg a célok és tartalmak meghatározásánál elsősorban a tanulót, illetve annak a jövő társadalmában betöltendő szerepét tartjuk szem előtt, addig a módszerek és eszközök mind a tanárt, mind a diákot feltételezik. Bár e szereposztás súlypontja a tartalom természetétől függően változhat, a tanárnak figyelembe kell vennie, amit már Arisztotelész is megállapított: *„Amit meg kell tanulnunk megtenni, azt a tett által tanuljuk meg."*

A jelen dolgozatban három új didaktikai módszert mutatunk be, amelyekkel a tanítás-tanulás-értékelés folyamat mindhárom elemének hatékonysága növelhető. Az első módszert (**Algoritmustervezés felülnézetből**) az informatikaoktatás egy sajátos területére, az algoritmustervezési stratégiák oktatásához dolgoztuk ki. A módszer alkalmazásához szükséges tartalom biztosítása érdekében a szerző kiadott – ugyanezzel a címmel - egy könyvet is [11]. Bár elsősorban *tanítás*i módszer, hangsúlyt fektet a tanulók aktivizálására is. Ezt főként kérdésfeltevésekkel éri el. Mindegyik stratégia esetén egy Pólya-féle kérdéssorozatot ajánl, amelyek megválaszolása lépésről lépésre elvezet a feladatok megoldásához. Miután elég sok feladat kapcsán a tanár teszi fel a kérdéseket, a tanulók eljutnak oda, hogy ugyanezt a megközelítést alkalmazva (kérdéseket tesznek fel maguknak), önállóan is képesek lesznek ismeretlen feladatokra alkalmazni az illető stratégiát. Maga a könyv is olyan stílusban íródott, hogy a diákok önállóan is tudják használni, és élvezettel mélyüljenek el benne.

A második módszer (**Látás, hallás és kinesztetikus érzékelés bevonása elemi algoritmusok tanításába**) az elemi algoritmusok *tanításának és tanulásának* hatékonyságát hivatott növelni. Ehhez a módszerhez egy szoftvert is kidolgoztunk, amely mint didaktikai eszköz lehetővé teszi – mind a tanárnak, mind a tanulóknak - a módszer hatékony alkalmazását a tanítás, illetve a tanulás során. Amint a módszer neve is utal rá, a tanulók szinte a teljes lényükkel vesznek részt a tanulásban. A diákok a szoftvert - mint e-learning eszközt - önállóan, akár otthon is használhatják.

Végül a harmadik módszerrel, amely egy hálózati szoftver **(„Legyél te is eminens"**) köré épül, az *értékelés* fázisát szeretnénk hatékonyabbá tenni. Egy olyan tesztalapú „feleltető szoftverről" van szó, amely szinte minden tantárgy keretén belül jól használható, és lehetővé teszi a tanítás-tanulás-értékelés folyamatra, mint egészre való rálátást. A „Legyél te is eminens" szoftverrel történő feleltetés egyik erőssége, hogy az osztály többi részét is aktívan bevonja a felelők megmérettetésébe.

Mielőtt részletesen bemutatjuk a három módszert és a hozzájuk kapcsolódó tanmenetet, illetve a szoftvereket, tekintsük át röviden az informatikadidaktika célrendszerét. Ez segíteni fog átlátni, hogyan visznek közelebb a kidolgozott módszerek és eszközök egyéb területeken is a hatékony informatikaoktatás céljához.

1.1.3 Az informatikadidaktika céljai

Az informatikatanítás, -tanulás céljait nyilván az általános tanítási és tanulási céloknak rendeljük alá. A The World Book Encyclopedia a következőképpen fogalmazza meg tömören az oktatás célját: „Az oktatásnak segítenie kell abban, hogy az emberek a

társadalom hasznos tagjaivá váljanak. Abban is segítségükre kell lennie, hogy nagyrabecsülést fejlesszenek ki kulturális örökségük iránt, és hogy elégedettebb életet éljenek."

A mai informatizált világban vitathatatlanul komoly szerepe van az informatikatanításnak abban, hogy a fiatalok a társadalom hasznos tagjaivá váljanak, és sikeres, elégedett életet éljenek. Nemcsak arról van azonban szó, hogy olyan tartalmakat közvetítsünk a tanulók felé, amelyeket később fel tudnak használni, hanem egy olyan gondolkodásmód kialakítására kell törekedni, amely időtálló az állandó változásokkal szemben, és amelyet az élet különböző területein hasznosítani tudnak.

Mivel a tartalmak a számítástechnika, az informatika fejlődésével állandóan változnak, a hangsúlyt az alapelvek elmélyítésére kell helyezni. Fontosabb például az algoritmikus gondolkodás, illetve a különböző programozási paradigmák megtanítása, mint egy konkrét programozási nyelv bemutatása, bár ez utóbbira is szükség van. Számos programozási nyelv mára már holttá vált, de a számítógépek programozásának alapelvei nem változtak. Ezért, ha valaki egy ma már holtnak számító programozási nyelvet tanult az egyetemen, hogy milyen könnyen tud áttérni egy új programozási nyelvre, az attól függ, mennyire sajátította el az alapelveket.

Fontos szerep jut tehát annak, amit mi „szakmai bölcsességnek" fogunk nevezni, és amit a következőképpen lehetne meghatározni: A bölcsesség ötvözi az ismeretet, a tisztánlátást és az értelmet, és kamatoztatja őket a gyakorlatban. Egy tisztánlátó ember képes felmérni a dolgok értékét, képes különbséget tenni fontos és lényegtelen között, érzékeli a dolgok közti különbségeket - még az árnyalatbelieket is -, és mivel a dolgok mögé lát, látja azt is, ami nem nyilvánvaló. Az értelem alatt itt azt értjük, hogy valaki látja, hogyan függenek össze egymással a tények, illetve átlátja a sok-sok részletből összeálló teljes képet. Mindezen fogalmaknak valamely szakterületre történő vonatkoztatása elvezet a szakmai bölcsesség fogalmához. Az oktatás egyik alapvető céljának kell tekinteni e bölcsesség közvetítését. Ugyanezt emeli ki Stephen [9], amikor a tanulási eredmények előtérbe helyezéséről beszél, vagy Schaffhauseri [5], amikor az információ tudássá alakításának képességét hangsúlyozza.

Amint már említettük, az informatikaoktatásnak két jól elhatárolható területe van: képezünk számítógép-használókat és képezünk programozókat. A mai társadalomban már elengedhetetlen, hogy minden diák legalább felhasználói készségekkel hagyja el az iskola padjait. Egyre több középiskola indít továbbá informatika tagozatos osztályokat, ahol programozást tanulhatnak a diákok. Sőt vannak iskolák, amelyek mind elemista, mind gimnazista tanulóknak bevezették a programozást választható tantárgyként. Ha nem programozóként fogja is valaki keresni a kenyerét, hasznos lehet számára, hogy szert tesz programozói készségekre (nem beszélve a gondolkodásmódjára gyakorolt jótékony hatásról): felhasználhatja felsőfokú tanulmányai során, a szakmájában, tudományos kutatásokhoz, stb. Mivel a bemutatandó módszerek főleg a programozás tanításának hatékonyságát növelik, ezért elsősorban annak a céljaira fogunk összpontosítani.

Stephen [9], az Egyesült Királyság bolognai szervezője Budapesten 2006. januárjában tartott előadásában Bloom módszertanára hívta fel a figyelmet, amelyet a XXI. század tanárának is követnie kellene. Bloom [12 ,13] minden tantárgyra használható taxonómiát dolgozott ki.

1.1.3.1 Kognitív célok

A Bloom-féle taxonómia kognitív célrendszere szintekre bontva a következő:

- megismerés;
- megértés;
- elemzés;
- szintézis;
- értékelés.

Az, amit társadalmi szinten az információ koráról a tudás korára való áttérésnek nevezünk, az individuum szintjén a megértéssel kezdődik. A megértést követően beszélhetünk először arról, hogy az információ alkalmazható tudássá alakult. Az elemzés, a szintézis és az értékelés a tudás magasabb szintjei, amelyek lépésről lépésre elvezetnek a megszerzett információ mind teljesebb hasznosításának képességéhez.

A második fejezetben kifejtett „Algoritmustervezés felülnézetből" didaktikai módszer kidolgozásánál a legmesszebbmenőkig igyekeztünk szem előtt tartani a Bloom által megfogalmazott kognitív célokat. A módszer segít a tanulóknak, úgymond felülnézetből látni öt alapvető programozási technikát (a mohó algoritmust, a visszalépéses keresést, az oszd meg és uralkodj elvet, a dinamikus programozást, az elágazás korláttal módszert), ugyanakkor megértik a technikák közötti elvi, alapvető, sőt árnyalatbeli különbségeket, illetve hasonlóságokat. A módszer másik erőssége, hogy segít a diákoknak a vizsgált technikák egymáshoz viszonyított „értékét" is felmérni. A tanulók előtt nyilvánvalóvá válnak az egyes technikák erős, illetve gyenge pontjai, valamint az, hogy adott helyzetben melyiknek az alkalmazása a legcélszerűbb és miért. Mindez ahhoz vezethet, hogy a tanulók képesek lesznek a leghatékonyabb algoritmus megtalálására egy adott probléma megoldásához.

Az elemi algoritmusok tanításának-tanulásának hatékonyságát a különböző érzékszervek bevonásával növelő módszer (3. fejezet) olyan fokú megértést, tisztánlátást, elemező-tervező képességet és problémamegoldó készséget alakít ki a tanulókban, hogy indokoltnak láttuk az „Elemi algoritmusok anatómiája" kifejezés használatát.

1.1.3.2 Affektív célok

A gyermekeknél az új dolgok iránti kielégíthetetlen kíváncsiság és a tudás iránti szenvedélyes vágy figyelhető meg. Hogy kielégítsék kíváncsiságukat, valóságos laboratóriummá változtatják maguk körül a világot. Minden érzékszervüket bevonják a vizsgálódásba, még az ízlelőbimbóikat is!

A gyermekek tudásvágya még nyilvánvalóbbá válik, amikor elkezdenek beszélni. Sok szülő türelmét próbára teszik a se vége, se hossza kérdezősködéssel. „Ismereteik túlnyomó részét kitörő lelkesedéssel és buzgalommal sajátítják el"- írja Holt [26].

Néhány évvel később a gyermekek egy új világba csöppennek: tanárok, tankönyvek, iskolapadok és talán több száz másik gyerek társaságában folytatják a tanulást. Sajnos, többévi iskolai oktatás után sokuk már nem ég annyira a vágytól, hogy ilyen módon

szerezzen ismereteket. Néhányukat egyenesen stresszeli az iskola, vagy kínszenvedésnek tartja a tanulást.

Az iskolai tanulással szemben kialakult negatív érzések némelyeket egészen felnőttkorukig vagy akár időskorukig is elkísérik, elkedvetlenítve őket mindentől, ami elmélyült gondolkodást, tanulást vagy kutatómunkát igényel. E valós jelenség szöges ellentétben áll azzal, amit a modern társadalomban a sikeres élet alapfeltételének tekintenek: az egész életen át való tanulás. Mindez elvezet az oktatás affektív céljainak fontosságához, valamint ahhoz a felfogáshoz, miszerint az iskolát úgy kell tekinteni, mint a tanulási feltételeket biztosító, illetve a teljesítményfokozó intézményt. A tanulók nem eleve motiváltak, hanem motiváltságuk, teljesítményük nagymértékben függ a tanulási környezetüktől is [14].

Affektív célokon azt értjük, hogyan gondolkodik, érez, áll hozzá, mennyire motivált a tanuló az illető tantárggyal kapcsolatban. A Bloom által kidolgozott affektív célrendszer minden tantárgyra az alábbi:

1. figyelembevétel;
2. aktív hozzáállás;
3. értékelés;
4. általános érdeklődés;
5. meggyőződés.

Wittmann [15] által a matematikatanításra kidolgozott affektív célrendszer egy lehetséges alkalmazása a programozás tanításának területére a következő:

1. A programozás okozta öröm, érdeklődés és pozitív hozzáállás fejlesztése.
2. Önbizalommal végzett önálló tanulói munka.
3. Közös munkára való készség, abban öröm lelése. Másoknak adott segítség, másoktól való tanulás.
4. Céltudatos, lelkes, koncentráltan végzett munka. Törekedés a megértésre.
5. A programozásban szellemi megelégedettség találása.
6. Annak az elégedettségnek a tapasztalása, amikor az ember látja a programozói munka gyakorlati értékét.
7. A sikeres programozói alkotómunka végén érzett öröm.
8. Az informatikának a társadalomban betöltött szerepének a méltányolása.
9. Annak felismerése és értékelése, ahogyan az informatika hozzájárul a modern élethez.
10. Egy algoritmus szépségében, egy felület felhasználóbarát voltában, stb. való gyönyörködés.

Az informatikát oktató tanár különösen előnyös helyzetben van, hiszen a diákok - függetlenül attól, hogy elemista, középiskolás vagy egyetemista diákról beszélünk - már úgy mennek iskolába, hogy vonzódnak az informatika iránt. Tehát a társadalom bizonyos mértékben megoldja a tanár helyett az egyik legnehezebb problémát, az érdeklődés felkeltését. A tanárnak csak fenn kell tartania és el kell mélyítenie a diákokban ezt a vonzódását az informatika iránt. A számítógép-felhasználók képzésében ezt nem annyira nehéz megtenni, de a programozás tanítására bizony oda kell figyelni. Nem ritka eset, hogy a diák megutálja az informatikát az iskolában.

Az affektív célok elérése nagyban függ attól, hogy maga a tanár miként gondolkodik és érez az informatika és az informatika tanítása iránt. A jó tanár fontosnak tartja és szereti azt, amit tanít. Tisztában van tantárgya gyakorlati értékével, és képes erről meggyőzni diákjait is. Fontosnak és kiváltságosnak kell tartania azt a szerepet is, amelyet tanárként betölt a jövő nemzedék oktatásában, és szeretnie kell diákjait. Az a tanár, akit nem lelkesít a tantárgya és a tanári szerep, nem valószínű, hogy képes lesz lelkesíteni a diákjait. Bár az, amit tanítunk, mindig elsődleges fontosságú, az affektív célok elérésénél különös figyelmet kell fordítani arra, ahogy tanítunk: beleéléssel, meggyőződéssel, stb.. A tanárt ragadja magával az, amit tanít, és ez látszódjon az arckifejezésén, taglejtésein, érződjön a hangszínéből, stb.. A lelkes tanár már a puszta jelenlétével is azt üzeni a tanulóknak, hogy szereti, amit tanít, és ez az értékes tulajdonság sugárzik róla, amikor tanít.

Amint az előbbi leírásból is kiderül, a sikeres tanár törődik azzal, hogy diákjai értik-e azt, amit tanít nekik, és hogy élvezik-e óráit. Ugyanis lehetetlen lelkesen tanítani egy olyan osztályban, ahol a tanulók vagy unottan figyelnek, vagy nem figyelnek. Ahogy Bolyai Farkas mondta: „Tanítani tanulni vágyót gyönyörűséges”. Mindez viszont hatékony és *érdekes* tanítói, illetve értékelési módszereket feltételez. Ha a tanár látja a diákok arcán, hogy módszerei fellelkesítik őket, továbbá, ha az eredmények is tükrözik a módszerei hatékonyságát, akkor ez pozitívan hat vissza rá. Tehát a tanítási-tanulási módszereknek és a hozzájuk kapcsolódó didaktikai eszközöknek a milyensége kétszeresen is kapcsolódik az affektív célok eléréséhez. A tanulók lelkesen, értelmes tekintettel, érdeklődéssel figyelnek, és ez fokozza magának a tanárnak a lelkesedését is, ami viszont ragályos a tanulókra nézve.

Ami a bemutatandó módszerek érdekessé tételét illeti, amint látni fogjuk, az algoritmustervezési stratégiákat „talk show”-ra hívjuk, az érzékszervek bevonásával az elemi algoritmusok tanítását tesszük játékossá, a „Legyél te is eminens” szoftver pedig a számonkérésnek ad versenyjelleget.

Egy tanulmány kimutatta, hogy amikor a tanulókat érzelmileg is bevonják jelentős feladatokba, akkor a tanulásért felelős agyterületeik sokkal aktiváltabbak, mint amikor ismétlődő, unalmas feladatokat kapnak, vagy ha túl sok információval árasztjuk el őket [16].

2 Algoritmustervezés felülnézetből

Comenius [6] a következő kijelentést tette a tanítási módszereket illetően: „Tanítani szinte nem is jelent mást, mint megmutatni, miben különböznek egymástól a dolgok a különböző céljukat, megjelenési formájukat, és eredetüket illetően. ... Ezért aki jól megkülönbözteti egymástól a dolgokat, az jól is tanít.” Az ebben a fejezetben bemutatott didaktikai módszer elsősorban erre az alapelvre épül. Egy olyan tanítási, illetve tanulási módszerről van szó, amely segít a tanulóknak, úgymond felülnézetből látni öt alapvető algoritmustervezési stratégiát. A módszer célja az, hogy e programozási technikák bemutatásán túl, olyan nézőpontba juttassuk a tanulót, amelyből feltárulnak előtte a technikák közötti elvi, alapvető, sőt árnyalatbeli különbségek, illetve hasonlóságok. A comeniusi alapelvvel összhangban ez nélkülözhetetlen, ha uralni szeretnénk a programozás e területét.

A szakirodalomban számos példát találunk a programozási technikák összehasonlítására. Cormen, Leiserson és Rivest például [17] referenciamunkájukban összehasonlítják a dinamikus programozás és a mohó stratégiákat. Más publikációk [18] azt tárgyalják, hogyan egészíthetik ki egymást a visszalépéses keresés és a mohó technikák. Andone és Garbacea [19] a dinamikus programozás és az oszd meg és uralkodj stratégiákról közöl összehasonlító elemzést.

A technikák párhuzamos bemutatására alkalmazott másik módszer, amikor ugyanazon feladatokat különböző technikákkal oldják meg [37]. Ezt az utat követve kidolgoztunk egy didaktikai módszert, amely lehetővé teszi mind az öt technika módszeres, párhuzamos vizsgálatát. [63] Célunk olyan felülnézet kialakítása a tanulók fejében, amelyből egyidejűleg láthatják a bemutatott stratégiákat. A módszer másik erőssége, hogy látva a teljes képet, a tanulók képesek lesznek felismerni az egyes technikák egymáshoz való viszonyát, és így a „nehezebb” stratégiák is elérhetőbbé vállnak számukra.

Romániában például az elmúlt években a középiskolai tanterv nem írta elő a dinamikus programozás tanítását. Ezzel szemben a tanulók már a tantárgyi versenyek megyei szakaszain is olyan feladatokkal találták szembe magukat, amelyek e technika alkalmazását feltételezték. A legtöbb tanár plusz órákon próbálta meg bevezetni versenyzőit ezen algoritmustervezési stratégia rejtélyeibe. Ettől az évtől kezdve azonban az új tanterv lehetővé teszi a dinamikus programozás tanítását is. Tekintettel e technika „mélységére”, igazi kihívást jelent a középiskolában egy teljes osztálynak tanítani. Amint látni fogjuk, a felülnézet módszer oly módon terjeszthető ki a dinamikus programozás területére, hogy az átlagos képességű tanulók is át tudják látni a lényegét.

Először a problémamegoldásról ejtünk néhány szót (az „Algoritmustervezés felülnézetből” módszer egyik célja a problémamegoldó gondolkozás fejlesztése), majd részletesen kifejtjük magát a felülnézet módszert, illetve annak a programozási technikák területére való alkalmazását. Ezt követően egy tanmenetet javaslunk, illetve megmutatjuk, hogyan implementálható a felülnézet módszer e tanmenet segítségével. Külön kitérünk a felülnézet módszernek a dinamikus programozás területére való sajátos alkalmazására. Végül beszámolunk egy kísérleti mérésről, amely empirikusan igazolja a módszer hatékonyságát.

2.1 A problémamegoldó gondolkodás fejlesztése

A XXI. század elején számos ország természettudományos oktatásának egyik megoldatlan gondja a problémamegoldás [20] és az ismeretek gyakorlati alkalmazásának nem megfelelő színvonala. A nemzetközi felmérések azt mutatják, hogy számos diák esetében nehézkesen működik a megszerzett ismeretek teljesítményképes tudássá való átalakítása. Ezt igazolja az is, hogy az 1995-ös TIMSS (Third International Mathematics and Science Study) felmérésben a 18 éves magyar tanulók a természettudományos ismeretek alkalmazása terén 21 ország közül a 18. helyen végeztek. Ezzel szemben a világban egyre nagyobb hangsúlyt kap korunk embereszménye, a rohanó élethez rugalmasan alkalmazkodó, kommunikatív, a problémákat felismerni és megoldani képes, gyors döntéskészséggel rendelkező személyiség.

Az iskola fontos feladata, hogy a tanulókat felkészítse az életre. Az ismeretek megszerzésén túl kialakítsa azokat a személyiségjegyeket, amelyek képessé teszik őket az életben való helytállásra, a problémák megoldására.

A feladatok makrostruktúráján azokat a stratégiai lépéseket értjük, melyeken a tanuló végighaladva eljut a megoldáshoz. A megoldási stratégiákra vonatkozóan több nézet vált ismertté [21, 22, 23, 24], melyek közül Pólya György elmélete ma is helytálló. Eszerint a problémamegoldás fő lépései:

- a feladat megértése;
- tervkészítés;
- a terv végrehajtása;
- a megoldás vizsgálata.

A feladatok *megoldási stratégiájára* lefordítva ez a következőket jelenti:

- A probléma felvetése és megértése, mely egybeesik a szükséges információk megadásával.
- Részproblémák megfogalmazása, megértése a szükséges háttérinformációk elemzésével.
- A cél, azaz a megoldás eléréséhez szükséges gondolkodási műveletek mozgósítása. Mindez egy feltételezett megoldáshoz vezet, és megerősítés hiányában az is marad. Ebben a szakaszban jelentős szerepe van a találgatásnak, illetve a rugalmas gondolkodásnak.
- A cél, a megoldás elérése. Ebben fontos szerep jut a belátásnak.
- Megerősítés, igazolás. Ez vagy megelégedettséggel jár, vagy egy újabb folyamatot indít be.

A bemutatandó módszer, az egyes programozási technikák területéhez tartozó feladatok makrostruktúrájából kiindulva, számos feladatmegoldó stratégiát javasol.

A problémamegoldási folyamat mikrostruktúráján azokat a *gondolkodási műveleteket* értjük, melyeket a probléma megoldása érdekébe alkalmazni kell. Ilyenek az analízis, szintézis, absztrakció, összehasonlítás, elvont adatok összehasonlítása, összefüggések felfogása, kiegészítés, általánosítás, konkretizálás, rendezés és analógia. E sorrend egyben összetettségük sorrendjét is jelenti. Például, az analógia (amikor valamely jelenséget összefüggésbe hozunk egy már régebben ismert tárggyal vagy jelenséggel azon

az alapon, hogy a két tárgy vagy jelenség hasonló tulajdonságokkal rendelkezik) magába foglalja az összefüggések felfogását és a kiegészítés gondolkodási műveleteit, amelyeket egymás után alkalmaz [25]. Az „Algoritmustervezés felülnézetből” módszer azzal, hogy szinte mindegyik gondolkodási műveletet aktivizálja, erőteljesen hozzájárul a problémamegoldó gondolkodás fejlesztéséhez.

2.2 A felülnézet módszer általános leírása

Mit jelent „felülről látni” valamit? Képzeljük el a következő helyzeteket: A rendőrségen, egy bűneset kapcsán a különböző forrásokból érkező bizonyítékokat feltűzik egy hirdetőtáblára. Miért? A polgármesteri hivatal városrendezésért felelős szakosztálya elkészíti a város makettjét és körbeállják. Miért? Azért, hogy átfogó képet kapjanak az „egész”-ről, valamint, hogy jobban érzékelhetőek legyenek az egyes „részek” közötti hasonlóságok, különbségek, illetve kapcsolatok.

A két esetben különböző módon alakították ki az illetékesek a felülnézetet. A rendőröknek szükségük volt egy „platformra” (a tábla), amelyen elhelyezve a bizonyítékokat, „egymás mellett” láthatták őket. A műépítészek kicsinyítést és absztrakciót használtak a makett elkészítéséhez. Az absztrakció azért fontos, mert elvonatkoztat attól, ami a tanulmányozás szempontjából lényegtelen.

A két módszer ötvözéséből látható, hogy a felülnézet kialakításához szükséges lehet egy úgynevezett „absztrakt platform”, amelyen az elemzett entitások úgy helyezhetők egymás mellé, hogy szembetűnővé váljanak a vizsgálat szempontjából lényeges tulajdonságok és kapcsolatok.

Azt, hogy mit fogunk felülnézet alatt érteni ebben a dolgozatban, a következőképpen lehetne összefoglalni:

1. „Egymás mellett” látjuk a vizsgálat tárgyát képező entitásokat.
2. Csak az látszik, ami a vizsgálat szempontjából lényeges.
3. Nyilvánvalóak a hasonlóságok és a különbségek, szembetűnők a kapcsolatok.

A módszer célja, olyan helyzetbe juttatni a tanulót, hogy rálátása legyen a tanulmányozás alatt álló anyagra. Természetesen, különböző felülnézetek alakíthatók ki attól függően, hogyan valósítjuk meg az említett lépéseket. Ez kívánatos is, hiszen minden újabb felülnézet egy másik szempontot jelent. A módszer egyik erősségének számít az is, hogy segít a diákoknak a vizsgált entitások egymáshoz viszonyított „értékét” is felmérni. Például, az algoritmustervezési stratégiák tanulmányozása esetén nyilvánvalóvá válnak az egyes technikák erős, illetve gyenge pontjai, valamint az, hogy adott helyzetben melyiknek az alkalmazása a legcélszerűbb és miért.

Az alábbi egyszerű kísérlettel jól ellenőrizhető a módszer hatékonyságát igazoló alapelv: Mutassunk fel egy papírlapot, és kérjük meg a tanulókat, hogy nevezzék meg minél több tulajdonságát. Ezután egy másik osztályban ismételjük meg a kísérletet, de úgy, hogy a papírlappal együtt egy másik alakzatot is felmutatunk (például, ami fából készült, nem síkidom, nem egyszínű stb.).

Mit fogunk tapasztalni? Az utóbbi osztályban a papírlapnak számottevően több tulajdonsága fogalmazódik meg a tanulókban. Nem biztos például, hogy az első

osztályban felfigyelnek arra, hogy az ív egyszínű, síkidom, összegyűrhető, nyersanyaga fa stb.. Ez az egyszerű kísérlet egy rég ismert igazságot emel ki: *Az ellentétek felhívják a figyelmet úgy magukra, mint a hasonlóságokra.*

Mi volt a tanár szerepe ebben a kísérletben? Az, hogy a diák tudatában kialakítsa a felülnézetszerű rálátást. Ez azt feltételezte, hogy az ív mellé helyezett tárgyat úgy válassza meg, hogy szembeötlővé váljanak azok a szempontok, amelyek alapján szeretné, hogy a tanulók az összehasonlítást elvégezzék.

2.3 Felülnézetek az informatikaoktatásban

Hogyan lehet felülnézeteket kialakítani informatikaórán? A legtöbb tanár megteszi ezt – még ha nem is így nevezi –, amikor a rendezéseket tanítja. A fejezet végén veszünk egy konkrét számsorozatot, amelyen kipróbáljuk, vagy kipróbáltatjuk a tanulókkal az összes megtanított rendezési algoritmust, és felhívjuk a figyelmet a hasonlóságokra, valamint a különbségekre. Amikor így járunk el, felülnézetet alakítunk ki a diákokban, hiszen:

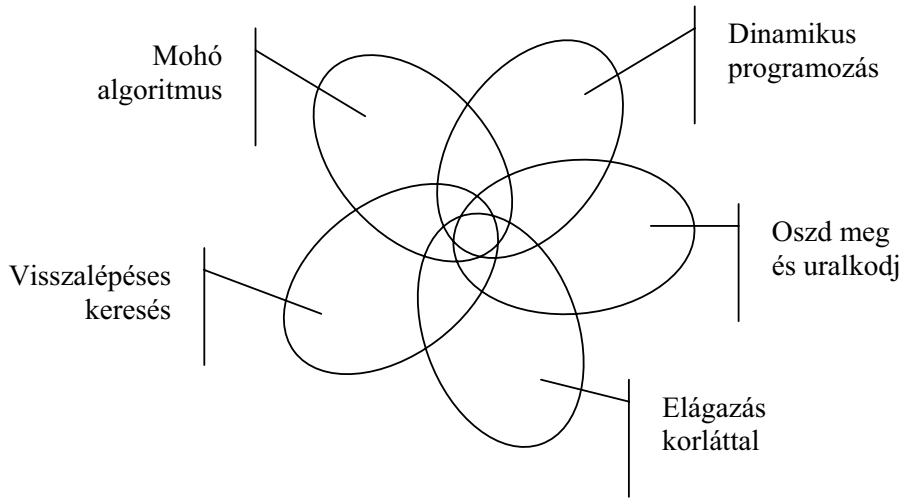
- „Egymás mellé" helyezzük a rendezési algoritmusokat azáltal, hogy ugyanazon a számsorozaton próbáljuk ki őket.
- Felhívjuk a diákok figyelmét arra, hogy egy adott felülnézetből mi lényeges és mi nem. Az összehasonlításon alapuló rendezések esetén például, arra összpontosíthatnak a diákok egy adott felülnézetből, hogy milyen stratégiák szerint történik az elemek hasonlítgatása.
- Segítünk a tanulóknak meglátni a hasonlóságokat és a különbségeket, az algoritmusok erősségeit és gyenge pontjait.

Kitűnő számítógépes szimulációk léteznek, amelyek megkönnyíthetik a felülnézet kialakítását a rendezési algoritmusok esetén.

2.4 A felülnézetmódszer és az algoritmustervezési stratégiák

Hogyan alakíthatunk ki felülnézeteket az algoritmustervezési stratégiák tanításakor? A kihívás abban áll, hogy amíg a rendezési algoritmusok ugyanazt a feladatot oldják meg, addig minden egyes algoritmustervezési stratégiának többé-kevésbé megvan a saját felségterülete. A *2.1 ábra* ezt szemlélteti. Az egyes körök azon feladatok halmazát jelentik, amelyek megközelíthetők az illető stratégiával, függetlenül attól, hogy optimális megoldást nyújtanak-e vagy sem, hatékony-e az algoritmus vagy nem.

A metszetek azt szemléltetik, hogy léteznek olyan feladatok, amelyek több technikával is megoldhatók, sőt egyesek mindegyikkel. Ebből arra következtethetünk, hogy léteznie kell egy közös „síknak", amelyen az egyes technikák feladathalmazai alapvető hasonlóságokat mutatnak. Melyik ez a „sík"?



2.1 ábra

2.4.1 Egy „absztrakt platform”

A visszalépéses keresés és az oszd meg és uralkodj technikák általában rekurzívan közelítik meg a feladatot. A rekurzió gondolatmenete a következő: A feladatot visszavezetjük hasonló, egyszerűbb részfeladatokra, majd ezeket további hasonló, még egyszerűbb részfeladatokra, egészen addig, míg triviális részfeladatokhoz nem jutunk. Ez a fajta lebontás azt feltételezi, hogy a feladatnak felépítésében fastruktúrája legyen. A gyökércsomópont nyilván magát a feladatot képviseli, az első szint csomópontjai azokat a részfeladatokat, amelyekre a feladat első lépésben lebontható, és így tovább. Végül a fa levelei fogják megadni a lebontásból adódó triviális részfeladatokat.

A mohó, a dinamikus programozás és az elágazás korláttal technikák egyik közös vonása, hogy általában olyan feladatok esetében alkalmazzuk őket, amelyek döntéssorozatként foghatók fel. Ez ismét egy fastruktúrához vezet, amelyben a gyökér a feladat kezdeti állapotát jelképezi, az első szint csomópontjai azokat az állapotokat, amelyekbe a feladat az első döntés nyomán kerülhet, a második szinten lévők azokat, amelyek a második döntésből adódhatnak stb.. Egy csomópontnak annyi fia lesz, ahány lehetőség közül történik a választás az illető döntés alkalmával.

Mindezt figyelembe véve elmondható, hogy az összes bemutatandó módszert elsősorban olyan feladatok esetében alkalmazzuk, amelyek valamilyen értelemben fastruktúrával rendelkeznek. A technikák szempontjából ez azt jelenti, hogy mindegyik úgy tekinti a feladatot, mint egy fát. Nos, ez a fastruktúra az a közös „sík”, vagy „absztrakt platform”, amelyen a technikák egymás mellé helyezhetők, és amely a felülnézet kialakításához szükséges.

2.4.2 Egy átfogó kép a felülnézet módszerről működés közben

Az alábbiakban egy példafeladaton keresztül bemutatjuk a módszer mindhárom szakaszát. Ez segíteni fog abban, hogy jobban átlássuk a módszer lényegét.

A felülnézet módszer alkalmazásának első lépése a vizsgálat tárgyát képező entitások – jelen esetben a programozási technikák – egymás mellé helyezése. A legegyszerűbben ezt úgy tudjuk megtenni, ha ugyanazt a feladatot oldjuk meg mindegyik technikával.

Sokatmondó felülnézet alakítható ki, ha feladatnak az alábbi optimalizálási[1] problémát választjuk, amelyet 1994-ben javasoltak megoldásra a Svédországban megrendezett Nemzetközi Informatikai Olimpián:

Egy *n* soros négyzetes mátrix főátlóján és főátló alatti háromszögében természetes számok találhatók. Feltételezzük, hogy a mátrix egy *a* nevű kétdimenziós tömbben van. Határozzuk meg azt a „leghosszabb" utat, amely az *a[1,1]* elemtől indul és az *n*-dik sorig tart, figyelembe véve a következőket:

- egy úton az *a[i,j]* elemet az *a[i+1,j]* elem (függőlegesen le) vagy az *a[i+1,j+1]* elem (átlósan jobbra le) követheti, ahol $1 \leq i < n$ és $1 \leq j < n$.
- egy út „hossza" alatt az út mentén található elemek összegét értjük.

Például, *n = 5* esetén az alábbi mátrixban a csúcsból az alapra vezető „leghosszabb" utat besatíroztuk. Ennek az útnak a „hossza" *37*:

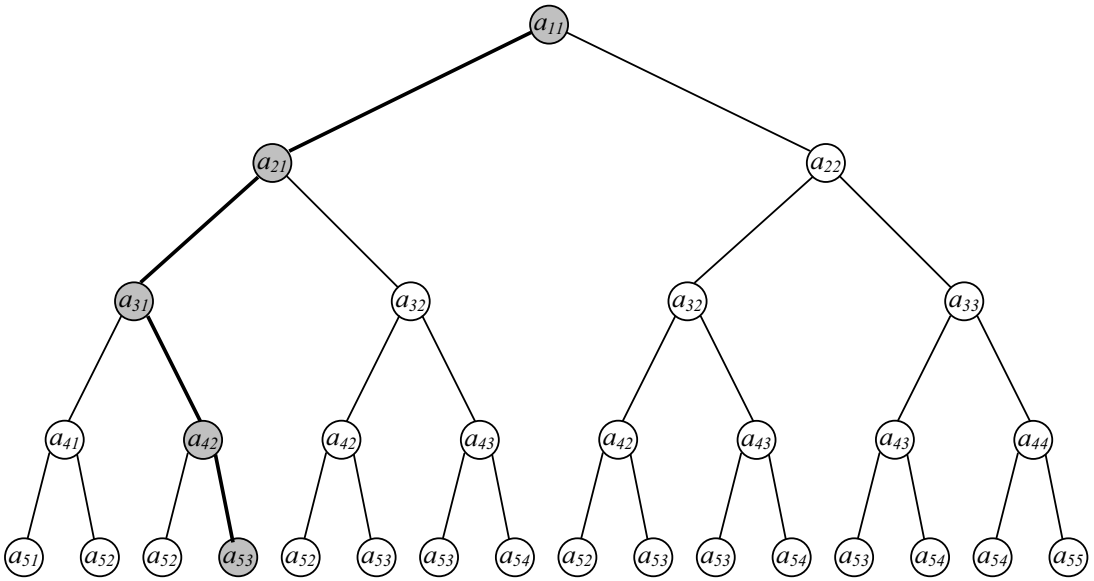
7				
5	9			
10	1	4		
2	7	3	1	
2	5	8	3	1

A felülnézet kialakításának második lépése az absztrakció. Amint azt az előbbiekben kifejtettük, ez megköveteli a feladat mögött meghúzódó fastruktúra azonosítását.

Elemezve a feladatot észrevehetjük, hogy egy olyan optimalizálási problémáról van szó, amelyben az optimális megoldáshoz *n-1* döntés után juthatunk el, és mindegyik döntésnél *2* választásunk van (melyik irányba lépjünk tovább: függőlegesen le vagy átlósan jobbra). Minden döntéssel a feladat hasonló, de egyszerűbb feladatra redukálódik. Tehát az „absztrakt platform" szerepét az alábbi bináris fa (*2.2 ábra*) tölti be (egy döntési fáról van szó, amelyet megoldásfának is fogunk nevezni):

Az optimális megoldás meghatározása az optimális döntéssorozat megtalálását jelenti. Úgy is mondhatnánk, hogy meg kell találnunk a megoldásfa 2^{n-1} darab, a gyökértől a levélhez vezető útjai közül a „legjobbat". Más szóval, meg kell keresnünk a „legjobb levelét" a megoldásfának, azt, amelyikhez a „legjobb út" vezet.

[1] Leginkább az optimalizálási feladatok között találunk olyanokat, amelyeket mind az öt technika meg tud oldani, hiszen ez egy olyan terület, amellyel mindenik foglalkozik valamilyen szinten.



2.2 ábra

A megoldásfának egy alaposabb vizsgálata további észrevételekhez vezethet:

1. A megoldásfa csomópontjainak száma $1 + 2 + 2^2 + ... + 2^{n-1} = 2^n-1$. Ez azt jelenti, hogy bármely algoritmus, amely bejárja a teljes fát az optimális út megtalálásához, exponenciális bonyolultságú lesz.
2. Amíg a fa a teljes feladatot képviseli, addig a részfái azokat a hasonló, de egyszerűbb részfeladatokat (a levelek a triviális részfeladatokat), amelyekre ez lebontható. Konkrétan: az a_{ij} gyökerű részfa az *a[i,j]* elemtől az alapra vezető „leghosszabb út” meghatározásának feladatát adja.
3. A fenti ábra azt is kiemeli, hogy különböző döntéssorozatok azonos részfeladatokhoz vezethetnek, ami azt jelenti, hogy a megoldásfának vannak azonos részfái. Nem nehéz átlátni, hogy a különböző részfeladatok száma azonos a mátrix elemeinek számával, azaz $n(n+1)/2$. Tehát az algoritmus, amelynek sikerül elkerülnie az azonos részfeladatok többszöri megoldását, négyzetes bonyolultságú lesz.

Most már mindent előkészítettünk a háromlépéses módszer alkalmazásához. **Első lépésként** úgy érhetjük el, hogy a tanulók egymás mellett lássák a technikákat, hogy megfogalmazzuk nekik, miként alkalmazható sorra mindegyik stratégia a szóban forgó feladatra. Ha szeretnénk ezt játékossá tenni, meghívhatjuk az öt technikát egy képzeletbeli „talk show”-ra, hadd mutatkozzanak be ők maguk, elmagyarázva, miként közelítenék meg a bemutatott feladatot.

Mohó algoritmus

A csúcsból indulunk. Minden lépésben két lehetséges elem közül kell választani. Melyiket válasszuk? Természetesen mindig a nagyobbik elemet!

Megjegyzés: Mi a következménye a mohóságának? Jelen esetben az, hogy egyáltalán nem biztos, hogy megtalálja a legjobb utat. A példamátrix esetében a mohó út *31* hosszú lesz, és ez a következő: *a[1,1], a[2,2], a[3,3], a[4,3], a[5,3].*

Visszalépéses keresés

Generáljuk az összes olyan utat, amely a csúcsból az alapra vezet, és kiválasztjuk közülük a „legjobbat”.

