SZAKDOLGOZAT

Csató Endre

Debrecen

2009.

Debreceni Egyetem

Informatikai Kar

Informatikai versenyek

Témavezető:

Dr. Papp Zoltán Lajos

egyetemi adjunktus

Készítette:

Csató Endre

Informatika tanár

Debrecen

2009.

Tartalomjegyzék

I. Bevezetés

E szakdolgozat témája alapján lehetne egy kiválasztott verseny mély elemzése, amely alapján a felkészítő tanároknak példán keresztül bemutatnám, milyen minta feladatokra/algoritmusokra való odafigyeléssel érdemes elindítani a versenyeken a diákokat.

De én ennél szeretnék többet. Ezért megpróbálom kategorizálni, összehasonlítani, elemezni a sok informatikai versenyt, lehetőséget nyújtva arra, hogy a megfelelő diákot a megfelelő versenyre küldhessük el. Természetesen a teljesség igénye nélkül, csak példákon keresztül.

A kedves olvasót arra kérem, hogy lapozzon a függelékbe (VII.1) és tanulmányozza át az Informatikai Diákolimpia tanmenetét. Meg fog lepődni, hogy a középiskolások számára mi minden ismeret szükséges a helytálláshoz. Szeretném elérni, hogy egy kis útmutatást adjon e dolgozat egy egyszerű iskolán belüli ad- hoc versenytől egészen a diákolimpiára készülőknek.

A dolgozat további célja, hogy azokra a problémákra is rávilágítson, ami hiányzik a gyerekek (és sokszor a felkészítő tanár) tarsolyából ahhoz, hogy eredményesen vegyen részt a versenyen az induló.

Miért merek ilyen összehasonlításokat végezni? Álljon itt egy történeti és személyes indíttatású áttekintés, amiért beleszeretem a versenyeztetésbe és a tanításba.

I.1. *Versenyzőként*

(1989-1991.) A középiskolai tanárom indíttatására eljutottam a megyei programozói versenyre. Tombolt a Commodore „korszak”. Amire azóta is emlékszem, hogy hajnalig tartó „*demo” író partivá* változott az első forduló utáni „várakozás”. Nem volt könnyű az első fordulós feladatsor, a vacsorát, mint kiéhezett farkas ettük meg, bármi is volt. Megragadott ugyanígy a régi kollégium hangulata is, a „líceumi sugárzás”. Az esélytelenek nyugalmával írtuk és másoltuk még hajnali 4-kor is a programokat (a másolásra ma ez úgy mondanánk illegálisan). E tájt egyszer csak *Dr. Papp Zoltán* szólt, hogy ideje aludni, készülni a második fordulóra. Amíg meg nem mutatták a továbbjutókat, el sem hittem. Továbbjutottam, és végül a 8-dik helyen végeztem. Ez a verseny annyira meghatározó volt, hogy már alig vártam a következő évet. Ekkor *Dusza Árpád* Tanár Úr vette át a terelgetésemet a programozói úton. Bár sokszor nézeteltérésünk volt, ekkor már sikerül a döntőben második helyen végeznem. Esélyem sem volt akkori Földesbeli konkurenciámmal szemben. Utolsó évben nyertem, ami hozzájárult, felsőfokú végzettségemet Debrecenben szerezzem meg.

I.2. *Zsűritagként*

1991 és 2006 között pár kihagyással talán, de sikerült Papp Zoltán Tanár Úrral (és Pusztai László barátommal) először csak a zsűriben segédkezni (több megye programozói és alkalmazói versenyén), később feladatokat is összeállítani. Erre azért is volt szükség, mert a csökkenő gyereklétszám és az érdeklődés megváltozása életre hívta a verseny alkalmazói kategóriáját. Így ismét szép számmal indultak versenyzők. Nem mondtuk, hogy könnyebb lett volna a feladatokat összeállítani, valamint zsűrizni, főleg a „művészi összbenyomás" miatt, de minden alkalommal törekedtünk az objektív értékelésre. Az első fordulós feladatok kezdetben papíron folytak, innen idézném a megnevettető ötletet, Basic nyelven:

```
REM EKMŰVEM ELSŐ SORA
```

Élveztük a gyerekek munkáját, például *Marhefka Istvánét* (akit majd később említek egy versennyel kapcsolatban), aki diákolimpiáig és szintén a zsűriig jutott, vagy *Sasvári Tamásét,* aki assemblyben majdnem elverte magasabb szintű nyelvet használó mezőnyt.

I.3. *Felkészítő tanárként*

2007-től, már gimnáziumi tanáraként vittem versenyezni a diákokat. Jó volt látni, hogy érdemes foglalkozni a gyerekekkel a megváltozott, felgyorsult világban is. Ugyanakkor célul tűztem ki, hogy az üzleti életben töltött évek tapasztalatával igyekszem felvértezni a tanulókat, hogy az újkori olimpiai idézet ne csak szlogen maradjon:

„Citius, Altius, Fortius (gyorsabban, magasabbra, távolabbra)"

Csató Endre

Informatika tanár

Debrecen, 2009.

II. Informatikai versenyek

Versenyeztessünk, ne versenyeztessünk? Hogyan mérjék a diákok a saját tudásukat, és hogyan hasonlíthatják össze azt másokéval? Adjunk-e és ha igen, milyen fokú sikerélményt? Mi a célunk? Hogyan nevelhető ki a következő generáció sokszor fanatikus, mai szóval élve „kocka” informatikusa? Ez a szó mennyiben változott meg az elmúlt évek és évtizedek alatt? Ki érzi magát informatikusnak, a diák hova sorolja magát?

Nem szeretném az Interneten fellelhető sok információ alapján tovább boncolgatni és feltenni a hasonló kérdéseket, mert sokat többen és nálam jobban megválaszoltak már. (1)

II.1. Szempontrendszer

A dolgozat szempontrendszere az első fejezetre korlátozva a következők szerint hasonlítja össze a különböző versenyeket.

- Szintek (területek)
- Korcsoportok
- Kategóriák
- Típusok
- Fordulók

A összehasonlítást minden területen kiegészítem a következőkkel, skálázva is őket 1-10 között:

- Erősségi szint, (általánosan, összegezve)
- Élményszint (mekkora élmény jelent a gyereknek, ha elmegy)
- Motiváltság (mennyire lehet motiválni vele, mennyire érzi fontosnak a gyerek)
- Érdekesség (a feladatok mennyire érdekesek)
- Tárgyi tudás (mekkora tárgyi tudást feltételez induláskor)
- Jutalom (mekkora jutalmat kaphat, ebbe beleértve a részvétel erkölcsi jutalmazását, oklevél, dicséret címén)
- Jövőbemutatás (mennyire fontos ez a következő lépcsőfok elérésében, a fejlesztésben)

Nem minden csoportosításnál tudom, akarom, lehet mindenben helyesen és értelmesen állást foglalni. Mégis az alfejezet végén egy összehasonlító táblázatban számszerűsítem a szubjektív

véleményemet, sőt, még grafikonon is próbálom ábrázolni, a jobb összehasonlíthatóság érdekében.

II.2. Szint (terület)

Sokszor halljuk, hogy a tanárok büszkék tanítványai akár iskolán belül elért vagy városi versenyen mutatott jó eredményeikre. Ez így is van rendjén, de azért rangsoroljuk, hogy milyen szinten érdemes indítani és besorolni a versenyeket és a hozzájuk rendelhető tanulókat, mert a diákolimpia, vagy annak felkészítése más munkát és energiát igényel.

II.2.1. Verseny egy csoporton, osztályon belül

Van ennek értelme? A válasz igen, ha sikerül egy olyan „tehetségekkel teli" csoporttal vagy osztállyal dolgozni, akik a tananyagon kívül érzik magukban a késztetést, hogy pluszt tanuljanak és arról számot is adjanak. Nagyon nehéz ugyanakkor ezt belehelyezni a jelenlegi oktatási környezetbe a minimalizált óraszámok miatt, mégis szükség lehet rá, hogy egy-egy eddig habozó diák a sikerélményen keresztül vonódjon be a versenyzésbe. Ne feledjük, hogy az elindítás, ha eddig még a diák nem foglalkozott a számítástechnikával, nagyon fontos, meghatározó, éppen ezért rengeteg plusz energiát igényel.

A legfontosabb, hogy érdekes, élmény-gazdag, az eddigi tudásából következő feladatokkal találkozzon a jutalmazásra nem is számító, de a jövőjét ezáltal önmaga nagymértékben befolyásoló versenyző.

Ne keverjük a normál órai munkával, itt a versenyek előtti szint és tudásfelmérést célzó feladatoknak van hely, általában belépő osztályokban, az év elején.

II.2.2. Iskolai versenyek

Melyik tanárnak van ideje az iskolán belül a kötelező óraszáma mellé saját versenyeket szervezni? Van-e elegendő gyermek létszám ahhoz, hogy érdemes legyen foglalkozni vele? Mikor jön jól egy ilyen verseny? Képzeljük el a következő lehetőségeket: egyrészt túl sok azonos szintű gyerek van, akiket a versenyekre szét kell osztani, ekkor egy iskolai forduló segíthet az igazságos, objektív döntésben. Másrészt adódnak olyan pillanatok (fordított napok, kulturális nap, stb.) amikor a gyerekek maguk szervezhetnek versenyt (kontrollal), és még így könnyebben fogadják el az eredményt. Ugyanakkor az iskolai versenyek lehetnek a többi tárgyi versenyek dömpingje előtt és után is, így folyamatosabb terhelés alá kerülhetnek a gyerekek. Ezeken a versenyeken már szabad akár előre témát kiadni, hogy önmaguk járjanak utána, de nem szabad csak tárgyi tudást számon kérni, mert akkor az évfolyamok és szintek

közti szakadék egyértelmű eredményre fog vezetni. Sőt, ha olyanoknak adjuk ki a verseny kiírását, feladatírást és zsűrizést, akik eleve egy adott csapat részei, és biztos a tagságuk adott versenyeken, akkor sokkal homogénebb eredményt, de még mindig mérhető eredményt kapunk.

Itt kapcsolódik már az iskolai számítástechnikai közössége, akár mint szakköri közösség egymáshoz, ezért egymás és annak tudásának ismeret minta és húzóerő lehet a többiek számára. Fontos azonban még az élmény-gazdag, érdekes feladatok kitalálása, és a jutalmazás lehetősége: más versenyeken való indulás.

II.2.3. Városi, települési versenyek

Ennek csak akkor van értelme, ha egy városban több olyan középiskola is van, akiknél folyik verseny szintű tárgyi felkészítés, szakkör, tanfolyam, stb.

Egyre inkább kimegy a divatból, mert a fenntartók az egyes oktatási formák szétterítésében, a tanerők optimalizálásában érdekeltek, főleg pénzügyi szempontból, ezért a tagozatok, fakultációk, szakkörök kevésbé versenyeznek egymással a településeken belül.

Itt már megjelenik a szaktanárok tanítási módszertanának eredményei is, amit elismerő oklevelet átadásakor ismerhet meg a közösség.

A városi versenyeken éppen ezért nehéz igazságos feladatsort kitalálni,ha azt egy szaktanár teszi. Csapatmunka eredményeként az ütköztetett feladatok már jól mérhető, iskolánként rangsor felállításához is elegendő tapasztalatot nyújthatnak.

Az erősségi szint már olyan kell legyen, hogy a legjobbaknak is jusson gondolkodtató feladat.

A városi és a továbbiakban felsorolt versenyekre is igaz az, hogy az élmény megvalósítása a nagyon könnyű feladattól indul, azaz ne legyen 0 pontos eredmény.

II.2.4. Megyei versenyek

Ezek a versenyek vették át rohanó világunkban a kisebb versenyek helyét, és első szintű megmérettetési helyszínné váltak (tapasztalataim szerint).

Lehetőséget nyújtanak a szaktanároknak a nevelés során adott fázisban jutalmazni a versenyen való részvétellel, vagy éppen ellenkezőleg, serkenteni a diákokat, hogy sokan vagytok, csak 2-2 fő juthat a versenyre.

Sokan vitatkoznak velem, de én fontosnak tartom, hogy a gyerekeket motiváljuk a versenyre és a versenyen. A versenyen az elért eredmény alapján olyan tárgyi vagy más ajándék érhető el, ami pénzszemléletű világunkban a gyerekre nézve is serkentő hatású. Ehhez természetesen szponzorok szükségesek, de ebből még addig nem volt hiány. És azt az arcot nem lehet elfelejteni, amikor egy kézzel fogható hardver, szoftver, könyv, utalvány, stb. átvételekor a büszkeség leolvasható az arcokról. (Később teszek majd megjegyzést az országos versenyekre a témában.)

Az erősségi szint nem kell, hogy megegyezzen az országos megmérettetéssel, de szórást kell biztosítson. Itt már elvárható olyan tárgyi tudás, amely mind a matematikai, mind az informatikai alapokat érinti, sőt, kategóriánként a felsőbb szintű versenyekből merít. Annyira legyenek „csak” érdekes a feladatok, hogy a gondolkodás, szövegértés, ügyesség, elvonatkoztatás területén fejlesszék a diákokat, az iskolában megszokott közvetlen és az ún „rejtett tanterv” segítségével. Utóbbihoz sorolom a feladatok közös megoldását és elemzését, bár idő hiányában ezek el szoktak maradni. Ne feledjük, hogy bármely tanári munkánál többet mond a tanoncoknak a másik diák produktumának megtekintése, algoritmusának ismertetése, gondolkodásának asszimilációja.

A versenyek ugrást jelentenek az országos szint felé, de sokszor nem érik el azt (sajnos).

A versenyek, amely alapján e gondolatokat leírtam a Borsod-Abaúj-Zemplén megyei Számítástechnika versenyen szerzett személyes tapasztalatok (2).

II.2.5. Országos versenyek

Vegyük sorra mely versenyekről is beszélünk, és mi a versenykiírás és a követelményrendszer. (Van, ahol a kategóriánál vagy a korcsoportnál részletezem.)

II.2.5.1. Logo Országos Számítástechnikai Tanulmányi Verseny (Logo)

(3) A verseny tárgya: számítástechnikai és programozási alapismeretek, felhasználói és kezelői felületek kialakí-tása, a Logo programozási nyelv alapvető elemei, rendszerszemléletű feladatmegoldás, algoritmusok kidolgozása, megvalósítása számítógépen.

Szükséges ismeretek: (a római számok a legkisebb korcsoport sorszámát jelölik, ahol ezt az ismeretet feltételezzük):

- *A Logo nyelv grafikai utasításainak ismerete. (I.)*

- *A Logo-szerű gondolkodásmód, Logo-szerű algoritmusok megértési és végrehajtási képessége. (I.)*
- *Elemi és összetett alakzatok megrajzolása, eljárások használata, eljárások paraméterezése. (I.)*
- *Ábrák eltolása, nagyítása, elforgatása. (I.)*
- *Sokszögek, csillagok, körök, körívek rajzolása. (I.)*
- *Sorminták, területminták (mozaikok) rajzolása. (I.)*
- *Rekurzív ábrák (fák, indák,spirálok, fraktálok) rajzolása. (II.)*
- *Zárt területek befestése. (I.)*
- *Érzékelős teknőc alkalmazása. (III.)*
- *A Logo nyelv szövegkezelő függvényei, alkalmazásuk szöveg-feldolgozási feladatokban. (III.)*
- *A Logo nyelv rajzolási és szövegkezelési lehetőségeinek összekötése: szöveges paraméterrel vezérelt rajzolás. (III.)*

Először találkozunk valamilyen fajta tanmenettel, követelményrendszerrel egy versennyel kapcsolatban. Ha jól szemügyre vesszük ez az a verseny, amely képes lehet játékos formában behálózni az ifjút még az általános iskolában. Innen már csak egy dolga van a középiskolának, ne engedje el ezt a szálat, sőt, fejlessze tovább és transzformálja a megszerzett tudást a későbbi Nemes és OKTV versenyekre. Nem is sejtjük, hogy milyen sok, bár nem meg nevezett ismeretet halmozhatnak fel a gyerekek. Az utasítások, függvényhívások, nyelvi elemek, kifejezések eljutnak a fa, fraktál és rekurzió használati szintű elsajátításához. Az algoritmusok és a programozás grafikával vegyített, magas élményindexű használata a vizuális befogadó készségre épül. Jutalom maga a siker, hogy működik, amit készítettünk, még motiválni sem kell a pszichológiai fejlettség alacsonyabb szintjén a gyerekeket, és az ön-motiváció átszivárog a nagyobb, felsőbb korcsoportba is.

Említsük meg, hogy a kicsiknek a kézzel fogható megoldásokhoz speciális robotok építhetők, ha persze van megfelelő anyagi támogatás az ilyen készletek megvásárlásához.

II.2.5.2. Nemes Tihamér Országos Középiskolai Tanulmányi Verseny (Nemes)

(4) A verseny elsődleges célja az, hogy az általános és a középiskolák tanulóinak lehetőséget adjon programozási ismereteik és képességeik összehasonlítására.

A versenyfeladatok a problémamegoldó, algoritmizáló, modellalkotó, modularizáló készséget mérik fel. A Pascal nyelv használatában való jártasságot elvárjuk a versenyzőktől, de fontosnak tartjuk más programnyelvek (pl. BASIC, Logo, C++, Prolog, assembly stb.) szemléletmódjának ismeretét is. A hangsúly nem az egyes nyelvek részleteinek, hanem a módszeres programozás fogalmainak, elveinek és gyakorlatának, a helyes programozási módszereknek és stílusnak géptől és nyelvtől független ismeretén van.