Oszd meg és uralkodj

Észrevehető, hogy bármely *a[i,j] (i<n)* elemtől induló „legjobb út” meghatározása visszavezethető az *a[i+1,j]*, illetve az *a[i+1,j+1]* elemektől induló „legjobb utak” meghatározására. Ugyanis, amennyiben ezek rendelkezésre állnak, nem marad más hátra, minthogy „belépjünk” az *a[i,j]* elemmel a hosszabbik elé. Más szóval, először lebontjuk rekurzívan a feladatot egyszerűbb, majd még egyszerűbb részfeladatokra, végül pedig a „visszaúton” – figyelembe véve az előbbi észrevételt – felépítjük a legjobb utat.

Dinamikus programozás

Az alapötlet az, hogy kiindulva a triviális részfeladatok kézenfekvő megoldásaiból, felépítjük az egyre bonyolultabb részfeladatok megoldásait, míg végül el nem jutunk a fő feladat megoldásához. Mivel el szeretnénk kerülni az azonos részfeladatok többszöri megoldását, az optimális részmegoldásokat eltároljuk egy *c* kétdimenziós tömbbe (a *c[i,j]* tömbelem az *a[i,j]* elemtől az alapra vezető „legjobb út” hosszát fogja tartalmazni). A *c* tömb főátlóját és a főátló alatti háromszögét lentről felfelé, soronként töltjük fel, figyelembe véve, hogy

$$c[i,j] = a[i,j] + max(c[i+1,j], c[i+1,j+1]),\ i < n.$$

Végül a *c[1,1]*-ben lesz az optimális út hossza. Ahhoz, hogy meglegyen maga az út is, az szükséges még, hogy végigmenjünk a *c* tömbön mohó módon.

37				
30	25			
25	16	15		
7	15	11	4	
2	5	8	3	1

Elágazás korláttal

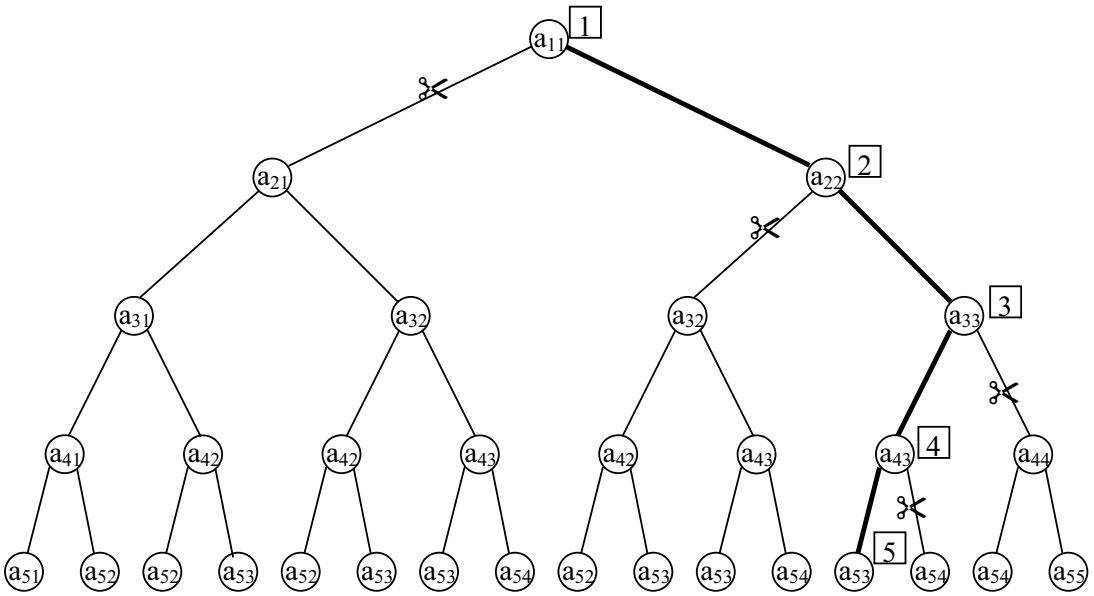
Párhuzamosan elindulunk minden csúcsból az alapra vezető úton lefelé, mindig abba az irányba lépve először, amelyik a legígéretesebb. Amelyik úton hamarabb érjük el az alapot, az a legjobb.

Megjegyzés: Bizonyítható, hogy e feladat esetében ez a megközelítés nem mindig vezet el az optimális megoldáshoz.

A második és a harmadik lépés: Elérkeztünk a felülnézet módszer csúcspontjához, a technikák közötti alapvető hasonlóságok és különbségek kiemeléséhez az „absztrakt platformnak” számító fastruktúrán. A következő két kérdéssel határozhatjuk meg az irányt a vizsgálódáshoz:

1.) Hogyan járják be és miként „metszik meg” a megoldásfát a különböző technikák?
(Az ábrákon négyzetekben megszámoztuk, hogy milyen sorrendben foglalkoznak, illetve metszik meg az egyes technikák a megoldásfa csomópontjait.)

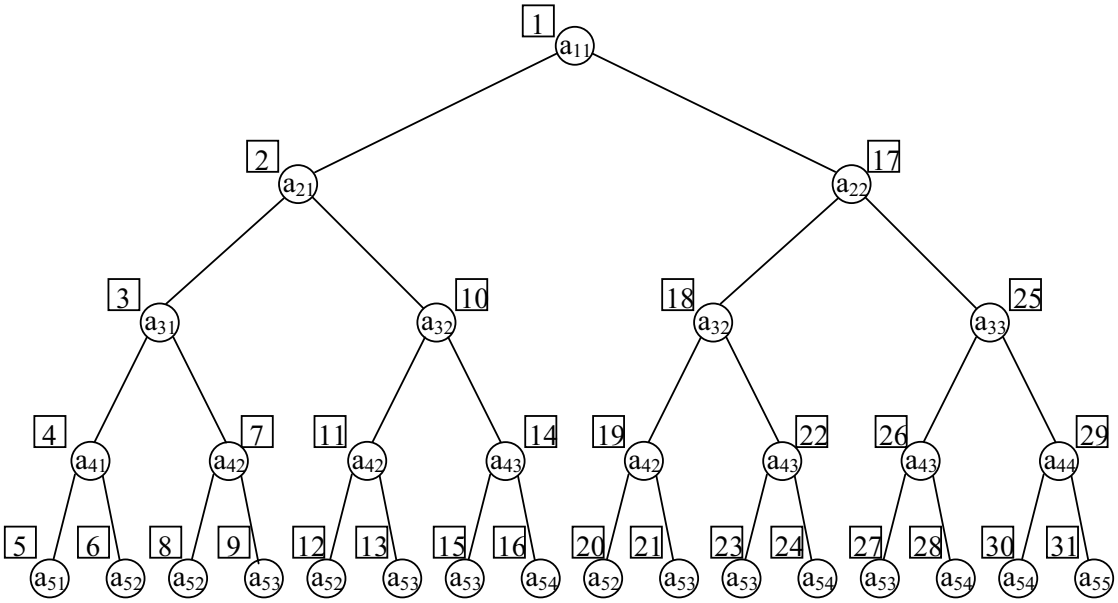
A *mohó* technika a gyökértől a levelek felé haladva metszi meg a fát, egész részfákat vágva le róla. Egyetlen gyökér-levél utat jár be, a leggyorsabb algoritmust adva ez által a kezünkbe (bonyolultsága lineáris), de sajnos - legalábbis jelen esetben - semmilyen garanciát nem tud nyújtani az optimális megoldás megtalálására. (*2.3 ábra*)



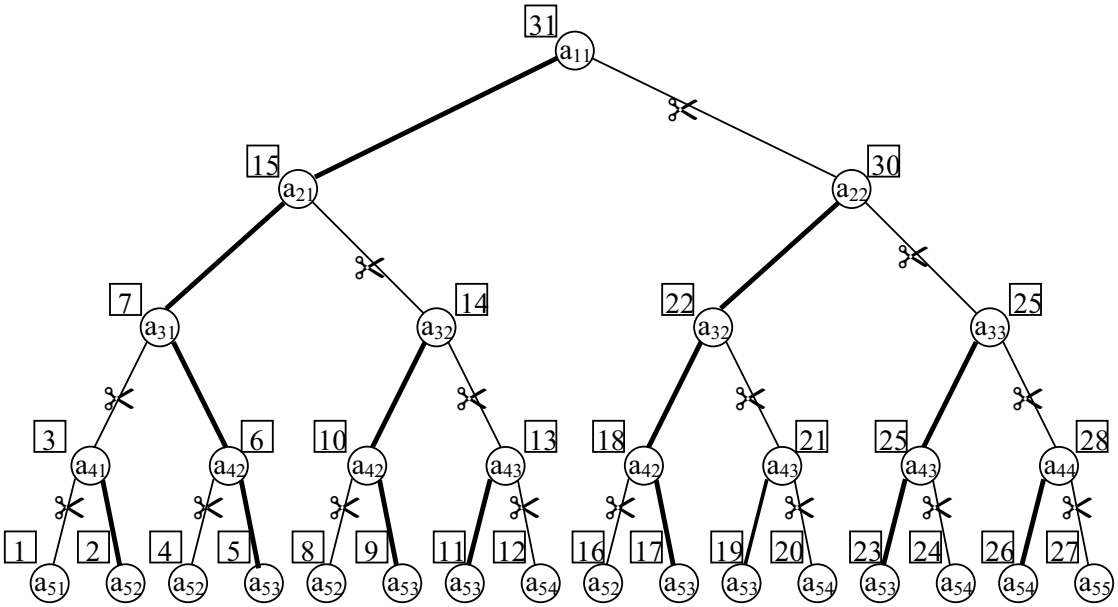
2.3 ábra

A *visszalépéses keresés* e feladat esetében egyáltalán nem metszi meg a fát, ugyanis mindegyik gyökér-levél útban egy lehetséges megoldást lát. Azért, hogy megtalálja ezeket, *mélységében* járja be a fát, *preorder* sorrendben foglalkozva a csomópontokkal. Mivel a fa csomópontjainak száma exponenciálisan függ *n*-től, algoritmusa egyértelműen exponenciális bonyolultságú lesz. (*2.4 ábra*)

Az *oszd meg és uralkodj* technika minden csomópontot megmetsz abban a sorrendben, ahogy a fa *postorder mélységi* bejárása szerint következnek. Mindegyik csomóponton pontosan egy ágat hagy meg: azt, amelyik az illető részfeladat optimális megoldását képviseli. Mivel a teljes fát bejárja, ezért az algoritmusa is nyilvánvalóan exponenciális bonyolultságú. (*2.5 ábra*)



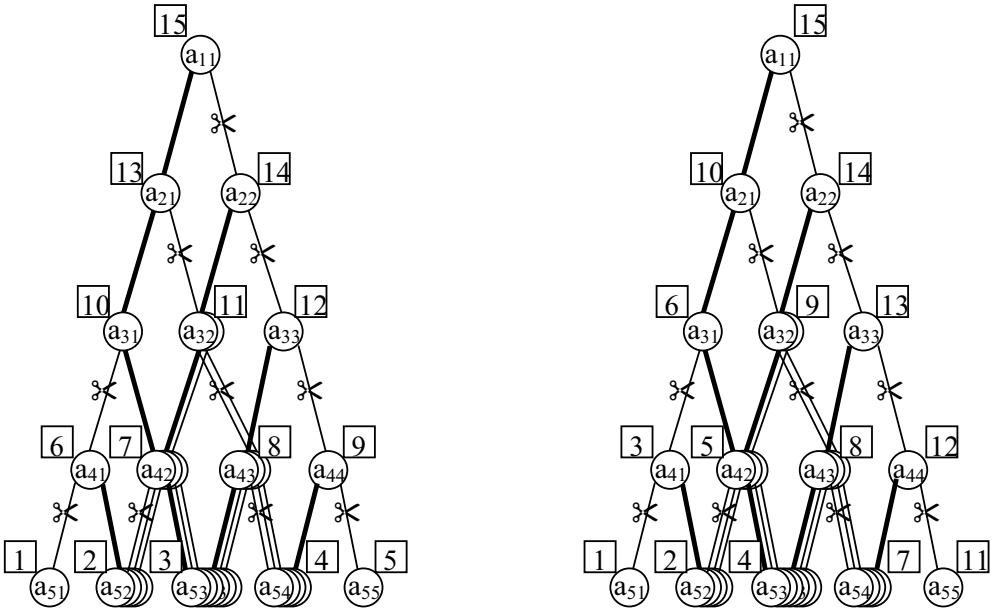
2.4 ábra



2.5 ábra

A *dinamikus programozás* technika először egymásra helyezi az azonos részfákat, létrehozva ezáltal egy „összevont döntési fát”, amely csak a különböző részfeladatokat képviselő csomópontokat tartalmazza (ezért a dinamikus programozás akkor tudja igazán értékesíteni erősségeit, ha sok az identikus részfeladat). Ezt követően mindegyik csomópontról lemetszi a száraz ágakat, akárcsak az oszd meg és uralkodj technika. (*2.6 bal ábra*)

Milyen sorrendben metszi meg a dinamikus programozás az összevont döntési fa csomópontjait? Hagyományos változatában lentről felfelé halad végig a fán, szintről-szintre. A rekurzív megvalósítás ugyanabban a sorrendben foglalkozik a csomópontokkal, mint az oszd meg és uralkodj technika, csak kihagyja az ismétlődőket. Tekintettel arra, hogy az összevont döntési fa csomópontjainak száma *n(n+1)/2*, algoritmusának bonyolultsága négyzetes. (*2.6 jobb ábra*)



2.6 ábra

2.) Hogyan építik fel a megoldást a különböző technikák?

A *mohó,* illetve a *visszalépéses keresés* technikák *fentről-lefelé* keresik meg a megoldást, illetve a megoldásokat, levélben adva meg az eredményt.

Az *oszd meg és uralkodj* és a *dinamikus programozás* technikák pontosan fordítva, *lentről felfelé* keresik meg az optimális megoldást. Mindkét technika a gyökérben adja meg az eredményt: a dinamikus programozás hagyományos változata felérkezve, az oszd meg és uralkodj pedig visszaérkezve ide. Az oszd meg és uralkodj és a dinamikus programozás rekurzív változatai között csak annyi a különbség, hogy az első egymástól függetlenül oldja meg a részfeladatokat, és ezért nem veszi észre, ha ismétlődnek, a második viszont valahányszor megold egy részfeladatot, eltárolja a megoldását, így ha újból találkozik ugyanazzal a részfeladattal, a megoldása már rendelkezésre áll.

A *visszalépéses keresés* és az *oszd meg és uralkodj* technikák közös vonásai, hogy mindkettő mélységébe járja be a megoldásfát. Mivel ellentétes irányban építik a megoldást, ezért az egyik preorder, a másik viszont postorder sorrendben látogatja meg a csomópontokat. Ez magyarázata annak is, hogy miért ad meg a visszalépéses keresés több lehetséges megoldást is (amelyek közül ki kell választania az optimálist), az oszd meg és uralkodj technika pedig miért épít fel csupán egyet, az optimálist: a fa lefelé terebélyesedik, felfelé viszont szűkül, sok levele, de egyetlen gyökere van.

Mi a másik alapvető különbség a *mohó* és a *dinamikus programozás* technikák között? Az első *optimális döntésekkel*, a második pedig *optimális részmegoldásokból* próbálja felépíteni az optimális megoldást. Ezért a mohó megközelítés csak akkor kielégítő, ha bizonyítani tudjuk, hogy az adott feladat esetében az optimális döntések tényleg optimális megoldáshoz vezetnek (a globális optimum lokális optimumokat feltételez). A dinamikus programozás alkalmazhatóságának feltétele pedig az, hogy a feladat optimális megoldása valamilyen módon felépíthető legyen a részfeladatok optimális megoldásaiból (érvényes legyen az optimalitás alapelve).

A fenti bemutató természetesen csak azt a célt szolgálja, hogy a diákok fogalmat alkothassanak a megvizsgálandó technikákról, és egymás mellett lássák őket, amint ugyanazzal a feladattal küzdenek.

2.4.3 Javasolt tanmenet

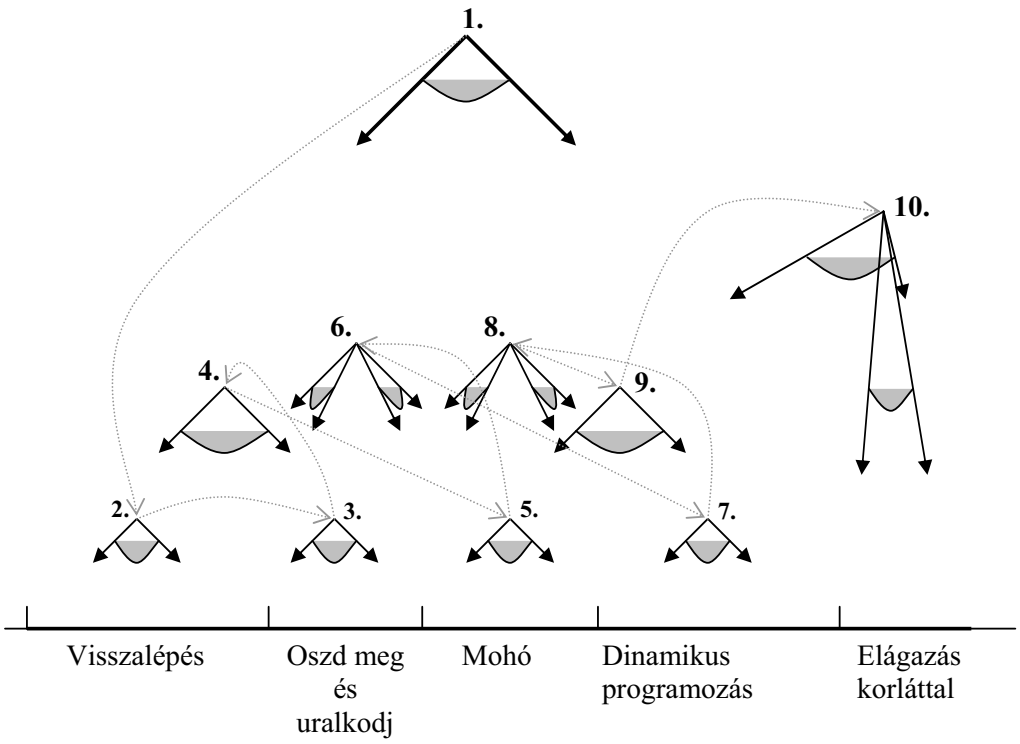
A következőkben a felülnézet módszer alkalmazásához egy tanmenetet javasolunk. A 2005/2006-os tanévtől a Sapientia Erdélyi Magyar Tudományegyetem marosvásárhelyi karán az alábbi tanterv szerint történik az algoritmus tervezési stratégiák oktatása egy teljes féléven (14 hét) keresztül. A marosvásárhelyi Bolyai Farkas Elméleti Líceumban már évek óta sikeresen alkalmazzák a felülnézet módszert, egyes osztályokban a visszalépéses keresés, az oszd meg és uralkodj és a mohó technikák tanítására, másokban pedig a dinamikus programozást is hozzávéve. A felülnézet módszernek a javasolt tanmenet szerinti alkalmazása feltételezi, hogy a tanulók rendelkeznek alapvető programozói készséggel, és tisztában vannak a rekurzió és a fastruktúra fogalmakkal. Különösen fontos, hogy ismerjék a gyökeres fák preorder, inorder és postorder mélységi bejárásait, valamint a fák szélességi bejárását. Ha a hivatalos tanterv e gráfelméleti fogalmakat későbbre teszi, célszerű egy-két órát szánni ezek bemutatására anélkül, hogy elmélyülnénk a gráfelméletben.

1. A rekurzió átismétlése, a fastruktúrák és bejárásaiknak bemutatása.
2. Egy feladaton keresztül, amely mindegyik algoritmussal megoldható, általános és átfogó képet nyújtunk a technikákról. Anélkül, hogy valóban megoldanánk a feladatot, az osztállyal történő beszélgetés formájában körvonalazzuk az egyes stratégiákat *(1. hét)*.
3. Bemutatjuk a visszalépéses keresés technikát *(2.-3. hét)*.
4. Elmélyítjük a visszalépéses keresés stratégiát.
   - Speciális, visszalépéses kereséssel megoldható feladatok gyakorlása.
5. Bemutatjuk az oszd meg és uralkodj technikát *(4. hét)*.
   - Elmélyítjük az oszd meg és uralkodj stratégiát.
   - Speciális, oszd meg és uralkodj módszerrel megoldható feladatok gyakorlása.
6. *Oszd meg és uralkodj vagy visszalépéses keresés (5. hét)*.
   - Olyan feladatokat választunk, amelyek mindkét módszerrel megoldhatók.
7. Bemutatjuk a mohó algoritmust *(6. hét)*.
   - Elmélyítjük a mohó stratégiát.
   - Speciális, mohó módszerrel megoldható feladatok gyakorlása.
8. *Visszalépéses keresés és mohó algoritmus (7. hét)*.
   - Olyan feladatokat választunk, amelyek kiemelik, hogyan egészítheti ki a két módszer egymást.

9. Bemutatjuk a dinamikus programozás módszerét *(8.-10. hét)*.
   - Elmélyítjük a dinamikus programozás módszerét.
   - Speciális, a dinamikus programozás módszerével megoldható feladatok gyakorlása.
10. *Oszd meg és uralkodj vagy dinamikus programozás (11. hét)*.
   - Olyan feladatokat választunk, amelyek mindkét módszerrel megoldhatók.
   - A dinamikus programozás rekurzív változata.
11. *Mohó módszer vagy dinamikus programozás (11. hét)*.
   - Olyan feladatokat választunk, amelyek mindkét módszerrel megoldhatók.
12. *Beágyazzuk az elágazás korláttal technikát az előbbi módszerek alkotta képbe (12.-13. hét)*.
   - Áttekintjük a négy bemutatott technika fő jellegzetességeit és rámutatunk, hogy milyen „űrt" tölt be az elágazás korláttal technika a kialakult képben.
   - Elmélyítjük a stratégiát specifikus, elágazás korláttal megoldható feladatokkal.
13. *Határesetek a programozási technikák világában (14. hét)*.
   - Áttekintjük felülnézetből a teljes anyag felépítését.
   - Néhány „határeset algoritmus" megvizsgálásával teljesebbé tesszük a képet.

Az alábbiakban bemutatjuk, miként tölthető meg e tanmenet tartalommal oly módon, hogy a tanulók valóban felülnézeti rálátást nyerjenek a programozási technikák világára.

2.4.3.1 A programozási technikák világa felülnézetből



2.7 ábra

A bevezető órán (*1. hét*) tekintsük át - mintegy operatőr kameráján keresztül -, hogyan pásztázza végig a programozási technikák jelképes világát a fentebb javasolt tanmenet. Ehhez használhatjuk a *2.7 ábrát* és a csatolt leírást (az ábra a képzeletbeli kameránk mozgását ábrázolja).

Az első egység madártávlatból ad átfogó képet az öt bemutatandó technikáról, egyelőre csak körvonalazva stratégiáikat. A második egységben a visszalépéses kereséssel megoldható feladatokra fókuszálunk, hogy a tanulók részletes képet kapjanak róla. A harmadik egységben képzeletbeli kameránk fókusza átkerül az oszd meg és uralkodj algoritmusokra. A negyedeik egység nyújtja az első igazi felülnézetet az első két technika párhuzamba állításával. Itt tárulnak fel a diákok előtt a visszalépéses keresés és az oszd meg és uralkodj módszer közötti alapvető, illetve árnyalati hasonlóságok és különbségek. Az ötödik egység a mohó algoritmust állítja a figyelem középpontjába, majd a hatodik egységben kameránk újra felemelkedik, hogy a diákok együttesen láthassák a mohó és a visszalépéses keresés stratégiákat. A hetedik egységben megpróbáljuk átkutatni a dinamikus programozás felségterületét. E mélyreható vizsgálódást további két felülnézet egység egészíti ki: először az oszd meg és uralkodj módszerrel, majd pedig a mohó stratégiával hasonlítjuk össze a dinamikus programozást. A tizedik egységben más „filmezési technikát" alkalmazunk. Kameránkkal újra végigpásztázva a már bemutatott négy technikán, a diákok figyelmét az optimalizálási feladatok világának egy olyan részére irányítjuk, amely egyik addigi országhoz sem tartozott igazán. Végül odafókuszálva úgy azonosítjuk e titokzatos területet, mint az elágazás korláttal feladatok külön világának csücskét.

Ezt követően -, hogy életet leheljünk ezen előzetes bemutatóba, – hagyjuk, hogy maguk a technikák mutatkozzanak be a diákoknak. Ehhez kitűnően felhasználható a 2.4.2 alfejezetben bemutatott feladat és az azt követő „talk show".

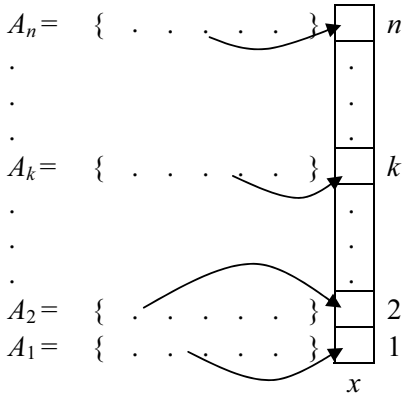
Az egyes technikákat bemutató órákon hangsúlyozni kell, mit jelent az illető stratégia szempontjából fastruktúraként felfogni egy feladatot. A felülnézet órákra (dőlt betűvel jeleztük a tanmenet e leckéit) olyan feladatokat választunk, amelyek megoldhatók mindkét összehasonlítandó módszerrel. Ezek azok az órák, amelyeken kihangsúlyozzuk a stratégiák közti hasonlóságokat, különbségeket és kapcsolatokat. Hogyan valósítható mindez meg? (Célunk itt, természetesen, csak ízelítőt adni.)

A feladat fastruktúrája „visszalépéses keresés szemmel" (2.-3. hét)

A visszalépéses keresés technika hagyományos változatában alapvetően a következő feladatot oldja meg: Egy ismert halmazsorozatból ($A_1, A_2, \ldots, A_n$) az összes lehetséges módon összeválogat elemeket - halmazonként egy-egy elemet -, szem előtt tartva, hogy az összeválogatott sorozatok (vektorok) elemei közt teljesülnie kell bizonyos feltételeknek. *(2.8 ábra)*

A visszalépéses keresés stratégiája jobban nyomon követhető, ha a feladatot fastruktúra formájában ábrázoljuk. A fa gyökeréből n_1 első szinti fiú csomópont ágazik le, amelyekhez az A_1 halmaz elemeit rendeljük. Minden első szinti választáshoz a második szintre n_2-féleképpen választhatunk elemet az A_2 halmazból. A fán ez abban tükröződik, hogy minden első szinti csomópontnak n_2 fia lesz a második szinten, amelyekhez az A_2 halmaz elemeit rendeljük. Ez összesen $n_1 \times n_2$ második szinti csomópontot eredményez. Végül az n-edik szinten $n_1 \times n_2 \times \ldots \times n_n$ csomópontja, más szóval levele lesz a fának.

Általános szabályként megjegyezhető, hogy ha a csomópont a fa *k*-adik szintjén található, és az az apa csomópontjának az *i*-edik fia, akkor az A_k halmaz *i*-edik elemét rendeljük hozzá. Mivel a fa bármely csomópontjához a gyökérből pontosan egy út vezet, ezért elmondható, hogy minden csomópont képviseli ezt az utat. A levelek mindegyike a halmazok Descartes-szorzatának valamelyik elemét képviseli, azt, amelyiket a gyökérből az illető levélhez vezető út ad meg.

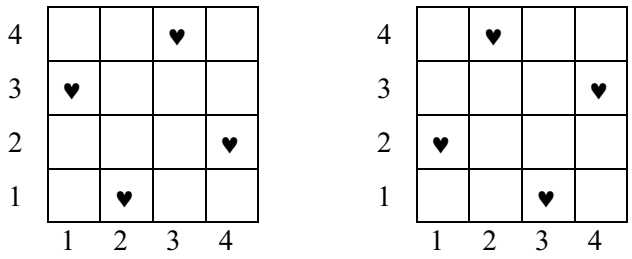


2.8 ábra

Mik adják a fán a feladat megoldásvektorait? Ha a megoldások a Descartes-szorzat adott tulajdonságú elemei, akkor ezeket a megfelelő gyökér-levél utak adják. Nevezzük ezeket megoldásutaknak vagy megoldáságaknak, a hozzájuk tartozó leveleket pedig megoldásleveleknek. Egyes feladatokban a megoldásokat nem feltétlenül gyökér-levél utak adják, hanem a gyökértől, adott tulajdonságú csomópontokhoz vezető utak. Tehát általános esetben megoldáscsomópontokról beszélünk. Ezek után nem nehéz belátni, miért nevezik a feladathoz rendelt fát a megoldások terének.

Szolgáljon példaként a „Királynők” feladata: *Helyezzünk el n királynőt egy n × n méretű sakktáblán úgy, hogy ne támadják egymást.*

A *2.9 ábra n = 4* esetén adja meg a megoldásokat:



2.9 ábra

Hogyan fogható fel visszalépéses keresés feladatként az n királynő feladata?

A megoldások *n* elemű vektorokként (x_1, x_2, ... , x_n) kódolhatók: az *i*-edik királynőt az *i*-edik sorkoordinátájú és az x_i-edik oszlopkoordinátájú négyzetre tesszük.

Például, ha $n = 4$, a megoldások kódjai:

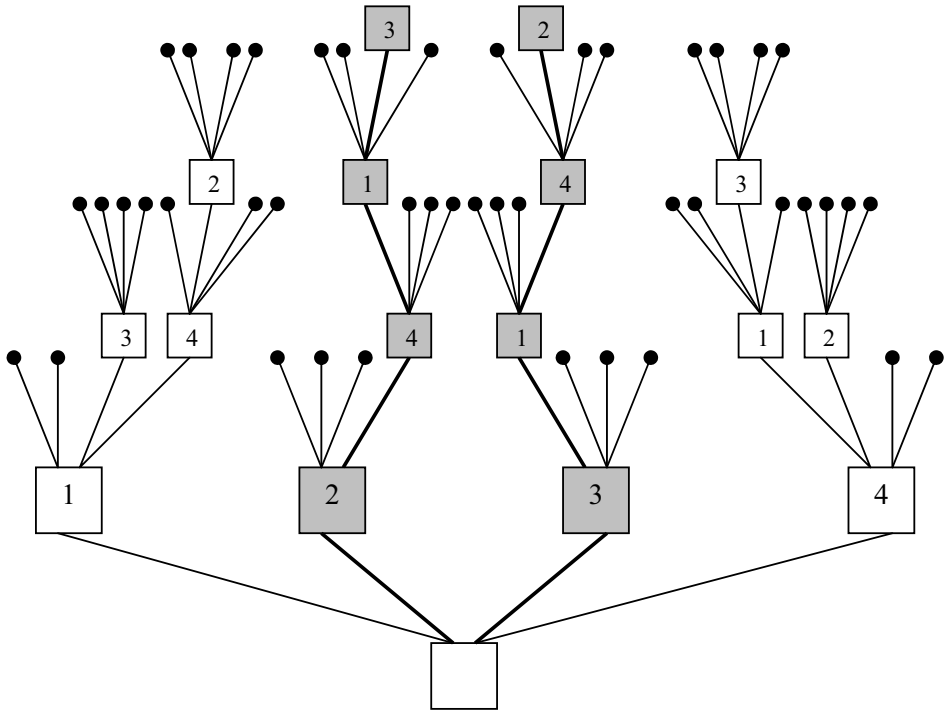
1. (2, 4, 1, 3)
    1. királynő: 1. sor 2. oszlop
    2. királynő: 2. sor 4. oszlop
    3. királynő: 3. sor 1. oszlop
    4. királynő: 4. sor 3. oszlop
2. (3, 1, 4, 2)
    1. királynő: 1. sor 3. oszlop
    2. királynő: 2. sor 1. oszlop
    3. királynő: 3. sor 4. oszlop
    4. királynő: 4. sor 2. oszlop

Az alapábra:

$A_4 = \{ 1, 2, 3, 4 \}$ 4
$A_3 = \{ 1, 2, 3, 4 \}$ 3
$A_2 = \{ 1, 2, 3, 4 \}$ 2
$A_1 = \{ 1, 2, 3, 4 \}$ 1
x

2.10 ábra

A feladat mögött meghúzódó fastruktúra:[2] *(2.11 ábra)*



2.11 ábra

[2] Helyhiány miatt, nem rajzoltuk le a teljes fát (a 4-edik szinten 256 levele van). Csak azt a részfát ábrázoltuk, amely a lehetséges megoldásleveleket tartalmazza (ezt nevezhetnénk *a fa élő részé*nek). A jobb áttekinthetőség kedvéért bejelöltük az „*élő* csomópontok" „*száraz* irányba" vivő fiait is [11].

Mindezeket figyelembe véve egy visszalépéses keresési feladat az alábbi módon is megfogalmazható: Keressük meg a feladathoz rendelhető fa megoldáscsomópontjait, illetve építsük fel a gyökérből ezekhez vezető megoldásutakat, amelyekhez éppen a megoldásvektorok vannak rendelve.

A feladat fastruktúrája „oszd meg és uralkodj szemmel” (4. hét)

Az oszd meg és uralkodj technika segítségével olyan feladatok oldhatók meg, amelyek visszavezethetők, más szóval lebonthatók két vagy több hasonló, de egyszerűbb (kisebb méretű) részfeladatra. Ezen részfeladatok hasonlóak lévén az eredeti feladathoz, maguk is visszavezethetők további hasonló, de még egyszerűbb részfeladatokra. Addig járunk így el, míg triviális részfeladatokhoz jutunk, amelyek tovább nem is bonthatók. Ismét egy fát kapunk. A gyökérben található az eredeti feladat. A fa első szintjén helyezkednek el azok a részfeladatok, amelyekre első lépésből bontható le a feladatot. Mely részfeladatok kerülnek a második szintre? Nyilván azok, amelyek közvetlenül adódnak az első szinti részfeladatok lebontásából. Végül a fa leveleiben lesznek a lebontásból adódó triviális részfeladatok.

Az imént bemutatott fa minden csomópontjában hasonló feladatok találhatók. Ezért beszélhetünk a részfeladatok általános alakjáról. Az eredeti feladat úgy tekinthető, mint az általános feladat határesete, abban az értelemben, hogy méretben a legnagyobb. A triviális részfeladatok a másik határesetként foghatók fel, hiszen méretben a lehető legkisebbek (tovább nem darabolhatók).

Szemléltessük mindezt a *„Hanoi tornyok”* feladatán keresztül: *Képzeljünk magunk elé három rudat, amelyeket a-val, b-vel és c-vel fogunk jelölni. Az a rúdon n különböző méretű korong van, méretük szerint csökkenő sorrendben (legalul található a legnagyobb átmérőjű). Helyezzük át az n korongot az a rúdról a b rúdra a c rúd felhasználásával, figyelembe véve a következőket:*

- *egy lépésben egyetlen korong mozdítható el valamelyik korongtorony tetejéről egy másik tetejére;*
- *tilos nagyobb átmérőjű korongot kisebb átmérőjűre helyezni.*

A fenti feladat első lépésben két hasonló, de egyszerűbb részfeladatra bontható (amelyek mindegyike további kettőre, és így tovább), ugyanis *n* korong áthelyezése *a*-ról *b*-re a *c* segítségével visszavezethető *n–1* korong kétszeri áthelyezésére:

1. áthelyezzük a felső *n–1* korongot *a*-ról *c*-re *b* segítségével;
2. áthelyezzük a legnagyobb korongot *a*-ról *b*-re;
3. áthelyezzük az *n–1* korongot *c*-ről *b*-re *a* segítségével.

A részfeladatok általános alakja: *k* korong áthelyezése az *f* (*forrás*) rúdról a *c* (*cél*) rúdra a *s* (*segítő*) rúd segítségével. Határesetek a méretet tekintve: ha $k = n$, akkor az eredeti feladatot kapjuk, $k = 1$ esetén pedig triviális feladattal van dolgunk.

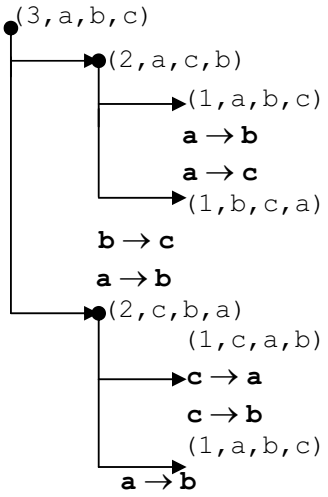
Példaként tekintsük az $n = 3$ esetet. A *2.12 ábra* szemléletesen mutatja be, hogyan bontható le az eredeti feladat részfeladatokra, és érzékelteti a feladat mögött meghúzódó fastruktúrát is. Félkövér stílust használtunk a feladatot megoldó lépéssorozat kiemelésére:

$$(a \to b, a \to c, b \to c, a \to b, c \to a, c \to b, a \to b).$$

A jelölések jelentése:

(k, f, c, s) – áthelyezünk k korongot az f rúdról a c rúdra a s rúd segítségével.

$s \rightarrow d$ – áthelyezünk *1* korongot az s rúdról a d rúdra.



2.12 ábra

Visszalépéses keresés vagy oszd meg és uralkodj? (5. hét)

Hasonlóságok:
Mindekét technika mélységében járja be a feladatok szerkezetét ábrázoló fát. Ez a fő ok, amiért a rekurzív megvalósítás annyira kézenfekvő mindkét esetben.

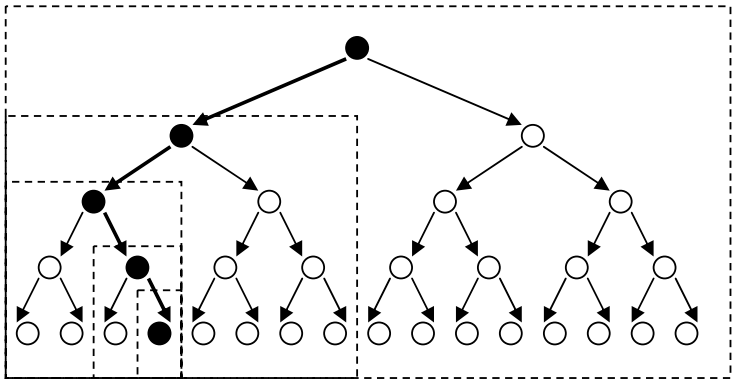
Különbségek:
A visszalépéses keresés algoritmusokban a megoldások felépítése a gyökértől a levelek irányába történik. Így a fa leveleiben adnak megoldást, pontosabban a megoldáslevelekben. Úgy is mondhatnánk, hogy a mélységi bejárás előremutató útszakaszain épít, a visszamutatókon pedig bont. Mivel a visszalépéses keresés technika a feladathoz rendelt fa csomópontjait első érintésükkor használja a megoldások építésében (ekkor kerülnek be a verembe, majd utolsó érintésükkor kerülnek ki onnan), ezért fogalmazhatunk úgy, hogy preorder mélységi bejárást alkalmaz.

Ezzel szemben az oszd meg és uralkodj algoritmusok a gyökértől a levelek irányába (fentről lefelé) lebontják a feladatot egyre egyszerűbb részfeladatokra, majd a visszaúton (lentről felfelé) a részfeladatok megoldásaiból felépítik az eredeti feladat megoldását. Egy részfeladat csak azután oldja meg, miután a részfeladatai már megoldotta. Milyen sorrendben oldja hát meg a fa csomópontjai által képviselt részfeladatok? Postoreder mélységi bejárás szerinti sorrendben.

Másik különbség, hogy a visszalépéses keresés a megoldáslevelekbe felérkezve, az oszd meg és uralkodj módszer pedig a gyökérbe visszaérkezve ad megoldást. Ez megmagyarázhatja azt, hogy a visszalépéses keresési feladatoknak általában miért van több megoldásuk is, az oszd meg és uralkodj feladatoknak viszont csak egy (egy fának sok levele van, de egyetlen gyökere).