Elvárt alapvető ismeretek középiskolásoktól:

- *A Pascal nyelv elemei. Szintaxisábrák, BNF-jelölés. Strukturált vezérlési szerkezetek: felsorolás (szekvencia), választás (elágazás), ismétlés (ciklus).*
- *Adattípusok: egész, valós, logikai, karakter, szöveg. Összetett adatok: tömb, halmaz, rekord, lista, verem, sor, fa, gráf, állomány stb. Láncolt ábrázolás.*
- *Fölülről lefelé haladó programozás lépésenkénti finomítással. Programok élesztése, tesztelése, hatékonysági megfontolások.*
- *Számábrázolás bináris, oktális, decimális, hexadecimális számrendszerben. Átalakítások, alapműveletek. Fixpontos és lebegőpontos ábrázolás. Pontosság, túlcsordulás.*
- *Boole-algebrai és matematikai logikai alapismeretek. Közelítő módszerek. Görbe alatti terület kiszámítása. A valószínűségszámítás alapelemei: gyakoriság, relatív gyakoriság, középérték, súlyozott középérték, hisztogram stb. Véletlenszámok és alkalmazásuk (kockadobás, lottószámok húzása stb.).*
- *Programozási típusalgoritmusok. Pl. rendezések, keresések, visszalépéses keresés, stb. Adatok beszúrása, törlése, keresése.*
- *Gráfalgoritmusok, gráfbejárás, fabejárás.*
- *Dinamikus programozás, mohó algoritmusok, kombinatorikus algoritmusok.*

II.2.5.3. Országos Középiskolai Tanulmányi Verseny (OKTV)

(5) Az OKTV, mint a Nemes III. korcsoport (alkalmazóiban II.) jelenik meg, és a lebonyolításában illeszkedik a más tárgyak országos versenyeihez, így a lebonyolítása is az OKÉV által szervezet.

II.2.5.4. Informatikai Diákolimpiák válogató versenyei (Válogató)

(6) Az IOI válogatóversenyen az Informatika OKTV Programozás kategóriájának első 20-25 helyezettje vehet részt (a résztvevők pontos számát a versenyek eredményének ismeretében az Országos Versenybizottság állapítja meg).

A CEOI válogatóversenyen az Informatika OKTV Programozás kategóriájának első 20-25 helyén végzett 11. osztályos tanulók, valamint a II. kategória első 3-6. helyén végzett 9-10. osztályos tanulók vehetnek részt (a résztvevők pontos számát a versenyek eredményének ismeretében az Országos Versenybizottság állapítja meg). Kiemelkedő eredmény esetén az I. kategória győztese is indulhat a válogatóversenyen.

Mindkét válogatóverseny jogosult résztvevői a korábbi diákolimpiák korcsoportban megfelelő magyar résztvevői.

II.2.5.5. Dusza Árpád Emlékverseny (Dusza)

(7) A programozás iránt érdeklődő tehetséges diákok számára lehetőséget szeretnénk teremteni egy olyan országos szintű megmérettetésre, amelyen az informatikai ismereteik mellett a csapattársakkal való együttműködésben is kipróbálhatják magukat. További célunk, hogy segítsük a felkészülést az emelt szintű informatika érettségire, a felsőfokú informatika tanulmányokra, illetve a különböző projektmunkákban való hatékony részvételre.

A versenyen olyan 3 fős csapatok vehetnek részt, amelyeknek tagjai az ország valamely oktatási intézményének 9-13. évfolyamos tanulói. Egy csapat tagjai különböző iskolák tanulói is lehetnek, ilyenkor bármelyik csapattag iskolája elküldheti a nevezési lapot.

A verseny tárgya, követelményei:

Komplex problémák megoldása részfeladatokra bontással. Függvények, eljárások készítése. Típusalgoritmusok, rendezés, keresés, rekurzió illetve fájlkezelés használata. Gráfalgoritmusok, gráfbejárás, fabejárás. A grafikus képernyő kezelése.

A feladatok megoldásához szükség lehet a középiskolában tanult matematikai és fizikai ismeretekre.

A verseny elsősorban csapatversenyként és komplex probléma és konkrét alkalmazás kifejlesztés, dokumentálás területén tér el a többi versenytől.

A verseny kiírás és zsűriztetés is eltér a megszokottól:

Regionális fordulóban: egy komplex feladat megoldása a versenyző csapat által választott feladatmegosztással. A feladatmegoldás tartalmazza a megfelelő kommentek elhelyezését is.

Döntőben: Egy komplex feladat megoldása a versenyző csapat által választott feladatmegosztással, valamint az elkészült program rövid, összefoglaló fejlesztői dokumentációjának elkészítése. Az elkészült munkákat a csapatok 10-15 percben bemutatják.

Tehát figyelni kell a produktum olvashatóságára (nem csak kód szinten, de ott sem elhanyagolható a munkamegosztás miatt), valamint az „eladhatóságra", bemutathatóságra.

A gyerekeket rá kell szoktatni a kódban a változó, függvény, eljárás elnevezések megfelelő használatára (pl.. magyar jelölés vagy összetett szavas jelölés). Ugyanígy a kommentek, behúzások, struktúrák, elágazások, utasítás-zárójelek megfelelő, szisztematikus és konzisztens és konzekvens használatára.

Dokumentáció: a gyerekek általában már annak is örülnek, ha megértik a feladatot. Azt, hogy le is írják, mit akartak megvalósítani, nehezen végzik el. Azt, hogy előbb legyen leírva, tehát megtervezve a program, az algoritmusok, kódrészletek, végképp nem hallottak és nem is gyakorolják. Ezzel - pofonként - először az „életben" találkoznak, ha programozói vagy programtervezői pályán helyezkednek el.

A verseny még csak kétéves. A gyermektől elvárható, hogy legyenek betegségei, de így is megpróbálja betölteni az űrt az algoritmizáló diákolimpia és versenytársai (Nemes, OKTV) és a szoftverfejlesztési ismereteket IS igénylő valódi ipari szoftverfejlesztés között. Erre a verseny egyik feladat kitalálója és zsűritagja, *Marhefka István* a garancia, aki diákolimpiai résztvevő is volt és jelenleg szoftverfejlesztéssel foglalkozó vezető fejlesztő.

II.2.6. Nemzetközi versenyek

Hangsúlyoznom kell, hogy a teljesség igénye nélkül! Így elsősorban csak a Nemes és az OKTV (programozás kategória) győzteseinek szól a felvételi pontszerzés mellett a további versenyzés és a kiemelkedő elismerés megszerzésének lehetősége. Itt a motiváltság nem más, mint maga a versenyen való részvétel, az elért eredmény már mindenképpen rangot és példaértékű teljesítményt jelent a diákok számára.

II.2.6.1. Közép-Európai Informatikai Diákolimpia (CEOI)

(8) A versenyt hivatalosan nyolc közép-európai ország kezdeményezte 1994-ben a Közép-Európai Informatikai Diákolimpiát (Central-European Olympiad in Informatics, CEOI), a

Nemzetközi Informatikai Diákolimpián (IOI) szereplők közvetlen utánpótlásának versenyeztetésére, ahol minden országot legfeljebb 4 versenyző képvisel.

Az alapítók: Ausztria, Csehország, Horvátország, Lengyelország, Magyarország, Románia, Szlovákia, Szlovénia. (Az elmúlt évben az alapítók köréhez való csatlakozást kérte Németország.) – ez is az egyik legfontosabb elismerése a verseny létjogosultságának.

A diákolimpia válogatóversenyén az Informatika OKTV Programozás kategóriájának első 20-25 helyén végzett 11. osztályos tanulók, valamint a II. kategória első 3-6. helyén végzett 9-10. osztályos tanulók vehetnek részt (a résztvevők pontos számát a versenyek eredményének ismeretében az Országos Versenybizottság állapítja meg). Kiemelkedő eredmény esetén az I. kategória győztese is indulhat a válogatóversenyen.

A csapatok összetétele a következő: maximum 4 tanuló (olyanok, akik 19 évesnél nem idősebbek az olimpia megrendezésének évében július 1-jén, és az előző tanévben még iskolába jártak), (ajánlott az IOI-s korosztálynál eggyel fiatalabbakat nevezni)

A résztvevők egyénileg versenyeznek, két fordulóban egy-három feladatot kell megoldaniuk, fordulónként 5-5 óra alatt. A feladatokat a nemzetközi zsűri választja ki a tudományos bizottság által előkészített feladatok közül. A feladatok algoritmikus jellegűek, a megoldásukhoz különleges gépi eszközre vagy programcsomagra nincs szükség.

Az olimpián valamennyi résztvevő rendelkezésére áll egy-egy IBM PC/AT kompatibilis számítógép a szükséges programokkal, fejlesztői környezettel. Semmilyen más segédeszköz nem használható.