A feladat fastruktúrája „mohó szemmel” (6. hét)

A mohó technikát optimalizálási feladatok megoldására használjuk. Általában arról van szó, hogy egy döntéssorozattal kell meghatározni az optimális megoldást. Újra egy fát kapunk, egy döntési fát, amely a feladathoz rendelhető. Minden döntéssel a feladat kisebb és kisebb méretű hasonló feladatokká redukálódik, amelyek egymásba ágyazott részfeladatainak tekinthetők. Az eredeti fa az első döntéssel leszűkül valamelyik fiú részfájára (azzal, hogy választunk, a többi fiú részfát mintha lemetszenénk a fáról), majd a következő döntéssel ennek valamelyik fiú részfájává, és így tovább, amíg csak egy levél marad belőle. *(2.13 ábra)*



2.13 ábra

Az optimális döntéssorozatot, amelyik elvezet az optimális megoldáshoz, a fa valamelyik gyökér-levél útja képviseli. Nevezzük el ezt az utat optimális útnak, az illető levelet pedig optimális levélnek. Ezek után úgy is megfogalmazható egy mohó feladat, hogy keressük meg a feladathoz rendelhető döntési fa optimális gyökér-levél útját.

A mohó algoritmus a gyökértől a levelek felé haladva metszi meg a fát, egész részfákat vágva le róla. Tehát egyetlen gyökér-levél utat jár be.

Visszalépéses keresés és mohó algoritmus? (7. hét)

Hasonlóságok:
Mindkét technika *fentről-lefelé* építi fel a megoldást, illetve a megoldásokat, levélben adva eredményt.

Különbségek:
A visszalépéses keresés technika megadja az ígéretes részfa összes gyökér-levél útját. Ezzel szemben a mohó technika egyetlen gyökérutat állít elő.

Együttes alkalmazás:
Egyes feladatok esetében úgy találhatjuk, hogy míg a mohó stratégia nem kielégítő, a visszalépéses keresés túl időigényes. Egy lehetséges (és jobb) megoldáshoz vezet a két módszer egymást kiegészítő kombinálása, amit a köztük lévő hasonlóságok, illetve különbségek tesznek lehetővé.

Foglalkozzon a visszalépéses keresés mohó sorrendben az $A_1, A_2, \ldots, A_n$ halmazok elemeivel. Ily módon a visszalépéses keresés algoritmus is elsőként éppen a mohó megoldást találja meg. Ezzel a megközelítéssel önmagában persze még nem növeltük az algoritmus hatékonyságát, csak azt biztosítjuk, hogy a lehetséges megoldásokat egy bizonyos esélyességi sorrendben generáljuk. Mindez azonban előfeltétele a következő optimalizálásnak: mielőtt a visszalépéses keresés bejárná mélységében valamely csomópont következő fiú részfáját, először megvizsgáljuk, hogy van-e rá esély, hogy a megoldáslevél az illető részfában legyen. Ezt úgy tudjuk megtenni, hogy a szóban forgó részfában egy mohó algoritmussal megbecsüljük (a feladat természetétől függően használunk alul- vagy felülbecsülést), mennyire jó lehetséges megoldásra számíthatunk, ha folytatjuk az illető részfa bejárását. Az optimalizálás alapötlete a következő: ha ez a becsült érték sem jobb, mint a visszalépéses keresés algoritmus folyamán addig talált legjobb megoldás, akkor az illető részfa biztosan nem tartalmazza az optimális levelet, tehát értelmetlen lenne bejárni (így teljesen le fogjuk metszeni a fáról).

A feladat fastruktúrája a „dinamikus programozás szemével" (8.-10. hét)

A dinamikus programozást is - akár a mohó technikát - elsősorban optimalizálási feladatok megoldására használjuk. Ebben az esetben is rendszerint arról van szó, hogy létezik egy célfüggvény, amelyet egy (optimális) döntéssorozattal optimalizálni kell. Tehát ismét döntési faként fogható fel a feladat. A 2.5 alfejezetben részletesen foglalkozunk a felülnézet módszernek a dinamikus programozás területére való sajátos alkalmazásával.

Oszd meg és uralkodj, mohó vagy dinamikus programozás (11. hét)

Hasonlóságok:
Amint láttuk a mohó, illetve a dinamikus programozási feladatokban általában arról van szó, hogy egy döntéssorozattal meghatározzuk az optimális megoldást. Minden döntéssel a feladat egy (bizonyos esetekben két) kisebb méretű hasonló feladattá redukálódik. Ezek szerint egy döntési fa is azt ábrázolja, hogy - akárcsak az oszd meg és uralkodj feladatok esetén - miként vezethető vissza a feladat hasonló, de egyszerűbb részfeladatokra. Következtetésként kijelenthető, hogy az oszd meg és uralkodj, a mohó és a dinamikus programozási feladatok szerkezete ebből a szempontból hasonló. Mindegyik technika úgy tekinti a feladatot, mint amely faszerkezet szerint bomlik részfeladatira. A nagy kérdés az, hogy mikor ki viszi el a pálmát, azaz melyik technika kapja meg a feladatot, amiért a legjobb algoritmust tudja felajánlani. A legjobb alatt azt értjük, hogy legyen helyes, hatékony és egyszerű.

Különbségek:
Míg a mohó technika egyetlen döntéssorozatot állít elő (bízva abban, hogy ez lesz az optimális), addig a dinamikus programozás több (optimális) rész döntéssorozatot is generál, amelyekből majd felépíti az eredeti feladatot megoldó (optimális) döntéssorozatot. Ez a különbség abból adódik, hogy a mohó algoritmus mohó döntések sorozatával, a dinamikus programozás pedig az optimalitás alapelve szerint építkezve oldja meg a feladatot. A fenti megállapítással összhangban a mohó stratégia fentről lefelé, a dinamikus programozás pedig lentről felfelé oldja meg a feladatot. Az első a döntési fa gyökerétől a levelei felé haladva minden mohó választással a feladatot kisebb méretű hasonló feladattá redukálja, míg triviálissá nem válik. A második, indulva a triviális

feladatokat ábrázoló levelektől, felépíti az egyre bonyolultabb részfeladatok megoldásait, végül pedig - felérkezve a gyökérbe - az eredeti feladat megoldását.

Az oszd meg és uralkodj módszer akkor nem hatékony, ha a feladat lebontásakor azonos részfeladatok jelennek meg, hiszen ezeket többször is megoldja. A dinamikus programozás viszont annál hatékonyabb, minél több az azonos részfeladat, mivel képes elkerülni ezek ismételt megoldását. Ez a különbség a stratégiáikból adódik. Egy oszd meg és uralkodj algoritmus először lebontja a feladatot (preorder mélységi bejárás szerint), majd a rekurzió visszaútján postorder sorrendben megold *minden* részfeladatot, amellyel a lebontás alkalmával találkozott. Mivel minden részfeladat megoldását a közvetlen fiú részfeladatai megoldásaiból építi fel, ezért csak ezeket tárolja el, és ezeket is csak ideiglenesen. Más szóval, nem vezet nyilvántartást a már megoldott részfeladatokról és azok megoldásairól. Ez a magyarázata annak, hogy az azonos részfeladatokkal - anélkül, hogy tudomása lenne róla - többször is találkozik, újra és újra megoldva őket. Ezzel szemben a dinamikus programozás lentről kezd neki a feladatnak. Mivel egymástól különböző részfeladatok megoldásaiból építkezik, ezért eleve kizárt, hogy találkozzon azonos részfeladatokkal. Nyilvántartást vezet továbbá (általában egy tömbben) azon részmegoldásokról, amelyekre potenciálisan szüksége lehet a későbbiekben.

Mindkét technika a gyökérben ad megoldást: az oszd meg és uralkodj a rekurzióból ide visszaérkezve, a dinamikus programozás iteratívan ide felérkezve. Tehát, míg az egyik algoritmus mindig rekurzív, a másik mindig iteratív.

Az elágazás korláttal technika beágyazva a többi négy alkotta képbe (12.- 13. hét)

Az elágazás korláttal technika sokkal összetettebb, hogy azt e tanment keretén belül, a teljesség igényével bemutathatnánk. Célunk csupán a felülnézet módszer kiterjesztése egy jellegzetesen elágazás korláttal technikára, az úgynevezett A^* algoritmusra.

Szemléletesen ezt a stratégiát úgy lehet elképzelni, mintha a szélességi bejárásnak adtunk volna egy olyan mélységi jelleget, amelyben céltudatosság van. Mintha párhuzamosan a fa mindegyik ágának irányába elindulnánk a hagyományos szélességi bejárással, ugyanazzal a sebességgel haladva minden levél felé. Ha súlyozzuk a csomópontokat (egy csomópont súlya a gyökér-csomópont távolság plusz a csomópont-megoldáslevél megbecsült távolság), és a szélességi bejáráshoz prioritássort használunk, akkor ahelyett, hogy minden levél irányába ugyanazzal a sebességgel haladnánk, az ígéretesebbnek tűnő irányokat előnyben részesítjük. De a pillanatnyilag kevésbé ígéretes irányokról sem mondunk le. Ott vannak a sor végén arra az eshetőségre, ha netalán tévednénk. Ne feledjük, hogy az ígéretesség meghatározása csak becslésen alapszik.

Összehasonlítva az előbbi technikákkal, elmondható, hogy az imént bemutatott elágazás korláttal stratégia céltudatosabb, mint a visszalépéses keresés, és óvatosabb, mint a mohó algoritmus. Az elágazás korláttal stratégia is és a dinamikus programozás is prioritássorral megvalósított szélességiszerű bejárást alkalmaz, és mindkettő elkerüli az azonos csomópontokkal való többszöri foglalkozást. A lényeges különbség abban van, ahogyan a csomópontok súlyozását végzik. A dinamikus programozás esetén a jelen kizárólag a múltban gyökerezik (a sorban lévő csomópontok aktuális súlya a *bejárt részfa* legjobb gyökér-csomópont útjának súlya). Ezzel szemben az elágazás korláttal stratégiák esetében a csomópontok súlyozásában, bizonyos értelemben, jelen van a jövő is azon meggondolásból, ahogy megbecsüljük a megoldáslevéltől való távolságukat. További

különbség köztük például az is, hogy az elágazás korláttal technika – a hagyományos szélességi bejáráshoz hasonlóan – minden csomópontot éppen a legrövidebb úton ér el először, míg a dinamikus programozás e változata általában többszöri finomítás révén állítja elő az optimális utakat.

Határesetek a programozási technikák világában (14. hét)

Az eddigiekben a felülnézetet általában úgy valósítottuk meg, hogy ugyanazt a feladatot több technikával is megoldottuk. Ezekben az esetekben ugyanarra a feladatra két különböző algoritmust adtunk.

Egy másik módja a stratégiák együttes tanulmányozásának a „határeset algoritmusok" tanulmányozása. Ahogy a határvárosokban keverednek a kultúrák, ugyanígy léteznek olyan algoritmusok, amelyekben két vagy több technika jellegzetes vonásai is fellelhetők. A felülnézet módszer alkalmazásának mintegy lezárásaként megvizsgálhatunk néhány ilyen algoritmust. Szolgáljon példa gyanánt a bináris keresés algoritmusa.

Bináris keresés: *Adott egy szigorúan növekvő számsorozat, amelyet az a[1..n] tömbben tároltunk el, valamint egy x szám. Írjunk* ***hatékony*** *algoritmust az x megkeresésére a számsorozatban. Ha megtalálható, adjuk vissza a sorszámát, különben nullát.*

Az algoritmus: Kiválasztjuk az *a[1..n]* tömb középső elemét, az *a[n/2]* elemet. Ha ez éppen a keresett elem, a keresést befejeztük. Ellenkező esetben, attól függően, hogy x kisebb vagy nagyobb, mint e középső elem, a keresést vagy a számsorozat felső (*a[1..n/2-1]*), vagy az alsó (*a[n/2+1..n]*) felében folytatjuk, ahol természetesen hasonlóan járunk el. Az algoritmus akkor ér véget, ha megtaláltuk a keresett elemet, vagy ha üres tömbszakaszhoz jutattunk.

Ezt az algoritmust hagyományosan az oszd meg és uralkodj stratégiának tekintik, és ezt nem is vitatjuk, hiszen minden lépésben a feladatot valóban ketté osztja. Mégis rá fogunk mutatni, hogy egy határeset algoritmusról van szó, „mohó ország" határához közel. Mielőtt rámutatnánk a mohó jegyekre, elevenítsük meg a feladat mögött húzódó bináris fastruktúrát.

A fa gyökere a teljes számsorozatot képviseli, a többi csomópont pedig a felezésekből nyert tömbszakaszokat. A fa levelei a feldarabolás nyomán adódó üres tömbszakaszokat ábrázolják, kivéve egyet, amennyiben *x* megtalálható a számsorozatban. Ez esetben, a szóban forgó levél egy olyan tömbszakaszt képvisel, amelynek *x* pontosan a középső eleme.

A mohó algoritmusokat optimalizálási feladatok megoldására használjuk. Fellelhető-e a feladat esetében az optimalizálási jelleg? Egy bizonyos értelemben igen! A feladatot lineáris kereséssel is meg lehetne oldani (sorban megvizsgáljuk az összes elemet anélkül, hogy kihasználnánk a sorozat rendezettségét), de tőlünk hatékony algoritmust kérnek. A keresésre nézve a legrosszabb eset, ha a keresett szám nem található meg a számsorozatban. Ilyenkor a lineáris keresés *O(n)* időigényű, a bináris viszont csak *O(lg(n))*. Tehát a feladat úgy is megfogalmazható, hogy írjunk olyan kereső algoritmust, amely esetében az összehasonlítások száma minimális. Ebből a szempontból a középső elem megadása helyi optimumnak tekinthető, hiszen csakis középen történő vágásokkal jutunk *lg(n)* magas fához. Tehát a mohó döntés nem azt jelenti, hogy a számsorozat

melyik felében folytatjuk a keresését (hiszen ez egyértelmű), hanem azt, hogy minden lépésben éppen a középső elemet vizsgáljuk meg. Bármely más választás a legrosszabb esetben *lg(n)*-nél több összehasonlítást jelentene.

Más szóval, a sorozat rendezettségéből adódóan egyetlen összehasonlítással – amennyiben nem a keresett elemet találtuk el - a vizsgált elemet megelőző, vagy az őt követő elemek kizárhatók a további kereséséből. A *legrosszabb eset* nyilván az, ha mindig a nagyobbik szakaszban *kell folytatnunk* a keresést (mert feltehetően ott található meg a keresett elem). Ha az éppen vizsgált részsorozat *m* hosszú, akkor a vágásból adódó nagyobbik rész az $\lceil (m-1)/2 \rceil$ ... (m-1) intervallumban lehet. A bináris keresés a középső elem választásával ezen intervallumnak éppen az alsó, a lineáris keresés pedig (mivel mindig az első elemet vizsgálja) a felső korlátját használja.

Az alábbiakban összefoglaljuk az oszd meg és uralkodj stratégia azon jellemzőit, amelyek hiányoznak a bináris keresés algoritmusból, és felsorolunk további mohó jegyeket, amelyek viszont jelen vannak benne:

- Bár az algoritmus a feladatot minden lépésben kettéosztja, nem oldja meg mindkét részfeladatot, és nem ezek megoldásaiból építi fel az apafeladat megoldását, mint ahogy a hagyományos oszd meg és uralkodj stratégia esetében szokásos.
- A feladat minden lépésben egy hasonló és egyszerűbb feladattá redukálódik, ami az algoritmus mohó jellegét hangsúlyozza.
- Az algoritmus már a fa valamelyik levelébe érkezve megoldást tud adni, nemcsak akkor, amikor visszaér a gyökérbe (rekurzív implementálás esetén). Ezért is van az, hogy a hagyományos változatában a bináris keresés iteratív algoritmus.

2.4.4 A felülnézet módszer és a problémamegoldás

Az „Algoritmustervezés felülnézetből” módszer végső célja, hogy segítse a tanulókat annak felismerésében, hogy mely feladatok melyik technikával oldhatók meg a leghatékonyabban, majd pedig ténylegesen megtalálják a megoldási algoritmust. A módszer rávezeti a diákokat az egyes technikák alkalmazásának stratégiai lépéseire. Ha kérdések formájában fogalmazzuk meg e lépéseket, akkor lehetővé válik a tanulók aktivizálása, gondolkodásra késztetése. A stratégia rögzítése miatt fontos – számos megoldott feladattal -, hogy ugyanazon lépéssorozaton többször is végigvezeti a diákokat. Mivel egyes feladatok a stratégia sajátos alkalmazásait feltételezhetik, a tanulók felismerik, hogy nem egy merev sémáról, hanem tényleg stratégiáról van szó. Minél többféle helyzetben látnak „működni” egy adott programozási technikát, annál jobban fogják ismerni azt. Szolgáljon példaként az a kérdéssorozat, amely az „Algoritmustervezés felülnézetből” című könyv [11] oszd meg és uralkodj technikával foglalkozó fejezetében található.

Lépéssorozat az oszd meg és uralkodj stratégia alkalmazásához:

1. Hogyan vezethető vissza a feladat hasonló, de egyszerűbb részfeladatokra?
2. Mi az általános feladat alakja? Mik a paraméterei? (Ezek lesznek az oszd meg és uralkodj eljárás formális paraméterei!)
3. Milyen paraméterértékek esetén kapjuk vissza az eredeti feladatot? (Ezekre hívjuk meg kezdetben az eljárást!)

4. Mi a trivialitás feltétele? Hogyan oldhatók meg a triviális részfeladatok?
5. Nem triviális esetben mik az általános feladat részfeladatainak a paraméterei? (Ezek lesznek a rekurzív hívások aktuális paraméterei!)
6. Hogyan építhető fel a részfeladatok megoldásaiból az általános feladat megoldása?

2.5 A felülnézet módszer és a dinamikus programozás

A dinamikus programozás nevet viselő programozási technika rendelkezik a legvonzóbb tulajdonságokkal a tárgyalt technikák között. Számos olyan feladat van, amelyet bár megold „valahogy" az oszd meg és uralkodj módszer is, a dinamikus programozás hatékonyan teszi ezt meg. Nem kevés azon optimalizálási feladatok száma sem, amelyek esetében, míg a mohó megközelítés nem kielégítő, a dinamikus programozás sikeresen alkalmazható. Az érem másik oldala az, hogy a dinamikus programozás igényli a legelmélyültebb gondolkodást. A nehézség többek között a dinamikus programozási feladatok sokszínűségében áll. A diáknak nagyon mélyen át kell látnia a dinamikus programozás alapelveit, hogy a legkülönbözőbb helyzetekben alkalmazni tudja. Gyakran panaszkodnak az informatikatanárok, sőt az egyetemi oktatók is, hogy a legtöbb diák nem tud eljutni a megoldott feladatok reprodukálásán túl olyan szintre, hogy ismeretlen feladatok esetében is tudja alkalmazni ezt a programozási technikát. A bemutatott felülnézet módszer egy újabb erőssége abban rejlik, hogy kiterjeszthető a dinamikus programozási feladatok területére, lehetővé téve ezek egyfajta osztályozását. [61]

Ugyanazt az absztrakt platformot használjuk, a feladatok mögött lévő döntési fát, és a feladatokat e döntési fa különböző típusai szerint osztályozzuk. Minden egyes feladatosztályhoz a dinamikus programozásnak mint algoritmustervezési stratégiának egy sajátos változata rendelhető. Az adott feladatnak a megfelelő osztályba sorolása után már sokkal elérhetőbb a technika valamelyik (az illető feladatosztályt megoldó) sajátos változatának alkalmazása. A felülnézet módszer kivetítése a dinamikus programozás területére azzal az előnnyel is jár, hogy árnyaltabbá teszi a dinamikus programozás és a mohó algoritmus, illetve az oszd meg és uralkodj technika közti különbségeket, hasonlóságokat és kapcsolatokat.

2.5.1 Kétféle döntési fa

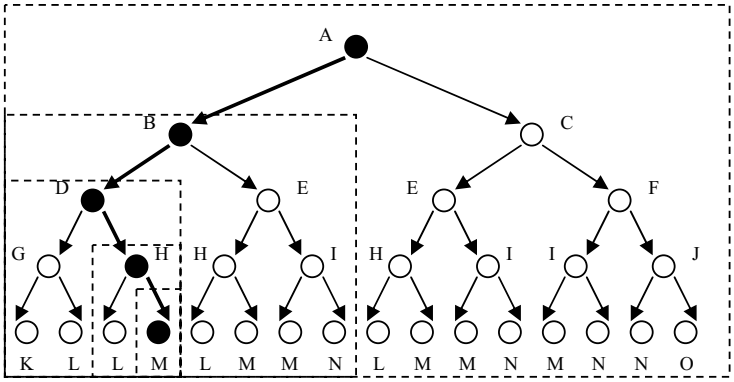
A döntési fa gyökere a feladat kezdeti állapotát jelképezi, az első szinti csomópontok azokat az állapotokat, amelyekbe a feladat az első döntés után kerülhet, a második szinten lévők pedig azokat, amelyek a második döntésből adódhatnak stb.. Egy csomópontnak annyi fia lesz, ahány lehetőség közül történik a választás az illető döntés alkalmával.

A *2.14 ábra* egy olyan helyzetet mutat be, amikor négy döntés révén jutunk megoldáshoz. Minden döntés alkalmával két lehetőség közül választhatunk.

Két esetet különböztetünk meg:

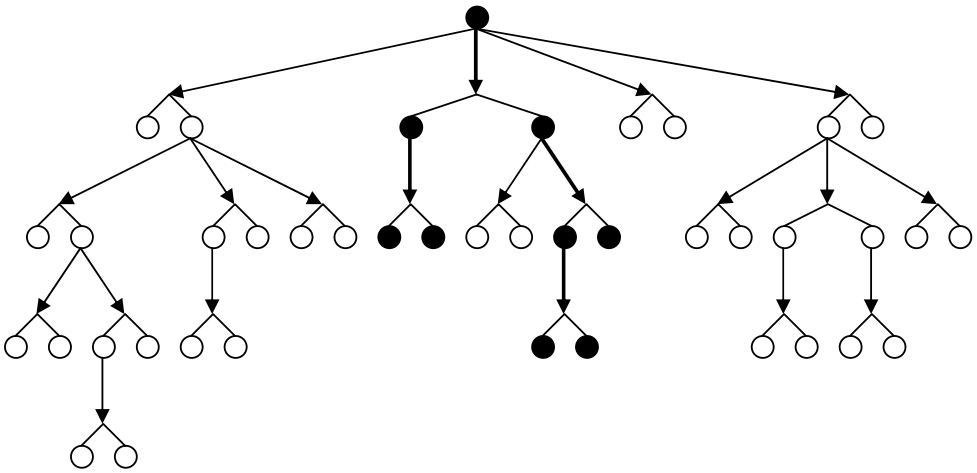
1. típusú döntési fa: *Minden döntéssel a feladat egy kisebb méretű hasonló feladattá redukálódik, amelyet az aktuális csomópont valamelyik részfája ábrázol.* Ilyenkor az optimális megoldást valamelyik gyökér-levél út fogja képviselni a döntési fában. Erre az esetre vonatkozik a *2.14 ábra* is. A szaggatott vonalú téglalapokkal azt érzékeltettük, hogy – amennyiben a vastagított nyilakkal jelölt döntéssorozat az optimális – miként

redukálódik általa a feladat egyre kisebb méretű részfeladataira.



2.14 ábra

2. típusú döntési fa: *Minden döntéssel a feladat két vagy több kisebb méretű hasonló feladatra esik szét, amelyeket az aktuális csomópont megfelelő részfái ábrázolnak.* A *2.15 ábra* egy olyan helyzetet mutat be, amikor minden döntéssel a feladat két részfeladatra bomlik. Feltételeztük, hogy a megvastagított bináris részfa ábrázolja az optimális részfeladatokra bontást.



2.15 ábra

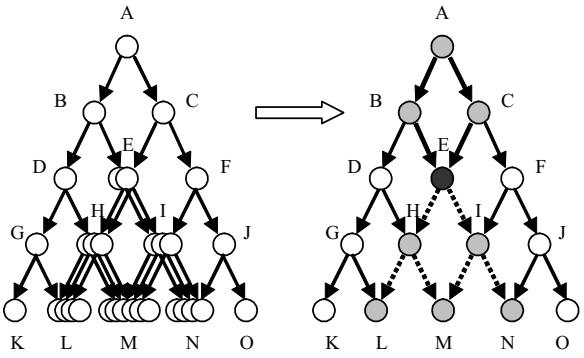
A döntési fa részfái azokat a részfeladatokat képviselik, amelyekké a feladat – az egyes döntéssorozatok által – redukálódhat (*1. eset*), illetve széteshet (*2. eset*).

2.5.2 Az összevont döntési fa

Bár a döntési fa csomópontjainak száma exponenciálisan függ az optimális döntéssorozat döntéseinek számától, gyakran megtörténik, hogy számos azonos csomópontot tartalmaz, amelyek nyilván azonos állapotokat ábrázolnak.

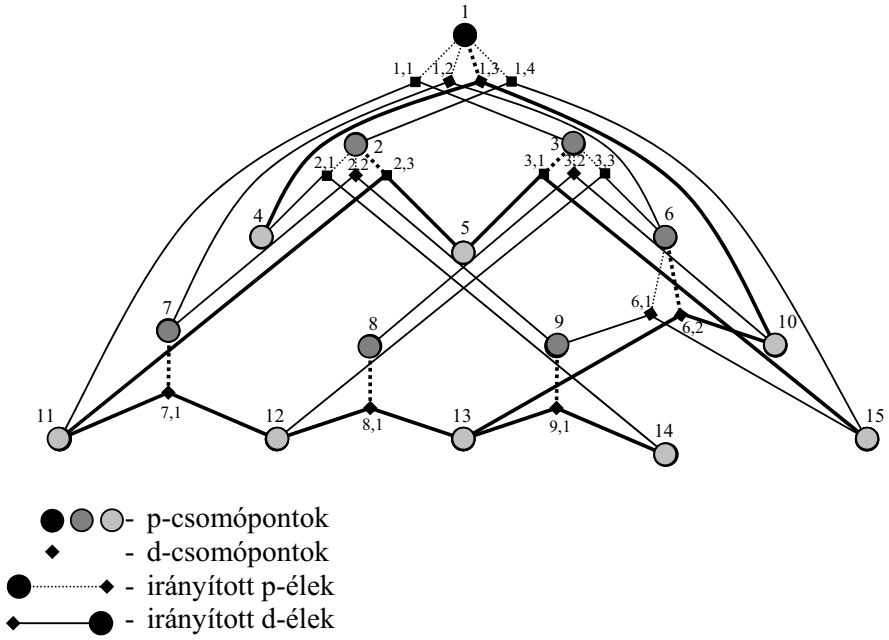
Minél több az azonos csomópontok száma, annál jobban látszanak a dinamikus programozás erősségei. Az első típusú döntési fák esetében ez a helyzet akkor alakul ki, ha a különböző rész-döntéssorozatok révén a feladat ugyanazon részfeladattá redukálódik. Egy ilyen helyzetet mutat be a *2.14 ábrán* látható döntési fa is. Csúsztassuk egymásra a fa azonos állapotait képviselő csomópontokat. Az így kapott adatszerkezetet

összevont döntési fának fogjuk nevezni *(2.16 ábra)*. Amint látható, az összevont döntési fa már nem fastruktúra, hanem egy irányított gráf[3]. Az összevont fának pontosan annyi csomópontja lesz, ahány különböző állapotba kerülhet a feladat (ez a szám rendszerint csak polinomfüggvénye az optimális megoldáshoz vezető döntések számának).



2.16 ábra

A 2. típusú döntési fák esetében az összevont döntési fa egy speciális gráf lesz, amelyet d-gráfnak neveztünk [59]. Egy ilyen d-gráfot (összevont 2. típusú döntési fát) láthatunk a *2.17 ábrán*. Egy d-gráf esetében megkülönböztetünk *d* és *p* típusú csomópontokat, illetve éleket. Egy d-csomópontot egy számpár azonosít aszerint, hogy *p* típusú apacsomópontjának hányadik fia.



2.17 ábra

[3] Valahányszor az összevont döntési fa gyökeréről, illetve leveleiről beszélünk, azokra a csomópontokra gondolunk, amelyek a döntési fában gyökér, illetve levél minőségben voltak jelen.

2.5.3 A dinamikus programozási stratégiák osztályozása

A dinamikus programozással megoldható feladatok a szerint osztályozhatók, hogy a mögöttük meghúzódó döntési fa első vagy második típusú, illetve, hogy az első esetben az összevont döntési fa körmentes-e vagy sem, és amennyiben tartalmaz kört, van-e negatív éle vagy nincs.

Az 1. típusú döntési fájú feladatok esetén a feladat minden döntéssel *egy* hasonló, egyszerűbb feladattá *redukálódik*. A „redukálódik" kifejezéssel azt szeretnénk érzékeltetni, hogy minden döntéssel megoldunk valamennyit a feladatból. Úgy is mondhatnánk, hogy a D_1, D_2, … , D_n döntéssorozat után a feladat lépésről lépésre megoldódott. A kérdés csak az, hogyan hozzuk meg a döntéseket, hogy optimális megoldáshoz jussunk. Az ilyen típusú feladatok első ránézésre „mohó feladatnak" tűnnek, csakhogy nem lévén érvényes rájuk a mohó választás alapelve, „kénytelenek vagyunk" a dinamikus programozáshoz folyamodni.

Mivel ebben az esetben az összevont döntési fa egy hagyományos súlyozott irányított gráf, amelyben az optimális megoldást az optimális gyökér-levél út képviseli, a feladat optimális út problémaként is felfogható, amelyre közismert gráfelméleti algoritmusok léteznek. Ezzel összhangban az egyes feladatosztályokra vonatkozó sajátos dinamikus programozási stratégiákat a szerint nevezhetjük el, hogy melyik optimális út algoritmus biztosítja hozzá a gráfelméleti hátteret.

1. *Topológikus sorrend alapú dinamikus programozás*: Az összevont döntési fa körmentes. Ebben az esetben létezik a csomópontok topológikus sorrendjére alapozó, *O(N+M)* futási idejű algoritmus a feladatra (*N* és *M* a gráf csomópontjainak, illetve éleinek száma)[17].

2. *Dijkstra algoritmus alapú dinamikus programozás*: Az összevont döntési fa tartalmaz ugyan kört, de nincs negatív éle. Erre az esetre vonatkozik *Dijkstra* algoritmusa, amely egy elsőbbségi sor segítségével az optimum értékek szerinti növekvő sorrendben határozza meg a minimális súlyú utakat. *Dijkstra* algoritmusának bonyolultsága $O(N^2)$, de javítható, ha az elsőbbségi sort bináris kupac ($O((N+M)lg(N))$), vagy Fibonacci-kupac ($Nlg(N+M)$) segítségével implementáljuk [17].

3. *Belmann-Ford algoritmus alapú dinamikus programozás*: Az összevont döntési fának van negatív éle is, de nincs a gyökérből elérhető negatív összsúlyú köre. Ezt a feladatot oldja meg *Belmann-Ford* algoritmusa ($O(NM)$) [17].

A 2. típusú döntési fájú feladatok inkább az oszd meg és uralkodj technikához állnak közelebb. Az oszd meg és uralkodj algoritmus a feladatot két vagy több hasonló, egyszerűbb részfeladatra osztja fel, és ezek megoldásaiból építi fel az eredeti feladat megoldását (a részfeladatok megoldásánál természetesen hasonlóképpen jár el, egészen addig, míg triviális részfeladatokhoz nem jut). Ha a feladat részfeladatokra bontása többféleképpen is elvégezhető, akkor felmerül az optimális felbontás kérdése. Ilyen esetben már többről van szó, mint egy egyszerű oszd meg és uralkodj feladatról. Ha minden lépésben meghatározható lenne mohó döntéssel, hogy hol van az optimális „vágás" akkor azt mondjuk, hogy a mohó technika dolgozott az oszd meg és uralkodj keze alá. Ha viszont nincs elegendő információnk mohó vágásokhoz, de a feladat

feldarabolására érvényes az optimalitás alapelve, akkor a dinamikus programozás segíthet az oszd meg és uralkodj technikának az optimális megoldás megtalálásában.

Tehát olyan stratégiáról van szó, amely az oszd meg és uralkodj technika „oszd meg" szakaszába beépíti a dinamikus programozást. Ez azt jelenti, hogy az optimalizálási feladatokra jellemző $D_1, D_2, \ldots, D_n$ optimális döntéssorozat alapvetően a feladat optimális részfeladatokra bontását jelenti. Az aktuális feladat minden döntéssel *szétbomlik* hasonló, egyszerűbb részfeladatokra (ellentétben azzal az esettel, amikor egyetlen hasonló, egyszerűbb feladattá *redukálódott*). Amikor optimális részfeladatokra bontásról beszélünk, arra gondolunk, abból a szempontból optimális, hogy az oszd meg és uralkodj algoritmus „uralkodj" szakaszában az optimális megoldást kapjuk. A dinamikus programozás e fajtáját „optimális megosztás–optimális uralom" névvel azonosíthatjuk. Az [59] cikk részletesen tárgyalja e sajátos dinamikus programozási stratégia gráfelméleti hátterét.

2.6 Kísérleti mérés

Az alábbiakban egy kísérletről számolunk be, amellyel lemértük, hogyan járul hozzá a bemutatott módszer és a javasolt tanmenet az algoritmus tervezési stratégiák hatékony tanításához-tanulásához. A kísérletet a marosvásárhelyi Bolyai Farkas Elméleti Líceumban végeztük el a 2003/2004-es tanévben. Ebben az iskolában minden évfolyamon (*IX.-XII.*) három párhuzamosan működő informatika tagozatos osztály van. A hivatalos tananyag X. osztályban írja elő a technikák tanítását (akkor még csak a visszalépéses keresés, a mohó, valamint az oszd meg és uralkodj módszereket). A kísérletbe mindhárom osztályt bevontuk: *X.g, X.h, X.i.* Az osztályok átlagos felkészültségét azonosnak tekinthettük, mert IX. osztály elején nem a felvételi jegyek alapján kerültek egy osztályba a tanulók, hanem aszerint, hogy ki milyen idegen nyelvet tanult. A *h* és *i* osztályokat ugyanaz a tanárnő, a *g* osztályt pedig egy másik tanárnő tanította. Mindkét pedagógus különbözött e dolgozat szerzőjétől, aki a módszert kidolgozta. A *g* és *i* osztályokat választottuk kísérleti csoportnak, a *h* osztályt pedig kontrollcsoportnak. A kísérleti csoportokban a tanárnők felülnézet módszerrel, a javasolt tanmenet szerint oktattak, a kontrollosztályban pedig a hagyományos módon (a technikákat különálló egységekként kezelték).

Mivel a módszer célja, hogy a diákok megértsék a stratégiák közti elvi, alapvető hasonlóságokat és különbségeket, ezért arra számítottunk, hogy maradandó nyomot kell hagynia a tanulókban. Azért, hogy ezt lemérhessük, a kísérleti mérést nem végeztük el a tanév végén, hanem elhalasztottuk közel egy évvel, a 2004/2005-ös tanév második félévére. Azt megelőzően, hogy a diákok a felmérőt megírták, tartottunk mindhárom osztályban egyetlen órát, amely alkalmával felfrissítettük ismereteiket. A tesztet úgy állítottuk össze, hogy mérhető legyen a tanulók tisztánlátása, és az, hogy azt hogyan tudják kamatoztatni a gyakorlatban. A felmérőn a kontrollcsoportban 26 diák, az első kísérleti csoportban 18, a másodikban pedig 26 diák vett részt. Az alábbiakban közöljük a tesztet, amelyet a tanulóknak 50 perc alatt kellett megoldaniuk:

1. Az adott *a[1..7,1..7]* tömb mely nulláit és milyen sorrendben írja át kettesekre az alábbi rekurzív visszalépéses keresési eljárás a *fest(3,3)* hívásra? *(3 pont)*

```
procedure fest(i,j:integer);
begin
     a[i,j]:=2;
     if a[i-1,j]=0 then fest(i-1,j);
     if a[i,j+1]=0 then fest(i,j+1);
     if a[i+1,j]=0 then fest(i+1,j);
     if a[i,j-1]=0 then fest(i,j-1);
end;
```

1	1	1	1	1	1	1
1	1	0	0	0	0	1
1	0	0	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	1	0	1
1	1	0	0	0	0	1
1	1	1	1	1	1	1

2. A mellékelt labirintusból (1=fal, 0=szabad) egy ember a *(4,3)* koordinátájú négyzetből a legrövidebb úton szeretne kijutni. Az oszd meg és uralkodj technika az embertől induló legrövidebb kiút meghatározását visszavezeti a szomszédos (fel, jobbra, le és bal sorrendben) szabad pozíciókból induló legrövidebb kiutak meghatározására (részfeladatok). Milyen sorrendben kell megoldani az egyes pozíciókhoz tarozó részfeladatokat? Hogyan lehetne megoldani a feladatot mohó algoritmussal, és melyik utat találná ez legrövidebbnek? *(4 pont)*

1	0	1	1	1	1	1
1	0	0	0	0	0	1
1	1	1	0	1	0	1
1	0	0	0	1	1	1
1	0	1	1	1	0	0
1	0	0	0	0	0	1
1	1	1	1	1	1	1

3. Milyen technikákkal oldanád meg az alábbi feladatokat?
   a. Adott egy *H* széles polc, és *n* könyv, amelyek vastagságai rendre *v1, v2, ... , vn*. Hányféleképpen tölthető ki pontosan a polc ezen könyvek segítségével, ha csupán egyetlen könyvből szabad kitépni lapokat. Melyek ezek a kirakási módok? *(0,5 pont)*
   b. Adott egy *H* széles polc, és *n* könyv, amelyek vastagságai rendre *v1, v2, ... , vn*. Melyik az a pontos kitöltése a polcnak ezen könyvek segítségével, amelyik a legtöbb könyvet használja? Csupán egyetlen könyvből szabad kitépni lapokat. *(0,5 pont)*

4. Adott egy négyzet és egy kör. Határozzuk meg a metszési felületük területét háromtizedes pontossággal! Melyik technika közelítené meg így ezt a feladatot: négy részre osztom a négyzetet, és meghatározom mindegyik négyzetecske metszési területét a körrel, majd összeadom ezeket? Mindegyik négyzetecskével hasonlóképpen járok el, míg elég kis négyzetekhez nem jutok. *(1 pont)*

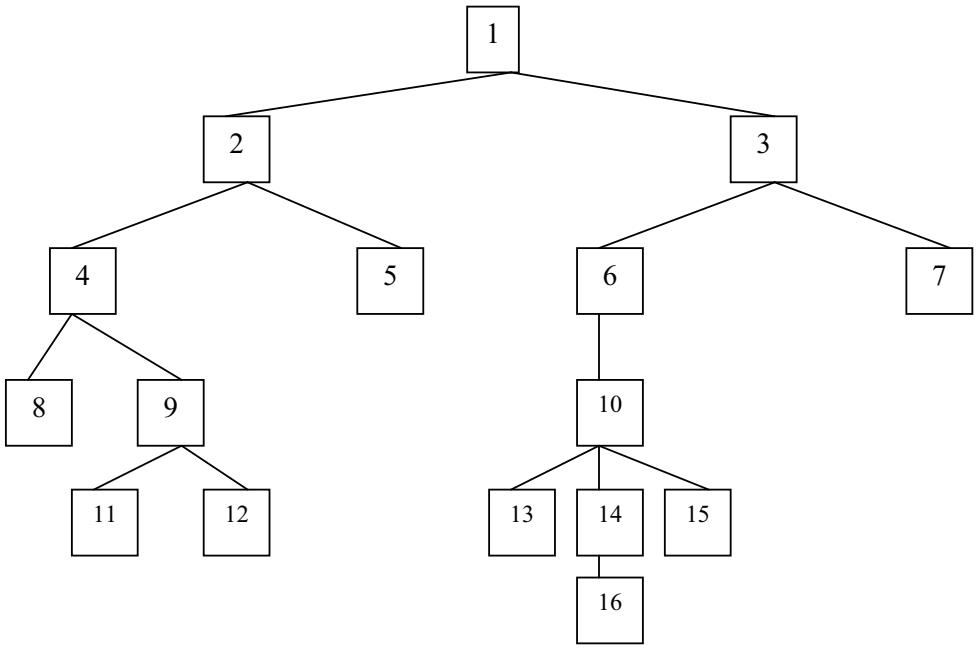
5. Egy *n* méter hosszú rudat daraboljunk fel az összes lehetséges módon *1* és *2* méter hosszú rudacskákra. Milyen megoldásokat ad az alábbi eljárás, és milyen sorrendben írja ki őket? Melyik technikához sorolnád be ezt az algoritmust? (A *kiir(A,k)* eljárás kiírja az *A* tömb első *k* elemét, és új sorba ugrik. Az eljárást *rud(4,1)* hívjuk meg.) *(3 pont)*

```
Procedure rud(n,k:integer);
Begin
   If n=0 then kiir(A,k-1);
   Else
        begin
        If(n>=1)then
             Begin
```

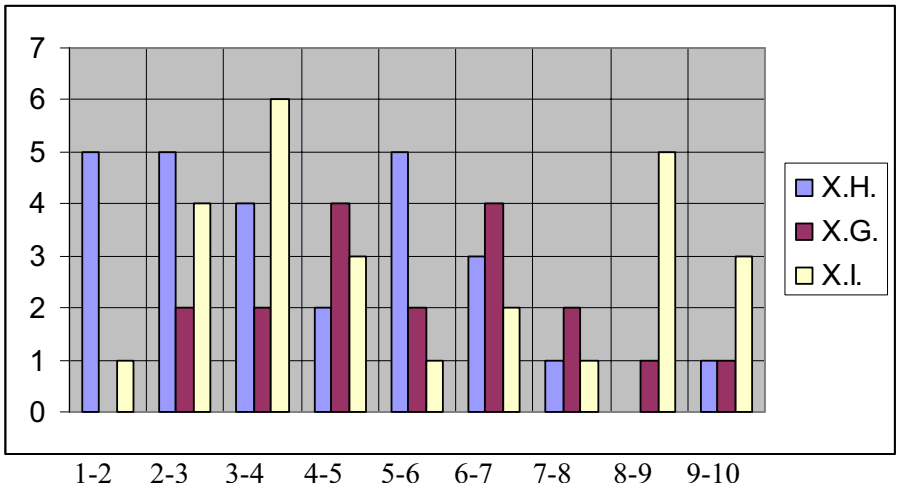
```
                A[k]:=1;
                rud(n-1,k+1);
                end;
          If(n>=2)then
                Begin
                A[k]:=2;
                rud(n-2,k+1);
                end;
          End;
    End;
```

6. Egy optimalizálási feladat kapcsán mely technikák „gondolkodnának” az alábbi irányelvek szerint? *(4 x 0,25 pont)*
   a. Biztos, ami biztos!
   b. Élj a mának!
   c. Amit ma megtehetsz, ne halaszd holnapra!
   d. A jó főnök megkeresi a megfelelő embert a megfelelő helyre!