Az informatikai diákolimpia (IOI) előszobájaként tekinthető, hasonló feltételekkel és színvonallal. Látható, hogy a bejutás és a versenyzés a Nemes és OKTV versenyeken elért eredmény következménye. A diákokat tehát nem különösen erre a versenyre, hanem a Nemesre és az OKTV-re kell felkészíteni, bár a feladatsorok áttanulmányozása, és a feltételek ismeret nem hátrány. Pl.: egy adott algoritmusnak idő és/vagy tárhatékonyságban is a korlátok között kell maradnia, hogy értékelhető eredményt kapjunk.

Felhívom a figyelmet arra, hogy a verseny, mint az IOI is, kötött nyelvi és fejlesztői környezettel dolgozik. Tehát nem árt, ha az ezt nem ismerő diákoknak ezeket ismertetjük és készség szintű elsajátítását javasoljuk és gyakoroltatjuk. Javasoljuk, mert szakkörök, iskolai

foglalkozások kevesek ahhoz, hogy ilyen szintre eljusson az ember. Sok olvasás, gyakorlás, fanatikus elmélyedés szükséges a diákok részéről. Ezt tudatosítani is szükséges bennük!

II.2.6.2. Nemzetközi Informatikai Diákolimpia (IOI)

(9) 1989-ben rendezték meg az első Nemzetközi Informatikai Diákolimpiát (International Olympiad in Informatics, IOI), az UNESCO támogatásával.

A versenyre a tematikát a függelék (VII.1) tartalmazza, az eredményeket pedig a (10).

A „nagyok" versenye. Követelményei hasonlóak a CEOI-hoz, az ott leírtak ide is vonatkoznak, csak világversenyről van szó.

Megjegyzem, hogy ide eljutni nem egyszerű, az OKTV 3, majd a válogató 4 fordulóján át lehet. Az olimpiára való közvetlen felkészülés az NJSZT (11) segítségével központilag zajlik.

II.3. Korcsoportok

A versenyek szintjétől függ, hogy hány és milyen korcsoportban indulnak versenyek. Általában elmondható, hogy nem jó keverni a „kicsik" és a „nagyok" tudására épülő számonkérést. Ezért legjobban a Logo verseny differenciál korcsoportjaival: 3-4,5-6,7-8,9-10.

A többi, (főleg országos) verseny legalább 9-10 és 11-13 elkülönítést végez. Nagyon fontos, hogy a nehézségi szint megfeleljen a pszichológiai, tudásbeli, logikai készségeknek, épüljön valamelyest a többi tantárgy (főleg a természettudományok) már tanult eseteire.

Ma hiába elvárható, hogy a gyerekek középiskolába már alapvető informatikai tudással érkezzenek. Érkeznek is. Sokszor játék, csevegés, internet használat, rajzolás az, ami ezt jelenti. Egy alkalmazás használatának mély elsajátítása már kevés gyermek hozománya, és ne is beszéljünk a programozói múlt kifejezés nagyon ritka jelenlétéről. Akinek otthoni (szülő, testvér) környezet ezt példázta, annak előnye van. De ez az előny egy érdeklődő, de kevesebb hozott tudással rendelkező diáknál behozható, ha Ő is így akarja. Ezért fontos a nem elriasztás, az eredmény, az élmény megléte az első körben a versenyekkel találkozók esetében.

Problémát okoz még az, hogy a megfelelő logikával megáldott gyerekek nem csak informatikából jeleskednek, így „versenylóként" más tantárgyakból, matematika, fizika, stb. is indítják Őket versenyeken. Ez nem is lenne baj, de sokszor hallom, hogy a hét minden napján szakkör, elfoglaltság, edzés, stb. az, amivel a gyerek foglalkozik. Este tanulás után a

számítógéphez ülve nem informatikai feladatok megoldására koncentrálnak, hanem kikapcsolódásként használják az eszközt, tisztelet a fanatikus kivételeknek.

A megyei és alsóbb szintű versenyeken, ahol eleve csekély a létszám, nem nagyon lehet és érdemes több korcsoportot felállítani. Helyette a feladatok sorrendisége által is meghatározott nehézségi szintekkel, különdíjakkal lehet és kell serkenteni az ifjúságot. Pl.: programozói kategóriában a legjobb lány vagy bármely kategóriában a legfiatalabb versenyző díjazása.

A Nemes Tihamér versenyek I. korcsoportjában (csak programozói kategóriában) papír és számítógépes feladat egyaránt szerepel, a legmagasabb korcsoport pedig az OKTV versenyek része.

II.4. Kategória

Alkalmazói (I.) vagy programozói (II.)

A programozás kategória régóta versenyelem, de az informatikai eszközök és az érettségire való felkészítés előhívott egy másik területet, az alkalmazói ismeretek mérését.

Példaként az Informatika OKTV kezdetektől fogva két kategóriás volt (programozás, illetve alkalmazás), a Nemes Tihamér OKTV-n belül pedig a 2005/2006-os tanévben indul el a 9-10. osztályosok alkalmazói kategóriája.

Tehát sokkal többen versenyeznek, ha több kategória van? Sajnos nem. A csökkenő programozói létszámot próbálja ellensúlyozni az alkalmazói kategória. Itt elvárás a szokásos szöveg és táblázatkezelés és egyéb „irodai” ismeret. A gyerekek, akik általában középszintű érettségi szinten használják ezeket az szoftvereket, hiszik, hogy versenyszinten is megállják a helyüket. Ez nem így van. Nem elegendő ismerni a szoftvert, annak minden apró, sebességet növelő eljárásával, trükkjével, technikájával rendelkezni kell, hogy a feladatokat időben el lehessen végezni. Itt az idő a legnagyobb ellenfél. Mert minden feladatra rá lehet jönni, csak megfelelő idő kell hozzá, ami meg nincs. A maximális pontszám még tapasztalt tanárok számára is kihívást jelent az időkorlát miatt. Persze a gyerekek gyors észjárásukkal „előrébb” vannak, de kapkodnak, figyelmetlenek, a verseny láz sem tesz mindenkinek jót. Na és a sikerélmény. Egy 200 pontos feladatból 50-60 megszerzése nem dicsőség, de a középszintű érettségin mégsem lenne olyan rossz. A továbbjutáshoz szükséges 80 pontos határt már nagyon szűrt létszám éri el. 100-150 pont felett már jó, e felett kiválónak mondható a teljesítmény. A döntőben olyan kérdések is előkerülnek, amely mélyebb matematikai ismeret,

akár programozói ismeretet is igényel. Ilyen példa a lineáris programozási példát/ismeretet igénylő „solver” használata táblázatkezelőben.

Egy kicsit nézzük másképpen is. Az országos versenyek mellett a megyei és alacsonyabb szintű versenyeken is megjelent ez a kategória, egyrészt a „lányok” megmozdítása, másrészt a versenyző létszám megtartása miatt. Ugyanis 10-15 diák miatt nem érdemes, és költséges versenyt rendezni. Így segít a kategória a programozói kategória életben tartásában is.

Mivel a Nemes és az OKTV verseny szintismertetőjénél nem tettem meg, álljon itt az alkalmazói kategória részletesebb feladatkiírása:

Általános követelmények:

- képek, ábrák számítógépes előállítása, transzformálása,
- szövegszerkesztési ismeretek,
- táblázatkezelési ismeretek,
- adatbázis-kezelési ismeretek,
- prezentáció,
- honlap-készítési ismeretek.

Megoldandó fordulónként egyre nehezedő és nagyobb ismeret-területet lefedő 4-7 kisebb/közepes/nagyobb feladat számítógépen, a fent említett általános követelmények alapján.

II.5. Típus

Egyéni vagy csapatverseny.

Általában minden verseny egyéni, még a diákolimpiai verseny is, habár a résztvevőket országonként, mint csapat is lehet tekinteni.

Egy kivételt említenék, amelyik nagyon másra sikerült, mint a többi verseny. Ez a Dusza Árpád Emlékverseny, amely olyan általam nagyra tartott középiskolai verseny, ahol csapatban kell helytállni. Meg kell tanulni a projektmunkát, csapatmunkát, feladatmegosztást, együttműködést, speciális szoftverek, fejlesztőeszközök, segédprogramok, szoftverfejlesztési technikákat. Ezek azért nem egyszerűek, mert habár az iparban bevált és elfogadott módszerekkel dolgoznak, ez általában – lásd az IOI tanmenet – nem szerepel semmilyen verseny követelményei között. Ugyanígy a megfelelő dokumentáció-készítés sem. Ennek

előnyét abban látom, hogy később könnyebb lesz a diákoknak a valódi életben munkát kapni és eredményt elérni.