7. Keressük a leghosszabb gyökér-levél utat a mellékelt fában. Hogyan közelíti mag a feladatot a három technika? *(2 pont)*



Az elért pontszámokhoz hozzárendeltük a jegyeket a Romániában használatos 1-10 skálának megfelelően. Az alábbi diagramok *(2.18 ábra)* azt mutatják, milyen eredmények születtek a kísérleti felmérés során. A vízszintes tengelyen a jegyeket ábrázoltuk, a függőlegesen pedig azt, hány tanuló esett a megfelelő jegyintervallumba.



2.18 ábra

A három osztály átlagai:

- Kontrollcsoport (h osztály): 4,05.
- Első kísérleti csoport (g osztály): 5,48.
- Második kísérleti csoport (i osztály): 5,52.

Látható, hogy mindkét kísérleti osztályban (bár más-más tanárnő alkalmazta a módszert) az átlagok (az 1-10 skálán) körülbelül másfél jeggyel magasabbak lettek, ami szignifikáns különbséget jelent.

Megjegyzés: Tekintettel a minták méretére, a kétmintás t-próbát használtuk. Ellenőriztük e próba alkalmazhatóságának feltételeit:

1. A minták nyilvánvalóan normális eloszlásúnak tekinthető populációból származnak.
2. Ellenőriztük az F-próba segítségével, hogy az ismeretlen populációeloszlások egyenlőnek tekinthetők-e. Például, a *h* és *i* osztályok összehasonlítása esetében a populációeloszlások 0,9999 szignifikancia szint mellett tekinthetők egyenlőnek.

Alkalmazva a kétmintás t-próbát a *h* és *i* osztályokra, azt találtuk, hogy átlagkülönbségük (a kísérleti osztály javára) 0,9877 valószínűséggel szignifikáns.

A módszer kimagasló eredményekhez vezetett a tehetséggondozásban is. A marosvásárhelyi Bolyai Farkas Elméleti Líceum mindig is szépen szerepelt országos és

nemzetközi informatikaversenyeken. A dolgozat szerzője 13 éven keresztül (1992-2004) évente juttatott el diákokat (egyet, legfeljebb kettőt) ezekre a versenyekre. A 2002/2003-as tanévben született eredmények viszont egyedülállóak voltak. Az informatikai olimpia megyei szakaszán X. osztályban (ezen évfolyam számára írja elő a romániai tanterv a programozási technikák tanítását) az első, a második, illetve a harmadik helyezett is abból az osztályból került ki, amelyben először alkalmazták a bemutatott módszert a programozási technikák oktatásában. Ezen diákok közül az egyik a romániai döntőn harmadik helyezést, a magyarországi Nemes Tihamér Verseny budapesti döntőjén pedig második helyezést ért el. Ezek az eredmények nemcsak a szerző tanári pályafutásában példátlanok, hanem a Bolyai Farkas Elméleti Líceum, sőt Maros megye informatikaoktatásának történetében is. Maros megyében általában öt középiskola képviselteti magát versenyképesen a megyei informatikai olimpián. A tanulók évfolyamonként körülbelül 10 különböző osztályból érkeznek. A 2002/2003-as tanév X. évfolyamát kivéve, soha nem történt meg, hogy ugyanabból az osztályból kerüljön ki az első három helyezett.

2.7 Következtetések

A fent bemutatott kísérlet bizonyítja, milyen hatékonyan lehet alkalmazni a felülnézet módszert a programozási technikák területén. A módszer didaktikai értéke már a felülnézet definíciójából nyilvánvaló:

1. „Egymás mellett” látjuk a vizsgálat tárgyát képező entitásokat.
2. Csak az látszik, ami a vizsgálat szempontjából lényeges.
3. Nyilvánvalóak a hasonlóságok és a különbségek, szembetűnők a kapcsolatok.

A javasolt tanmenetet követve, az egyes stratégiákat bemutató órákon a tanár elsősorban analízist, absztrakciót és konkretizálást alkalmaz, illetve a tanulókban ezeket a gondolkodási műveleteket tudatosítja. Ezek segítségével értik meg a tanulók az adott technikához tartozó feladatok fastruktúráját, majd az ezen a struktúrán megoldandó problémát, illetve a fastruktúra - megoldáshoz szükséges - megfelelő bejárását. Egy feladat ilyen típusú „átvilágítása” révén a tanulók közelebb kerülnek ahhoz, hogy a feladat makrostruktúrája is megelevenedjen előttük, azaz megértsék azokat a stratégiai lépéseket, amelyeken végighaladva eljuthatnak a megoldáshoz. Ha – több feladat kapcsán - a tanulókat újra és újra végigvezetjük (mintegy kézen fogva őket) a megoldás kulcslépésein, akkor a stratégia bevéződik a fejükbe. Ahogy a szülő gyermekét kézen fogva tanítja járni, úgy a tanár is hasonló hatást érhet el, ha kérdések formájában fogalmazza meg a stratégia kulcslépéseit.

Mivel a következő technikát általában a már bemutatottakra támaszkodva tárgyaljuk, ezért a felülnézet módszer nagy hangsúlyt fektet a hasonlóságra, amely az összefüggések megértése és a kiegészítés gondolkodási műveleteket egymás után alkalmazza. Például, a visszalépéses keresés és oszd meg és uralkodj technikák között az alapvető összefüggés abban áll, hogy mindkettő mélységében járja be a megoldásfát, de az egyik preorder, a másik pedig postorder sorrendben foglalkozik a csomópontokkal. Vagy mind a mohó, mind a dinamikus programozás elsősorban döntési fán dolgozik, de az első optimális döntésekkel, a második optimális rész-döntéssorozatokkal építi fel az optimális megoldást. A kiegészítés érthetőbbé tételéhez célszerű úgy bevezetni a következő technikát, hogy kiemeljük, milyen pluszt ad az előzőekhez képest. Különös módon

hivatkozunk az összefüggés-kiegészítés párosra „A visszalépéses keresés és mohó algoritmus” részben, amikor a két technika összekapcsolási lehetőségét mutatjuk be.

A felülnézet órákat hangsúlyosan a szintézis és az összehasonlítás gondolkodási műveletek jellemzik. Az elágazás korláttal technikának a többi technika alkotta képbe való beillesztése az összes főbb gondolkodási műveletet igényli: analízis, szintézis, absztrakció, összehasonlítás, összefüggések megértése, kiegészítés, általánosítás, konkretizálás és analógia. Például, az „Algoritmustervezés felülnézetből” könyv [11] e témával foglalkozó fejezete általánosítást alkalmaz, amikor egy olyan algoritmus vázat javasol, amely a legtöbb A^* algoritmussal megoldható feladat esetében igen jól alkalmazható.

A gondolkodási műveletek e széles skálája magyarázattal szolgál az „Algoritmustervezés felülnézetből” tanítási-tanulási módszer hatékonyságára. Arra is rámutat továbbá, hogyan finomítja a módszer a diákok általános problémamegoldó gondolkodását, képességét is.

A módszer másik erőssége, hogy a különböző fogalmakat egymáshoz kapcsolva mutatja be, ami tartósabb mentális reprezentációt eredményez. Ez összhangban van azzal, ahogy a modern tudomány az emberi memóriát látja: „A memóriát ne úgy képzeljük el, mint egy tartályt, ami fokozatosan megtelik. Inkább egy olyan fához lehetne hasonlítani, amelyen kampók nőnek. Ezekre a kampókra kerülnek az információk. Minden elraktározott információ újabb kampókat jelent, melyekre még több információ kapcsolódhat. Ez azt jelenti, hogy a memória fokozatosan nő. Minél nagyobb ismerete van az embernek, annál nagyobb ismeret befogadására képes” [38]. Bár Russel ezen szavai elsősorban a memória növekedéséről szólnak, a szemléltetés azt is mutatja, milyen fontos az újonnan elsajátított dolgokat összekapcsolni a már ismertekkel. A felülnézet órákon éppen ez történik. A tanmenet végére a diákok fejében az algoritmustervezési stratégiákról kialakult kép nemcsak átfogó, tiszta és mély lesz, hanem egyben tartósan eltárolt is.

3 Látás, hallás és kinesztetikus érzékelés bevonása elemi algoritmusok tanításába

Az utóbbi évek izgalmas felfedezései az agykutatás, valamint a kognitív pszichológia területén megváltoztatták azt szemléletet, ahogyan a tudósok az emberi agyat, a tanulás központi szervét tekintették. A XVIII. század közepén, az ipari forradalom kezdetén divatossá vált az agyat egy géphez hasonlítani. Később, amikor a telefonközpontok lettek a haladás jelképei, az emberek egy nagy forgalmat lebonyolító központhoz hasonlították az agyat, aminek van kezelője, aki a döntéseket hozza. Most, hogy a számítógépek a bonyolult műveletvégzés szimbólumai, sokan a számítógéphez hasonlítják az agyat. A legújabb kutatások hatására viszont Edelman [27] dzsungelhez hasonlította az emberi agyat a maga körülbelül 50 milliárd neuronjával, egymillió milliárd szinapszisával, amelyben másodpercenként 10 millió milliárd ingerület keletkezik.

A tudósok körében ma már általánosan elfogadott tény, hogy az ember által tervezett fix szerkezetekkel és elektromos rendszerekkel szemben az emberi agy egy rendkívül rugalmas szerkezet, amely a tanulás és tapasztalás hatására folyamatosan változik mind strukturális, mind funkcionális szempontból, jó vagy rossz irányba. Természetesen mindenkinek más az élettapasztalata, és így a különböző kulturális, társadalmi, gazdasági és neveltetési háttérrel rendelkező emberek esetében nagyon más lehet az, ahogy gondolkodnak, tanulnak, kommunikálnak és viselkednek. Ezt a felismerést minden tanárnak és szülőnek a komolyan figyelembe kell vennie, ha sikeres szeretne lenni tanító-nevelő munkájában. Ugyancsak e kutatási eredmények szolgálnak alapul a különböző tanuláselméletek kidolgozásához [28].

A Harward Egyetemen a Project Zero kutatást vezető Gardner munkássága kimutatta, hogy különböző típusú tanulási készségek keveréke van jelen minden emberben. Az első könyvében [29] Gardner hét „intelligenciát” azonosított, amelyekhez a közelmúltban csatolt egy nyolcadikat. A megnevezett intelligenciák közé tartoznak például a muzikális és kinesztetikus érzékek, vagy a matematikai logika. Gardner kiemelte, hogy születésünkkor az összes intelligenciatípussal rendelkezünk, de a legtöbb ember az élete folyamán legfeljebb két intelligenciát fejleszt ki teljesen. Gardner kutatásai azt is megmutatták, hogy bár beszélhetünk egyéni „intelligenciáról”, minden ember domináns intelligenciája szorosan összefügg a többivel. Fontos tehát, hogy miután valaki azonosította, melyik számára a leghatékonyabb tanulási mód, ne csak erre az egyre támaszkodjon. Gardner munkásságának egyik fő mondanivalója: „Ha szeretnénk teljes lénnyé válni, törekednünk kell arra, hogy oly módokon is tanuljunk, amelyek nem kézenfekvőek és nem természetszerűek. Például, az elsősorban vizuális típusú emberek, vagy a logikára alapozva tanulók jól teszik, ha nem csak a legkényelmesebb intelligenciájukra támaszkodnak. Az ideális iskola számos különböző intelligenciatípus tekintetében biztosít tanulási lehetőségeket a diákok számára [30].

A fentebb említett kutatási eredmények értelmezése összhangban van azzal a ténnyel, hogy az emberi agy két féltekéje szorosan együttműködik. Az egyéni tanulási stílusok és az agyféltekék közötti kapcsolatra vonatkozó elképzelések gyakran figyelmen kívül hagyták, hogy a két agyfélteke mennyire összedolgozik. A legújabb kutatások egyik meghatározó hatása az oktatásra az, hogy a diákok nem kategorizálhatók kizárólag jobb vagy bal agyféltekével tanulókra [27].

Végkövetkeztetésként elmondható, hogy nincs két teljesen egyforma elme, ezért mindenkinek másként kell tanulnia. Ami az egyik embernél beválik, az a másiknál nem annyira eredményes. Levine [31] írja: „Ha minden gyermekkel egyformán bánnánk, akkor nem bánnánk velük egyenlően. A különbözőségük miatt más-más módszerre van szükségük ahhoz, hogy el tudjanak sajátítani valamit, és joguk van ahhoz, hogy ezt a szükségletüket kielégítsék."

3.1 Az érzékszervek és a tanulás

„A környezetünkkel olyan hatékonyan és könnyűszerrel létesítünk kapcsolatot, hogy fel sem fogjuk, milyen sok műveletnek köszönhetjük még a legegyszerűbb érzékelést is" [32].

Képzeljük el, hogy egy csendes úton biciklizünk. A lábunkban lévő érzékelők képessé tesznek bennünket arra, hogy akkora erővel nyomjuk a pedált, amekkora a sebesség tartásához szükséges. Egyensúlyozó szerveinknek köszönhetően nem borulunk fel, orrunkkal érezzük az illatokat, szemünkkel látjuk a tájat, fülünkkel halljuk a madarak csicsergését. Amikor szomjasok vagyunk, az ujjainkon lévő receptorok segítik, hogy meg tudjuk fogni a kulacsunkat. Ízlelőbimbóink, valamint hőérzékelőink tudatják velünk az ital ízét és hőmérsékletét. A bőrünkben lévő, valamint a szőrszálainkhoz kapcsolódó érzékelők tudomásunkra hozzák, hogy mennyire erős a szél, a szemünkkel együttműködve pedig azt, hogy mennyire gyorsan haladunk. Bőrünk a közvetlen környezetünk hőmérsékletét és páratartalmát is érzékeli, és eközben az időérzékünk jelzi, hogy körülbelül mióta vagyunk úton. Végül belső szervi érzékeink arra késztetnek, hogy pihenjünk és együnk. Igen, az élet az érzékek remek összjátéka!

3.1.1 Több mint öt érzékszerv

Biciklizés közben hány érzékszervünk működik? Csupán az az öt – látás, hallás, szaglás, ízlelés és tapintás -, melyet általában ismernek az emberek? Az Enciklopedia Britannica szerint Arisztotelész beszélt először arról, hogy ötfajta érzékelés van. Ez az ókori filozófus „olyan nagy hatással volt az emberek gondolkodására, hogy sokan még mindig öt érzékről beszélnek, mintha nem is lenne több".

E szerint az enciklopédia szerint azonban elég csak a bőrérzékelésről készült tanulmányokat megvizsgálni, és máris „bebizonyosodik, hogy az embernek ötnél több érzékszerve van". Hogyan lehetséges ez? Bizonyos funkciókhoz, mint például a tapintás, amelyeket korábban egy érzéknek tekintettek, ma már különböző érzékeléseket társítanak. A fájdalomérző receptorok különbséget tesznek a mechanikai, a kémiai és a hőmérsékleti hatások között, és másként reagálnak rájuk. Bizonyítékok támasztják alá, hogy legalább kétféle nyomásérzékelő receptorunk van: egyik az enyhe, felületi nyomást érzékeli, a másik a mélyebbre jutó ingert.

A belső szervi érzékelés a testünkben lejátszódó változásokat ismeri fel. Minthogy tudatosan érzékeljük az idő múlását, vannak, akik az időérzékünket is szeretnék bevonni az érzékek sorába. Van ezen kívül vesztibuláris, vagyis egyensúlyérző rendszerünk, mely a belsőfülben található; ez reagál a gravitációra, a gyorsulásra és a forgásra. Mi több, van kinesztetikus érzékelésünk is, mely képessé tesz minket az izmok feszítettségének az

érzékelésére, és ennek köszönhetően még akkor is érzékeljük végtagjaink mozgását és helyzetét, amikor csukva van a szemünk.

Persze nemcsak az emberekre jellemző az érzékelés. Az állatoknak is számos érzékük van, köztük néhány igazán bámulatos is, ami nincs meg bennünk. Mindezek ellenére, ahogyan az emberben az érzékek egyensúlyban vannak, az kétségkívül lehetővé teszi, hogy kiemelkedő lény legyen. Egy másik szempont, ami az embert egyedülállóvá teszi, az érzékszervek szerepe a tanulásban. Mindez Poincaré, francia tudós és matematikus szavait juttatja eszünkbe: "A tudós … azért tanulmányozza a természetet, mert örömét leli benne, és azért leli örömét benne, mert a természet gyönyörű."

Minden, amit tudunk a minket körülvevő világról, az érzékszerveink révén szereztük. Így érzékeinknek kitüntetett szerepük van a tanulás folyamatában. Mivel egy olyan társadalomban élünk, amelyben a legtöbb információ vizuális vagy auditív formában ér bennünket, az ismeretanyagunk 75%-át látás, 13%-át hallás és a többi 12%-át a többi érzékszervünk révén nyerjük. Bár a kutatók véleménye megoszlik, ami a számokat illeti, abban azonban egységesek, hogy a tanulás hatékonysága növelhető az érzékszervek kombinálásával [34].

3.1.2 Az érzékszervek és a memória

A tanulás folyamatának egyik legfontosabb eleme a memorizálás. Mi hasznát vennénk a tanultaknak, ha nem tudnánk visszaemlékezni rájuk, hogy különböző élethelyzetekbe felhasználhassuk ismereteinket. „Tantum scimus quantum memoria tenemus" (Annyit tudunk, amennyit emlékezetünkben megtartunk). A kísérletek kimutatták, hogy az olvasottaknak 10%-ára, a hallottaknak 20%-ára, a látottaknak pedig 30%-ára tudunk visszaemlékezni. Ha kombináljuk a látottakat a hallottakkal, akkor ez az arány 50%-ra emelkedik. Ha kérdésekkel ráirányítjuk a diákok figyelmét arra, hogy mit kell megjegyezniük a látottakból és a hallottakból, és megkérjük őket, hogy fogalmazzák is meg a tanultakat, akkor elérhető akár a 70%-os érték is. Sőt, a 90%-os emlékezési arányhoz is eljuthatunk, ha sikerül elérni, hogy a tanuló az összes érzékszervével *részt vegyen* a tanulásban. Ez viszont már művészi tanítást igényel [35].

Érdekes megjegyezni, hogy amíg a legtöbb ember a látására támaszkodva tanul, nem a látás útján szerzett ismeret memorizálása a „legtartósabb", hanem a szaglással érékelt. Igaz, rövid ideig képesek lehetünk a látottaknak akár a 100%-át is megjegyezni, de ennek 50%-át rendszerint elfelejtjük 3 hónap alatt. Ezzel szemben a szaglás útján nyert ismeret 80%-ára akár egy év múlva is visszaemlékezünk. Egyesek szerint ez talán azzal magyarázható, hogy nincs védőgát a szaglósejtek és a környezet ingerei között, ahogy a szemben és a fülben védetten levő érző idegsejtek esetében van. Ehelyett a szaglóidegek magában az agyból indulnak ki, és a külvilággal közvetlenül érintkeznek, így kevesebb „adatfeldolgozásra" van szükség. Az orr az a hely, ahol az agy és a környezet találkozik [36].

Másik igen hatékony érzékszervünk a memorizálás szempontjából az ízlelés, mely érzékelés szorosan kapcsolódik a szagláshoz. Bartoshuk a királynőnek nevezte az ízlelést az öt érzék birodalmában. Az ízlelés jó példa arra, ahogy az érzékszervek összedolgoznak. Amit mi laikusként ízlelésnek nevezünk, az valójában számos érzékelés

bonyolult összjátéka: a szaglás, az ízlelés, az érintés, az anyag állagának érzékelése, a látás, a kémiai hatás és a hőmérséklet érzékelése.

A hallásnak különleges szerepe lehet a tanultakra való visszaemlékezésben. Például, ha háttérzene szólt, amikor tanultunk, akkor ez a zene segíthet a tanultak későbbi felidézésében. A hallás szerepét a memorizálásban az is kiemeli, hogy a szavakat általában mind a hangzásuk, mind a jelentésük alapján eltároljuk agyunkban. Ezért tanácsolják a szakemberek, hogy az idegen szavak tanulásakor, vagy olvasás közben mondjuk is ki a szavakat [36].

A kettőskódolás elmélete [39] szerint a tanulási folyamat hatékonyabb, a mentális reprezentáció pedig tartósabb, ha az információközvetítés mind verbális, mind képi kódolással megtörténik [40]. Ez összhangban van az agyműködés agyfélteke-specializáció modelljével, mely szerint a szöveges, verbális kódolású információkat a bal, a képi kódolásúakat a jobb agyfélteke dolgozza fel. Egyes kutatási eredmények a képi kódolást helyezik előtérbe [41], ami különösen fontossá teszi a látás bevonását a tanulásba.

Összetett információtartalmak közvetítése során különösen célszerű a kettőskódolás, illetve a duplaszenzoros bemutatás alkalmazása. Így a terhelés több érzékszerven oszlik el, illetve az információ feldolgozása során segíthetjük az érzékszervek hatékony együttműködését. Például, ha komplex képeket és képsorokat auditív módon is értelmezünk (hangos szövegelmondásos magyarázat), a vizuális érzékelés a képekre koncentrálódhat, és a szöveges kommentár egyúttal irányíthatja a szemet, optimális sorrendet és tempót diktálva. [42] Egyféle kódolású információközvetítés esetén is célszerűbb két alapvető érzékszerv bekapcsolása. Például, ha egy olvasott szöveget hallgatunk is [43].

Minél több érzékszervet vonunk be a tanulásba, annál valószínűbb, hagy a tanulók vissza tudnak emlékezni a tanultakra. Az érzékszervek együttes jelenléte a memorizáláskor azt eredményezi, hogy az eltárolt információ több úton is elérhető a visszaemlékezéskor.

3.2 Az érzékszervek együttes bevonása a számítógépek programozása tanításába

Foglaljuk össze a fentebb felvázolt ismeret gyakorlati értékét az ebben a fejezetben bemutatandó didaktikai módszer szempontjából. Legalább öt érvet említhetünk meg, amelyek mind ugyanahhoz a következtetéshez vezetnek: a tanítás-tanulás folyamata annál hatékonyabb, minél több érzékszervet vonunk be.

- Több érzék, több információt jelent.
- Az, hogy mindenkinek más a tanulási stílusa, azt jelenti, hogy az egyes tanulóknak más-más lehet a domináns érzékszerve. Minél több érzéküket veszik igénybe a diákok, annál valószínűbb, hogy hatékonynak fogja találni az illető módszert.
- Gardner munkássága a nyolc „intelligenciáról” és más kutatások arra is rámutattak, hogy növeli a tanulás hatékonyságát, ha nem csak a domináns érzékszervünkre támaszkodunk. Más szóval, az érzékszervek kombinált alkalmazásukkal erősíthetik egymást.

- A kettőskódolás elmélete rámutat annak előnyére, ha a terhelés megoszlik a különböző érzékszervek között.
- Minél több érzékszervet vontunk be a tanuláskor, annál több úton érhető el az adott információ a visszaemlékezéskor.

E szempontok lényegét a következőképpen szemlélteti Levine [31]: „Úgy lehet a legjobban memorizálni valamit, ha megváltoztatjuk, ha valahogyan átalakítjuk az információt. Ha képekről és ábrákról van szó, öntsük az információt szavakba. Fordított esetben ábrázoljuk, készítsünk róla képet." Ez a megközelítés nemcsak eredményesebbé, de élvezetesebbé is teszi a tanulást.

A számítógép-programozás elsajátítása, ahogy a nyelvtanulás a csecsemőknél, legalább háromszorosan elvont folyamat. A programozási nyelv - ahogy az emberi nyelv is - maga is absztrakció a valósághoz képest, a nyelv segítségével megírt számítógépi program pedig kétszeres absztrakció. Háromszoros absztrakciót jelent, ha egy elvont algoritmust kell megérteni. A programozás tanulásakor ehhez hozzáadódik egy további absztrakciós szint, ugyanis a számítógéppel való kommunikáció megköveteli, hogy a programozó a gép „primitív gondolkodásmódjához" igazodjon, azaz, gépi szinten „magyarázza el" (a program által), hogy mi a gép teendője. Mindezeket figyelembe véve, a számítógép programozásának hatékony tanítása, illetve tanulása olyan módszereket és eszközöket feltételez, amelyeket átitat a szemléletesség didaktikai alapelve.

A szemléletesség lényege az, hogy kapcsolatot teremtünk az érzéki megismerés és az elvont gondolkodás között azzal a céllal, hogy megkönnyítsük az elvont ismeretek megértését, illetve segítsük az ismeretek rögzítését. Amint láttuk, annál hatékonyabb a szemléltetés, minél több érzékszervét foglalkoztatja egyszerre a diáknak. Sőt, mivel a legújabb felfedezések értelmében az egész testünkkel érzékelünk, így beszélhetünk a mozgást alkalmazó szemléletességről is.

3.2.1 A matematika ritmusa

Szolgáljon példaként, ahogy számos új-mexikói iskolában a matematikaoktatást összekapcsolják a zene- és tánctanítással. Tapasztalataik azt mutatják, hogy e látszólag teljesen különböző két terület sokat segíthet egymásnak. Mind a matematikára, mind a zenére jellemző a számok és a „mintázatok" alkalmazása. E „mintázatokat" a zenében és a táncban ritmusnak nevezik [30].

A programban résztvevő tanulók a dobolás révén megtanulták összekapcsolni a ritmust, a zenét és a mozgást az alapvető matematikai fogalmakkal. Például, az ütemek eltapsolása közelebb viszi őket a törtek matematikai fogalmához. Továbbá, a fél, negyed és nyolcad hangok dobolását megtanulva a tanulók eljuthatnak oda, hogy már a csontjaikban érzik a törteket, ami jó alapul szolgál összetettebb matematikai fogalmak elsajátításához. Ahogy táncolás közben a diákok összekapcsolják a lépéseket, hangokat és mozdulatokat, az további matematikai fogalmakhoz vezethet. E kezdetleges koreográfia lehetőséget nyújt a tanárnak, hogy kapcsolatot teremtsen a táncoló tanuló által leírt alakzatok és a geometriai alakzatok között.

Ahogy az új-mexikói iskolák tapasztalata is mutatja, az érzékszervek kombinált bevonása az olyan tantárgyak oktatásába, mint a matematika, de az informatika is, az elvont

fogalmak jobb elsajátításán túl egyéb haszonnal is jár, amelyek az oktatás affektív céljaihoz visznek közelebb:

- Mivel minden év végén sor kerül egy előadásra is, a tanulók már kicsi korban megtanulják, mit jelent kitartó erőfeszítéseket tenni, illetve együttműködni egy cél érdekében.
- A zene és a tánc izgalmassá teszi az órákat, ezzel együtt fogékonyabbak a tanulókat az absztrakt fogalmak elsajátítására.
- A zene és a matematika összekapcsolása révén tudatosult a diákokban, hogy mennyire jelen vannak a körülöttünk lévő világban az úgynevezett elvont tudományok is. Az oktatók azáltal is erősíthetik ezt a felismerést, hogy egyéb „mintázatokra" is felhívják a tanulók figyelmét. Például, a legtöbb növény az aranyszögnek nevezett sajátos szögben rendezi el új hajtásait, hogy azok csigavonalat alkossanak az optimális kihasználtságot érdekében. Továbbá, a csigavonalak száma rendszerint megegyezik a Fibonacci-sor valamelyik elemével. Ezekben a példákban is jelen van a művészet, hiszen régóta ismert tény, hogy az aranymetszés a legkellemesebb a szemnek. Az ilyen és ehhez hasonló példák rádöbbenthetik a tanulókat a tanultak gyakorlati értékére, és érdekesebbé is tehetik a tantermi órákat.

Az új-mexikói példa a zene és a tánc bevonására a matematikaoktatásba nem egyedi jelenség. Egyre több iskola megpróbálkozik összekapcsolni a reáltárgyak oktatását a művészetek tanításával. Törekvéseikben az agykutatás és a kognitív pszichológia területén elért legújabb eredményekre alapoznak, valamint arra az alapelvre, amire mi is: nevezetesen, hogy a tanulás annál hatékonyabb, minél több érzékszerv jut szerephez benne.

A következőkben egy olyan módszert és egy olyan szoftvereszközt mutatunk be, amelyek a szemléletesség elvével összhangban egyedülálló módon vonják be a látást, hallást és a kinesztetikus érzékelést az elemi algoritmusok programozásának tanítása-tanulása folyamatába. A dolgozat tartalmazza egy kísérlet leírását és kiértékelését is, amellyel empirikusan bizonyítjuk a módszer hatékonyságát. [62]

3.3 Elemi algoritmusok anatómiája

Bármely algoritmus[4] esetén beszélhetünk annak ciklusvázáról (csontvázáról), amely alatt az algoritmus ciklusutasításainak a szerkezetét értjük. A ciklusok magját képező utasításokat pedig úgy tekinthetjük, mint az algoritmus „húsos részeit". [64] Az alábbi kétlépéses módszert elemi algoritmusok[5] tanításához, illetve tanulásához ajánljuk:

1. A feladat elemzési szakaszában a megoldási algoritmus ciklusvázának a meghatározása.
2. A ciklusváznak a megfelelő utasításokkal való feltöltése.

[4] Csak olyan algoritmusokkal foglalkozunk, amelyek tartalmaznak ciklus utasítást és nem rekurzívak. Egyébként is, a rekurzió feloldható ciklus utasítások segítségével.

[5] Elemi algoritmus alatt olyan egyszerű algoritmusokat értünk, mint például számlálás, keresések, rendezések, stb. és ezek kombinálása.

Mivel a második lépés helyes megvalósítása feltételezi az első lépés helyes megtételét, a tanár – ha az irányítása vagy a felügyelete alatt folyik a feladatmegoldás - ne engedélyezze a második lépést, amíg a diák nem látja teljesen át az algoritmus ciklusvázát.

A következőkben egy példafeladaton szemléltetjük a módszert.

Feladat: *Írj C/C++ programot, amely természetes számokat olvas be nulla végjelig, és ellenőrzi, hogy a prímek számjegyei szorzatainak összege prím-e.*

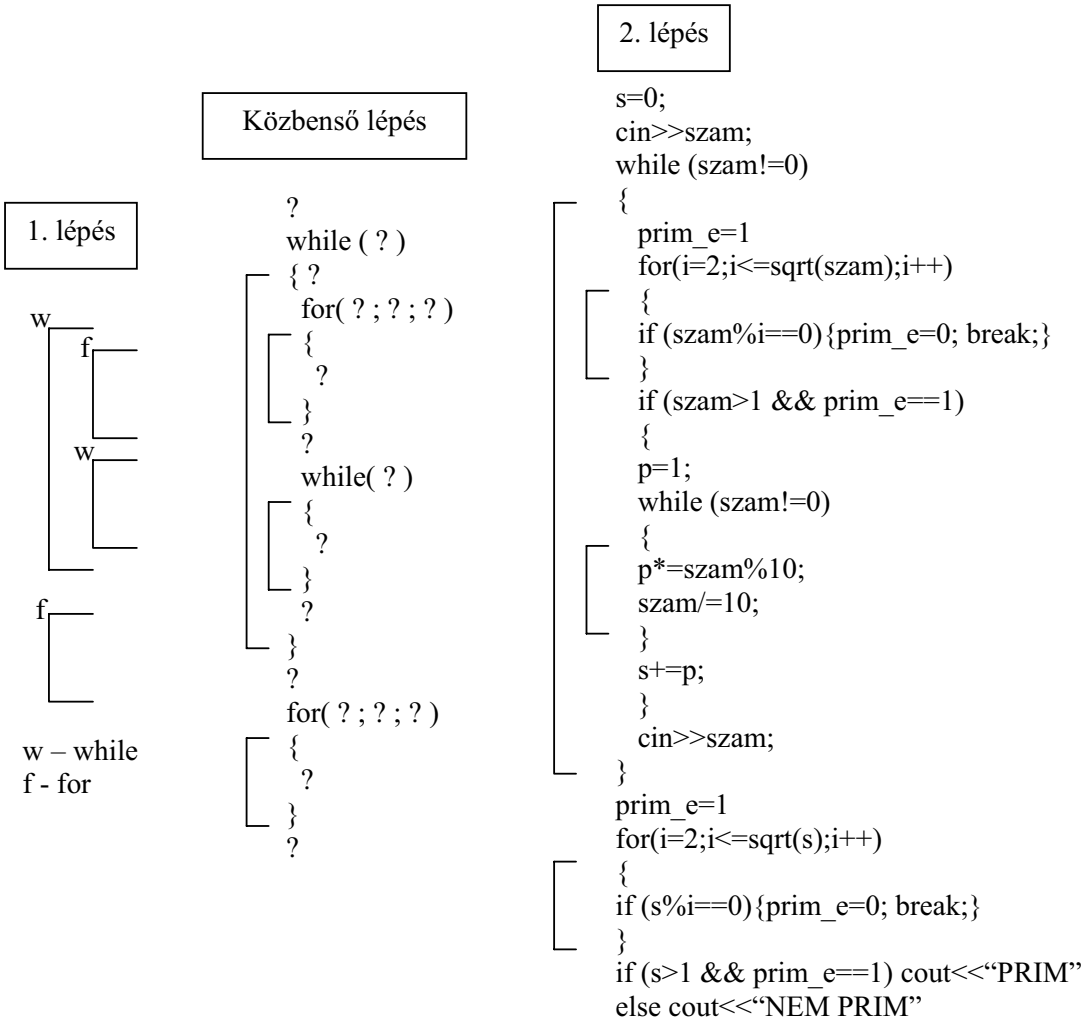
A feladat elemzése a ciklusváz meghatározása érdekében: Megkérjük a tanulót, hogy próbálja meg a számítógép helyébe képzelni magát és átlátni, milyen műveleteket feltételezne a gép részéről a feladat megoldása.

- Először sorra be kell olvasni a természetes számokat. Mivel az olvasás végjelig történik, egy *while* ciklussal utasíthatjuk erre a számítógépet.
    - Minden beolvasott számról mindjárt a beolvasása után jó lenne eldönteni, hogy prím-e. Mivel a prímvizsgálat feltételezi annak ellenőrzését, hogy a számnak van-e osztója kettőtől a szám gyökéig, ide egy újabb ciklusutasításra lesz szükség, például egy *for* ciklusra. Tekintettel arra, hogy *minden* számra el kell végezni a prímtesztet, a *for* ciklust be kell ágyaznunk a *while*-ba.
    - A prímellenőrzés befejezésével *minden* prímnek talált számhoz ki kell számolni számjegyei szorzatát. Ez azt jelenti, hogy a külső *while* ciklus, amely számról-számra lép, a prímtesztet végző *for* ciklus *után* tartalmaz egy újabb *while* ciklust, amely az egyes számok számjegyekre bontását végzi.
- Miután végigmentünk az összes számon, és megvan a prímek számjegyei szorzatainak összege, akkor erről az értékről még el kell dönteni, hogy prím-e. Ez a prímteszt egy - a számról-számra vivő *while* ciklust *követő* - *for* ciklussal valósítható meg.

A *3.1 ábrán* látjuk - a módszer első lépéseként - a feladatot megoldó algoritmus ciklusvázát.

A második lépés megtételében az alábbi Pólya-féle [24] rávezető kérdésekkel segíthetünk a tanulóknak:

- Mik a feladat input/output adatai?
- Milyen változókban (adatszerkezetekben) tároljuk el őket?
- Az algoritmusvázban hol és hogyan valósítsuk meg az input adatok beolvasását, illetve az output adatok kiírását?
- Milyen részfeladatokat kell megoldaniuk a „belső ciklusoknak”?
- Milyen segédváltozókat igényelnek a részfeladatok?
- Mit (ciklusmag) és meddig (ciklusfeltétel) ismételjenek az egyes ciklusutasítások?
- Az egyes részfeladatokhoz tartozó kezdőértékek megadásokra hol kerüljön sor (mely ciklusok előtt)?



3.1 ábra

Mindez, a fokozatos finomítás módszerével együtt, a következőképpen alkalmazható a példafeladatra:

- A bemenő adatok a természetes számok.
- Mivel mindegyik szám közvetlenül a beolvasása után feldolgozásra is kerül, használhatjuk ugyanazt a változót mindegyik számra a számítógép memóriájában való ideiglenes eltárolására. Legyen ez a *szam* nevű változó.
- A számok beolvasásának nyilván a fő *while* ciklusban kell történnie, amelyik megy számról-számra. Mivel végjelig olvasunk, ennek a módja:

```
cin>>szam;
while (szam!=0)
{
<a szam változóban eltárolt természetes szám feldolgozása>
cin>>szam;
}
```

- Minden szám prímtesztjét a belső *for* ciklus végzi. Mivel a *prim_e* változó inicializálása hozzátartozik a prímvizsgálathoz, ezért annak közvetlenül e *for* ciklus előtt kell történnie.
- Tekintettel arra, hogy csak a prímeknek kell kiszámolni a számjegyeik szorzatát, az ezt végző *while* ciklus egy *if* utasítás igaz ágára került. A szorzatszámításhoz szükséges *p* változó közvetlenül e ciklus előtt kap kezdőértéket.
- Mivel minden prímszám számjegyei szorzatának kell az összege, ezért az összegszámításra a fő *while* ciklusban kerül sor. Így az *s* összegváltozó e *while* ciklus előtt kap kezdőértéket, és a *p* változóban kialakuló számjegyszorzatok a belső *while* ciklus után adódnak hozzá.

A kezdő programozókat a feladat megoldására - a fenti elemzéssel - rávezetni önmagában is nagy kihívás. Milyen messze van még ettől, hogy a diákok képesek legyenek maguk is elvégezni ezt az elemzést!

A következőkben a módszer első lépésére összpontosítunk, pontosabban arra, hogyan segíthetünk a diákokban kifejleszteni azt a készséget, hogy felismerjék az algoritmus ciklusvázát. Egy szoftvert is készítettünk ehhez, amit az alábbiakban bemutatunk.

3.4 A szoftver

A kidolgozott szoftver célja, hogy lehetővé tegyük a látás, a hallás és a kinesztetikus érzékelés bevonását a ciklusváz felismerési készség kifejlesztésének folyamatába. Az alkalmazás Visul C++-ban készült és négy fő modulja van. A *cod_creator*, *cod_beautifier*, *cod_buherator* és a *run_code* modulok.

3.4.1 cod_creator modul

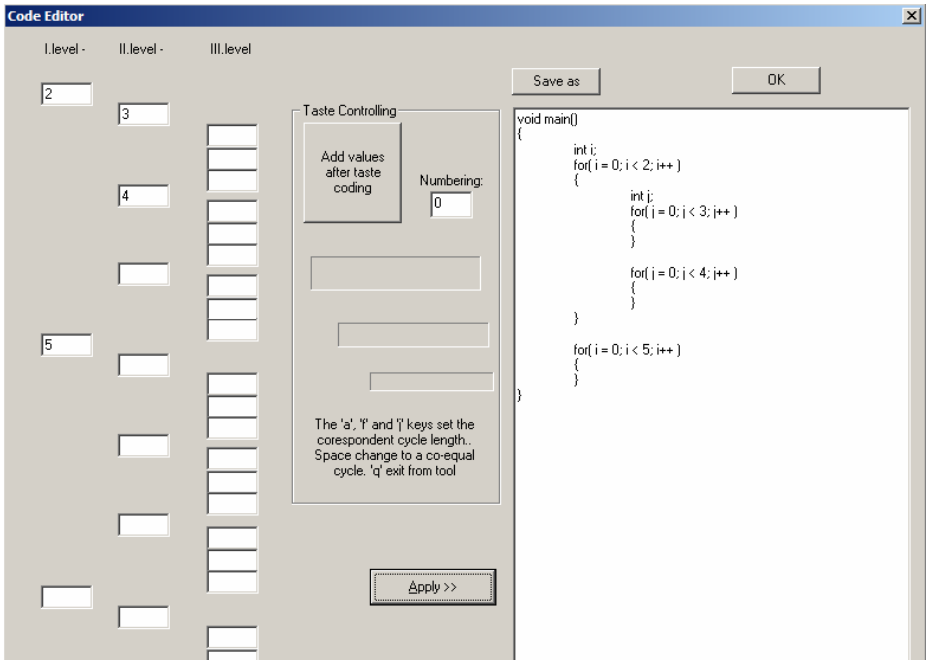
A *cod_creator* modul (mellékelve láthatjuk a felhasználói felületét; *3.2 ábra*) különböző ciklusvázú programok generálását teszi lehetővé. Két üzemmódban használható:

- *A ciklusváz paramétereinek megadása:* Egyszerűen beírjuk a I., II., illetve III. szint elnevezésű oszlopokba, hogy hány első szintű ciklust szeretnénk, és hogy ezekbe hány második, illetve harmadik szintű ciklus legyen beágyazva. Ezen kívül megadjuk, hogy az egyes ciklusokat hányszor kell végrehajtani. Az ábrán látható példában a kódgenerátortól olyan programot várunk, amelyben két (2-szer, illetve 5-ször végrehajtandó) első szintű ciklus van, és amelyek közül az elsőnek van két (3-szor, illetve 4-szer végrehajtandó) második szintű beágyazott ciklusa.
- *A ciklusváz „bedobolása":* A dialógus ablak *Drumming* területe felügyeli ezt az üzemmódot, lehetővé téve, hogy „begépeljük" a program ciklusvázát, mintha eldobolnánk a ritmusát, mintáját. Az első, második és harmadik szintű ciklusok „bedobolására" alapértelmezésben az *A*, *F* és *J* billentyűket használhatjuk. Az imént leírt ciklusváz „ritmusa" a következő:

afff_ffff afff_ffff aaaaa

ahol ’_’ karakterrel jelöltük a szóköz billentyűt, amit az egymást közvetlenül követő, azonos szintű ciklusok ritmusmintája között kell leütni.

A módszer alkalmazásának ennél a pontjánál vonjuk be a kinesztetikus érzékelést az elemi algoritmusok tanításába.



3.2 ábra

Az *Apply* feliratra/gombra kattintva a jobb oldalon látható – *for* ciklusokat alkalmazó - C/C++ program generálódik.

3.4.2 code_beautifier modul

Ezzel a modullal bármely C/C++ program forráskódja „megszépíthető”, azaz a szoftver oly módon rendezi át a kódot, hogy jól nyomon követhető legyen a ciklusváza. Különösen a látás bevonása szempontjából fontos ez a művelet. Például, a ciklusmagokat közrezáró kapcsos zárójelek külön sorokba kerülnek. Továbbá, a külső ciklusok kintebb, a belsők pedig bentebb kerülnek, hogy nyilvánvaló legyen a köztük fennálló alárendeltségi viszony. A *cod_creator* modul által létrehozott programvázak már eleve megfelelően tördelve jönnek létre (lásd fentebb). A *cod_beautifier* modult a diákok által írt programoknak a szoftver igényeihez való igazítására használjuk. Az alábbi példa a *cod_beautifier* modul szerepét hivatott szemléltetni:

„Szépítés” előtt:

```
s=0;cin>>szam;
while(szam!=0){
p=1;while (szam!=0){p*=szam%10;szam/=10;}s+=p;cin>>szam;}
cout<<s;
```

„Szépítés” után:

```
s=0;
cin>>szam;
while (szam!=0)
{
     p=1;
     while (szam!=0)
     {
          p*=szam%10;
          szam/=10;
     }
     s+=p;
     cin>>szam;
}
cout<<s;
```

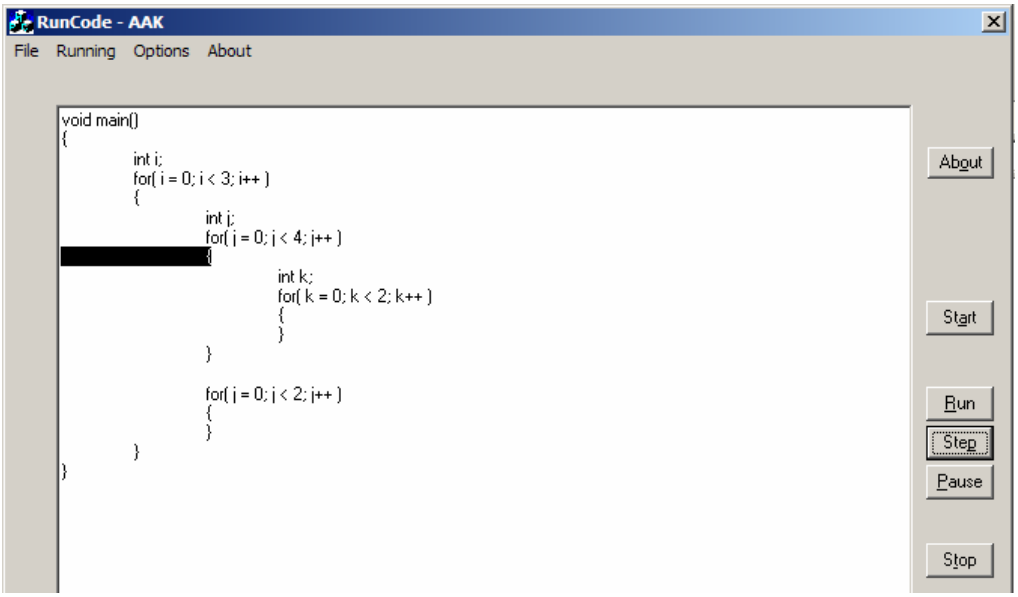
3.4.3 code_buherator modul

A *code_buherator* modul – átírva a forráskódot - az algoritmus minden ciklusmagjába hangokat megszólaltató, illetve a programfutást késleltető rutinokat épít be. Olyan, mintha „kihangosítanánk” az algoritmus ciklusvázát. A ciklusmagok minden egyes végrehajtásával megszólal egy zongorahang. A külső ciklusok mélyebb, a belsők magasabb hangokon „szólalnak meg”. Továbbá, a különböző szintű ciklusok különböző mértékű késleltetésével elérjük, hogy a külső ciklusok végrehajtásait jelző hangok kisebb, a belső ciklusoké pedig nagyobb frekvenciával kövessék egymást. Például, a fentebb bemutatott ciklusváz a következőképpen fog szólni (az 1. szintű ciklusokba a dó, a 2. szintűekbe a fá, a 3. szintűekbe pedig szí hangot építettünk be; az ’_’ karakterekkel a szünetek hosszúságát szemléltettük):

dó fá_fá_fá___fá_fá_fá_fá dó fá_fá_fá___fá_fá_fá_fá dó__dó__dó__dó__dó

3.4.4 run_code modul

E modul dialógus ablakában *(3.3 ábra)* megjelenik az aktuális algoritmus megszépített C/C++ kódja. Ez lehet a *cod_creator* modul által létrehozott valamilyen programváz, vagy a *cod_beautifier* modullal javított valódi algoritmus. A *Run* feliratra/gombra kattintva a háttérben elindul a *cod_buherator* modul által átdolgozott (kihangosított és lelassított) program. Miközben a tanuló hallja – a futó program révén - az algoritmus ciklusvázát szemléltető hangjegysort, a *run_code* ablakban követheti a szemével is, hol tart a végrehajtás. Természetesen, amit lát, az csak egy szimuláció.

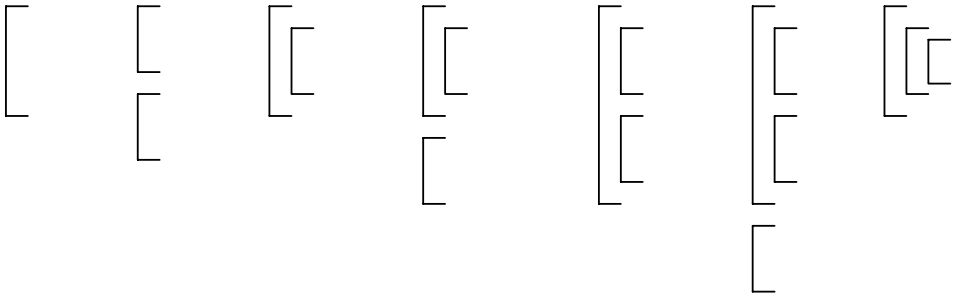


3.3 ábra

3.5 Javasolt tanmenet

A bemutatott szoftver alkalmazásához az alábbi tanmenetet javasoljuk:

- A tanár különböző ciklusvázakat „hallgattat meg” a tanulókkal, miközben a diákok szemükkel követik az algoritmus végrehajtását. Tanácsos végigmenni az alábbi ciklusváz mintákon:



- A tanulókat megkérjük, hogy próbálják meg hallás után felismerni a „lejátszott” ciklusvázakat. Rajzolják le a „meghallgatott” ciklusváz ábrát, és mondják meg, hogy az egyes ciklusokat hányszor hajtotta végre a gép.
- „Kihangosított” alapvető elemi algoritmusok végrehajtásának nyomon követése füllel és szemmel. Hogyan „szól” egy kereső algoritmus? És egy rendezés? Ha például a diákoknak a fülében marad a rendezések „hangja”, talán nem fognak megpróbálni egyciklusú rendező algoritmusokat írni.
- A diákok különböző ciklusvázakat „gépelnek be”. Ez egy nagyon fontos mozzanata a módszer alkalmazásának. Ahogy az új-mexikói gyerekek a zene és a tánc által úgymond a csontjaikban érezték a törteket, úgy a mi diákjaink is eljuthatnak oda, hogy, miközben ujjaik ütemes, zongorázásra emlékeztető

mozdulataival újabb és újabb ciklusvázakat gépelnek be, végül már bennük cseng az algoritmusok ritmusa. A módszer e szakaszát elméleti órákon is alkalmazhatjuk, ha nem is áll rendelkezésre számítógép és a szoftver. Először a tanár, majd a diákok a kezeiket és lábaikat használva ledobolhatják a katedrán, illetve a padokon a ciklusvázak ritmusát.

- Tipikus hibákat tartalmazó algoritmusok ciklusváz hibáinak felismerése hallás után. Tegyük fel, hogy a feladat azt kéri, számoljuk meg, hány prím található *n* darab természetes szám között. A diák érzékeli ugyan, hogy az *n* szám beolvasásához kell egy ciklus, és a prímteszthez egy másik, de nem ágyazza egymásba őket. A látás nincs, hogy segítsen neki észrevenni a hibát, különben nem írta volna már eredetileg sem hibásan a programot (a látás jelen van a programíráskor). Viszont, ha „meghallgatja" hibás algoritmusát, akkor nagyobb valószínűséggel rájön, hogy nem úgy hangzik, mintha minden számra megtörtént volna a prímvizsgálat. Ha megkérjük, hogy dobolja el, ahogy elképzelte az algoritmust, rögtön rájön, hogy az implementált, két egymás utáni ciklus nem úgy szól.

3.6 Kísérlet

A bemutatott módszer és szoftver hatékonyságának empirikus bizonyítására az alábbi kísérletet végeztük el. A marosvásárhelyi Bolyai Farkas Elméleti Líceum két IX. osztályát (*h* és *g*) vontuk be. Mindkét osztály az idei tanévtől (2005/2006) kezdte a programozást tanulni C/C++ nyelven. A kísérletre az első félév végén került sor. A két osztály ugyanazon tanterv szerint tanulta a programozást, de más-más tanár tanította őket.

A kísérlet előkészítéséhez végeztünk egy előzetes felmérést, amely eredményeként a *h* osztály átlaga közel egy jeggyel jobbnak bizonyult a *g* osztályénál a Romániában használatos 1-10 jegyskálán. Mindkét osztályt két-két (egy kísérleti és egy kontroll), egyenlő erősségű csoportba osztottuk. A *h* osztályból 13+13, *g*-ből 12+12 diák vett részt a kísérletben. Az osztályonkénti szétosztást nem a felmérés eredménye, hanem a diákok első félévi teljesítménye alapján valósítottuk meg. Miután a diákokat a félévvégi átlagaik alapján sorba állítottuk, az „egy ide – egy oda" módszert alkalmaztuk a szétválogatásukra. Ezt követően a tanulókat a csoportjukon belüli sorszámuk alapján azonosítottuk. Az elosztási módból adódóan mindkét osztályban olyan diákpárok alakultak ki (az *i*-edik párt az illető osztály *i*-edik kísérleti, illetve *i*-edik kontrolldiákja alkotta), amelyek tagjai majdnem egy szinten álltak a kísérlet előtt (a félévi átlagaik között legfeljebb egy jegy különbség volt). Azt vizsgáltuk, hogy milyen hatással lesz a kísérlet a párokon belüli tudásszint különbségre. Mindkét osztály esetében sikerült mind a kísérleti, mind a kontrollcsoportot oly módon kialakítani, hogy az egyes csoportokhoz tartozó diákok félévvégi átlagaik átlaga pontosan 8 legyen[6].

Két héten keresztül mindkét osztály kísérleti csoportjában a fentebb leírt tanmenet és kétlépéses módszer szerint történt az oktatás. A szoftver használata hozzájárult ahhoz is, hogy a tanórákat Shank szavainak szelleme jellemezze: „A tanár szerepe, hogy a tanulást érdekessé tegye, ami azt jelenti, hogy a tanulók élvezni fogják azt, amit tesznek." A

[6] A felmérés eredményével ellentétben (amely szerint a *h* osztály jobbnak bizonyult a *g*-nél) a két osztály félévvégi átlaga azonos volt. Ez valószínűleg a két tanár különböző értékelési rendszerével magyarázható. A felmérőt egységes kulcs szerint értékeltük ki.

kontrollcsoportokban a hagyományos módszereket használták anélkül, hogy törekedtek volna az érzékszervek bevonására, és persze anélkül, hogy használták volna a szoftvert.

A tesztet szándékosan nehéznek terveztük, hiszen elsősorban azt szerettük volna vizsgálni, hogy vezethet-e a módszer csupán kéthetes alkalmazása is jelentős eredményekhez. Az alábbiakban ismertetjük, milyen típusú feladatokat adtunk a kísérleti, illetve a kontrollcsoportok diákjainak:

- A tanulók különböző ciklusvázakat kaptak, és ki kellett találniuk olyan feladatokat, amelyek megoldási algoritmusa az adott ciklusvázak szerinti.
- Egy feladatnak úgy adtuk meg a megoldási algoritmusát, hogy közben összekevertük a sorait. A diákoknak rekonstruálniuk kellett a helyes algoritmust.
- Egy bonyolult feladat megoldási algoritmusának meg kellet határozniuk a ciklusvázát.
- Meg kellett találniuk egy viszonylag nehéz feladat megoldási algoritmusának a ciklusvázát, majd meg kellett írniuk magát az algoritmust is.

Az alábbiakban közöljük magát a tesztlapot:

1. Találj ki olyan feladatokat, amelyek megoldási algoritmusa

A) két ciklusutasítást tartalmaz egymás után;

B) egy külső ciklusban tartalmaz egy belsőt;

C) egy külső ciklusban tartalmaz két, egymást követő belsőt;

D) tartalmaz két egymásba ágyazott ciklust, majd ezeket követően egy következőt.

Indokold meg a válaszodat!

2. Legyen egy *n* elemű egész értékeket tartalmazó számsorozat. Hány eleme számjegyeinek összege páros, és hánynak páratlan? A feladatot megoldó C++ program sorait összekevertük. Állítsd helyre az algoritmust!

```
int szam, osszeg, parosok, paratlanok, n, i;
osszeg=0;
szam=szam/10;
parosok=0;
while (szam>0)
cin>>szam;
osszeg=osszeg+szam%10;
for (i=1;i<=n;i++)
paratlanok=0;
cout<<paratlanok<<parosok;
if (osszeg%2==0) parosok++;
else paratlanok++;
cin>>n;
{
```

```
{
}
}
```

3. Pozitív egész számokat olvasunk be 1 végjelig. Hány köztük a prím? A feladatot megoldó C++ program sorait összekevertük. Állítsd helyre az algoritmust!

```
int szam, prim_e, primek, i;
cin>>szam;
cin>>szam;
cout<<primek;
primek=0;
prim_e=1;
while (szam!=1)
for (i=2;i<=sqrt(szam);i++)
if (szam%i==0) prim_e=0;
if (prim_e==1) primek++;
{
{
}
}
```

4. Adott *n* darab természetes szám. Írj programot, amely kiírja a tükörképét azoknak a számoknak, amelyek számjegyeinek összege prím. Rajzold fel az algoritmus ciklusvázát is (használd az első feladatbeli jelöléseket).

5. Adott *n* darab számpár. Írasd ki a legkisebb közös többszöröseit azon számpároknak, amelyek tükörképei relatív prímek. Használva az első feladatbeli jelöléseket, mi lenne e feladat megoldási algoritmusának a ciklusváza (az algoritmust nem kell megírni)?

3.6.1 A teszteredmények és kiértékelésük

A teszteredményeket az alábbi grafikonokon mutatjuk be *(3.4, 3.5 ábrák)*. A diákokat – amint már említettük – a csoportjukban kapott sorszámuk (félévi átlaguk szerint növekvő sorrendben) azonosítja. Ezt ábrázoljuk a vízszintes tengelyen. Az elosztási módból adódóan, az egyes osztályok különböző csoportjainak azonos sorszámú diákjai között a félévi eredményeik (zöld és narancssárga oszlopok) szerint legfeljebb egy jegy különbség van. A bordó és kék oszlopok a kísérlet utáni felmérés eredményeit ábrázolják. Mindenik diákpár esetében feltüntettük - a kísérletből adódott - relatív szintkülönbség változást. Ezeket az értékeket az alábbi képlettel számítottuk ki:

$$\text{Félévi\_Kontroll}_i\text{-Félévi\_Kísérlet}_i\text{+(TESZT\_Kísérleti}_i\text{-TESZT\_Kontroll}_i\text{)}$$

Mivel mindkét osztály esetében az egyes párok között fellépett relatív szintkülönbség változások normális eloszlást mutatnak, ezért az egymintás t-próbát alkalmaztuk, és a következő eredményekhez[7] jutottunk:

[7] A két osztály által elért különböző eredmények a tanárok közötti különbözőségekkel magyarázható.

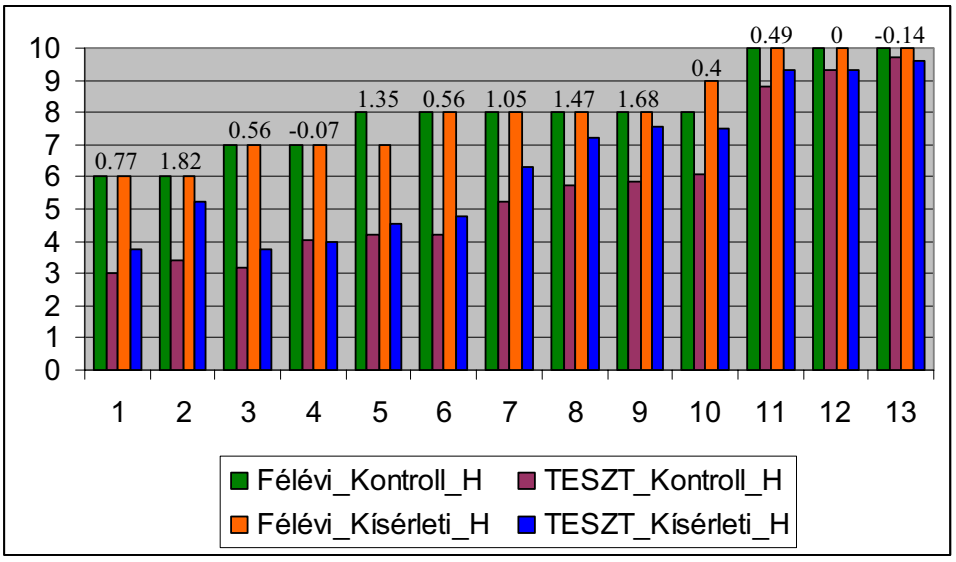
- A h osztály kísérleti eredményei azt mutatják, hogy az új módszer alkalmazása várhatóan (0.9804 valószínűséggel) 0.76 jeggyel fogja emelni az osztály átlagot.
- A g osztály esetében ez az érték 1.35, 0.9897 valószínűséggel.
- Ha a két mintát együtt vizsgáljuk akkor 1.05 értékű átlagos javulásra számíthatunk 0.9957 valószínűséggel.

E szignifikáns eredmény úgy tekinthető, mint a módszer hatékonyságának kísérleti bizonyítéka.

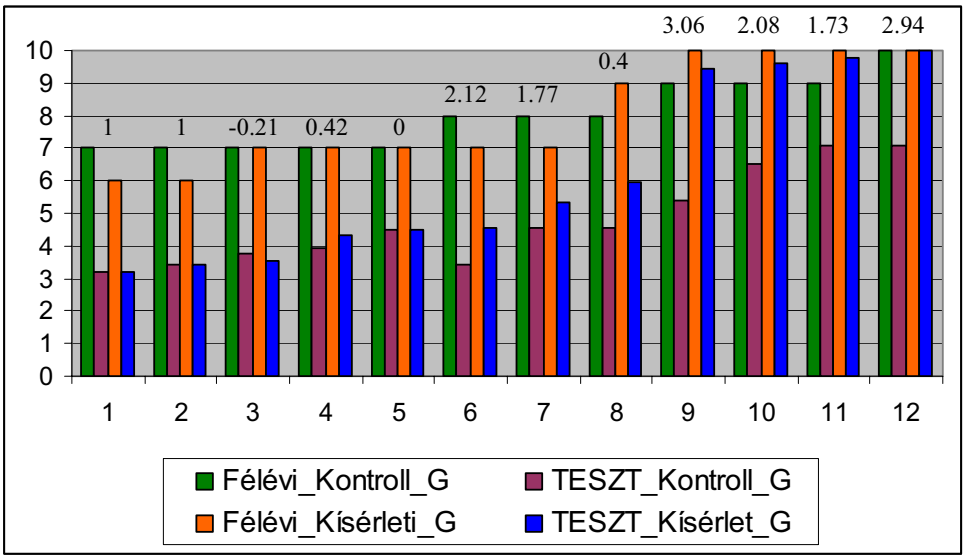
Érdekes megfigyelni, hogy az előzetes felmérés alapján jobbnak bizonyult (*h*) osztály legjobb tanulói (10-13 sorszámú párok), függetlenül attól, hogy részvettek-e a kísérletben, nagyon jó eredményeket értek el. Ez aláhúzza azt a megállapítást, hogy a nagyon jó diákok esetében kevésbé fontos a módszer.

A félévi átlagokhoz képest alacsonyabb eredmények a teszt nehézségi fokával, az osztályok teljesítménye közötti különbség pedig a felmérővel kimutatott szintkülönbséggel magyarázható.

	Félévvégi átlag	A teszt eredménye
g - kontrollcsoport	8	4.77
g – kísérleti csoport	8	6.13
h - kontrollcsoport	8	5.6
h - kísérleti csoport	8	6.36



3.4 ábra



3.5 ábra

3.7 Következtetések

A szoftver alkalmazása lehetővé tette, hogy a diákok úgymond ki tudják tapintani az algoritmusok pulzusát. Valóban egész testtel való tanulást biztosít. A tanulók láthatták, halhatták és – a ciklusvázak bedobolásával - érezhették is az algoritmusok lüktetését. Mindez - a bevezetőben említett kutatási eredményekkel összhangban - egyértelműen hatékonyabb tanulást jelent. A több érzékszerv nemcsak több információt és jobb memorizálást jelent, hanem hozzájárul a más-más domináns érzékszervű diákok „esélyegyenlőségéhez" a tanítás-tanulás folyamatában.

Egy számítógépi programban megtestesülő algoritmus elemzése a kód alapján hangsúlyosan a bal agyféltekével történik. A bemutatott módszer oly módon vonja be a tanulásba az érzékszerveket, hogy e vizsgálódás a jobb agyféltekét is igénybe vegye. Például, az algoritmusok ciklusvázának kihangosítása miatt a lassított futtatás alatt a „megszépített" kód ciklusutasításait, mint szöveges kódolású információkat, a bal agyfélteke dolgozza fel. Ezzel párhuzamosan, a különböző hangmagasságú és frekvenciájú hangjegyek, valamint az aktuális programsornak a hangjegysorral összehangolt kiemelése révén, a jobb agyfélteke jut információhoz az algoritmus ciklusvázának felépítéséről. Tehát egy olyan kódolásról van szó, amikor az információt három különböző formában közvetítjük az agynak: szöveges formában a programkód utasításai révén, hangjegysorként, valamint képi kódolású „szövegkiemelés" által. A háromféle reprezentáció harmonikusan fonódik össze, hiszen miközben fülünk érzékeli a hangjegyeket, a szövegkiemelés vezeti szemünket a futtatott kódon. Mindez hatékonyabbá teszi a tanulási folyamatot, illetve tartósabbá a mentális reprezentációt [40, 42, 43].

Amint a fentiekben már említettük, azok az új-mexikói iskolák, ahol a „Matematika ritmusa" módszert alkalmazzák [30], a tanév végén műsort szerveznek a szülőknek. Ebből az ötletből kiindulva megkérhetjük a „zenész diákokat" (akik játszanak valamilyen hangszeren), hogy „a következő órára" különböző ciklusváz ritmusokkal készüljenek,

amelyeket az osztálynak majd hallás után kell azonosítania. Nem nehéz elképzelni az affektív síkon kifejtett jótékony hatását annak, amikor a tanulók saját osztálytársukat hallgatják, amint az különböző elemi algoritmusok ciklusvázának ritmusát furulyáján fújja, vagy gitárján pengeti.

A bemutatott módszer a problémamegoldó képesség fejlesztéséhez, valamint az algoritmikus gondolkodás kialakulásához is hozzájárul. Emlékezzünk Pólya elméletére [24], miszerint a problémamegoldás fő lépései: a feladat megértése; a tervkészítés; a terv végrehajtása; a megoldás vizsgálata. Gyakori probléma a diákoknál, hogy a feladatmegértési fázist elnagyolják, a tervkészítést pedig egyszerűen kihagyják. Az általunk ajánlott módszer kényszeríti a tanulót a Pólya-féle stratégia szoros követésére. A tervkészítés a ciklusváz azonosításával kezdődik, ami viszont a feladat alapos megértése és elemzése nélkül lehetetlen. A tervkészítés folytatásaként, az elemzés egy következő szakaszában, azonosítjuk a részfeladatokat, ami azután elvezet a feladat ciklusvázának, a megfelelő utasításokkal való „feltöltéséhez". Mivel a belső ciklusok általában a részfeladatok vázai is, ezért a ciklusváz meghatározása, és a részfeladatok azonosítása műveletek gyakran összekapcsolódnak. A Pólya-módszer tervkészítési fázisát az elemi algoritmusok területére konkretizálva mintegy algoritmizáltuk az e feladatcsaládba tartozó problémák megoldási lépéseit.

A tanulók problémamegoldó készségének javulását az is alátámasztja, ahogy a módszer különböző gondolkodási műveleteket aktivizál bennük. Például, a ciklusváz azonosítása analízist és absztrakciót, az utasításokkal való feltöltése pedig kiegészítést feltételez.

A módszert alkalmazó tanárok elmondják, hogy a programírás folyamatának két jól elhatárolt szakaszra bontása (a ciklusváz azonosítása, majd ennek utasításokkal való feltöltése) megértette a diákokkal, milyen hasznos már az elemi algoritmusokat is megtervezni. Így lényegesen kevesebb hibás algoritmus született. Csökkentette például azon hibák számát, amikor a diák tudja ugyan, hogy milyen típusú utasításokat kell használnia, de nem a megfelelő helyekre írja be őket az algoritmusba.

A kísérleti eredmények mintegy aláhúzzák a módszer ezen elméleti megalapozását.

4 „Legyél te is eminens” - Értékelési módszer és szoftver

Ebben a fejezetben a tanítás-tanulás-értékelés folyamat értékelés fázisára helyezzük a hangsúlyt. Egy olyan didaktikai módszert és eszközt mutatunk be, amely az oktatás e területén vezet előrelépéshez.

Először röviden áttekintjük az értékelés pedagógiai szerepét, majd az ellenőrzésre, illetve a számítógépi tesztelésre - mint a tanulók értékelésének napjainkban népszerű módjára - összpontosítunk. Mivel hiányoznak az igazán hatékony módszerek és a modern eszközök, amelyek képesek érdekessé is tenni az értékelés folyamatát, áttekintünk néhány próbálkozást a „Legyen ön is milliomos” című népszerű TV-játéknak didaktikai eszközként való implementálására. Ezt követően bemutatjuk a „Legyél te is eminens” hálózati szoftvert, valamint rámutatunk arra, hogy miként nyújt többet az általunk kidolgozott értékelési módszer és eszköz az eddigi megvalósításoknál.

4.1 Értékelés a pedagógiában

Az értékelés az a folyamat, amelyben szisztematikusan vizsgálják, illetve megítélik valamilyen jelenségnek, eseménynek vagy tárgynak az értékét. A XX. század második felének pedagógiájára nagy hatást gyakorolt Tyler értelmezése, miszerint az értékelés információt ad a célok megvalósulásának a mértékéről. Más szóval, az értékelés alkalmával összefüggéseket keresünk a célok, a folyamat és a megvalósult végállapot között. Az elvárás és a valóság közötti különbségből nyert visszacsatolás alapján javíthatunk a célon, illetve optimalizálható a folyamat és így az elért eredmény is. Ebből a szempontból az értékelés már nemcsak nevelési és oktatási módszer, hanem „rendszerszabályozó elem” [44, 45].

A neveléstudomány magyar és osztrák jeles képviselői 2003-ban ugyanezzel a koncepcióval adták közre a „Tanítás, tanulás, értékelés” című tanulmánykötetet, nevezetesen, hogy a tanítás, a tanulás és a vizsgáztatás az oktatási folyamat komponensei. A célok ugyanis azonosak. „Ebből a szemléletből következik, hogy a vizsgáztatás szerepe elsősorban nem az, hogy a szelekció iskolai, illetve társadalmi eszköze legyen, hanem az, hogy egy visszacsatolási mechanizmus szükséges elemeként működjön a tanítási, tanulási és vizsgafolyamatban” [33].

Az utóbbi évtizedekben Magyarországon is teret nyert a pedagógiai szakirodalomban a tanítás-tanulás rendszer szintű modellezése. A nemzetközi kutatásokban felhasznált modellek közös jellemzője megint csak az, hogy a pedagógiai értékelés a tanítás-tanulás modelljeinek alapvetően fontos rendszereleme.

A visszacsatolás elsődleges célja a szabályozás. Az értékeléssel szerzett információk felhasználásával szabályozható úgy a tanítói, mint a tanulói tevékenység. A tanár esetében e szabályzás a közvetített tartalmak, valamint a használt módszerek és eszközök felülvizsgálata és módosítása révén történhet meg. Az értékelés a tanulóknak képet nyújt arról, hogy milyen mértékben sikerült elsajátítaniuk az átvett anyagrészt, illetve milyen hatásfokkal sikerült ismeretüket használható tudássá alakítani. Mindez serkentőleg hathat

rájuk. Előnyös, ha gyakran keletkezik visszacsatolt információ, hiszen így a tanítás-tanulás folyamata finomabban szabályozható [46].

Az értékelés folyamatát különböző megközelítésekben írják le. A pedagógiai szakirodalomban, gyakori a következő leírás [45]:

1. az értékelési probléma megértése;
2. az értékelés tervezése (funkció, célok, kritériumok, folyamat, módszerek stb. meghatározása);
3. adatgyűjtés;
4. az adatok elemzése, értelmezése;
5. javaslatok megfogalmazása.

Az adatgyűjtésnek számos módszere használatos: megfigyelés, szóbeli és írásbeli feleltetés, tesztek, dokumentumok elemzése, esettanulmányok stb.

Ma már az oktatásban alkalmazott módszerek sokfélék: részben továbbélnek a hagyományos értékelési formák, részben kibővültek a tudományos kutatás módszereivel és az azokhoz kötött módszertani követelményekkel (objektivitás, validitás, megbízhatóság). A hagyományos értékelési formák egyik hiányossága az, hogy az oktató csak viszonylag kevés és nehezen feldolgozható (elemezhető és értelmezhető) visszacsatoló információhoz jut. Továbbá e számonkérési formák objektivitása is vitatható.

A számítógépek, valamint a számítógép-hálózatok megjelenése a tesztre irányították a figyelmet, mert kiküszöböli a hagyományos módszerek számos hiányosságát. Mivel az általunk kidolgozott módszer is ebbe a kategóriába tartozik, az alábbiakban részletesebben bemutatjuk a tesztelést, mint a pedagógiai értékelés eszközét.

4.2 A tesztekről általában

4.2.1 Mi a teszt?

Tesztelni annyit tesz, mint próbára tenni. Az angol teszt szó eredeti jelentése próba. A népi fejtörőket tekinthetjük az első teszteknek. Teszten manapság elsősorban olyan kérdéssort értünk, melyben minden kérdésre megadnak néhány lehetséges választ, és ezek közül kell a helyeset kiválasztani. A többféle válasz lehetősége miatt nevezzük az ilyen teszt kérdéseit *feleltválasztós* típusúnak. A legtöbb teszt kérdéseinek egyik további jellegzetessége, hogy a lehetséges válaszok közül csupán az egyik helyes. Ezekre példák az igaz-hamis választ váró kérdések, az üres hely kitöltésére vonatkozó kérdések, a kakukktojás keresése, vagy akár a párosítás is. [47]

4.2.2 A jó teszt feltételei

Binet [48] teszttel kapcsolatos feltételei:

- lehetőségekhez képest sokoldalú legyen;

- magasabb rendű képességekre vonatkozzon;
- megoldása rövid időt vegyen igénybe;
- legyen változatos, ne fárasztó és unalmas;
- illeszkedjen abba a környezetbe, amelyben a vizsgált személy él;
- ne igényeljen drága műszert.

A tesztnek objektívnek, megbízhatónak és érvényesnek kell lennie.

4.2.3 A feleletválasztós tesztek pontozási lehetőségei

1. Minden helyes válasz egy pontot ér. A helytelen válaszokat nem büntetjük. Nyilvánvaló, hogy ebben az esetben a tesztet kitöltő érdeke az, hogy minden kérdésre (arra is, amit nem tud) adjon valamilyen, akár véletlenszerű választ.

2. A helyes válaszokért egy pont jár, a helytelenekért valamennyi (*n)* pontot levonunk a tanulótól. Ebben az esetben a vizsgázónak az az érdeke, hogy csak azokra a kérdésekre adjon választ, melyeket biztosan tud. Ha $n = 2$, csak akkor érdemes válaszolnia, ha 66%-ban biztos a válaszában. Megmutatható, hogy az $n = 2$ választás a legjobb, ugyanis nagyobb n-ekre a büntetés gátolja a válaszadást. A tapasztalatok azt mutatják, hogy ilyen pontozás mellett a pontszám és a valódi tudás között nagyobb a korreláció, mint az első pontozási rendszert használva. Nagyon fontos a tesztet kitöltőkkel megismertetni, hogy milyen lesz a pontozás, és így a kiértékelési elveknek megfelelően válaszoljanak. Az első ilyen jellegű tesztek az átállás után nem adnak megfelelő eredményt, mert még nem tudatosult a vizsgázókban, mi is az érdekük. A későbbiekben azonban alkalmazása a korábban említett előnyökkel jár. Ez a módszer $n = 0$ esetén azonos az első módszerrel.

3. Az egyes kérdésekre adott válaszokat különböző súllyal kívánjuk figyelembe venni. Nagyobb tudásanyag ellenőrzésekor az objektív érdemjegy megállapításához súlyozásra nyilvánvalóan szükség van, mert a kérdések között lehetnek fontosak és kevésbé lényegesek, valamint olyan kérdések is, melyekre adott hibás válaszok alapvető hiányosságokra utalnak.

4. Minden kérdésre adott válasz mellett fel kell tüntetni, hogy mennyire biztos benne a vizsgázó. Így olyan mennyiségeket határozhatunk meg, melyek nem csupán a tudás mennyiségét, de annak biztosságát, vagy akár a félreértéseket is felfedik. Aki tudásában biztos - bár többet hibázott, és a hibásakban sem volt biztos -, értékesebb, használhatóbb tudással rendelkezik, mint aki sok kérdésre adott helyes, de bizonytalan választ. Jellegzetes tudásproblémák:
   - *Félreinformált*. Legalább 10%-ban olyan választ adott, amit biztosan helyesnek tekintett, pedig nem volt az, azaz sok a „biztos, de rossz" válaszainak száma.
   - *Bizonytalan*. A helyes válaszok kevesebb mint 50%-ban voltak biztosak. Hiába adott sok jó választ, nem elég magabiztos.

   Ez a típusú teszt pontosabb képet ad a tanuló használható tudásáról, és segíti a hatékony tanulást, önképzést. Segítségével megállapítható, mely területeken

bizonytalan vagy félreinformált a vizsgázó, így ezekre lehet koncentrálni, és így jobb eredményeket elérni a pontosabb visszacsatolás révén.

5. A tesztekben arra adunk lehetőséget a tanulónak, hogy ha több válasz közül nem tudja eldönteni, melyiket válassza, a helytelent is kiválaszthatja. A pontozás 3 lehetőség közül való választás esetén:
   - megjelöli a helyes választ: +3 pont;
   - megjelöl egy helytelen választ: +1 pont;
   - megjelöl egy rossz választ: -10 pont;
   - nem ad választ: 0 pont.

   Ez a pontozási rendszer jutalmazza a hiányos tudást is, de a helytelen ismereteket keményen bünteti. Alaposabb vizsgálattal megállapítható, mely területek tanulására, tanítására kell több időt szánni.

6. A kérdésekre egy helyes válasz van, és ezt kell megadni. Egészítsük ki a lehetőségeket a következőkkel, és vizsgáljuk meg az erényeiket:
   - Egyik válasz sem helyes. Ez a lehetőség a tanulót arra ösztönzi, hogy ha csak valószínűnek tart egy választ, elgondolkozzon annak helyességén. Általában, az ilyen típusú kérdések megválaszolásával telik el a legtöbb idő.
   - Minden válasz helyes. Ezzel a vizsgázót arra kényszerítjük, hogy minden választ végiggondoljon, ne álljon meg az első helyesnél.
   - Hiányzó ismeret. Például, a "Hány éves volt Neumann János?" kérdés megválaszolásához tudni kellene, melyik évben. A lehetőség arra sarkallja a diákot, hogy elgondolkozzon a kérdés jelentésén, és felfedezze, hogy milyen információ hiányzik a megválaszoláshoz. Az ilyen kérdések nagyon közel állnak a valós élethez, hiszen gyakran szükség van újabb vizsgálat elvégzésére, hogy eredményt mondhassunk.
   - Logikátlan kérdés. Például, "Miért tiltakozott Pascal, hogy róla nevezzék el a nevét viselő programozási nyelvet?" Bizonyos esetekben az ilyen kérdések alkalmasak lehetnek a „mély megértés” vizsgálatára. Ezt a típusú kérdést válaszolják meg a leggyorsabban és legjobb eredménnyel a tanulók.