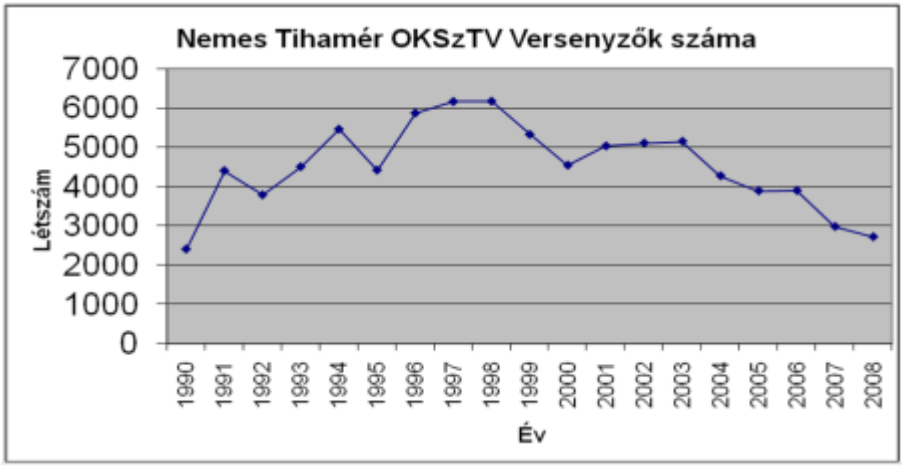
Mit oktassunk az előbb felsorolt rengeteg tudnivaló közül? Én azt tartom a legfontosabbnak, hogy gyakoroltassunk velük együtt előző évi hasonló feladatot, alakuljon valódi csapat, legyen vezető, ismerjék egymás tudását, erősségét, gyengéjét, tanuljanak meg bánni az idővel.

II.6. Fordulók

I.-II.-(III.)-(IV.)

Általánosságban elmondható, hogy a versenyek fordulói egyre nehezednek, egyre mélyebb tudás, tapasztalat, matematikai és informatikai, algoritmus vagy eszköz (alkalmazói) ismeretet igényelnek. A döntőkben nem ritka az egyetemi szintű ismeret követelményként való beszivárgása sem. A versenyek iskolai vagy városi szinten egy, megyei szinten kettő, országos szinten általában három, olimpiai válogatón 4-5 és az olimpián több fordulósak, hogy folyamatosan konvergáljon a mérés a valódi tudáshoz.

Habár országos szinten még nem nagyon jutalmazzák az I. fordulóban való részvételt, mégis fontos, mert itt kerül először tapasztalatszerzési szándékkal a versenyző éles helyzetbe. Ez általában döntési pont is, hogy akarja/tudja-e folytatni a következő fordulóban vagy a következő évben. És itt fontos megjegyezni, hogy a gyerekek egy része azonnal feladja, mert vagy a matematikai részével nincs tisztában és esélytelennek tartja magát, vagy nem tartja képesnek magát arra (=nem hisz benne), hogy a megfelelő tudást (amit nem adnak ingyen) elsajátítsa. Így nem is csoda, ha évről évre csökkenő versenyző létszámot mutatnak a statisztikák.



1. ábra Nemes OKSzTV versenyzok száma

Az persze nem baj, ha a mennyiség helyett a minőségre fektetjük a hangsúlyt és a nemzetközi szintekhez kapcsolódó feladatsorainkat nem adjuk alább (Nemes – OKTV – IOI), de meg kell teremteni a többi versenyen a fokozatos felzárkózás és utánpótlás lehetőségét.

Ha megnézzük például a függelék (VII.1) IOI tematikáját, akkor elszomorodhatunk, mivel a középiskolai matematika és informatika tanítás alapján esély sincs arra, hogy a gyerekek megoldják, vagy egyáltalán megértsék a feladatokat és a mögöttük rejlő algoritmusokat. Például 9-10-dik osztályban a gráfok ismerete matematikából még nem igény, habár az informatikai versenyeken nem lehet megélni nélküle. Fordulónként ráadásul a feladatok komplexitása természetes módon növekszik, így a logikai rész, ha nem épül alapvető tárgyi tudásra, akkor több idő alatt oldható meg, ha megoldható egyáltalán.

Tekintsük most meg fordulókon alkalmazott technikákat. Sok verseny első fordulója papír alapú és főleg a gondolkodást ellenőrzi. Ilyen a Nemes programozói kategória 9-10 korcsoportja, vagy az OKTV programozói kategóriája. Ez egyfelől jó, mert pl. a felismert tehetséges gyerekek, akiket 2 hónap alatt nem lehet algoritmizálásra és programozásra hatékonyan felkészíteni belépő évfolyamon, azok is kapnak esélyt. Ugyanakkor ellenállást vált ki azokból, akik már valamilyen kezdő szinten tudnak programozni, de a matematika vagy az algoritmizálás nem annyira kedvencük. Így aztán létrejött a mérés egy kevert formája, amit a Nemes programozói kategória 7-8 korcsoportjában ismertünk meg 1-2 éve. Papír és számítógépes feladat együttesen szerepel a verseny 1. fordulójában.

Végül tekintsük az értelemszerűen számítógépnél megoldandó feladatokat, mint a Nemes Alkalmazói, OKTV alkalmazói, és persze a programozói versenyek nem első fordulói. Itt a gyerekek számítógépnél adnak számot tudásukról. Nem elég bekapcsolni tudni a gépet, az adott feladatnak megfelelően alkalmazói és/vagy fejlesztői szoftver és nyelv alapszintű ismerete is szükséges.

II.7. Összefoglalás

Folyton változó világunkban nem elég ismerni, tudni is kell, hogy melyik verseny mit kínál, melyik gyermeket melyikre küldjük el, és ott siker vagy kudarc vár-e rá.

A függelékben találjuk a szubjektív összefoglaló táblázatomat (VII.2.1).

III. Versenyfeladatok elemzése

Mielőtt rátérünk egy-két valódi feladat elemzésére, tekintsük át azokat a problémákat, amelyekkel találkozunk versenytől függetlenül.

A gyerekek általában nem törekszenek egy-egy feladat megoldásánál arra, hogy bármi áron pontokat szerezzenek. Akik több fordulón és versenyen átestek már, azoknak egyre jobban érződik, hogy igazából pontvadászat folyik. Nem a csak a tökéletes megoldásra kell törekedni, hanem a legtöbb pont megszerzésére. Nem szép, teljes, hanem jó, vagy megfelelő, vagy éppen a részben megfelelő megoldások kell, hogy szülessenek, mivel a zsűri ezeket pontozza. Részmegoldásokat pontoznak, ami arra is lehetőséget ad, hogy egy-egy komoly problémát a legegyszerűbb technikával, kereséssel, rekurzióval, trükkös adattárolással, stb. is meg lehet oldani. Így a feladat elején kapunk pontot, de az összetettebb, sok memóriát, vagy nagy időigényű megoldásokat már lehet, hogy nem fogják értékelni.

A feladat bonyolultságából adódóan egyszerűbb megoldást nem csak akkor érdemes keresni, amikor nem jövünk rá az optimális algoritmusra, hanem akkor is, amikor már csak percek vannak az „utolsó" feladat megoldására. Az is fontos, hogy képesek legyünk a feladatokat rangsorolni magunkban. Nem arra gondolok, hogy a legelsőt utoljára, mert általában elől vannak a könnyebb, de fontos pontszerző feladatok. Hanem arra, hogy időkezelés (*time management*) legyen a versenyzők agyában. Nem szabad egy feladatnál sokáig elidőzni, ugyanakkor ugrálgatni sem célszerű. Általában ki kell jelölni egy adott időkeretet, ami alatt

- ismert a feladat, az algoritmus, ezért „a megoldást ismerem", „el tudom kezdeni a konkrét megvalósítást",
- nem ismert az algoritmus, gondolkodni kell, vagy ismert algoritmust nem ismert módon módosítani, kiegészíteni,
- teljes zsákutca az optimális megoldás keresése, ezért vissza kell lépni egy egyszerűbb megoldásra.
- feladom, egyáltalán nem tudok mit kezdeni a feladatot. Ekkor sem kell feladni, hanem a legvégére rakni, és ott az előző pontban megfogalmazni valamilyen megoldást.

Az időkeret 5-15 perc gyakorlással választható, embere válogatja. Egy feladathoz azonban a pontszámnak és a teljes időkeretnek megfelelően kiszámolhatjuk a készítők által optmiláisan elegendő megoldási időt. Pl.: 15 pontos feladat egy 75 összpontos, 3 órás feladatsornál 36 percnyi tervezett feladat-megoldási időt jelent.

A zsűri fejével is gondolkodni kell. Hogyan? Hát mit néznénk mi is meg, ha egy program/algoritmus helyességét bizonyítani szeretnénk? Mivel a versenyek nagy részén automatikus programjavítás van, így előre tudjuk a bemenetet, és a várt kimenettel lehet összehasonlítani. Anélkül, hogy a konkrét feladatot ismernénk, a feladatnak meg tudjuk határozni a határértékeit, speciális bemeneteit, amikre mindenképpen ki kell próbálni a beadandó alkalmazást. Így a 0, 1, kezdő, vég, egyenletes eloszlású, növekvő vagy csökkenően rendezett bemenet biztosan szerepel majd a próbákban. Ugyanígy véletlenszerűen generált adatokkal közepes és nagy elemszámmal is kipróbálják, hogy helyesen fut-e le, adott időn belül.