4.2.4 A teszt életciklusa

4.2.4.1 Előkészítés

A tesztet megíratás előtt elő kell készíteni. A feladatsorban szereplő elemeket (egyedi kérdéseket) előzetesen meg kell vizsgálni, jók-e. De milyen a jó kérdés? A válasz látszólag könnyű: a kérdés jó, ha az előre elkészített értékelési szempontok szerint pontozott tesztet az írja meg jól, aki tényleg elsajátította a visszakérdezni kívánt anyagot. Ezt azonban nem egyszerű elérni! Például, fontos, hogy a tesztkérdések lefedjék az adott anyagrészt, ugyanakkor az egyik kérdésből ne következzék a másikra a válasz, mert ekkor az, aki szinte mindent tud - csak erre nem készült –, duplán nehéz helyzetbe kerül, és tényleges tudása alatt teljesít. A teszt nehézségi fokát, azaz a legkönnyebb és legnehezebb kérdéseket, is jól kell megválasztani.

Széles körben végzett felmérés esetén egy előzetesen gondosan kiválasztott csoporton érdemes a tesztet kipróbálni, és a kapott eredményeket értékelni. A teljes felmérés eredményét így ennek a csoportnak a normájához tudjuk hasonlítani. Természetesen, csupán olyan esetekben lehet az így kapott eredményt értékelni, ha a tesztet megoldó körülménye hasonlít a normacsoportéhoz.

4.2.4.2 Értékelés

A teszteket megírásuk után értékelni kell. Ez történhet gépi, emberi vagy kevert módon. A módszert a teszt értékelésének jellege határozza meg. A feleletválasztós tesztek értékelését nyugodtan rábízhatjuk a számítógépre. Az ellenőrzés, értékelés alapvető célja, hogy visszajelzést adjon az oktatás eredményességéről. De lehetnek további célok is:

- A *van* és *kell* értékek összehasonlítása, és ennek alapján az oktatási folyamat befolyásolása.
- A hallgató, a szülők és a társadalom tájékoztatása.
- A tanítás tartalmának, szervezetének, eszközeinek és módszereinek fejlesztése.

A korábban meghatározott értékelési szempontok, pontozási módszerek segítségével a teszt első információs szintjéhez jutunk. Sokan megelégszenek ennyivel, pedig még számos hasznos tapasztalatot, eredményt kaphatunk!

A fenti tesztek segítségével meg lehet vizsgálni a populáció fejlődését. Az eredményeket össze lehetne hasonlítani más területen, hasonló körülmények között végzett tesztek eredményével. Vizsgálni lehet az oktatás hatékonyságát, stb.. Előzetesen azonban meg kell néznünk, vajon a teszt megbízható volt-e, az eredmény reális-e, mi az érvényességi köre. Erre jól kidolgozott matematikai statisztikai módszerek állnak rendelkezésre.

4.2.4.3 A kapott értékek értelmezése

Vegyük példaként az intelligenciateszteket. Egy roppant fontos szempont, ugyanakkor a laikus nagyközönség és más szakterületek képviselői is elfeledkeznek róla, hogy a kapott IQ pontoknak megbízhatósági intervallumuk van. Ez azt jelenti, hogy az általánosan használt Stanford-Binet és Wechsler teszt eredménye 95%-os valószínűséggel esik az eredmény körüli (-9,+9) intervallumba, azaz aki 110 pontot ért el, annak a valódi IQ-ja 101 és 119 között van. A mérési hibákból szükségképpen adódó bizonytalanságoknak különösen akkor van jelentősége, ha az eredményétől sorsok függnek, például kisegítő iskolai felvételik. Két tanulót megbízhatóan csak akkor rangsorolhatunk, ha a mért értékek által meghatározott intervallumok nem fedik egymást.

4.2.4.4 Megbízhatóság és érvényesség

A teszt megbízhatósága azt jelenti, hogy a teszt jól méri-e és azt-e, amire tervezték. Hasonló tudást mérő kérdésekre, hasonló választ kapunk-e. A vizsgázó csupán bemagolta az anyagot, és már ismerte a kérdésre a választ, vagy valóban a helyszínen kellett-e rájönnie. Ugyanazon kérdéssorban hasonló kérdésre azonos választ adott-e, mert ha nem,

akkor valamelyik válasza nem a tudását tükrözi (vagy elnézte az egyiket, vagy jól tippelt a másiknál). Ezt párhuzamos formulákkal vizsgáljuk, és az elért pontszám korrelációját nézzük. Az értékelési rendszer kis változtatásának nem szabad az eredményben jelentős változást okoznia. Azt, hogy valaki egy adott témában jártas, a tesztnek objektíven kell kimutatnia, az eredmény nem múlhat szubjektivitáson.

Az érvényesség alatt több dolgot értünk. Ezek bizonyítása elméletekkel, statisztikai analízissel történhet. Fontos megnézni, mennyire fedik le a teszt feladatai a felmérni kívánt tudást. A teszt mennyire mér független elméleti konstrukciókat, a várakozásokkal megegyező eredményt adott-e. Vizsgálni kell, hogy a kérdések a tanuló számára annak látszottak-e, aminek szánták. Tudatni kell, milyen probléma megoldását várják el, milyen szempontok szerint fog történni az értékelés; vajon volt-e félreérthető kérdés; a teszt mennyire tudja megítélni a vizsgázónak a jövőben, hasonló szituációban adott válaszát; a meghatározott érték mennyire esetleges, konkrét időhöz kötött.

4.3 Hagyományos értékelési módszerek vagy tesztrendszerek

A hagyományos értékelési módszerek leggyakrabban felhozott hiányosságai a következők:

- Szubjektivitás.
- Az osztály gyakran csak tétlen szemlélője a szóbeli feleltetésnek.
- Általában negatív érzést, stresszt eredményez a tanulókban.
- Nehéz a visszacsatolás; az értékelésből nyert információ nehézkesen használható fel az oktatási rendszer hatékony szabályozására (a szóbeli feleltetés csak néhány tanulóról ad információt; az írásbeli dolgozatokat csak nagyobb anyagrészek átvétele után, tehát meglehetősen nagy időközökkel alkalmazzák).
- A számítógép legfeljebb az adatok feldolgozásánál lehet jelen.

Ami a szubjektivitást illeti, kimutatták például, hogy egy felmérésben résztvevő tanárok 70%-a ugyanazokat a dolgozatokat három hét különbséggel különbözőképpen értékelte. A szóbeli feleletek megítélésénél a helyzet még rosszabb [46].

A tesztrendszerek kiküszöbölik a hagyományos értékelés számos hiányosságát, de sajnos ezeknek is megvannak a maguk gyenge pontjai.

A teszt néhány előnye:

- Objektivitás.
- Könnyen és egyértelműen értékelhetőek.
- Kevésbé időigényes, ezért gyakrabban alkalmazható.
- A tipikus hibák hamar felismerhetőek.
- Lehetővé teszik a számítógépi teszt.
- Átfogó képet nyújtanak a tanítás-tanulás folyamatáról.

A teszt néhány hátránya:

- Lehetővé teszik a véletlen találgatást.
- A zárt kérdések nem adnak teret az eredetiségnek, kreativitásnak, stb..
- A nyílt felépítésű kérdések nehezen értékelhetőek számítógéppel.

4.4 Az értékelés és a számítógép

A digitális számítógép feltalálását követően hamar elkezdődött az új csodagép nyújtotta lehetőségek kamatoztatása az oktatásban. Bitzer és Alpert az elsők között voltak, akik felismerték, mennyire hatékony didaktikai eszköznek bizonyulhat a számítógép. Az ő hozzájárulásával 1960-ban létrehozták az első számítógépes oktató rendszert (Programmed Logic for Automatic Teaching Operation).

A hardver és szoftver eszközök fejlődése azt eredményezte, hogy a számítógépeknek a tanítás-tanulás-értékelés folyamatában való felhasználásának a hatékonysága is folyamatosan nőtt. A személyi számítógépek, a lokális hálózatok, valamint az internet megjelenése tovább gyorsította e folyamatot.

Amint az a fenti fejtegetésből is kiderült, az értékelés az oktatási folyamat egy fontos, és egyben összetett része. A számítógépet már a kezdetektől fogva bevonták az oktatási folyamat e szakaszába is. Leginkább a tesztalapú értékelésben használhatók ki a számítógépek előnyei. Ahogy az oktatóprogramok újabb és újabb generációi jelentek meg, úgy fejlődtek a tanulók értékelését segítő szoftver eszközök is.

4.4.1 Miért jobb a modern gépi teszt?

A számítógép a teszt folyamat szinte mindegyik fázisát képes hatékonyabbá tenni. A tesztszoftver tartalmazhat egy külön modult a tesztelemek szerkesztéséhez, valamint az elembank létrehozásához és tárolásához. A számítógépi tesztkészítés magába rejti annak lehetőségét, hogy a tesztkérdésekbe olyan multimédiás elemeket építsünk be, mint például mozgó képek és hangok. Továbbá a számítógépes szerkesztés és tárolás megkönnyíti a régi tesztek felújítását.

Ha az értékelést hálózati szoftver segítségével végezzük, akkor a tesztkérdéseket a tanulók nem lapon, hanem a képernyőjén megjelenítve kapják meg. A szoftver hálózati jellege miatt képes hatékonyan összehangolni a folyamatot, amit a tanár folyamatosan követni tud a képernyőjén. A multimédiás elemek és az interaktivitás lekötik a tanulók figyelmét, élvezetessé teszik az értékelést. Egy másik fontos szempont, hogy az internet lehetővé teszi a tanulók egyidejű összehangolt vizsgáztatását egy terembe összegyűjtésük nélkül.

A számítógépi program képes elkészíteni a teljes folyamat részletekbe menő elektronikus jegyzőkönyvét. E jegyzőkönyvek tárolásával a nyers adatoknak egy olyan adatbázisa hozható létre, amely alapján később mélyreható kutatások végezhetők. Például, nyomon követhető az egyes tanulók fejlődése, vagy a diákok korára, nemére, stb. vonatkozó összehasonlító elemzések végezhetők. Az értékelés befejeztével azonnal rendelkezésre állnak a közvetlen eredmények. Azonnal láthatjuk továbbá táblázatok és grafikonok formájában az értékelés folyamatáról készült különböző statisztikai kimutatásokat.

Fontos szempont a mai rohanó világban, hogy a számítógép jelenléte az értékelés különböző szakaszaiban jelentősen felgyorsítja a folyamatot, és ezért gyakrabban célszerű alkalmazni. Minél gyakrabban nyerünk tartalmas visszacsatolást a tanítás-tanulás folyamatáról, annál finomabban szabályozható az oktatási rendszer.

A számítógépi teszt lehetővé teszi az adaptív tesztelést. Ennek feltétele, hogy rendelkezzünk egy adatbázissal, melyben minden elemnek ismeretes a nehézségi szintje. Mivel a mérendő tanulók különböző képességűek, nehéz olyan tesztet készíteni, ami mindenkit egyformán mér. Az a legelőnyösebb, ha személyre szabottan kapják a megfelelő kérdéseket, így kevesebb elem megválaszolása pontosabb eredményt ad, mivel az elembankból mindig a kellő nehézségű kérdésre kérünk választ.

Mivel az interaktivitás a számítógépi teszt egyik legfontosabb jellemzője, vizsgáljuk meg közelebbről. A legelterjedtebb rendszerekben a vizsga folyamata a következő:

1. A tanulót minden fontos információval ellátják: mennyi ideje van, hány kérdésre kell válaszolnia, és ezek milyen nehézségűek. Továbbá, hogy miként adhatja meg a helyes választ, és hogyan korrigálhat. A gép folyamatosan jelzi az igénybe vehető erőforrásokat, illetve a még rendelkezésre álló időt.
2. Megjelennek sorban a kérdések. A gép méri a reakcióidőt. A tanuló megadja válaszát és biztossági szintjét, ezeket módosíthatja, majd továbbléphet. Mivel a képernyőn látja a kérdést, az lehet részletesen kidolgozott, nem kell egy mondatba sűríteni a feladatot; tartalmazhat multimédiás elemeket is:színes képet, mozgó alakzatot, sőt hangot is.
3. A gép azonnal visszaírja a helyes választ és az addig elért eredményt. Ezt az azonnali visszacsatolást kedveli sok vizsgázó, bár nem feltétlenül hat ösztönzőleg.
4. Ha még van kérdés, az következik, különben a gép (kérésre) kinyomtatja az eredményt. Dokumentálja a vizsgát egy adatbázisban, illetve hagyományos módon papíron is, ha szükséges [47].

4.5 Értékelés „Comenius-módra”

Ahogy erre már többször is utaltunk, Comenius [6] az ideális tanítás-tanulás folyamatot gyakorlatiasnak és egyben élvezetesnek képzelte el. A kérdés, amelyet elemeztünk az, hogy ki lehetne-e terjeszteni e comeniusi alapelvet az értékelés folyamatára is? Játékossá és élvezetessé lehetne-e tenni a felelést a diákok számára?

Törekvésünk semmiképpen sem számít úttörőnek, hiszen számos kutató és pedagógus ért már el szép eredményeket e területen. A számítógépek, majd a számítógép-hálózatok megjelenése új lehetőségeket hozott magával. Másik segítségnek éppen a média bizonyulhat, amelyet gyakran – és jogosan - azért korholunk, mert elvonja a fiatalokat a minőségi tanulástól. Számos olyan TV-játék vált híressé, amely a játékon túl arra hivatott, hogy oktasson és ezzel egyidejűleg lemérje a játékosok tudásszintjét. E TV-játékok sikere azt mutatja, hogy nem lehetetlen izgalmassá tenni a feleltetést sem. Persze az osztálytermek és a stúdiók között van egy alapvető különbség: a játékos legfeljebb nem nyer semmit, a diák számára viszont a felelés rossz jeggyel, a vizsga pedig bukással is végződhet. Ezért gyakoribb e TV-játékokkal inspirált didaktikai szoftvereknek a tanult anyagrész átismétlésére, mint vizsgáztatásra való felhasználása.

Ilyen irányú sikeres alkalmazások a Jeopardy nevű népszerű TV-játék osztálytermi implementációi. A Journal of Management Education [49] 2004-ben beszámolva egy változatról, amit a Bloomsburg egyetemen dolgoztak ki, az alkalmazás alábbi előnyeit emeli ki:

- Motiválja a hallgatókat az aktív részvételre, valamint arra, hogy komolyabban vegyék a tanulást.
- Lehetőséget biztosít az osztálytermi csoportmunkához.
- Vizsga előtt felfrissíti a diákokban a tanult fogalmakat.
- Változatossá és szórakoztatóvá teszi az órát.

Grabowski és Price [50] a Pittsburgh-i egyetem kémia karán ugyancsak elkészítették a Jeopardy játék egy osztálytermi változatát. Beszámolójukban beszélnek arról, hogy bár a játékok egyedülálló módon képesek dinamizálni az osztálytermi órákat, miért idegenkedik mégis a használatuktól sok tanár:

- A keret elkészítése, valamint ennek tartalommal való feltöltése igen időigényes lehet.
- A diákok ma már nagyon igényesek, ami a szoftverek minőségét illeti.

Talán nem tévedünk, ha azt állítjuk, hogy nézettségét illetően a „Legyen ön is milliomos" TV-játék jelenleg a legnépszerűbb a didaktikai értékű médiajátékok területén. Nagyon sok pedagógust ihletett meg már e TV-játék és dolgozott ki számítógépi tananyagot ismétlő, illetve feleltető szoftvereket. Az, hogy egy feleltető szoftvernek van egy befutott TV-játék elődje, abban is segít, hogy elvégzi a pedagógus helyett az érdeklődés felkeltés egyáltalán nem könnyű feladatát. Úgy gondoljuk, hogy az általunk készített „Legyél te is eminens" hálózati feleltető szoftver úttörőnek számít e területen, és egyben az említett TV-játék egyik legsikeresebb osztálytermi implementációjának tekinthető.

4.6 „Legyen ön is milliomos" típusú didaktikai eszközök

A „Legyen ön is milliomos" TV-játék szereplői a játékvezető, a vele szemben ülő versenyző, a mögötte elhelyezkedő nézőközönség és a telefonkapcsolatban lévő „segítőkész barátok". A játékost 10 jelölt közül választják ki az alapján, hogy melyikük válaszol helyesen egy bemelegítő kérdésre. Az úgynevezett „forró székbe" jutott versenyzőnek 15, egyre nehezebb kérdésre kell válaszolnia, hogy elnyerje a nagydíjat. A kérdéseket számítógép generálja véletlenszerűen egy kérdésbankból, és mind a versenyző, mind a játékvezető, mind a közönség előtt megjelennek. A játékosnak minden kérdésre négy válasz közül kell kiválasztania a helyeset. A helyes válasz egy bizonyos pénzösszeget ér: a könnyebbek kevesebbet, a nehezebbek többet. A játékos háromszor kérhet segítséget: egyszer kérhet felezést (a számítógép kitöröl két helytelen választ), egyszer megkérdezheti a közönséget (statisztikát kap arról, hogyan választanának a nézők), egyszer 30 másodpercre telefonösszeköttetésbe kerülhet valamelyik ismerősével. Van két küszöb az ötödik és tizedik kérdések után. Az aktuális kérdés végkimenetele háromféle lehet: 1. a játékos, nem lévén biztos válaszában, megáll; 2. a játékos helyes választ ad; 3. a játékos helytelen választ ad. Az első esetben a játékos számára a játék befejeződik, és az addig megszerzett pénzösszeggel távozik. A második esetben a játék a következő kérdéssel folytatódik. Nyilván, a harmadik eset a legrosszabb. A játékosnak csak a legközelebbi küszöbértékig megszerzett pénze marad meg, és természetesen távoznia kell a játékból.

A „Legyen ön is milliomos" TV-játék néhány pedagógiai erőssége a következő:

- Aktív tanulást biztosít, dinamikus és izgalmas.

- Interaktív, ami lehetővé teszi a versenyző, a játékvezető, a nézők és az ismerősök összjátékát. Nem érezhető az, mint gyakran az iskolában, hogy az osztály unottan várja, legyen vége a feleltetésnek, és jöjjön már az új lecke.
- A játékvezető és a játékos közötti beszélgetés egyik célja, hogy betekintést nyerjünk a játékos gondolkodásmódjába. Ez a beszélgetés „tanári segítséget" jelent a „felelőnek", és igen tanulságos lehet a közönségnek.
- Mivel a továbblépés - amelyért egyre nagyobb jutalom jár - egyedüli módja a kérdés helyes megválaszolása, ezért óriási az ösztönzés, hogy a játékos mindent megtegyen annak érdekében. Nem jellemző az a hozzáállás, hogy „nekem elég annyi, hogy átmenjek", miért fárasszam magam.
- Mivel a rossz válasz komoly veszteséggel járhat, nagyon elrettent a találgatástól, ami különben a tesztrendszerek egyik gyenge pontja.
- E, sokszor igen nehéz döntési probléma feszültségét enyhíti a megállás lehetősége, ami kiegyensúlyozza a játékot.
- Mivel a mindennapi életben gyakran kerülhetünk hasonló választások elé, ezért a „Legyen ön is milliomos" játék rendkívül életszerű. Talán ebben is rejlik a varázsa. A kialakult helyzetek kezelése sok mindenre megtaníthatja mind a játékost, mind azokat, akik követik gondolkodásmódját.
- A három segítséggel való bölcs gazdálkodás (mikor melyiket használja), megint csak komoly kihívást jelent, és sokat elárul a játékos gondolkodásmódjáról. A segítségek és a küszöbértékek is hozzájárulnak a játék kiegyensúlyozásához. Nélkülük a játék túl kemény lenne, és alkalmatlan osztálytermi alkalmazásra.
- Bár a versenyző, a barátai, illetve a nézők bizonyos értelemben egy csapatot alkotnak, a pálmát csak a játékos viszi el. Ez lehetőséget ad az önzetlenség és a hála kimutatására.

Elsőként azt a prezentációt mutatunk be, amelyet a GCSE (General Certificate of Secondary Education) középfokú matematikakurzus anyagának az átismétlésére terveztek, és melegen ajánlották az alkalmazását a Ferl Practitioners' Award 2001 konferencián.

Marsh [51], a szoftver készítője, heti három órában tartja az említett egyéves előadást. Mivel nagy anyagot kell átvenni, kénytelen elég nagy iramban haladni, aminek gyakran az előadások pedagógiai aspektusai látják kárát. Miközben emiatt nyugtalankodott, diákjai azzal tetézték aggodalmát, hogy egyre hamarabb elkérezkedtek a „Legyen ön is milliomos" játékot nézni a TV-ben. E körülmények vezették rá Marsh-ot, hogy készítse el a TV-játék osztálytermi változatát.

A prezentáció egy laptopon futott, amelynek a képernyőjét projektorral az egész osztálynak kivetítette. A diákok papírlapon válaszoltak a kérdésekre, és sajnos csak a felezés segítséget sikerült beépítenie az alkalmazásba.

Milyen visszhangra talált a „Legyen ön is milliomos" TV-játéknak már e kezdetleges osztálytermi implementációja is? Marsh arról számolt be, hogy a GCSE összes matematikacsoportja nagyon ösztönzőnek, élvezetesnek találta, ami megújította a tanulási kedvüket. A szerző különböző iskolákban több tudományos előadáson is bemutatta e didaktikai eszközt. A visszajelzések biztatóak voltak, ami abból is látszott, hogy számos iskola készséget mutatott a prezentáció saját igényeinek megfelelő átírására.

Cochran-t [52], aki a Louisiana-i Technical University egyetem Computer Information Systems & Analysis College of Administration and Business karán tanít, az foglalkoztatta, hogyan lehet a diákok figyelmét folyamatosan fenntartani a 110 perces előadásai alatt. Számos kutató foglalkozott ezzel a témakörrel. Hartley és Davies [54] a kutatási eredményeket átvizsgálva arra a következtetésre jutottak, hogy a tanulók figyelme az előadás első 10 percében fokozatosan nő, majd csökkenni kezd. McKeachie [55] a jelenség kivédésére a környezetváltoztatást javasolja. E javaslat hagyományos implementálása a rövid szünetekben áll, csakhogy ennek számos mellékhatása van. Például, jelentős idő veszhet el kihasználatlanul az órából, vagy egyes tanulóknál, főleg a kisiskolásoknál, nehéz lehet a visszakapcsolódás.

Brown és Atkins [53] azt javasolják, hogy aktív tanulási gyakorlat beiktatásával biztosítsuk a környezetváltozást. Ha ez a gyakorlat a kapcsolódik az anyaghoz, sőt ha kiemeli ennek valamely fontos elemét, akkor a „szünet" nem elveszített idő, hanem a tananyag jobb megértését előmozdító momentum lesz [55, 56]. Ezt sikerült 2001-ben Cochran-nak elérni egy „Legyél te is milliomos" prezentáció osztálytermi változatával. Arról számol be, hogy sikerült: 1. folyamatosan megőrizni, fenntartani az előadás lendületét; 2. folyamatosan fenntartani a hallgatók érdeklődését; 3. a rövid szünetek felfrissítették a hallgatók figyelmét. Conchran az alábbi forgatókönyv szerint használja az említett prezentációt.

Körülbelül ötven perc eltelte után véletlenszerűen kiválaszt egy csoportot játékra. Mivel az előadását körülbelül nyolcvanan hallgatják, arra kéri a hallgatókat, hogy négytagú csoportokban játszanak. Ily módon az év végére mindenki egyszer résztvevő lehet. A nézőközönségben maradt diákokat ugyancsak megkéri, hogy tegyék félre jegyzeteiket és kövessék a játékot. Ezzel hangsúlyozza, hogy szünettartásról van szó.

A tanár az osztállyal szemben ül, a képernyő előtt ülő játékosok vele szemben, az osztály pedig háttal (a súgás elkerülése végett). A csapatnak négy kérdésre kell válaszolnia, pontosabban, négy lehetséges válasz közül kell az egyetlen jót kiválasztania. A kérdéseket a nézőknek is kivetítik. A játékosok megbeszélik a kérdéseket, és egyetértésre kell jutniuk, mielőtt válaszolnak.

A csapat 2/3 ajándékponttal indul, és minden jó válasz további 1/3 pontot ér. Így összesen 2 pontot lehet a játék végére összegyűjteni. A játékosok minden kérdés esetén három lehetőség közül választhatnak:

1. Megállnak a játékban, mert nem biztosak a helyes válaszban. Így megmarad az addig szerzet pontszámuk.
2. Válaszolnak. A helyes válasz 1/3 pontot ér, de helytelen válasz esetén elveszítik a nyert pontjaikat (ha már van egy pontjuk, ezt nem veszítik el).
3. Egy-egy alkalommal használhatják a három segítséget: megkérdezik valamelyik társukat, kikérik a nézők véleményét, felezést kérnek.

A tanár azzal is ösztönözheti a tanulókat a figyelésre, és a játék komolyan vételére, hogy a megszerzett pontjaikat bizonyos százalékban beleszámítja az év végi vizsgajegyükbe.

Conchran a hallgatók igen pozitív visszajelzéséről számol be. Íme néhány ezek közül:

- „A játék valóban segített nekem. Ha jól válaszoltam, tudtam, hogy megértettem az adott anyagrészt. A rossz válaszok tudatosították bennem, mivel kell többet foglalkoznom."
- „Élveztem, mert lehetőséget biztosított arra, hogy megtudjam, mely anyagrész igényel további elmélyülést. Továbbá, a játék jó lehetőség volt az előadás alatti pihenésre is."
- „Mindegyik órát érdeklődéssel vártam."
- „A játék valóban gondolkodásra ösztönzött, és izgalmassá tette az anyagot."
- „Ha tudtam mindegyik kérdésre a választ, azt az érzést keltette bennem, hogy okos vagyok, és növelte az önbizalmamat."
- „Ugyanúgy tanulságos volt, ha én vettem részt a játékban, és amikor csak megfigyelője voltam."
- „Jó lehetőség volt arra, hogy pontokat szerezzek, illetve átismételjem az anyagot."

Conchran azzal összegzi a tapasztalatairól szóló beszámolót, hogy a fenti (és más ehhez hasonló) visszajelzések meggyőzően bizonyítják, a „Legyen ön is milliomos" TV-játék osztálytermi változata elérte azokat a pedagógiai célokat, amelyeket a fentiekben mi is felvázoltunk.

Collins [57], a Bloomsburg egyetem földrajz karának oktatója, 2004-ben ugyancsak kidolgozta a „Legyen ön is milliomos" TV-játék osztálytermi változatát. Ő kifejezetten az év végi felmérő előtti összefoglaláshoz és ismétléshez készített prezentációt. Megfigyelte, hogy a hagyományos ismétlési módszerek – amikor az anyag egy rövid összefoglalása után a tanulók kérdéseket tesznek fel - nem igazán aktivizálja, főleg a gyengébb diákokat. A diákok rendszerint előre megkaptak egy ismétlési útmutatót, hogy előkészített kérdésekkel érkezzenek az év végi összefoglalásra és ismétlésre. Sajnos, általában csak néhány diák tett fel kérdéseket. Következésképpen Collins célja a TV-játék osztálytermi implementációjával az volt, hogy sarkallja a tanulókat felkészülésre, továbbá, hogy minél több diákot aktívan bevonjon az ismétlésbe az óra kreatívabbá és érdekesebbé tételével. Egy „Legyél te is meteorológus" típusú szoftver alkalmazása azért is kézenfekvő volt Collinsnak, mert az ismétlésen túl a tanulók hozzászokhattak a teszthez, ugyanis a vizsga – akárcsak a TV-játék - éppen feleletválasztós tesztrendszeren alapult.

Az alábbiakban kiemeljük a „Legyél te is meteorológus" szoftverrel történő ismétlés pedagógiai erősségeit:

A kvíz hasonló kérdéseket tartalmazott – de nem ugyanazokat -, mint a vizsgateszt. Ez a tanulóknak azt sugallta, hogy a sikeres vizsgához többre van szükség, mint a játékkérdések ismerete.

A tanulókat bevonta a kérdések összeállításába. Angelo és Cross [60] kutatók kimutatták, hogy mialatt a tanulók fontolgatták, milyen tesztkérdéseket készítsenek, máris elkezdtek készülni a vizsgára. Továbbá, a diákok nagyobb kedvvel vettek részt a játékban, ha tudták, hogy az általuk javasolt kérdések is bekerültek a kérdésbankba.

A teljes osztály bevonása érdekében a húsz-harminctagú osztályt két csoportba osztotta. A játékvezető nyilván a tanár volt, és a játékot kétszer játszották el. Az első játékban az egyik csoport tagjai a játékosok voltak, a másik csoporté pedig a nézők, majd szerepcsere következett. A játszó csoportból mindegyik diák egy-egy kérdésre próbálhatott meg válaszolni.

Mindhárom segítség felhasználható volt: a felezés, a nézők megkérdezése, egy néző megkérdezése. Ha valamelyik játékos elhasználta az egyik segítséget, a többi csapattag számára az már nem volt elérhető. Ez növelte a játék csapatjellegét. A segítségek kibővíthetők voltak például azzal, hogy meg lehetett kérdezni a tanárt, stb.. Ez ellensúlyozta azt, hogy egy csapatban több játékos vett részt.

A tanár azzal is növelhette a játék oktató jellegét, hogy a miért kérdésekre magyarázatokat fűzött mind a helyes, mind a helytelen válaszokhoz.

A tanár azon is elgondolkodhatott, hogy célszerű-e díjazni a győztes csapatot. Collins tapasztalatai szerint az hatott a legösztönzőbben a diákokra, ha eredményeiket valamilyen módon beleszámították a vizsgajegyükbe.

A „Legyél te is meteorológus" didaktikai módszer és eszköz hatékonyságát a szerző kétféleképpen is igazolva látja: javultak a vizsgaeredmények, és jók voltak a diákok visszajelzései. Íme egy példa: „A játék nagyon élvezetes volt és hasznos, mivel tudatosította bennem, mely anyagrészeket ismerem, és melyeken kell még dolgoznom. Tisztábban láttam, mire kell összpontosítani, amikor a tesztre készülök."

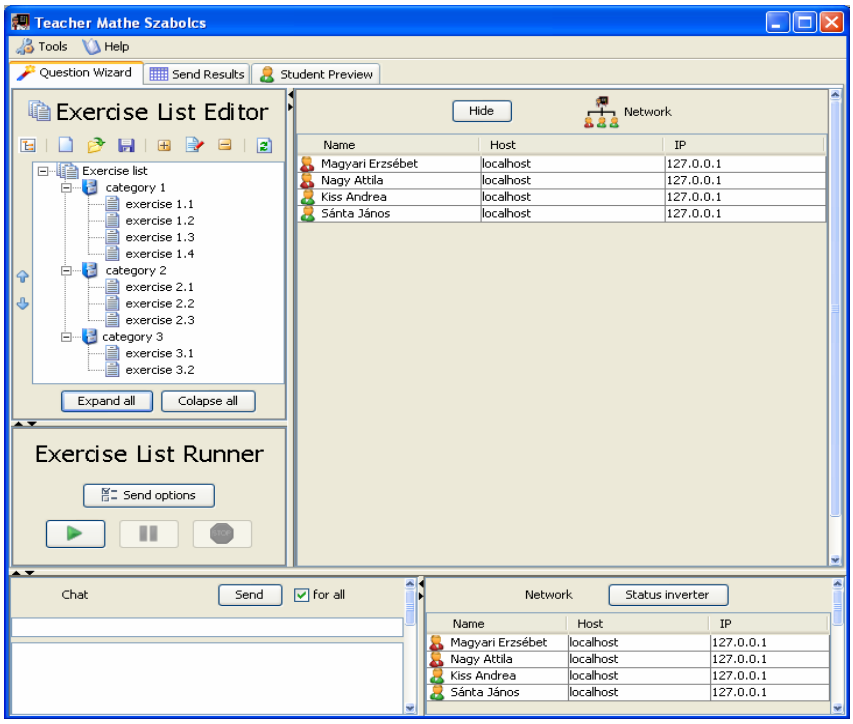
Bár a bemutatott alkalmazások jól szemléltetik, hogy a különböző játékok osztálytermi implementációja miként növelheti a tanítás-tanulás folyamatának hatékonyságát, mégis messze alulmaradnak a számítástechnika, az informatika biztosította lehetőségektől.

4.7 A „Legyél te is eminens" feleltető szoftver

A szoftver Java programozási nyelven írt, felhasználóbarát hálózati alkalmazás. Használata tehát számítógép-hálózat létezését feltételezi. A legelőnyösebb természetesen az, ha minden diák saját számítógéppel rendelkezik. A XXI. században már egyre több iskola tudja ezt biztosítani, legalábbis a felmérések alkalmával. Ma már szaktól függetlenül minden diák elsajátítja a számítógép-kezelés alapfogalmait, és ugyanez igaz szinte minden tanárra is. Az iskolák egyre felszereltebbek számítógép-hálózatok tekintetében, ami miatt lehetővé válik, hogy ne csupán az informatikaóráknak adjanak helyet a számítógépes laborok. A szoftverhez interneten keresztül is kapcsolódhatunk, ami megengedi a vizsgáztatást a diákok összegyűjtése nélkül is. Bár a szoftvert elsősorban feleltetésre terveztük, használható vizsgáztatásra, ismétlésre is.

A szoftver háromféleképpen használható: bejelentkezhetünk feleltető tanárként (jelszóval védett), felelő diákként vagy „néző" diákként. Lentebb láthatjuk a három felhasználói felületet. Bejelentkezéskor minden diák néző, majd ezt követően a tanár úgy tudja kijelölni a felelőket, hogy egyszerűen megváltoztatja a státuszukat. A TV-játéktól eltérően több felelő is lehet.

Tanári felület: *(4.1 ábra)*



4.1 ábra

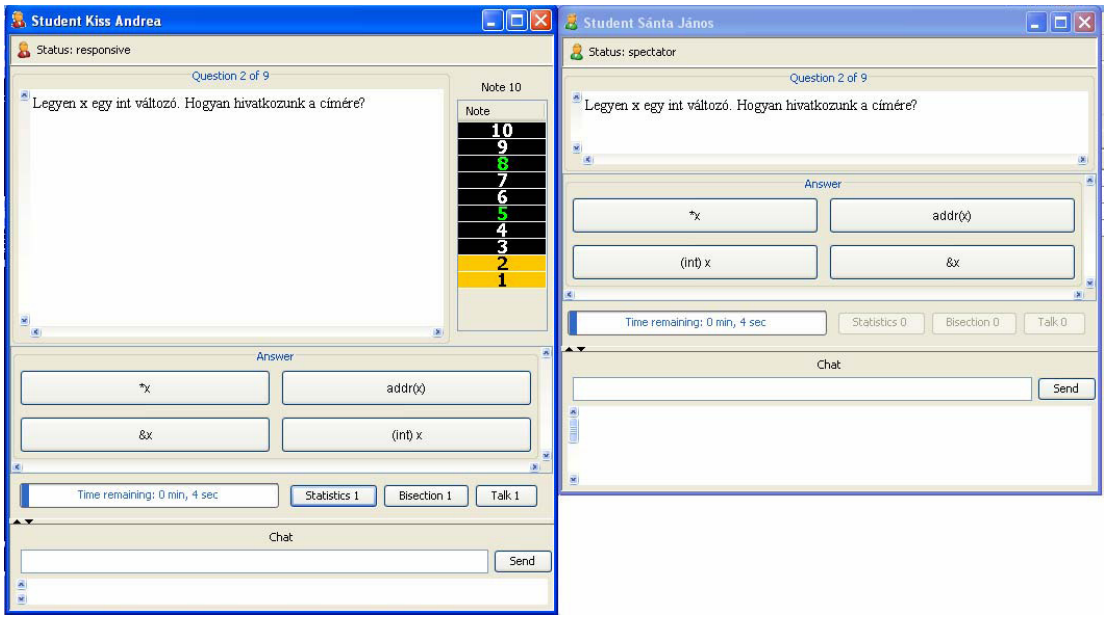
A tanári felület egy háromoldalas párbeszédablak. A *Question Wizard* oldal *Exercise List Editor* ablak a tesztek összeállítására szolgál. Látható, a kérdések nehézségi kategóriákba rendezhetőek és súlyozhatóak. A szerkesztő lehetővé teszi multimédiás elemek beépítését a kérdésekbe. A lehetséges válaszok száma nem kötött, és a helyes válaszok száma is akármennyi lehet. Ahogy az ikonok is jelzik, minden szerkesztési alapműveletet beépítettünk a kényelmes munka érdekében. A szerkesztő modul önálló egységként is működik, hogy a diákok hozzáférjenek. Ez lehetővé teszi, hogy a diákok javaslatai alapján is kerüljenek be tesztkérdések az elembankba.

Az *Exercise List Runner* ablakból indítjuk el, függesztjük fel vagy állítjuk le a feleltetést. A *Send options* gomb megnyit egy ablakot, ahol a feleltetés menetét meghatározó feltételek adhatók meg (például üzemmód, pontozási rendszer, időhatárok, stb., lásd lentebb). Az ablak jobb oldalán a tanár látja a bejelentkezett tanulók listáját. Státuszuk attól függően felelő vagy néző, hogy a megfelelő ikon piros vagy zöld színű. Az ablak alján lévő terület lehetővé teszi, hogy a tanár üzeneteket küldjön/fogadjon a tanulóknak/tól, illetve, hogy chat-eljen valamelyikkel közülük.

A *Send Results* oldal (lásd lentebb) a feleltetés mentét követi, folyamatosan jelezve, hogy az egyes kérdésekre milyen válaszok érkeztek, illetve az egyes tanulók milyen segítséget vettek, illetve éppen vesznek igénybe.

A *Student preview* oldal megmutatja a tanárnak, hogy éppen mit látnak a diákok a saját képernyőiken.

Felelői felület: *(4.2 ábra)*



4.2 ábra

A felelő diák (*status: Responsive*) felülete tartalmazza az aktuális kérdést, a kérdés számát (például *Question 2 of 9*) és a lehetséges válaszokat. A jobb felső sarokban a felelő folyamatosan követheti az elért pontszámát (sárga háttér), látja a küszöbértékeket (zölddel), és rendelkezésére áll a *Stop* gomb, amennyiben meg szeretne állni (ha nem elég biztos a válasza helyességében).

Az ablak alsó része a segítségeket ellenőrzi, illetve jelzi, mennyi ideje van még a felelőnek a kérdés megválaszolásához. A tanuló láthatja, hogy milyen segítséget hányszor vehet még igénybe (*Statistics*: a nézők válaszainak statisztikája, *Bisection*: felezés, *Talk*: Chat-elés valamelyik nézővel).

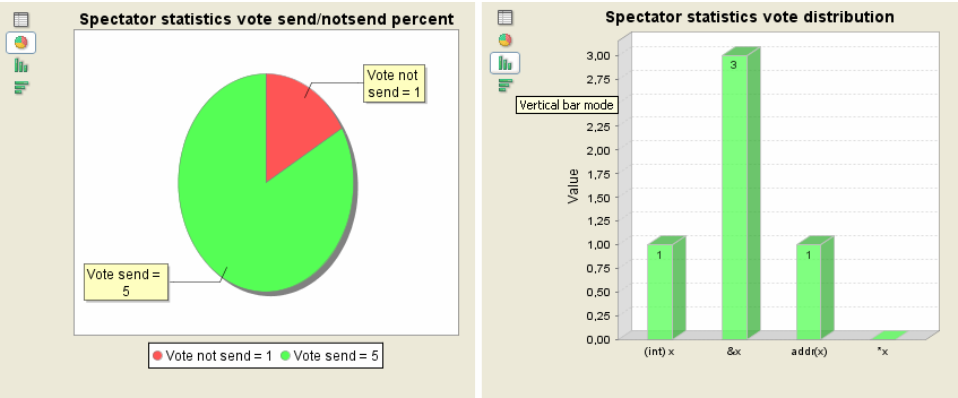
Nézői felület:

A néző diák (status: *Spectator*) felülete nagymértékben hasonlít a felelők felületére, csak nem jelenik meg pontszámkijelző, és inaktívak a segítségek.

A szoftver különböző üzemmódokban működtethető. Implementáltuk a hagyományos tesztelési módot is, amikor a diákok egymás után kapják a kérdéseket, és a szoftver végül összesíti a jó válaszokat. Az alapértelmezett „eminens” üzemmódban a diákoknak kilenc kérdésre kell válaszolniuk, amit ha helyesen tesznek meg, akkor az ajándékba kapott 1 ponttal együtt ez 10-es jegyet jelent nekik (összhangban a Romániában használatos 1 – 10 jegyrendszerrel). Két küszöbértéket építettünk be az 5-ös és 8-as jegyeknél. A kérdéseket három nehézségi kategóriába soroltuk be. A szoftver az első négy kérdést a legkönnyebb kategóriából (5-ös jegyig), a következő hármat a második kategóriából (8-as jegyig), végül az utolsó két kérdés a harmadik kategóriából (10-es jegyig) véletlenszerűen választja. A felelő diákok bármikor megállíthatják a saját kiértékelésük folyamatát, ha

meg vannak elégedve az összegyűjtött pontszámmal, és nem biztosak a következő kérdésre adandó válaszukban. Ilyenkor átkerülnek a nézők közé. Téves válasz esetén jegyük visszaesik a legközelebbi küszöbértékre. Az „eminens” üzemmód e jellegzetessége pedagógiailag azért hasznos, mert „igen veszélyessé” teszi a találgatásokat, ami elismerten a tesztrendszeres feleltetések egyik hátránya. Persze, a diák szempontjából éppen az a hátrányos, hogy kizárjuk a találgatás lehetőségét. Nos e „hátrányát” a diáknak ellensúlyozza az, hogy minden felelőnek rendelkezésére áll a három segítség:

1. Kérhet felezést, amikor is a számítógép véletlenszerűen bejelöl két helytelen választ a négy lehetségesből, és így a diáknak már csak két lehetőség közül kell kiválasztania a helyest.
2. Lekérheti a nézők válaszainak a statisztikáját: hány néző válaszolt, és a válaszok hány százaléka volt A, B, C illetve D (lásd a *4.3 ábrát*).
3. Chat-elhet valamelyik nézővel.



4.3 ábra

Többek között e segítségforrások adják meg az értékelés játékos jellegét.

4.8 Egy feleltetés forgatókönyve

A tanár és a diákok bejelentkezése, illetve a felelők kijelölése után a tanár beállíthatja, hogy az egyes kategóriákba tartozó kérdések megválaszolására mennyi időt engedélyez. (További lehetséges beállítások: hagyományos vagy „eminens” üzemmód; pont- vagy jegyrendszer szerinti feleltetés; legyenek-e küszöbök, és ha igen, hány és hol; a kérdések véletlenszerűen legyenek kiválasztva, vagy előre meghatározott sorrendben; hány segítség álljon rendelkezésre az egyes fajtákból; stb.)*(4.4 ábra)*

A beállítások után a tanár elindítja a feleltetést. Minden diák megkapja a kérdéseket függetlenül attól, hogy felelő vagy néző, és mindegyiküknek kell válaszolnia is. Ily módon az egész osztály foglalkoztatása biztosítva van. A másolás megakadályozása végett a program véletlenszerűen határozza meg, hogy az egyes képernyőkön hol jelenjen meg a helyes válasz. Ha már minden felelő válaszolt, és még nem telt le az adott kérdésre kiszabott idő, a nézőknek 10 másodpercük van, hogy válaszoljanak, amennyiben még nem tették meg (nem várunk a nézők után).

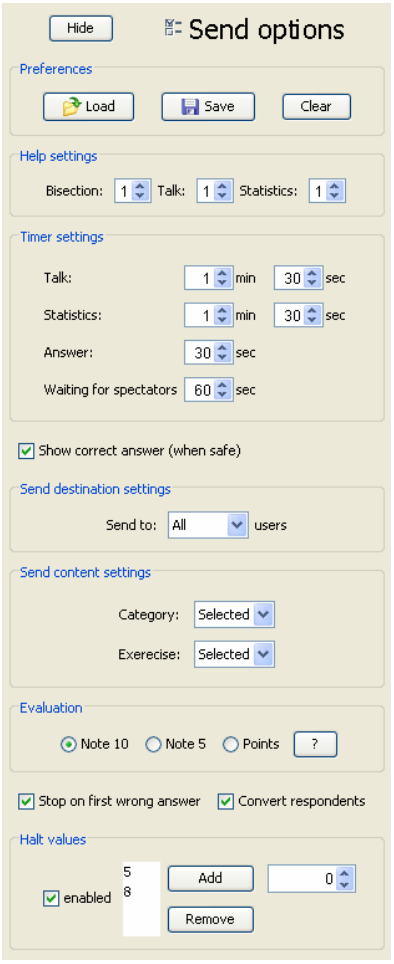
Ha valamelyik felelő a statisztika segítséget választja, a nézők képernyőjén megjelenik

egy üzenet, hogy egy adott időn belül válaszoljanak neki. A nézők e válasza különbözhet attól, amit saját válaszként adnak az adott kérdésre. Ha később ugyanazon kérdés kapcsán egy másik felelő is statisztikát kér, ő is ugyanazt az összeállított statisztikát kapja meg.

Chat-eléskor a felelő az elérhető (nem chat-el éppen egy másik felelővel) nézők listájából kiválaszthatja, hogy kitől szeretne személyes segítséget kérni.

Rossz válasz esetén, vagy ha a felelő megállította a saját feleltetési folyamatát, a szoftver kijelzi a kapott jegyét, és státusza automatikusan nézőre változik, hogy ezt követően se unatkozzon az óra végéig. Úgy döntöttünk, hogy az első küszöb alatti jegyek esetén a téves válasz nem eredményezi az 1-es jegyhez való visszaesését (Romániában 5-ös a legkisebb jegy, amivel már nem lehet megbukni).

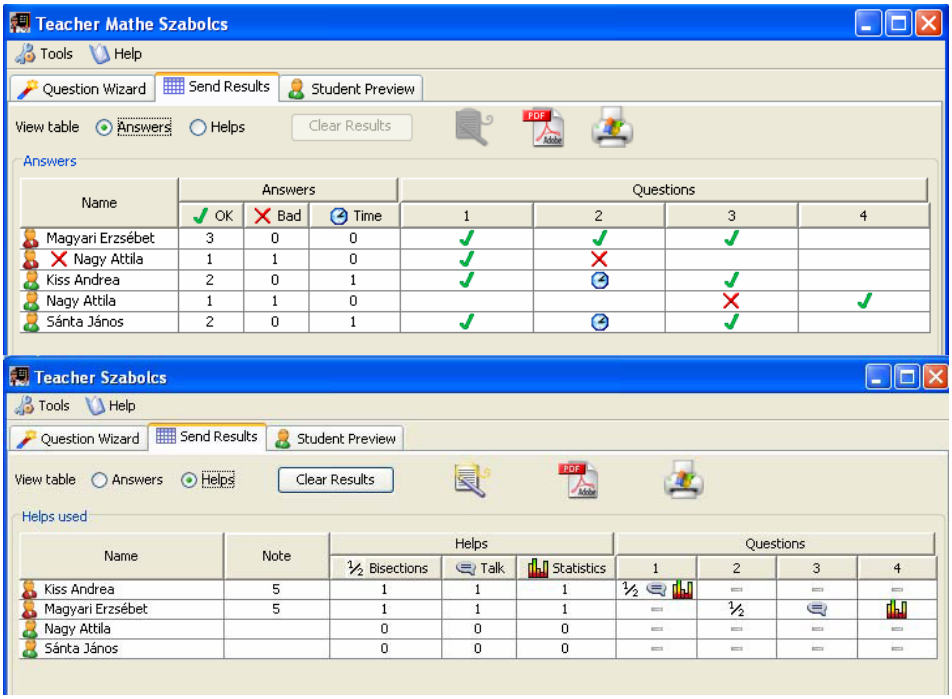
A kiértékelés teljes ideje alatt a feleltetés minden mozzanata ott van a tanár képernyőjén. Ennek egy részletét láthatjuk lentebb a tanári felület *Send Results* oldaláról. Ha az *Answers* gombot kapcsoljuk be, akkor a diákok válaszait láthatjuk: zöld pipa (helyes válasz), piros kereszt (helytelen válasz), óra (nem érkezett válasz az adott időn belül). A diákok neve után folyamatosan frissített összesítő táblázat áll a tanár rendelkezésére. Ha a felelő diák rossz választ ad, vagy túllépi az időt, akkor a neve, illetve az eltévesztett kérdés mellett piros kereszt jelzi a kizárását. A *Helps* gombot választva egy hasonló kimutatást kapunk arról, hogy az egyes felelők melyik kérdésnél, melyik segítséggel éltek. A kimutatások különböző formátumokban lementhetők, illetve kinyomtathatók. *(4.5 ábra)*



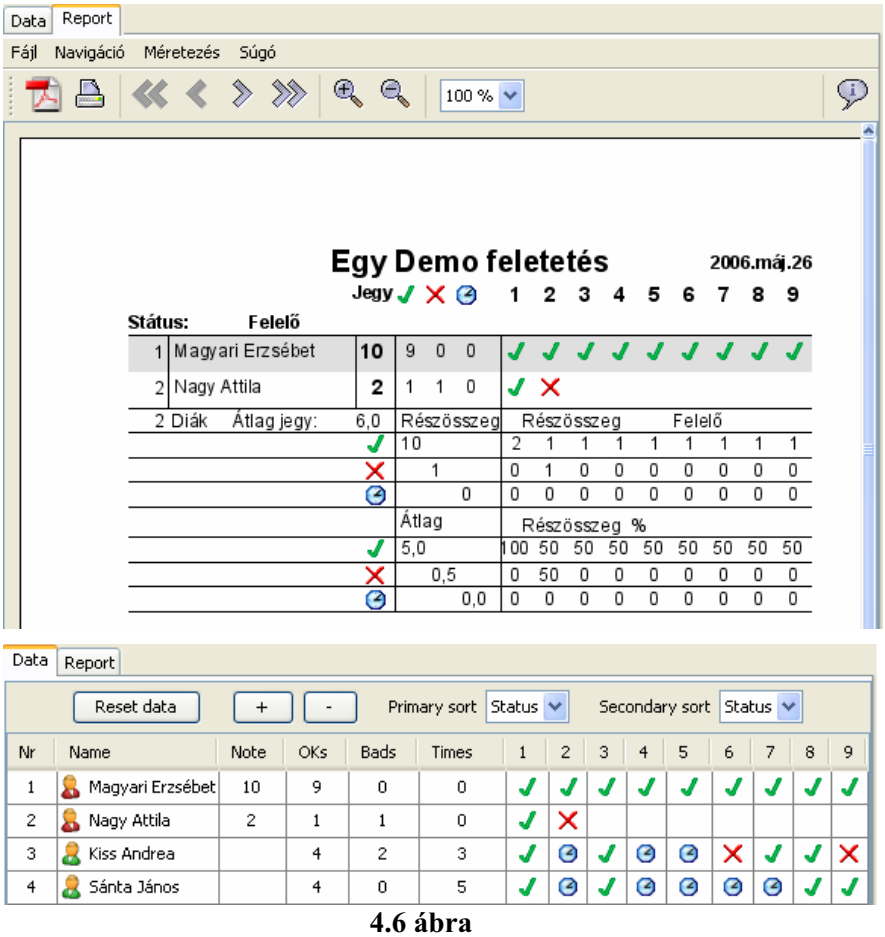
4.4 ábra

Részletes elektronikus jegyzőkönyv készül a teljes folyamatról, ami elmentett dokumentumként, illetve a későbbiekben pedagógiailag feldolgozható adathalmazként szolgál. A tanár regisztrálja saját, illetve az osztály az adatait, valamint a feleltetés általános jellemzőit. Minden diák bejelentkezéskor regisztrálja a személyes adatait. A feleltetés folyamatáról többek között az alábbi adatokat tároljuk el:

- a diák számítógépének az azonosítója (IP);
- a kérdések megjelenésének az időpontjai;
- a válaszok időpontjai, indexei, helyességei;
- a segítségkérések időpontjai.



4.5 ábra



4.6 ábra

A feleltetés befejeztével a szoftver teljes kimutatást készít a tanár részére az összes diák ténykedéséről. Ez lehetővé teszi a tanárnak, hogy megdicsérje (esetleg jutalmazza is) a kitűnő eredményeket elért nézőket, és figyelmeztesse a nagyon gyenge pontszámúakat. Ezen túlmenően a regisztrált adatok alapján számos statisztikai kiértékelés is készül a feleltetés eredményeiről. A *4.6 ábrán* láthatók a tanári felület *Result Report* oldalán megjelenő különböző típusú kimutatások.

A szoftvernek nevelő hatása is van. A tanár felhasználhatja a „Legyél te is eminens" szoftverrel való feleltetés nyújtotta lehetőségeket arra is, hogy önzetlenségre, segítőkészségre, törődésre, stb. nevelje diákjait. Például, a nézők olyan komolyan fogják tanulmányozni a kérdéseket, mintha ők felelnének, hogy a lehető legjobbat tudják nyújtani magukból, ha majd valamelyik felelő statisztikát kér, vagy chat-elni szeretne velük? Vagy csak azután nézik meg komolyan a kérdést, ha megjelent a segítségkérő üzenet a képernyőjükön és „már ketyeg az óra"? Továbbá, miután egy felelő megkapta a jegyét, vajon továbbra is bele fogja élni magát a „játékba", hogy segítsen a többi felelőnek? Vagy nyugodtan fog ülni a babérjain, mint aki már elvégezte a dolgát?

4.9 Vizsgáztatás a „Legyél te is eminens" szoftverrel

A „Legyél te is eminens" szoftver vizsgáztatásra is használható olyan tantárgyaknál, amelyek lehetővé teszik a tesztrendszeres értékelést. A Sapientia Erdélyi Magyar Tudományegyetemen a 2005/2006-os tanév első félévi C programozás vizsga „írásbeli" részét ezzel a szoftverrel bonyolítottuk le, az alábbi vizsgamenet szerint:

- A szoftvert - kétszer egymás után – „eminens" üzemmódban futtattuk le. Először az évfolyam egyik feléhez tartozó diákok voltak a felelők és a többiek a nézők, majd fordítva.
- Ezután a program a hagyományos üzemmódban futott le, amikor nem voltak sem küszöbök, sem segítségek. Minden diák felelő minőségben vett részt a kiértékelésben, és a jegye mondta meg, hány kérdésre adott helyes választ. Ez a módszer csak abban különbözött a papíron végzett teszttől, hogy a felelő nem láthatta előre az összes kérdést, és nem térhetett vissza már megválaszolt kérdéshez. Ezt a hátrányt ellensúlyozta, hogy a kérdéseket nehézségi sorrendben kapták.
- Végül a kapott két jegy közül a jobbat írta be a tanár a vizsgajegyzőkönyvbe.

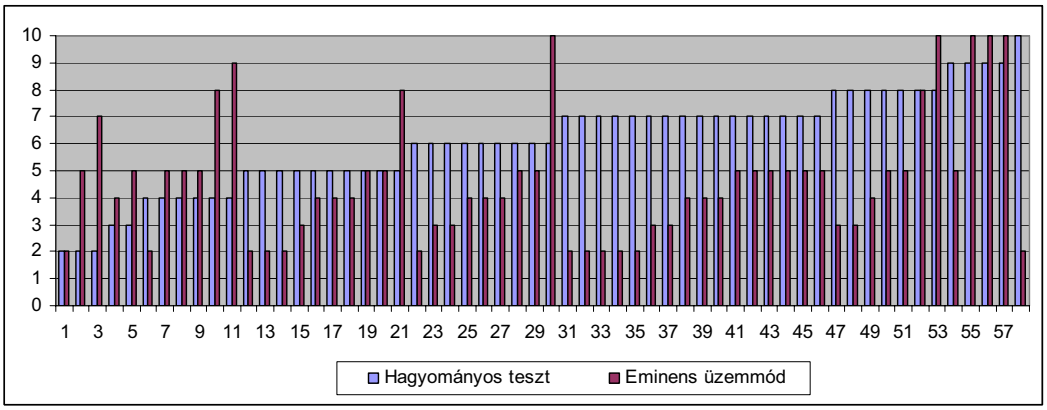
Miért szükséges ez a vizsgamenet? A TV-műsor játékos jellege abból is adódik, hogy a játékosnak nincs mit veszítenie, legrosszabb esetben nem nyer semmit. Ezzel szemben egy vizsga akár bukással is járhat. Ha a vizsgáztatás csak „eminens" üzemmódban történne, akkor a módszer elveszítené egyik legfontosabb erősségét, a játékos jellegét. Azzal, hogy a vizsga tartalmazza a hagyományos tesztet is (ami általánosan elfogadott, és amire mindenképpen sor kerülne), és mivel a jobb jegyet kapja a diák, biztosítani tudjuk, hogy az „eminens" üzemmódból csak nyerhessen a diák.

Az is fontos, hogy elsőként kerüljön sor az „eminens" üzemmódra. Ez a versenyjelleg megőrzéséhez fontos. Milyen tétje lenne az „eminens" üzemmódnak egy olyan tanuló esetében, aki maximálisan teljesített a hagyományos teszten?

A verseny- és játékjelleg fokozható a dobogósok díjazásával. Kaphatnak például egy-egy

üdítőt, hogy enyhítsék szomjukat, illetve a felgyülemlett izgalmat a hagyományos teszt előtt. Az „eminens” üzemmódnak a vizsgarendszerbe történő beépítésével jelentősen növelhető a diákok motiváltsága a vizsgára való felkészülésben.

A „Programozás C nyelven” vizsgán ötvennyolc diák vett részt. A hagyományos teszteredmények átlaga 6,1, az „eminens” üzemmódbelieké pedig 4,63 lett. A vizsgajegyek átlaga: 6,7. Összesen ötvenöt diáknak sikerült átmenő jegyet szerezni (legalább 5-öst). Tizenöt volt azok száma, akik jobb jegyet kaptak „eminens” üzemmódban. Ezek közül nyolcat mentett meg ez az üzemmód a bukástól. *(4.7 ábra)*



4.7 ábra

Bár a többség jobb eredményt ért el hagyományos üzemmódban, az „eminens” üzemmód pozitív izgalmat vitt a vizsgáztatásba. A gyengébb eredmények azzal is magyarázhatók, hogy a hallgatók még tapasztalatlanok voltak ezen üzemmódot illetően, és többségükben nem éltek bölcsen a segítségek nyújtotta lehetőségekkel (anélkül estek ki, hogy valamelyik segítségforrást használták volna). További pedagógia kísérleteket tervezünk középiskolás diákokkal és más szakos egyetemi hallgatókkal is. Előkészületek folynak egy kiterjedt felmérésre azt illetően, hogy milyen eredményekhez vezet a szoftver használata más-más korosztályú, más-más szakirányú diákok, illetve különböző tantárgyak esetében.

4.10 Feleltetés a „Legyél te is eminens” szoftverrel

Az osztálytermi felelés, ahogy a vizsgázás is, alapvetően nem játék. Nem is lenne jó, ha a diákok játéknak tekintenék. Csak arról van szó, hogy egy *játékszerű* kerettel feloldjunk valamennyit a feleléssel együtt járó stresszből, és motiváljuk a tanulókat a rendszeres tanulásra. Mivel az osztálytermi felelésekből származó osztályzatok a tanulók félév végi átlagainak csak egy részét teszik ki, ezért úgy gondoljuk, önmagában az a tudat, hogy a jegy bekerül a naplóba, nem zárja eleve ki annak lehetőségét, hogy az eminens üzemmód révén izgalmat vigyünk a felelésbe.

Mivel az alapértelmezett eminens üzemmódban már az első rossz válasz végzetes, a diákok úgy érezhetik, hogy nem arra kapták a jegyet, amit tudnak, hanem arra, amit nem tudtak. Ez a tudat nemhogy oldaná, inkább fokozná a felelés stresszes jellegét. A szoftver e „szigorúságát” hivatottak ellensúlyozni a segítségforrások, illetve a küszöbértékek. Csakhogy a rossz válasz lehet figyelmetlenség miatt is (ilyenkor a felelőnek meg se

fordul a fejében segítséget kérni), vagy a segítségforrások nem teljesen megbízható volta miatt (a legjobb néző is tévedhet, a nézők összvéleménye lehet akár megtévesztő is). Többek között ez a magyarázata annak, amiért a „Legyen ön is milliomos" TV-játék legtöbb osztálytermi alkalmazása nem általános feleltetésre, illetve vizsgáztatásra, hanem csak pozitív értékelésre (az eredmények csak pozitívan számítanak be a végső osztályzatba) készült. A 4.9 alfejezetben bemutatott kombinált vizsgamenet a példa erre. Az alábbiakban egy olyan lehetőséget mutatunk be a hagyományos és az eminenes tesztek ötvözésére, amely jól alkalmazható – élesben is – feleltetésre.

Ebben az üzemmódban az első négy kérdést hagyományos tesztrendszerben (a hibás válaszok nem járnak kieséssel; nincsenek segítségek) kapják meg a diákok. A felelők nem kapnak minden válasz után visszajelzést azok helyességéről, csupán a negyedik kérdés megválaszolását követően jelzi ki a szoftver, hány pontot sikerült addig összegyűjteniük (legalább 1-et, legfeljebb 5-öt). Az a tudat, hogy az első kérdések esetén a helytelen válasz nem végzetes, segíthet feloldani a kezdeti stresszt, amelynek bénító hatásai lehetnek. Elkerüljük azt is, hogy a tanulókban az az érzés alakuljon ki, a szoftver olyan tanárt modellez, amelyik az első téves válasz után a diákba folytja a szót: „Sajnálom fiam, de ha *ezt* nem tudod, nincs értelme, hogy folytassuk."

A hagyományos teszt révén szerzett pontszám az első küszöb (a második küszöb automatikusan három ponttal magasabb). Ettől a pillanattól (az első küszöbtől) a szoftver átvált eminens tesztre, ami azt jelenti, hogy a következő öt kérdésnél rendelkezésre állnak a segítségek, meg lehet állni a felelésben, illetve a rossz válasz kieséssel és a küszöbre való visszaeséssel jár. Az eminens teszt lehetővé teszi, hogy az osztály „törvényes keretek között" segíthessen a felelőknek: saját tudásukból merítve, a szoftver biztosította módokon (statisztika kérésekkel, illetve chat-eléssel). Ez a lehetőség plusz motivációforrás lehet ahhoz, hogy a tanulók minden órára készüljenek, nemcsak akkor, amikor felelés várható.

Egy másik fontos szempont, hogy a „kiesett" felelők nézőkként továbbra is részt vesznek a tesztben: „post eminens" teszt. A feleltetés befejeztével a tanár eldöntheti, hogy figyelembe veszi-e valamilyen módon a felelők azon jó válaszait, amelyeket - kiesésüket vagy megállásukat követően - nézőként adtak. Például, ha az egyik felelő a kiesését követő a nehezebb kérdésekre jól válaszol, a tanár úgy ítélheti meg a helyzetet, hogy hűebben tükrözné a diák tudás szintjét az a jegy, amelybe - valamilyen százalékban - e pontokat is beleszámítaná. Beállítható, hogy a szoftver milyen „algoritmus" szerint javasoljon osztályzatot. A post eminens teszt lehetőséget ad olyan jelenség áthidalására, amikor a tanuló nem tudja, amit tőle kérdeznek, de tudja a választ a másik felelőnek feltett kérdésekre. Amíg számos tanár erre úgy reagál (ha egyáltalán lenne is módja erről megbizonyosodni), hogy „Sajnálom fiam, de ez már csak eső után köpenyeg", addig a „Legyél te is eminens" szoftver lehetővé teszi e tudás figyelembe vételét is.

Összegzésként elmondhatjuk, hogy olyan összetett feleltetési módszerről van szó, amely a következő három tesztrendszer erősségeit igyekszik ötvözni:

- A hagyományos teszt – a tévedések nem „végzetesek".
- Eminens teszt – a diákoknak segítségforrások állnak rendelkezésükre.
- „Post eminens" teszt – a felelők nézőkként folytatják a tesztet. Ez lehetővé teszi, hogy a tanár figyelembe vegye a jegy megítélésekor azt a tudást (vagy ennek egy hányadát) is, ami a felelés alatt rejtve maradt.

Ahogy ezt egyik diák meg is fogalmazta, ez a módszer egy olyan tanárt szimulál – a gépi teszt adta korlátok között -, amelyik bár szigorú, arra törekszik, hogy segítsen a tanulóknak a lehető legtöbbet kihozni magukból.

4.11 Következtetések

A „Legyél te is eminens" didaktikai szoftver kidolgozásával az volt a célunk, hogy olyan módszert és eszközt biztosítsunk az értékelés folyamatához, amely: képes izgalmat vinni a feleltetésbe; aktivizálja az egész osztályt; a mérésen túl tanít és nevel is; rendelkezik a teszt előnyeivel a hagyományos módszerekkel szemben; maximálisan kihasználja a számítástechnikai nyújtotta jelenlegi lehetőségeket; olyan adatbázist hagy maga után, ami mélyreható pedagógiai kutatásokat tesz lehetővé. Arra is odafigyeltünk, hogy az eredeti ötletek mellett beépítsük a szoftverbe a már létező alkalmazások erősségeit. Az alábbi összefoglalása a módszer előnyeinek arról tanúskodik, hogy elértük célunkat. Reméljük, hogy a további kísérletek és kutatások megerősítik majd ezt a következtetést.

- Versenyhelyzetet alakít ki, és izgalmat nyújt, amelyek további motivációforrások, ami a tesztre való készülést illeti.
- A 4.10 alfejezetben leírt feleltetési üzemmód azt a tanárt „modellezi", amelyik tekintettel van a kezdeti plusz stresszre, lehetővé teszi a segítséget, amikor a felelő elakad, kész figyelembe venni abból a tudásból is, ami a feleltetés befejezte után kerül felszínre.
- Egyszerre több diák is „felelhet". Ha az első négy kérdésre egy-egy percet szánunk, a következő háromra kettőt-kettőt, és az utolsó kettőre egyenként két és fél percet, akkor 15 perc alatt kész is a feleltetés.
- A program kétszeri lefuttatásával (először az osztály egyik fele a felelő, majd a másik) osztályfelmérők „íratása" is lehetséges. A szoftver figyeli, hogy a tanulók ne találkozzanak ugyanazzal a kérdéssel többször is.
- A szoftver lehetővé teszi az eminens teszt erősségeinek kamatoztatását vizsgáztatás alkalmával. (4.9 alfejezet)
- Az egész osztályt foglalkoztatjuk, hiszen mindegyik tanuló képernyőjén megjelennek a kérdések, és válaszolniuk kell. Továbbá, a segítségek révén az egész osztály aktívan részt vesz a felelők megmérettetésében. Az eminens üzemmód „törvényes" keretet biztosít ahhoz, ahogy az osztály a felelők segítségére siethet. Mindezek plusz motivációforrást jelentenek a minden órára való tanuláshoz. Az eminens üzemmód „törvényes" keretet biztosít ahhoz, ahogy az osztály a felelők segítségére siethet. Mindezek plusz motivációforrást jelentenek a minden órára való tanuláshoz.
- A szerver folyamatosan mutatja a tanárnak az összes tanuló tevékenységét. Ez kizárja annak lehetőségét, hogy észrevétlen maradjon a felelést megúszó tanulók felkészületlensége, vagy a felelésről lemaradók felkészültsége. Ezek után a tanár úgy dönthet, hogy valamilyen módon jutalmazza a kimagaslóan teljesítő „nézőket"is, illetve figyelmeztetheti a nagyon gyenge eredményt elérőket (használhat piros/fekete, azaz plusz/mínusz pontrendszert).
- A „felelés" befejeztével azonnal számos statisztikai kimutatás áll a tanár rendelkezésére nyomtatható formában.
- A tanulók bevonhatók a tesztkérdések összeállításába. Ez legalább három haszonnal jár:

    - A kérdések elkészítésekor a tanulók – lehet ugyan, hogy tudtukon kívül, de - máris tanulnak.
    - Az, hogy az ő kérdéseik is jelen vannak az elembankban, érdekeltebbé teszi őket az órán való aktív részvételben. Sajnos egyre általánosabb jelenség a teljes érdektelenség az iskolai tevékenységekkel, főleg a tanulással szemben.
    - Megkönnyíti a tanár munkáját.
- Elektronikus jegyzőkönyv készül a feleltetések összes mozzanatáról. Ezen adatbázis lehetővé teszi a tanítás-tanulás folyamatának egy későbbi átfogó tanulmányozását.
- Tekintettel e tesztforma kis időigényére, gyakori alkalmazásával lehetővé válik a tanítás-tanulás folyamatának finomabb szabályozása.
- Mivel ötvözi a feleltetés és dolgozatírás bizonyos jellegzetességeit, a tanár úgy nyerhet rendszeresen információt a teljes osztály tudásszintjéről, hogy közben a diákoknak nincs az az érzésük, hogy „minden órán" tesztet írnak (nem mindenki felelő).
- Az értékelés mellett nemcsak a tanítást és tanulást támogatja, hanem a nevelést is.
    - Olyan tulajdonságokat hoz felszínre a tanulókban, mint az önzetlenség, segítőkészség, együttérzés. A néződiákok úgy érvényesítik e tulajdonságokat, hogy teljesen beleélik magukat a kérdésekbe, mintha ők felelnének, hogy a lehető legjobbat tudják nyújtani, ha a segítségüket kérik.
    - Előtérbe kerül az önbizalom, illetve az osztálytársakban, valamint az osztályban, mint egészben való bizalom kérdése.
    - Hozzájárulhat az osztályszellem megerősödéséhez.
- A rossz válasz szigorú büntetése, a megállás lehetősége, a küszöbértékek, valamint a segítségforrások együttes jelenléte gyakran olyan választások elé állítja a felelőket, amelyek próbára teszik, de egyben csiszolják is a bölcsességüket.
- A „Legyél te is eminens" szoftver felhasználóbarát, rugalmas és biztonságos. Például, használható mind hagyományos tesztre, mind „eminens" üzemmódban, és biztonsági másolat készül az eredményekről.
- Moduláris felépítéséből adódóan könnyen fejleszthető.

Természetesen, mint általában mindennek, úgy a „Legyél te is eminens" is vannak hátrányai is, mind a hagyományos feleltetési módszerekkel, mind a megszokott teszttechnikákkal szembe. A számítógépi teszt alapvető hiányosságai, amelyek a szoftver hagyományos üzemmódban való működtetésekor is fennállnak:

- A tanulók nem szabhatják meg, hogy milyen sorrendben foglalkoznak a kérdésekkel.
- A tanulók nem térhetnek vissza a megválaszolt kérdésekre, és nem láthatják a következőket.
- A tanulók nem határozhatják meg, hogy mennyi időt szánnak az egyes kérdésekre.

Úgy gondoljuk, hogy e „hátrányokat" „eminens" üzemmódban ellensúlyozzák a segítségforrások, valamint az a tudat, hogy a tanár megítélése szerinti nehézségi sorrendben követik egymást a kérdések.

Az alábbi irányelvek segíthetik a szoftver feleltető/vizsgáztató eszközként való alkalmazását:

- Adjunk egyértelmű kérdéseket.
- A kérdések ne azt mérjék, hogy a diák mennyire figyelmes, hogy észrevesz-e valamit, hanem azt, mennyire érti az anyagot.
- Csak azután használjuk élesben feleltetésre, miután a tanulók hozzászoktak az új módszerhez, és megtanulták ügyesen menedzselni a segítségforrásokat.

További üzemmódok beépítését tervezzük a jövőben:

- Hagyományos teszt esetén ugyanazt a kérdéssort kétszer olvashassák el a diákok. Az első lefutás során nem kapnak minden kérdésre azonnali visszajelzést a válaszaik helyességét illetően. Ezen felül, választani lehet majd az alábbi opciók között:
    - Az első lefutás után a felelő megtudja, hány kérdést rontott el, és a második lefutásban megpróbálja azonosítani és kijavítani ezeket.
    - A tanuló úgy megy át másodszor a kérdéseken, hogy nem tudja, hányat tévesztett el először. Beállítható lesz az is, hogy a második lefutáskor a válaszok azonnal értékelődjenek ki vagy sem. Bár a diákok általában szeretnék az azonnali visszacsatolást, rossz válasz esetén ez nem feltétlenül hat pozitívan rájuk.
- Minden tanuló saját ütemben haladhasson végig a kérdéseken (hagyományos teszt). Folyamatosan tájékoztatást kapnak a maradék idejükről, illetve a többi vizsgázó részeredményeiről (ki hányadik kérdésnél tart, és hány jó választ sikerült adnia). Az utóbbi információ egyeseket sokkolhat, míg másoknak növelheti az eredményességét. A program figyelmeztethet, ha túl sok időt vesztegetett el a tanuló valamelyik kérdés megválaszolásával. A végén a diák részletes elemzést kap a tesztkitöltés alatti munkájáról.
- A tanulóknak meg kell adniuk, mennyire biztosak válaszaikban (százalékban kifejezve).
- A rossz választ nem kieséssel, hanem negatív ponttal bünteti.
- Az eminens üzemmódnak tervezzük egy olyan változatát is, amikor a tévesen válaszoló diák „kegyelmet” kérhet. Ahogy a kérés következtében a diák válasza megjelenik a tanár képernyőjén, a tanár eldöntheti, hogy figyelmen kívül hagyja a kérést, vagy egy mentőkérdést ad. Mivel ez szubjektivitást vihet a feleltetésbe, a tanárnak körültekintően kell majd élnie e lehetőséggel.
- További terv, valamilyen szintű tanári segítség elérhetővé tétele. A TV-játékban a játékvezető-játékos párbeszéd egyik célja a figyelmetlenségből adódható tévedések elkerülése. A szóbeli feleltetések egyik jellemzője, hogy a tanár segíthet a tanulónak abban, hogy valódi tudásszintjének megfelelően teljesítsen.

Összegzésként kijelenthetjük, hogy a „Legyél te is eminens” didaktikai módszer és eszköz értelmes használata jelentősen hozzájárulhat az oktatás hatékonyságának növeléséhez. A szoftver különböző üzemmódokban, más-más diákanyagon való alkalmazása egy olyan adathalmazt hoz létre, amely nyersanyagként szolgálhat további kutatásokhoz.

5 Saját eredmények összegzése

A bemutatott mindhárom didaktikai módszer, ahogy a hozzájuk kapcsolódó szoftver eszközök is, saját eredménynek számítanak.

Az „Algoritmustervezés felülnézetből” módszer azzal nyújt többet, mint az eddig ismert egyéb módszerek, ahogy a programozási technikák egységes tanulmányozását lehetővé teszi. Az öt alapvető algoritmustervezési stratégiával megoldható feladatok mögött meghúzódó fastruktúrák jelenléte a technikák bemutatásában, önmagában is egy új megközelítésnek számít. A javasolt tanmenet a módszer alkalmazásával arra törekszik, hogy a megtárgyalt technikák egy egységes kép különböző részeiként rögzüljenek a tanulók fejében. Az a sajátos mód, ahogy a felülnézetet lépésről lépésre kialakítjuk (2.4.3.1 alfejezet, ábra), lehetővé teszi, hogy az átfogó képen túl, a diákok tisztán lássák az egyes technikák közti árnyalatbeli, de alapvető különbségeket, illetve hasonlóságokat. Továbbá, ahogy egy puzzle elemei egymáshoz illeszkednek, úgy – a felülnézet órákon történő összehasonlító elemzések révén - az egyes technikák is egymáshoz kapcsoltan tárolódnak el a diákok memóriájában. Következtetésként elmondható, hogy a felülnézet módszer alkalmazásával nyert stabil mentális reprezentáció, illetve rálátás és tisztánlátás, ami a programozási technikák világát illeti, egyedülálló módon járul hozzá az „egész osztály” problémamegoldó készségének növeléséhez.

Az érzékszerveknek az ismertetett módon való bevonása az elemi algoritmusok tanításába-tanulásába teljesen eredeti ötleten alapszik. Az első lépés annak felismerése volt, hogy nagymértékben növelheti a diákok feladatmegoldó készségét, ha a tervezés kezdeti szakaszában felvázolják a keresett algoritmus ciklusvázát. Ez azonban viszonylag magas absztraháló képességet feltételez. Mivel a legtöbb kezdő programozóban ez még meglehetősen hiányos, felmerült a kérdés, hogy nem lehetne-e fokozni a tanulók ciklusváz felismerő képességét az érzékszerveknek a hangsúlyosabb bevonásával a tanítási-tanulási folyamat e szakaszába. Egy kezdetleges próbálkozásként - a tanári katedrán, illetve az osztálytermi padokon - kézzel doboltuk el különböző algoritmusok ciklusváz ritmusát. Innen viszont már csak egy lépésre volt egy olyan szoftver kigondolása, amely képes kihangosítani az algoritmusok ciklusvázát, illetve lehetővé teszi különböző ciklusvázak „bedobolását” a billentyűzetről. Az egész testtel való tanulás lehetőségének e megteremtése affektív síkon is többet nyújt a hagyományos megközelítésekhez képest.

A „Legyél te is eminens” értékelési módszer és szoftver esetében az számít saját eredménynek, ahogy a TV-játék osztálytermi alkalmazását megvalósítjuk. A didaktikai eszköz hálózati szoftverként való megvalósítása lehetővé tette a TV-játékban szereplő segítségforrások hiteles osztálytermi implementálását. Az is pluszt jelent egyéb alkalmazásokhoz képest, ahogy az eszköz a tananyag átismétlésen túl feleltetést és vizsgáztatást is lehetővé tesz. Valódi eredménynek számít, ahogy e módszer képes verseny- és játékjelleget adni a különben stresszes feleltetéseknek és vizsgáztatásoknak, illetve, ahogyan az egész osztályt aktívan bevonja a felelők megmérettetésébe. Mindkét tényező révén hatékony motivációforrások kiaknázása történik meg. Annak érdekében, hogy megőrizzük a módszer keltette pozitív izgalmat a vizsgáztatással együtt járó stresszel szemben, az „eminens” üzemmódot követi egy hagyományos teszt is. További eredménynek tekinthető az is, ahogy a módszer az osztályszellemet erősíti, illetve önzetlenségre nevel.

Összefoglaló

„Tanítani szinte nem is jelent mást, mint megmutatni, miben különböznek egymástól a dolgok a különböző céljukat, megjelenési formájukat, és eredetüket illetően. ... Ezért aki jól megkülönbözteti egymástól a dolgokat, az jól is tanít.” [6] Az **Algoritmustervezés felülnézetből** fejezetben bemutatott didaktikai módszer elsősorban erre az alapelvre épül. Egy olyan tanítási, illetve tanulási módszerről van szó, amely segít a tanulóknak, úgymond felülnézetből látni öt alapvető algoritmustervezési stratégiát (visszalépéses keresés, oszd meg és uralkodj, mohó, dinamikus programozás, elágazás korláttal). A módszer célja az, hogy e programozási technikák bemutatásán túl, olyan nézőpontba juttassuk a tanulót, amelyből feltárulnak előtte a technikák közötti elvi, alapvető, sőt árnyalatbeli különbségek, illetve hasonlóságok. A módszer másik erőssége, hogy látva a teljes képet, a tanulók képesek lesznek felismerni az egyes technikák egymáshoz való viszonyát, és így a „nehezebb” stratégiák is elérhetőbbé vállnak számukra.

Felülnézet alatt ebben a dolgozatban azt értjük:

1. „Egymás mellett” látjuk a vizsgálat tárgyát képező entitásokat.
2. Csak az látszik, ami a vizsgálat szempontjából lényeges.
3. Nyilvánvalóak a hasonlóságok és a különbségek, szembetűnők a kapcsolatok.

A felülnézet kialakításához szükséges egy úgynevezett „absztrakt platform”, amelyen az elemzett entitások úgy helyezhetők egymás mellé, hogy szembetűnővé váljanak a vizsgálat szempontjából lényeges tulajdonságok és kapcsolatok.

Az összes tárgyalásra kerülő módszert elsősorban olyan feladatok esetében használjuk, amelyek valamilyen értelemben fastruktúrával rendelkeznek. A technikák szempontjából ez azt jelenti, hogy mindegyik úgy tekinti a feladatot, mint egy fát. Ez a fastruktúra az az absztrakt platform, amelyen a technikák egymás mellé helyezhetők, és amely a felülnézet kialakításához szükséges.