Ha már a zsűri fejével gondolkodunk, akkor nézzük meg azt is, hogy milyen célok vezérlik a zsűrit a feladat létrehozásakor. Mérni szeretné a tárgyi tudást és a gondolkodást, szinte mindig a kettőt ötvözve. Bár versenyenként eltér, hogy mi az elvárt tudás, mégis igaz, hogy

- objektíven mérhető,
- a tárgyi tudásra épülő,
- a tárgyi tudástól többet igénylés esetén azt megmagyarázó,
- a mindennapi életből vett minták alapján megfogalmazott,
- szöveges feladat

kitalálása a cél.

Az, hogy a feladat megoldásához ki, milyen segédeszközt használhat, versenyenként különbözik, de versenyenként rögzített. Az országos versenyek általában semmilyen eszközt nem engedélyeznek, de a számítógépen elérhető információforrás (*help*) elérhetőségét nem tiltják.

A feladat előállítójának ki kell dolgoznia a szerinte optimális megoldást, és az értékeléshez mintaadatokat is kell szolgáltatnia, azok eredményeivel együtt. E sokszor nem egyszerű feladat, mert a kevés adattól egészen a határértékekig kell generálni ezeket.

A megoldáshoz mellékelni kell rövid javítási útmutatót, aminek azonban egyértelműnek, objektívnek, érthetőnek, útmutatónak kell lennie. Ez nem egyezik meg a megoldás részletes ismertetésével, amely azonban már nem minden verseny követelménye, pedig a feladat kitalálója megtehetné.

Ezt a problémát meg is fogalmazzák a gyerekek: a megoldások alapos, részletes elemzése, a használt algoritmusok bemutatása, tanulás a hibákból, a megoldási útmutatótól eltérő, összegyűjtött ötletes megoldások közzététele. Ennek hiányában szélmalomharcnak tartják, hogy megértsék a feladat kidolgozók gondolkodásmódját – habár végre létezik IOI tanmenet, mint támpont.

A feladatoknak ráadásul le kell fedniük a számon kérendő tudás és gondolkodási képesség 50-70%-át a feladat sorszámának, és a fordulónak megfelelő nehézségi szinten. Nem kizárt, hogy ismert matematikai probléma átültetését kell informatikai eszközökkel megoldani, de nem mindig várható el a versenyzőktől, hogy a matematika nyelvét is olyan jól használják és értik, mint pl. egy programozási nyelvet. Ez sokszor bonyolulttá és félreérthetővé teszi számukra a feladatot.

Az IOI tematika nemcsak a versenyzők, hanem a feladat kitalálók részére is útmutatás, tehát azonos anyagból dolgoznak. Amíg a diák tanulja az algoritmusokat, a verseny kitalálója pedig annak tudatában és komplexitásának függvényében írja le az adott feladatot, alkalmazva egy-két ismert, eredeti, vagy módosított algoritmust.

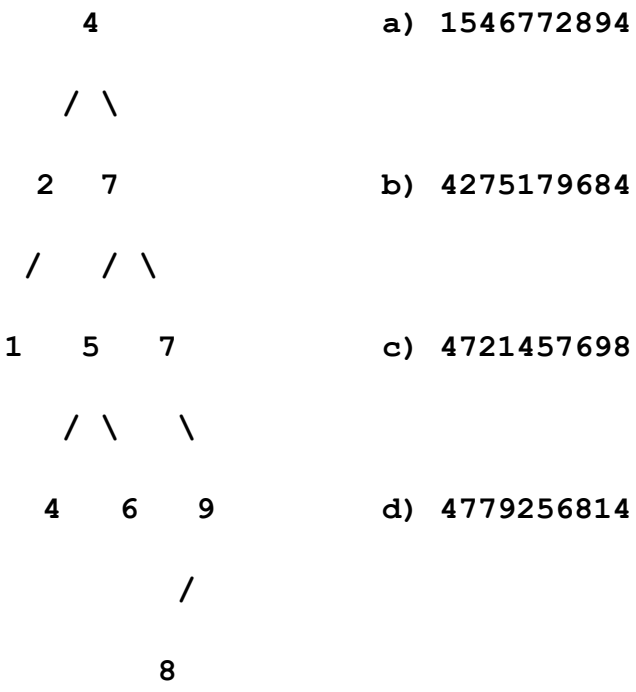
Erre a kérdésre részben a regionális és a központi NJSZT szakkörök jelenthetnek gyógyírt, de nem biztos, hogy a régi feladatok elemzésében képesek a feladat kitalálóinak gondolatmenetét és optimális megoldásmenetet vázolni a diákoknak. Szakköri füzetek formájában eljut a követelmény elsajátításához szükséges információ a diákokhoz. A „jó” tanítás sokkal több annál, mint egy előre elkészített anyag „leadása”. Stabil algoritmuselmélettel, tudással, gyakorlattal kell hitelesnek maradnia a tanárnak ahhoz, hogy képes legyen a gyengébb képességűeket magasabb szintre emelni, és a haladókat is ellátni új, érdekes, kihívásokkal teli feladatokkal.

Én az algoritmusok elemzésekor és a megoldások kifejlesztésekor a következő irodalmat ajánlom az olvasónak: (12)

III.1. Megyei programozói

III.1.1.1998 programozói I. forduló 5. feladat

Az alábbi ábra számok egy listája alapján készült: a soron következő számot valamilyen általános elv szerint helyeztük egy új leágazás végére. Mely számsor vagy számsorok alapján készülhetett az ábra? Milyen fa keletkezik a `3, 5, 7` számok további elhelyezése után.



Amire itt rá kell jönni, hogy milyen fa ez. Ha a nevét nem is tudjuk, de tehetünk megállapításokat:

- két felé ágazik maximum (bináris)
- egy csúcsban és az alatta levő csúcsokban levő számokat ha összehasonlítjuk, akkor észrevehetjük, hogy
    - balra alatta a csúcsnál kisebb számok
    - jobbra alatta a csúcsnál nagyobb vagy egyenlő számok vannak

Ha tehát ismerjük, hogy mi a szisztéma, akkor könnyen meg is oldhatjuk a feladatot. Először ellenőrizzük, hogy az a-b-c-d számsorokból felépítő-e a gráf helyesen.

a) esetben az 1 még elhelyezhető, de az 5 már csak jobbra alá, ami miatt meg kell állni, nem a rajznak megfelelő ábrát épít ez számsor
b) és d) esetben felépíthető, csak más sorrendben a fa
c) 4721-ig rendben is volnánk, de a következő 4-es a nem létező 5-ös alá kellene kerüljön

Most tekintsük a feladat másik részét. Ha folytatjuk a faépítést a 3-5-7 számokkal, többféle lehetőség közül választhatunk. A 3-ast elhelyezhetjük a 2-estől jobbra, a szabad 4-estől balra (és esetleg, mivel az általunk felállított egyetlen szabályt sem szegünk, a 6-ostól és a 8-astól

balra is). Ezután az 5-ös és a 7-es lerakása is hasonlóan, vagy balra egy nagyobb csúcs alá, vagy jobbra egy kisebb alá.

Általánosítsuk a feladatot, mit kell ebből megtanulni? Nem árt a gráfok ismerete, de a feladat megoldásához aki gondolkodik, annak a rajz is elég. Persze a számokat érdemes ismerni ☺ A számok összefüggéseinek ismerete, a gráf-tulajdonságok (pl. bináris), és a több megoldás mind-mind új fogalom vagy technika lehet első versenyes diákoknak. Ezért az a feladatunk, hogy olyan példákat mutassunk nekik a felkészítéskor, amelyek után nem ijednek meg a hasonló esetek láttán.

III.1.2.1998. programozói II. forduló

Készítsen programot, amely a rádióadók vételi viszonyaiban segít tájékozódni: adott földrajzi helyhez megadja azt, hogy az egyes adások milyen frekvenciákon, milyen erősen foghatók. A vétel erőssége egyenesen arányos a sugárzás energiájával, és fordítottan arányos a távolság négyzetével. A szükséges információt szöveges állományokban találja:

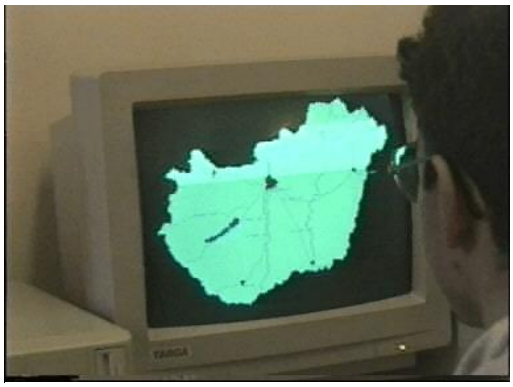
- *A radioado.txt a rádióadók neveit, földrajzi (grafikus) koordinátáit, a sugárzás energiáját és az adások frekvenciáit tartalmazza.*
- *A magyaro.txt, dunatisz.txt, balaton.txt, varosok.txt, utak.txt vonalábrákat tartalmaznak. Minden ilyen ábra kezdőkoordinátákból és iránykódok sorozatából áll: a továbblépés nyolc lehetséges irányát a koordinátageometriában szokásos sorrendben (jobbra, jobbra-fel, ..., le-jobbra) az a, ... , h betűk mint iránykódok jelölik.*
- *A varosxy.txt egyes városok centrumainak a koordinátáit tartalmazzák.*

A koordináták a grafikus képernyők koordinátakezelésének felelnek meg. A feladatban szereplő ábrák koordinátatartománya: vízszintesen 1-383, függőlegesen (felülről lefelé) 1-242.

Ha valaki a karakteres képernyőn oldja meg a feladatot, akkor adott koordinátákhoz vagy városokhoz számítsa ki és jelenítse meg a vételi viszonyokat.

Aki a grafikus képernyőt választja, az jelenítse meg Magyarországot, a zárt alakzatokat színezze ki, szemléltesse az adókat és a sugárzás energiáját, továbbá az utakon végighaladva folyamatosan jelenítse meg az adások vételi viszonyait. Az értékeket (frekvencia, erősség) logaritmikusan jelenítse meg.

A függelékben (VII.3) megtaláljuk a szövegfájlok tartalmát, hogy jobban értsük, milyen bemeneti adatokkal kell működjön a kifejlesztendő alkalmazás.



2. ábra - munka közben

A megoldáshoz sokfajta tudásra és ötletre lehet szükség. Ilyen a nyelvi és a grafikus képernyő használatának ismeret, de teljes értékű megoldásnak számít a karakterekkel való zsonglőrködés. A jobb oldali képen látszik, hogy volt olyan versenyző, aki nem fél a grafikus rendszertől és használta a feladat során. A megjelenítés kecsegtetőnek látszik, de még nem látszik a jelerősség és a mozgás megvalósítása.

A feladatban mégsem a megjelenítés a lényeg, hanem az, hogy az egyes pontokban hogyan számítjuk ki az energiát, valamint hogyan illesztünk útra a megjelenítéskor.

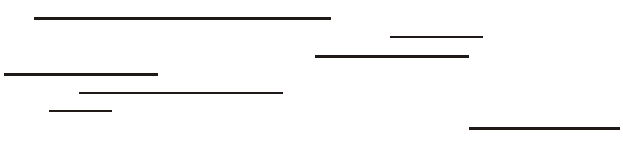
III.2. OKTV programozói

III.2.1.2009. II. forduló 4. feladat - Staféta

Az olimpiai lángot egy kiindulási városból a cél városba kell eljuttatni. A két város távolsága ***K*** kilométer. Sok futó jelentkezett, mindegyikről tudjuk, hogy hányadik kilométertől hányadik kilométerig vállalja a futást. Ha egy futó az ***x*** kilométertől az ***y*** kilométerig vállalja a futást, akkor minden olyan futó át tudja venni tőle a lángot, aki olyan ***z*** kilométertől vállalja a futást, hogy $x \le z \le y$.

Írj programot (`stafeta.pas`, …), amely kiszámítja, hogy legkevesebb hány futó kell ahhoz, hogy a láng eljusson a cél városig.

A `stafeta.be` szöveges állomány első sorában két egész szám található: a két város távolsága $(10 \le K \le 1000)$ és a jelentkezett futók száma $(2 \le N \le 20000)$. A további ***N*** sor mindegyikében két egész szám van ***I*** és ***E*** $(0 \le I < E \le K)$ ami azt jelenti, hogy egy futó az ***I***-edik kilométertől az ***E***-edik kilométerig vállalja a láng továbbítását. A futókat a sorszámukkal azonosítjuk, a bemenet ***j+1***-edik sorában a ***j***-edik futó adata van. Feltételezhetjük, hogy a láng eljuttatható a cél városig a jelentkezett futókkal.



3. ábra - intervallumok

A `stafeta.ki` szöveges állomány első sorába a láng célba juttatásához minimálisan szükséges futók ***M*** számát kell írni. A második sor pontosan ***M*** számot tartalmazzon (egy-egy szóközzel elválasztva), azon futók sorszámait, akik teljesítik a feladatot. Tehát a felsorolásban a ***j***-edik futó a ***j+1***-edik futónak adja át a lángot. Több megoldás esetén bármelyik megadható.

Példa:

`stafeta.be`	`stafeta.ki`

`40 7`
`2 21`
`25 35`
`20 34`
`0 10`
`5 18`
`3 7`
`34 40`
	`4`
	`4 1 3 7`

A feladat megoldásához először azt vegyük figyelembe, hogy pl. csak minimális előképzettséggel rendelkezünk. (Gondolok itt egymásba ágyazott ciklusok és/vagy rekurzió ismeretére). De tekintsük messzebbről a feladatot. Nagyban segít az N és K nagyságrendjének ismerete, amit megadnak. Eddigi tapasztalatok alapján a feladatok megoldhatók egy egyszerű 16 bites Turbo Pascal környezetben, így a rendelkezésre álló egyszerű adatszerkezet használatában gondolkodhatunk. Tehát ne gondoljunk pl. 1000x20000-es táblázatra, mert az létre sem hozható e nyelvi környezetben. (a 64kB-os szegmenskorlát miatt).

A másik, hogy általában egy 3-4-szeresen egymásba ágyazott N-ig vagy K-ig futó ciklikus próbálgatás végrehajtási ideje negyedik, ill. harmadik hatványú is lehet. Megnézve az elmúlt évek javítási útmutatóit, láthatjuk, hogy kevés közepes és nagy értékekre is kipróbálják az algoritmusunkat.

Értékelés:

Kisméretű bemenet, egyértelmű megoldás	*1+1 pont*
Kételemű megoldás	*1+1 pont*
Sok átfedő intervallum	*1+1 pont*
Sok egymásba skatulyázott intervallum	*1+1 pont*
Közepes véletlen teszt	*1+1 pont*
Nagy véletlen teszt	*1+1 pont*
Nagy véletlen teszt	*1+2 pont*

Tehát ha egy egyszerű algoritmust adunk, aminek nagy a futási ideje, akkor a kevés számú tesztadattal még sikeresek lehetünk, de a komplex, nagy tesztadatnál nem fut le mérhető

időben az algoritmusunk. (Ez nem baj, sőt, a pontvadászat miatt jobb a valamilyen, mint a semmilyen megoldás!)

Mi lehet egy ilyen algoritmus? Ennél a feladatnál egy minden lehetőséget végignéző visszalépéses vagy összes esetet figyelembe vevő technika is megfelelő lehet, az előbb említett kompromisszummal. Mennyi ekkor a lépésszám? $O(N^2)$ Más problémába nem futhatunk bele a lépésszám nagysága mellett? Dehogynem, hogy hogyan választjuk meg azt az adatszerkezetet, amiben gyűjtjük és számoljuk a megoldásokat és az odavezető utat.

Éppen ezért nem ezt a megoldás mutatom meg, hanem egy, szerintem optimális megoldást mohó algoritmus alkalmazásával.

A mohó algoritmus, vagy inkább stratégia, mint a dinamikus programozás egyik módosítása a következő lépésekből áll:

III.2.1.1. A feladat lépésekre bontása

A mi esetünkben sorba vesszük a futókat, és eldöntjük, hogy ha az érkezése óta még nem volt korábbi stafétaátadás és ekkor ezt feljegyezzük.

III.2.1.2. „Mohó választás

Helyi minimumra törekedve válasszuk mindig a legkorábban „belépő” futót.

III.2.1.3. A megoldás

A mohó algoritmus alkalmazásakor felmerül, hogy eleve egy hatékony és elégséges tárolási módot válasszunk. Ilyen a „`legkorabbi`” tömb, amely maximum `K` elemű és tartalma egy adott futónál mi a legkorábbi kezdés. A mohó stratégia alapján azonban szükségtelen minden elemnél az aktuális kezdő értéket tárolni, csak az érdekes, legkorábbi időpontra vagyunk kíváncsiak egy-egy befejezésnél. Így a tömbben csak ott lesz értékünk, ahol befejezéshez tartozik „legkorábbi” kezdés. Szükségünk van a kiíratás miatt a futó indexek mind bemeneti, mind kimenetű gyűjtéséhez, ezt a másik két tömb valósítja meg.