A felülnézet módszer alkalmazásához egy tanmenetet javasolunk:

0. A rekurzió átismétlése, a fastruktúrák és azok bejárásainak bemutatása.
1. Egy feladaton keresztül, amely mindegyik technikával megoldható, általános és átfogó képet nyújtunk a technikákról (1. hét).
2. Bemutatjuk a visszalépéses keresés technikát (2.-3. hét).
3. Bemutatjuk az oszd meg és uralkodj technikát (4. hét).
4. Oszd meg és uralkodj vagy visszalépéses keresés (5. hét).
5. Bemutatjuk a mohó algoritmust (6. hét).
6. Visszalépéses keresés és mohó algoritmus (7. hét).
7. Bemutatjuk a dinamikus programozás módszerét (8.-10. hét).
8. Oszd meg és uralkodj vagy dinamikus programozás (11. hét).
9. Mohó módszer vagy dinamikus programozás (11. hét).
10. Beágyazzuk az elágazás korláttal technikát az előbbi módszerek alkotta képbe (12.-13. hét).
11. Határesetek a programozási technikák világában (14. hét).

A dinamikus programozás az egyik leghatékonyabb programozási technika, csakhogy a legelmélyültebb gondolkodást feltételezi. A nehézség többek között a dinamikus programozási feladatok sokszínűségében áll. A diáknak nagyon mélyen át kell látnia a dinamikus programozás alapelveit, hogy a legkülönbözőbb helyzetekben alkalmazni tudja. A felülnézet módszer egy másik erőssége, hogy kiterjeszthető a dinamikus programozási feladatok területére, lehetővé téve ezek egyfajta osztályozását [61].

Ugyanazt az absztrakt platformot használjuk, a feladatok mögött lévő döntési fát, és a feladatokat e döntési fa különböző típusai szerint osztályozzuk. Minden egyes feladatosztályhoz a dinamikus programozásnak mint algoritmustervezési stratégiának egy sajátos változata rendelhető. Az adott feladatnak a megfelelő osztályba sorolása után már sokkal elérhetőbb a technika valamelyik (az illető feladatosztályt megoldó) sajátos változatának alkalmazása. A felülnézet módszer kivetítése a dinamikus programozás területére azzal az előnnyel is jár, hogy árnyaltabbá teszi a dinamikus programozás és a mohó algoritmus, illetve az oszd meg és uralkodj technika közti különbségeket, hasonlóságokat és kapcsolatokat.

Bevezettük az *első* (minden döntéssel a feladat egy kisebb méretű hasonló feladattá redukálódik, amelyet az aktuális csomópont valamelyik részfája ábrázol), illetve *második* (minden döntéssel a feladat két vagy több kisebb méretű hasonló feladatra esik szét, amelyeket az aktuális csomópont megfelelő részfái ábrázolnak) típusú döntési fák elnevezést, valamint az *összevont döntési fa* (egy súlyozott irányított gráf, amelyet úgy kapunk, hogy egymásra csúsztatjuk a döntési fa identikus állapotokat képviselő csomópontjait) fogalmát.

A dinamikus programozással megoldható feladatok a szerint osztályozhatók, hogy a mögöttük meghúzódó döntési fa első vagy második típusú, illetve, hogy az első esetben az összevont döntési fa körmentes-e vagy sem, és amennyiben tartalmaz kört, van-e negatív éle vagy nincs.

Az 1. típusú döntési fájú feladatok esetében három dinamikus programozásos stratégiát különítettünk el:

1. *Topológikus sorrend alapú dinamikus programozás*: Az összevont döntési fa körmentes.
2. *Dijkstra algoritmus alapú dinamikus programozás*: Az összevont döntési fa tartalmaz ugyan kört, de nincs negatív éle.
3. *Belmann-Ford algoritmus alapú dinamikus programozás*: Az összevont döntési fának van negatív éle is, de nincs a gyökérből elérhető negatív összsúlyú köre.

A 2. típusú döntési fájú feladatokhoz tartozó stratégia azonosítására az *„optimális megosztás – optimális uralom”* elnevezést javasoljuk [59].

A látás, hallás, valamint a kinesztetikus érzékelés bevonása elemi algoritmusok tanításába című fejezetben felsorakoztatott kutatási eredményekből legalább öt érvelési vonal emelhető ki, amelyek mind ugyanahhoz a következtetéshez vezetnek: a tanítás-tanulás folyamata annál hatékonyabb, minél több érzékszervet vonunk be.

- Több érzék, több információt jelent.
- Az, hogy mindenkinek más a tanulási stílusa összecseng azzal a felismeréssel, hogy az egyes tanulóknak más-más lehet a domináns érzékszerve. Minél több érzéküket veszik igénybe a diákok, annál valószínűbb, hogy hatékonynak fogják találni az illető módszert [28, 31].
- Howard Gardner munkássága a nyolc „intelligenciáról", és más kutatások arra is rámutattak, hogyan növeli a tanulás hatékonyságát, ha nemcsak a domináns érzékszervünkre támaszkodunk. Más szóval: az érzékszervek a kombinált alkalmazásukkal erősíthetik egymást [29].
- A kettőskódolás elmélete rámutat annak előnyére, ha a terhelés megoszlik a különböző érzékszervek között [39].
- Minél több érzékszervet vonunk be a tanuláskor, annál több úton érhető el az adott információ a visszaemlékezéskor.

Bármely algoritmus esetében beszélhetünk annak ciklusvázáról (csontvázáról), amely alatt az algoritmus ciklusutasításainak a szerkezetét értjük. A ciklusok magját képező utasításokat pedig úgy tekinthetjük, mint az algoritmus „húsos részeit" [64]. Az alábbi kétlépéses módszert ajánljuk elemi algoritmusok tanításához, illetve tanulásához:

3. A feladat elemzési szakaszában a megoldási algoritmus ciklusvázának a meghatározása.
4. A ciklusváznak a megfelelő utasításokkal való feltöltése.

Kidolgoztunk egy szoftvert, amely javítja a diákok ciklusváz-felismerő készségét. A szoftver célja, lehetővé tenni a látás, hallás, valamint a kinesztetikus érzékelés bevonását e folyamatba. Az alkalmazásnak négy fő modulja van. A cod_creator, cod_beautifier, cod_buherator és a run_code modulok.

A cod_creator modul különböző ciklusvázú programok automatikus létrehozását teszi lehetővé, és két üzemmódban használható:

- A ciklusváz paramétereinek megadásával: Megadhatjuk, hogy hány első szintű ciklust szeretnénk, és hogy ezekbe hány második, illetve harmadik szintű ciklus legyen beágyazva. Az is beállítható, hogy az egyes ciklusok hányszor legyenek végrehajtva.
- A ciklusváz „bedobolásával": „Begépeljük" a program ciklusvázát, mintha eldobolnánk a ritmusát, mintáját. A módszer alkalmazásának ennél a pontjánál kerül sor a kinesztetikus érzékelés bevonására.

A code_beautifier modullal bármely C/C++ program forráskódja megszépíthető, azaz a szoftver oly módon rendezi át a kódot, hogy jól nyomon követhető legyen a ciklusváza. Különösen a látás bevonása szempontjából fontos ez a művelet.

A code_buherator modul – átírva a forráskódot - az algoritmus minden ciklusmagjába hangokat megszólaltató, illetve a programfutást késleltető rutinokat épít be. Olyan, mintha kihangosítanánk az algoritmus ciklusvázát. A ciklusmagok minden egyes lefutásakor megszólal egy zongorahang. A külső ciklusok mélyebb, a belsők magasabb hangokon „szólalnak meg". A különböző szintű ciklusok különböző mértékű késleltetésével elértük, hogy a külső ciklusok lefutását jelző hangjegysorok hangjegyei kisebb, a belsőkéi pedig nagyobb frekvenciával kövessék egymást.

A run_code modul párbeszéd ablakában megjelenik a vizsgálat alatt álló algoritmus megszépített C/C++ kódja. A Run gomb lenyomására a háttérben elindul a cod_buherator modul által átdolgozott (kihangosított és lelassított) program futtatása. Miközben a tanuló hallja – futó program révén - az algoritmus ciklusvázát szemléltető hangjegysort, a run_code ablakban szemével követheti, hol tart a végrehajtás.

A szoftver alkalmazásához egy tanmenetet is javasolunk:

- A tanár különböző ciklusvázakat „hallgattat meg" a tanulókkal, miközben a diákok szemükkel követik az algoritmus végrehajtását.
- A tanulókat megkérjük, hogy próbálják meg hallás után felismerni a „lejátszott" ciklusvázakat.
- „Kihangosított" alapvető elemi algoritmusok végrehajtásának nyomon követése füllel és szemmel.
- A diákok különböző ciklusvázakat „gépelnek be". Miközben a tanulók ujjaik ritmikus, zongorázásra emlékeztető mozdulataival újabb és újabb ciklusvázakat gépelnek be, végül már bennük cseng az algoritmusok ritmusa. A módszer e szakaszát elméleti órákon is alkalmazhatjuk, még ha nem is áll rendelkezésre számítógép és a szoftver. Először a tanár, majd a diákok, a kezeiket és lábaikat használva, szó szerint eldobolhatják a katedrán, illetve a padjaikon a ciklusvázak ritmusát.
- Tipikus hibákat tartalmazó algoritmusok ciklusváz-hibáinak hallás utáni felismerése.

A szoftver használata lehetővé teszi, hogy a diákok úgymond le tudják fogni az algoritmusok pulzusát. Tényleg egész testtel való tanulást biztosít. A tanulók láthatják, hallhatják és – a ciklusvázak bedobolása révén - érezhetik is az algoritmusok lüktetését. Mindez egyértelműen hatékonyabb tanulást jelent. A több érzékszerv nemcsak több információt, jobb megértést és jobb memorizálást jelent, hanem hozzájárul a más-más domináns érzékszervű diákok „esélyegyenlőségéhez" a tanítás-tanulás folyamatában.

Az ideális tanítás-tanulás folyamatot gyakorlatiasnak és egyben élvezetesnek kell lennie. [6] A kérdés, amelyet a **„Legyél te is eminens" - Értékelési módszer és szoftver** fejezetben elemeztünk az, hogy ki lehetne-e terjeszteni e comeniusi alapelvet az értékelés fázisára is?

A „Legyél te is eminens" szoftver felhasználóbarát hálózati alkalmazás, amihez interneten keresztül is kapcsolódhatunk. A szoftver három felhasználói minőségben használható: feleltető tanárként, felelő diákként vagy „néző diákként". A TV-játéktól eltérően több felelő is lehet.

A szoftver különböző üzemmódokban működtethető. Implementáltuk a hagyományos tesztet is, amikor a diákok egymásután kapják a kérdéseket, és a szoftver végül összesíti a jó válaszokat. Az alapértelmezett „eminens" üzemmódban a diákoknak kilenc kérdést kell megválaszolniuk. Ha minden kérdésre helyesen válaszolnak, akkor a hivatalból kapott 1 ponttal együtt ez 10-es osztályzatot jelent nekik (összhangban a Romániában használatos 1 – 10 jegyrendszerrel). Két küszöbértéket építettünk be: az 5-ös és 8-as osztályzatokat. A kérdéseket három nehézségi kategóriába soroltuk. A szoftver az első négy kérdést véletlenszerűen az első (legkönnyebb) kategóriából választja (5-ös

osztályzatig), a következő hármat a második kategóriából (8-as osztályzatig), végül az utolsó két kérdést a harmadik kategóriából (10-es osztályzatig). A felelő diákok bármikor megállíthatják a saját kiértékelésük folyamatát, ha elégedettek az összegyűjtött pontszámmal, és nem biztosak a következő kérdésre adandó válaszukban. Ilyenkor átkerülnek a nézők közé. Téves válasz esetén osztályzatuk visszaesik a legközelebbi küszöbértékre. Minden felelőnek rendelkezésére áll három segítség: kérhet felezést, kérheti a nézők válaszainak a statisztikáját, illetve chat-elhet valamelyik nézővel.

A „Legyél te is eminens" szoftver jól használható vizsgáztatásra olyan tantárgyak esetén, amelyek lehetővé teszik a tesztrendszeres értékelést. Az alábbi vizsgamenetet javasoljuk:

- A szoftvert kétszer egymás után eminens üzemmódban futtatjuk le. Először az osztály egyik feléhez tartozó diákok a felelők és a többiek a nézők, majd fordítva.
- Ezután a program hagyományos üzemmódban fut le.
- Végül a két osztályzat közül a jobbat írjuk be a vizsgajegyzőkönyvbe.

Azzal, hogy a vizsga a hagyományos teszttel fejeződik be (ami már általánosan elfogadott, és amire mindenképpen sor kerülne) és a jobb osztályzatot írjuk be, biztosítani tudjuk, hogy az eminens üzemmódból (mint a TV-játékból) csak nyerhessen a diák.

Azért, hogy osztálytermi feleltetésre is használható legyen a szoftver, kidolgoztunk egy ötvözött üzemmódot is. Ilyenkor az első négy kérdést hagyományos tesztrendszerben kapják a diákok. A negyedik kérdés megválaszolását követően a szoftver kijelzi az elért pontszámot, ami első küszöbnek számit az ebben a pillanatban induló eminens teszthez (a második küszöb automatikusan három ponttal magasabb). A „kiesett" felelők nézőkként továbbra is részt vesznek a tesztben: „post eminens" teszt.

A „Legyél te is eminens" didaktikai szoftver megírásával az volt a célunk, hogy olyan módszert és eszközt biztosítsunk az értékelés folyamatához, amely: képes izgalmat vinni a feleltetésbe; aktivizálja az egész osztály; a mérésen túl tanít és nevel; rendelkezik a teszt előnyeivel a hagyományos módszerekkel szemben; maximálisan kihasználja a számítástechnikai nyújtotta jelenlegi lehetőségeket; olyan adatbázist eredményez, ami mélyreható pedagógiai kutatásokat tesz lehetővé. Arra is odafigyeltünk, hogy az eredeti ötletek mellett beépítsük a szoftverbe a már létező alkalmazások erősségeit. Az alábbi összefoglalása a módszer előnyeinek azt igazolja, hogy elértük célunkat. (Reméljük, hogy a további kísérletek és kutatások megerősítik majd ezt a következtetést.)

- Versenyhelyzetet alakít ki és izgalmas, amik további motivációforrások a tesztre való készülésnél.
- A feleltetési üzemmód azt a tanárt „modellezi", amelyik tekintettel van a kezdeti fokozott stresszre, segítséget ad, amikor a felelő elakad, és kész figyelembe venni azt a tudást is, ami a feleltetés során nem kerül felszínre.
- Egyszerre több diák is „felelhet". Ha az első négy kérdésre egy-egy percet szánunk, a következő háromra kettőt-kettőt, és az utolsó kettőre egyenként két és fél percet, akkor 15 perc alatt be is fejeződik a feleltetés.
- A szoftver lehetővé teszi az eminens teszt erősségeinek kamatoztatását vizsgáztatás alkalmával.
- Az egész osztályt foglalkoztatjuk, hiszen mindegyik tanuló képernyőjén megjelennek a kérdések, és válaszolniuk kell. Továbbá, a segítség-opciók révén, az egész osztály aktívan részt vesz a felelők megméretésében. Az eminens

üzemmód „törvényes” keretet biztosít ahhoz, ahogy az osztály a felelők segítségére siethet. Mindezek plusz motivációforrást jelentenek a rendszeres tanuláshoz.

- A szerver folyamatosan megjeleníti a tanulók tevékenységét. Ez kizárja annak lehetőségét, hogy észrevétlen maradjon a felelést megúszó tanulók felkészületlensége, vagy a felelésről lemaradók felkészültsége.
- A „felelés” befejeztével azonnal számos statisztikai kimutatás áll a tanár rendelkezésére nyomtatható formában.
- Mivel a kérdésszerkesztő modul önállóan is használható, a tanulók bevonhatók a tesztkérdések összeállításába, ami számos előnnyel jár.
- Elektronikus jegyzőkönyv készül feleltetések minden mozzanatáról. Ezen adatbázis lehetővé teszi a tanítás-tanulás folyamatának egy későbbi, igen átfogó tanulmányozását.
- Tekintettel e tesztforma kis időigényére, gyakori alkalmazásával lehetővé válik a tanítási-tanulási folyamat finomabb szabályozása.
- Az értékelés mellett nemcsak a tanítást és tanulást teszi lehetővé, hanem a nevelést is.
    - Olyan tulajdonságokat alakít ki a tanulókban, mint az önzetlenség, segítőkészség, együttérzés. A néző diákok úgy mutatják ki e tulajdonságokat, hogy teljesen beleélik magukat a kérdésekbe, mintha ők felelnének, hogy a lehető legjobbat tudják nyújtani, ha a segítségüket kérik.
    - Előtérbe kerül az önbizalom, illetve az osztálytársakban, valamint az osztályban, mint egészben való bizalom kérdése.
    - Hozzájárulhat az osztályszellem megerősödéséhez.
- A rossz válasz szigorú büntetése, a megállás lehetősége, a küszöbértékek, valamint a segítségforrások együttes jelenléte gyakran olyan választások elé állítja a felelőket, amelyek próbára teszik, de egyben csiszolják is a bölcsességüket.

Summary

„To teach means scarcely anything more than to show how things differ from one another in their different purposes, forms, and origins. . . . Therefore, he who differentiates well teaches well." [6] In chapter **"Upperview Algorithm Design"** we are going to present a teaching – learning method and suggest a syllabus that help the high school students look at the algorithm design strategies from a so called „upperview": greedy, backtracking, divide and conquer, dynamic programming, branch and bound. The goal of the suggested syllabus is, beyond the presentation of the techniques, to offer the students a view that reveals them the basic and even the slight differences and similarities between the strategies. By this means it becomes possible to integrate all the four techniques into a frame which forms a whole. If the students recognize the position of certain techniques related to the others, then the so called „more difficult" strategies become available for them.

We use the notion of „upperview" in the following sense:

1. We can see the entities being analyzed „next to each other".
2. Only those elements can be seen which are essential for the analysis.
3. The similarities and differences are obvious, the connections are striking.

In order to carry out an „upperview", a so called „abstract platform" might be necessary, where the entities being analyzed can be laid down next to each other in such a manner that the features and connections essential for the analysis become obvious.

We apply all the techniques we are going to present especially in the case of problems which in some consideration have the structure of a tree. From the point of view of the techniques this means that each one considers the problem as a tree. Well, this common tree structure is that common plane or abstract platform – necessary for the „upperview" – where the techniques can be laid next to each other.

As an application of the „upperview" method we suggest the following syllabus:

0. Revision of the recursion, presentation of the tree structures and their traversal.
1. Through a demo problem solvable with each of the strategies, we offer a general and comprehensive image of the techniques.
2. We present the backtracking technique.
3. We present the divide and conquer technique.
4. Divide and conquer and backtracking from „upperview".
5. We present the greedy technique.
6. Backtracking and greedy from „upperview".
7. We present the technique of the dynamic programming.
8. Divide and conquer and dynamic programming from „upperview".
9. Greedy and dynamic programming from „upperview".
10. We embed the "branch and bound" technique in the picture given by the above-mentioned methods.
11. Boundary cases in the world of programming techniques.

In this chapter we also present a study and classification of the dynamic programming strategies. By presenting the characteristics of certain dynamic programming strategies on the decision tree hidden behind the optimizing problems, we offer such a clear tool for their study and classification, which can help in the comprehension of the essence of this programming technique. [61]

We introduced the notions: *I. type decision tree* (By each decision the problem is reduced to a similar problem of smaller size, represented by one of the subtrees of the current node); *II. type decision tree* (By each decision the problem is divided into two or more smaller sized subproblems, represented by the corresponding subtrees of the current node); *contracted decision tree* (We obtain this data structure, if we overlap the nodes representing the identical states of the tree.)

In case of I. type decision tree we have distinguished three different dynamic programming strategies , for each of which there are well-known graph-algorithms:

1. The contracted decision tree is cycle free. In this case, there is an algorithm based on the topological order of the nodes.
2. The contracted decision tree contains circles, but it has no negative edges. For this case, there is the Dijkstra algorithm, which determines the minimal weight paths with a priority queue in the increasing order of the optimal values.
3. The contracted decision tree has negative edges, but has no negative total weight circle reachable from the root. The Belmann-Ford algorithm solves this problem.

In case of II. type decision tree we have called the specific dynamic programming strategy *„Optimal division – Optimal conquest”*. [59]

From the presented research results in chapter **“Involvement of the sight, hearing and the phenomenon of kinaesthesia in the teaching-learning process of elementary algorithms”** we can emphasize at least five reasoning lines, which all lead to the same conclusion: the process of teaching-learning is more efficient the more senses are involved in it:

- more senses - more information
- different students - different dominant senses [28, 31]
- different students - different "intelligences" [29]
- multiple senses - more pathways of locating the stored information
- multiple senses - distributed loading [39]
- combined senses – more efficient learning process

Any algorithm has a “loop-skeleton”, its structure of loops. The instructions that represent the nucleus of the loops can be seen as the algorithm’s “meat-parts”. [64] In what follows, we recommend two-steps method for teaching and learning elementary algorithms:

- Analyzing the task we establish the loop-skeleton of the algorithm that solves the problem.
- We fill up the loop-skeleton with the adequate instructions.

We have developed a new didactical software tool to help the in establishing the loop-skeleton. The software has four main modules: cod_creator, cod_beautifier, cod_buherator, run_code.

The cod_creator module makes it possible to create program-skeletons with different loop structure in an automatic way. It runs in two modes:

- Giving the loop-skeleton's parameters: We introduce how many loops we want on the first, second and third level, and which is subordinate to which. Additionally we have to give the number of iterations of each loop.
- Drumming the loop-skeleton in: This mode makes it possible to type the program's loop-skeleton in, as if we have drummed its rhythm-pattern in. The phenomenon of kinaesthesia is involved especially at this stage of the learning process.

By the code_beautifier module module every C/C++ source file can automatically be reorganized ("beautified") in such a way that its cycle skeleton should easily be noticed. This operation gets an important role because of the sight involvement.

The code_buherator module – rewriting the source file- plants sound and delay procedures in nucleuses of each loop instruction. This will have such an effect upon the algorithm as if its loop-skeleton would have been spoken up. A piano sound will be heard every time when a loop's nucleus is traversed. The outer loops will be audible in lower, and the inner ones in higher sounds. Additionally, applying different length delays in case of the loops, which are situated on different levels, the result will be that the outer loops will have smaller frequency sound-sequences than the inner ones.

In the dialog box of the run_code module module the "beautified C/C++ cod" of the analyzed algorithm appears. Pushing the Run button starts the slow motion running of the program. While the students "are listening to the algorithm's cycle-skeleton" represented by its sound-sequence, they can keep their eyes on the program's running (as we can see the instruction which is being executed is highlighted).

We suggest the following syllabus as application of the presented software.

- The students "are listening to" several loop-skeletons while they keep their eyes on the slow motion running of the program generated by the code_creator module.
- The students are invited to recognize some unknown loop-skeletons, only by hearing their "piano accompaniment".
- The students are following, with their eyes and ears, the running of some well-known elementary algorithms in the run_cod module's dialog box.
- The students are invited to "drum in" divers loop-skeletons.
- The students are listening to the "piano accompaniment" of algorithms that contain typical errors. Then they are invited to compare these sound-sequences with the "piano accompaniment" of the correct algorithms.

The use of this software in the teaching-learning process of elementary algorithms has made it possible for students to feel the algorithm's pulse. It has really been learning with the "whole body". The students could see, hear and – drumming the loop-skeleton's in – feel the pulsation of algorithms. The more senses involved doesn't only mean more information, better perception and more efficient memorising, but ensures the same chance for students with different dominant senses.

The ideal teaching-learning process should be practical and in the same time enjoyable. [6] The question we have analyzed in chapter **"Who wants to be eminent – Assessment method and software"** is whether we can extend this basic principle of Comenius to the assessment phase as well.

The "Who wants to be eminent" software is a user-friendly application, which you can access through the Internet as well. The software can be used as three different types of users: we can log in as teacher, responsive student or "spectator student". Contrary to the TV-game, there can be several responsive students.

The software can be exploited in different operational modes. The classical testing mode has been implemented, when the students get the questions one after the other one and the software sums up the good answers at the end. In the implicit "eminent" mode the students have to answer 9 questions and if they answer all of them correctly, with the 1 point received officially means a 10 mark (in accordance with the marking system 1-10 used in Romania). There are two margin values as the 5 and the 8. The questions are grouped in three categories of difficulty. The software chooses the first four questions in a random order from the first (easiest category), the next three from the second category (up to mark 8), finally the last two questions are chosen from the third category (up to mark 10). The responsive students can any time stop the process of their assessment, if they are contented with the result they have obtained and are not sure in the answers to the following questions. In this case they are transferred among the spectators. Should they give a wrong answer, their mark falls back to the closest marginal value. Every responsive student has the possibility of the three helps: 50-50, can ask for the statistics of the spectators' answers, can chat with one of the spectators.

The "Who wants to be eminent" software can be well used for examination, for subjects which allow the assessment based on testing. We suggest the following course of the examination:

- Firstly we run the software – twice consecutively – in the "eminent" operational mode. At first the students from the firs half of the class are responsive students and the others spectators, and secondly we invert the roles.
- Then the program runs in classical testing operational mode
- Eventually we write the higher mark out of the two the student received, into the examination register.

By ending the examination with a traditional testing (which is generally accepted and which would have been performed anyhow) and by writing the higher mark, we ensure the fact that the student can only win from the "eminent" operational mode.

In order to be able to use this software in the classroom as well, we have developed an alloyed operational mode, too. In this case the responsive students get the first four questions in a traditional testing system. After answering the fourth question, the software displays the points reached by the student, which will be considered the first margin for the "eminent" mode starting at this moment (the next margin will automatically be set for three points higher). The students who have "fallen out" will take part in the following testing as spectators: "post eminent" testing.

By developing the "Who wants to be eminent" educational software our aim was to ensure such a method and tool for the assessment process, which can bring excitement

into testing, activates the whole classroom, beyond the evaluation it teaches and educates as well, has all the advantages of testing compared to the traditional way, exploits maximally the advantages given by the computers and computer sciences and leaves behind a data base which makes extensive educational research possible. We have also paid attention on building into the software the advantages of the already existing applications, next to the original ideas. The following summing up of the method's advantages ensure us that we have reached our goal. We hope that further experiences and research will enforce this conclusion.

- It creates spirit of contest and offers excitement, which are further resources concerning the preparation for the test.
- The classroom variant of the software "models" the teacher, who pays attention to the stress at the start, makes help possible when the student has problems, is ready to take into account the knowledge revealed after the end of the test.
- Several students can answer in the same time. If we allow one minute for each of the first four questions, two minutes for each of the following three and two minutes and a half for each of the last two questions, we can finish the testing in 15 minutes.
- The software makes it possible to use the strong points of the "eminent" testing at examinations.
- The whole class is working, as the questions appear on the screen of each student and they have to answer. Furthermore, with the help options the whole class is part in the assessment of the students. The "eminent" mode gives a "legal" frame to the way the whole class can help the responsive students. All these mean an extra resource in the preparation for the next class.
- The server monitors the activity of every student for the teacher. This eliminates the possibility, that the lack of knowledge or the very high level of knowledge of the student not being tested remains unnoticed.
- After the end of the testing the teacher already has some printable statistical figures.
- As the module editing the questions can be used independently, the students can be involved in editing the questions, which has several advantages.
- There is an electronic minute made about every second of the testing. This database makes possible a later analysis of the teaching-learning process.
- Regarding the low time needed for this testing, it can be used quite often and thus a fine-tuning of the teaching-learning process becomes possible.
- Beyond assessment, it makes education possible, not only teaching and learning.
    - It activates such features in the students as selflessness, helpfulness, sympathy. The spectator students express these features by living thoroughly the testing as though they should answer, in order to do their best with their help.
    - Self-confidence becomes evident, respectively the confidence in the classmates and the class as a whole.
    - Can contribute to the strengthening of the class spirit.
- The strict punishment of the wrong answers, the possibility to stop, the common presence of the margins and the help resources often put the students in front of choices which test them but in the same time it develops their wisdom.

Irodalomjegyzék

[1] R. Kotulak, *Inside the Brain*, Andrews McMeel, Kansas City, 1996
[2] R. L. Atkinson, R. C. Atkinson, *Pszihológia*, Osiris, Budapest, 1997
[3] P. Drucker, *Knowledge-worker Productivity: The Biggest Challenge*, California Management Review 41, S. 79–94, 1999
[4] P. Drucker, *Beyond the Information Revolution*, The Atlantic, 1999. 12 www.theatlantic.com/issues/99oct
[5] F. Schaffhauseri, *Társadalom-iskolaelmélet – iskolaszervezet*, Bábosik I., R. Olechowski (Szerk.), Tanítás-Tanulás-Értékelés, Peter Lang Tudományok Európai Kiadója, 2003
[6] Comenius, *Orbis sensualium pictus*, 1653
[7] Z. Krygowska, *A matematikadidaktika jelenkori kutatásainak főbb irányzatai és problémái*, Dydaktyka Matematyki, Varsó, 1/1982
[8] H. Braun, *Reflections on the future of assessment,* Konfernciaelőadás, EARLI (European Association for Research on Learning and Instruction), Assesment and evaluation, 2000. szeptember 13–15., Maastricht
[9] A. Stephen, *Hogy kell a tanulás eredményeit használni?* „Csoportkapcsolatok", Bolognai Konferencia, Budapest, 2006. január 25.
[10] K. Garnitschnig, G. Khan-Svik, *Aktív tanulás. A sikerese oktatásszervezés aspektusai*, Bábosik I., R. Olechowski (Szerk.), Tanítás-Tanulás-Értékelés, Peter Lang Tudományok Európai Kiadója, 2003
[11] Kátai Z., *Algoritmustervezés felülnézetből*, Sapientia Kiadó, Kolozsvár, 2006
[12] B. S. Bloom, *Taxonomie von Lernzielen im kognitiven Bereich*, Weinheim, Base, 1972
[13] B. S. Bloom, D. R. Krathwohl, *Taxonomie von Lernzielen im affektiven Bereich*, Weinheim, Base, 1975
[14] E. Persy, *Motiváció a tanítás-tanulás folyamatában*, I. Bábosik, R. Olechowski (Szerk.), Tanítás-Tanulás-Értékelés, Peter Lang Tudományok Európai Kiadója, 2003
[15] E. Ch. Wittmann, *Grundfragen des Mathematikunterrichts*, Vieweng, 1981
[16] L. Eliot, *What's Going On In There? How the Brain and Mind Develop in the First Five Years of Life*, Bantam Doubleday Dell Pub, 2000
[17] T. H. Cormen, C. E. Leirserson, R. L. Rives, *Introduction to Algorithms,* by The Massachusetts Institute of Technology, 1990, 266-270, 287-289.
[18] I. Odagescu, C. Copos, D. Luca, F. Furtuna, I. Smeureanu, *Metode şi tehnici de programare*, Intact, Bucuresti, 1994, 95-108.
[19] R. Andone, I Garbacea, *Algoritmi fundamentali. O perspectivă C++*, Libris, Cluj-Napoca, 1995, 185-187, 219-221.
[20] Revákné Markóczi I., Máthé J., *A természettudományos problémamegoldó gondolkodás fejlesztése a középiskolában*, Új Pedagógiai Szemle, 2002/10
[21] J. Dewey, *How we think,* New York, Holt, Rinehart and Winston, 1910
[22] J. Rossman, *The psychology of the inventor,* Washington, Inventor's Publishing Co, 1931
[23] A. Osborne, *Applied imagination,* New York, Scribner, 1963
[24] Pólya Gy., *A problémamegoldás iskolája,* Budapest, Tankönyvkiadó, 1979
[25] Lénárd F., *A problémamegoldó gondolkodás,* Budapest, Akadémiai Kiadó, 1963
[26] J.Holt, *How Children Fail*, New York, Pitman, 1994.
[27] *Research on the Brain*, Classroom Compass, 1997, Volume 3, Number 2

[28] D. Dickinson, Washington State Art Commission, winter 2003 issue
[29] H. Gardner, *Frames of Mind*, 1985
[30] *Rhythm of Mathematics*, Classroom Compass, 1998, Volume 4, Number 2
[31] M. D. Levine, *A Mind at a Time*, Simon & Schuster, 2002
[32] H. C. Hughes, *Sensory exotica – a world beyond human experience*, Bradford Book, 2001
[33] Bábosik I., R. Olechowski (Szerk.), Tanítás-Tanulás-Értékelés, Peter Lang Tudományok Európai Kiadója, 2003
[34] D. Laird, *Approaches to training and development,* Addison-Wesley, Reading, Mass., 1985
[35] *Navy Instructor Manuals*, www.tpub.com/content/administration
[36] *Out of Memory*, http://library.thinkquest.org/C0110291
[37] http://www.cis.upenn.edu/~matuszek/cit594-2004/Lectures/44-dynamic-programming.ppt
[38] P. Russel, *The Brain Bokk*, Dutton, New York, 1979
[39] Komenczi B., *Orbis sensualium pictus. Multimédia az iskolában*, Iskolakultúra, 1997/1
[40] Paivio, *Mental representations. A dual coding approach*, New York, Oxford University Press, 1986
[41] D. L. Nelson, V. S. Reed - J. R. Walling, *Pictorial superiority effect*, Journal of Experimental Psychologi: Human Learning and Memory, 1976/2, 523-528
[42] M. Paechter, *Sprechende Computer in CBT: eine didaktische Konzeption*, Arbeiten aus der Seminar für Pädagogik, Bericht 1/93. TU, Braunschweig, 1993
[43] M. Pyter, *Textrepresentation in Hypertext. Empirische Analyse von visuellen versus audiovisuellen Sprachdarbietungen in Hypertext*, Papier zum Kongress der DGPs, Hamburg, 1994
[44] Báthory Z., *Értékelés a pedagógiában*, Pedagógiai Szemle, 1972/3, 212-220
[45] Golnhofer E., *A pedagógiai értékelés*, Falus Iván (szerk.), Didaktika. Nemzeti Tankönyvkiadó, Budapest. 392-414.
[46] H. Brückner, *Számítógépek az oktatásban, Számítógépes oktatás*, KSH Nemzetközi Számítástechnikai Oktató és Tájékoztató Központ, Budapest 1978
[47] Bálya D., *Az informatika kihívása a teszt-technológiában*, TDK dolgozat, BME TIO 1997
[48] A. Binet, *A propos de la mesure de l' intelligence*, Paris, 1905
[49] J. Benek-Rivera, V.E. Mathews, *Active learning with Jeopardy: Students ask the questions*, Journal of Management Education, 28 (1), 104-118, 2004
[50] J. J. Grabowski, Michelle L. Price, *Simple HTML Templates for Creating Science Oriented Jeopardy! Games for Active Learning*, Department of Chemistry, University of Pittsburgh, 2002
[51] D. Marsh, *Who wants to be a maths millionaire?*, http//ferl.becta.org.uk, 26 April 2002
[52] J. J. Cochran, *„Who Wants To Be A Millionaire: The Classroom Edition",* INFORMS Conference, San Antonio, TX., 2000
[53] G. Brown, M. Atkins, *Effective Teaching In Higher Education*, Methuen, London, 1988
[54] J. Hartley, I. K. Davies, *Note Taking: A Critical Review,* Programmed Learning and Educational Technology, Vol. 15, 207-224, 1978
[55] W. J. McKeachie, *McKeachie's Teaching Tips: Strategies, Research and Theory for College and University Teachers*, Houghton-Mifflin, Boston, MA, 1999

[56] T. Sutherland, C. Bonwell, *Using Active Learning In College Classes: A Range Of Options For Faculty*, New Directions for Teaching and Learning, Vol. 67, 1996
[57] J. M. Collins, *Who wants to be a millionaire? An educational game for learning enhancement*, Innovative Teaching Practices at Bloomsburg University, Teaching and Learning Enhancement Center, 2004
[59] Kátai Z., *Dynamic programming and d-graphs*, Studia - Informatics, Cluj, 2006
[60] T. Angelo, P. Cross, *Classroom assessment techniques: A handbook for college teachers* (2nd edition), San Francisco, 1993
[61] Kátai Z., *Dynamic programming strategies on the decision tree hidden behind the optimizing problems*, Informatics in Education, Institute of Mathematics and Informatics, Lithuania, 2006 (elfogadva)
[62] Kátai Z., *Involvement of the sight, hearing, touching and moving in teaching basic algorithms*, Computers & Education, 2006 *(benyújtva) (Társszerzők: Nyakóné Juhász Katalin, Adorjáni Alpár Károly)*
[63] Kátai Z., *"Upperview" algorithm design in teaching computer science in high schools,* Teaching Mathematics and Computer Science, 3 (2005) 2
[64] Kátai Z., *Programozás C nyelven*, Sapientia-EMTE, Scientia Kiadó, Kolozsvár, 2004

Publikációs jegyzék

Referált nemzetközi folyóiratban megjelent cikkek:

[1] *"Upperview" algorithm design in teaching computer science in high schools,* Teaching Mathematics and Computer Science, 3 (2005) 2, (221-241)

[2] *Dynamic programming and d-graphs*, Studia Universitatis Babes-Bolyai -- Series Informatica, 2006 *(elfogadva)*

[3] *Dynamic programming strategies on the decision tree hidden behind the optimizing problems*, Informatics in Education, Institute of Mathematics and Informatics, Lithuania, 2006 *(elfogadva)*

[4] *Involvement of the sight, hearing, touching and moving in teaching basic algorithms*, Computers & Education, 2006 *(benyújtva) (Társszerzők: Nyakóné Juhász Katalin, Adorjáni Alpár Károly)*

Lektorált egyetemi jegyzetek:

[1] *Programozás C nyelven*, Sapientia-EMTE, Scientia Kiadó, Kolozsvár, 2004

[2] *Algoritmustervezés felülnézetből, Sapientia-EMTE, Scientia Kiadó, Kolozsvár, 2006*

Tudományos konferenciákon elhangzott előadások:

[1] *Hogyan tanítsuk a programozási technikákat,* XIV Számítástechnika az oktatásban nemzetközi konferencia, EMT, Kolozsvár 2004. március 25-28

[2] *Programozási technikák felülnézetből,* XV Számítástechnika az oktatásban nemzetközi konferencia, EMT, Kolozsvár 2005. március 17-20

[3] *Algoritmus tervezés (Didaktikai szempontok),* Informatika a felső oktatásban nemzetközi konferencia, Debrecen, 2005, augusztus 24-26

Egyéb publikációk:

Tudományos konferencia kiadványok:

[1] *Hogyan tanítsuk a programozási technikákat?,* XIV Számítástechnika az oktatásban nemzetközi konferencia, EMT, Kolozsvár, 2004, (50-56)

[2] *Programozási technikák felülnézetből*, XV Számítástechnika az oktatásban nemzetközi konferencia, EMT, Kolozsvár, 2005, (139-146)

Egyéb tudományos folyóiratokban megjelent cikkek:

[1] *Programozási technikák felülnézetből (I rész)*, Firka, 2003/2004-4, EMT, Kolozsvár (145-148)

[2] *Programozási technikák felülnézetből (II rész)*, Firka, 2003/2004-5, EMT, Kolozsvár, (190-192)

Szakmaszpecifikus produktumok

[1] "Legyél te is eminens", Hálozati értékelési szoftver, (Társszerző: Máthé Szabolcs)

[2] "Elemi algoritmusok anatomiája", Didaktikai szoftver, (Társszerző: Adorjáni Alpár Károly)

Módszerek és eszközök az informatikaoktatás hatékonyságának növelésére

Értekezés a doktori (PhD) fokozat megszerzése érdekében informatikadidaktika tudományágban

Írta: Kátai Zoltán okleveles mérnök

Készült a Debreceni Egyetem TTK Matematika- és Számítástudományok Doktori Iskolája (Didaktika programja) keretében

Témavezető: Dr. Nyakóné dr. Juhász Katalin .....

A doktori szigorlati bizottság:

Elnök: Dr. .....
Tagok:
Dr. .....

Dr. .....

A doktori szigorlat időpontja: .....

Az értekezés bírálói:

Dr. .....

Dr. .....

Dr. .....

A bírálóbizottság:

Elnök: Dr. .....
Tagok:
Dr. .....

Dr. .....

Dr. .....

Dr. .....

Az értekezés védésének időpontja: .....