A működés pedig nem más, mint az inicializálás, a módosított beolvasás, a stratégia alapján az értékek „lineáris” megkeresése és végül a minimális érték és az ezt megvalósító futószám kiíratása.

```
 1: program stafeta;
 2: var f : text; {be-kiementi szövegfájl}
 3:     K,N,I,E,M: integer; {beolvasandó és kiírandó értékek}
 4:
 5:     { itt tároljuk az adott int. véghez tartozó legkorábbi kezdéseket }
 6:     legkorabbi : array[0..1000] of integer;
 7:     { itt gyąjtjük, hogy a legkorábbi kezdés melyik futónál fordult elő= index }
 8:     legkorabbi_futo : array[0..1000] of integer;
 9:     { a futók indexe a kimenethez, M-el indexeljük}
10:     futosorszam : array[0..1001] of integer;
11:
12:     x: integer; {ciklusváltozó}
13:     utolso : integer; {futócsere utolsó helye}
14: begin
15:   {inicializálás}
16:   for x:=1 to 1000 do
17:   begin
18:     legkorabbi[x]:=-1;
19:     legkorabbi_futo[x]:=-1;
20:   end;
21:
22:   {beolvasás fájlból}
23:   assign(f,'stafeta.be');
24:   reset(f);
25:   readln(f,K,N); {első sor - határértékek: K,N}
26:
27:   for x:= 1 to N do
28:   begin
29:     readln(f,I,E); {aktuális kezdő és végérték beolvasása}
30:
31:     {csak azokkal az elemekkel foglalkozunk, amelyekkel még nem számoltunk,
32:      vagy pedig a beolvasott intervallum kezdet korábbi az eddigieknél}
33:     if (legkorabbi[E]=-1) or (legkorabbi[E]>I) then begin
34:       legkorabbi[E]:=I;      { E. végpontban az új kezdet.}
35:       legkorabbi_futo[E]:=x; { tároljuk el, hogy az adott végpont melyik futóhoz tartozik}
36:     end;
37:   end;
38:   close(f);
39:
40:   {számolás, feltételezve, hogy nem rendezettek a legkorábbi tömb elemei}
41:   M:=0;    {eddig ennyi eredményt találtunk}
42:   utolso:=0; {itt jártunk utoljára, ahol futócsere volt}
43:
44:   while utolso<K do  {amig nem az utolsó elemnél vagyunk, keressünk}
45:   begin
46:     for x:=K downto 1 do {minden elemen menjünk végig}
47:     begin
48:       {ha van vége, és az az utolsónak kezeltnél korábbi, dolgozzuk fel}
49:       if (legkorabbi[x]<=utolso)and(legkorabbi[x]>=0) then
50:       begin
51:         utolso:=x; {legközelebb már csak idáig nézzük, megvan az új utolsó}
52:         M:=M+1; {plusz 1 futó kell}
53:         futosorszam[M]:=legkorabbi_futo[x]; {és megjegyezzük az indexét}
54:         break;
55:       end;
56:     end;
57:   end;
58:
59:   {eredmény kiírás}
60:   assign(f,'stafeta.ki');
61:   rewrite(f);
62:   writeln(f,M); {elsď sor, a legkevesebb futó száma.}
63:
64:   for x:=1 to M do
65:   begin
66:     write(f,futosorszam[x]); {a futók sorszáma}
67:     if x<M then write(f,' ');
68:   end;
69:   writeln(f); {sor bezására}
70:   close(f);
71: end.
```

Felmerül a kérdés, hogy lenne-e más megoldás? Igen. Ha például eleve a befejező kilométer szerint rendezve kezelnénk az adatokat (rendezéssel, beszúró tömb feltöltéssel, stb.). Ekkor a „számoló” rész legrosszabb $O(N^2)$ lépésszáma „vándorol át” a rendezésbe, és maga a kiválasztó algoritmus *O(N)*-nel becsülhető lesz, bár egy gyorsrendezésnek csak a legrosszabb esete ennyi, általában ennyél jóval „szebb” az eredmény.

III.3. OKTV alkalmazói

III.3.1.2009.III.forduló 5. feladat

Táblázatkezelés: Regionális forduló[1]

A Neumann János Számítógép-tudományi Társaság által szervezett Nemes Tihamér verseny néhány, egymáshoz viszonylag közel eső iskolájának legjobban szereplő versenyzői négy helyen írhatják meg a regionális forduló dolgozatait. Az A, a B és a C vizsgahelyen 15 fő számára van lehetőség, míg a D vizsgahely 20 versenyző részére tud férőhelyet biztosítani. A `tovabb.txt` *szövegfájl az iskolák nevét, az onnan továbbjutott tanulók számát tartalmazza, továbbá azt is megmutatja, hogy hány Ft-ot tesz ki egy-egy versenyző utazási költsége rendre az A, a B, a C ill. a D vizsgahelyre. Egy-egy iskola versenyzői együtt, ugyanarra a vizsgahelyre utaznak.*

Készítsd el az `NJSZT` *munkafüzet* `beosztás` *munkalapján a regionális forduló beosztását, azaz határozd meg, hogy melyik iskola versenyzői melyik vizsgahelyen oldják meg a regionális forduló feladatait, ha az utazási költség minimalizálása a cél! Természetesen tetszőleges számú segédcellát fölhasználhatsz, de valami olyan elrendezést várunk, mint amit az alábbi* `regford.jpg` *képen látsz. Világosan tűnjön ki, hogy az egyes iskolák számára melyik vizsgahelyet jelölöd ki (az ábrán szereplő beosztás természetesen csak illusztráció), továbbá az is, hogy mennyi lesz összesen a minimális utazási költség:*

[1] ez a feladat neve, nem a fordulóra utal

	Versenyzők száma az iskolából						Vizsga-hely:
Andrássy Gyula Gimnázium	5						C
Arany János Szakközépiskola	7						C
Deák Ferenc Gimnázium	7						A
Hajós Alfréd Gimnázium	7						D
Illyés Gyula Szakközépiskola	3						B
Jékely Zoltán Gimnázium	3						B
Kodály Zoltán Szakközépiskola	3						C
Leőwey Klára Szakközépiskola	9						C
Madách Imre Szakközépiskola	2						A
Ond Vezér Gimnázium	5						A
Rákóczi Ferenc Gimnázium	6						D
Szent István Gimnázium	5						D
Zrínyi Miklós Szakközépiskola	3						B
Összes költség:							

4. ábra - feladat

A nagy számítási igényre tekintettel most természetesen nem várjuk el, hogy a kiindulási adatok bármely változása esetén azonnal helyes eredményt kapj[2]*, de a táblázatkezelő eszközeinek segítségével gondoskodj azokról a feltételekről, hogy megoldható esetben, kellő idő és kellő számítási kapacitás birtokában megoldásra jussunk!*

Értékelés:

A könnyebb érthetőség kedvéért az értékeléshez az alábbi elrendezés alapján azonosítjuk az `NJSZT` *munkalap egyes részeit:*

[2] Már csak azért sem, mert a létszámadatok függvényében kialakulhat eleve megoldhatatlan feladat, pl. ha egyazon iskolából húsznál több, vagy egy kivételével minden iskolából páros számú versenyző jut tovább.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1			A	B	C	D		A	B	C	D				A	B	C	D		
2			vizsgahely					vizsgahely							vizsgahely					
3			15	15	15	20									15	15	15	20		
4			férőhely												férőhely					
5		Versenyzők száma az iskolából	Útiköltség Ft/fő												Versenyzők száma					Vizsga-hely:
6	Andrássy Gyula Gimnázium	5	140	200	130	150		0	0	0	1		1		0	0	0	5		D
7	Arany János Szakközépiskola	7	190	370	330	260		1	0	0	0		1		7	0	0	0		A
8	Deák Ferenc Gimnázium	7	370	250	260	120		0	0	0	1		1		0	0	0	7		D
9	Hajós Alfréd Gimnázium	7	260	370	100	360		0	0	1	0		1		0	0	7	0		C
10	Illyés Gyula Szakközépiskola	3	120	240	310	240		1	0	0	0		1		3	0	0	0		A
11	Jékely Zoltán Gimnázium	3	220	120	320	210		1	0	0	0		1		3	0	0	0		A
12	Kodály Zoltán Szakközépiskola	3	370	240	390	160		0	0	0	1		1		0	0	0	3		D
13	Leőwey Klára Szakközépiskola	9	230	230	260	310		0	1	0	0		1		0	9	0	0		B
14	Madách Imre Szakközépiskola	2	120	320	250	370		1	0	0	0		1		2	0	0	0		A
15	Ond Vezér Gimnázium	5	220	330	130	330		0	0	1	0		1		0	0	5	0		C
16	Rákóczi Ferenc Gimnázium	6	130	330	280	180		0	1	0	0		1		0	6	0	0		B
17	Szent István Gimnázium	5	180	350	200	250		0	0	0	1		1		0	0	0	5		D
18	Zrínyi Miklós Szakközépiskola	3	110	160	120	130		0	0	1	0		1		0	0	3	0		C
19																				
20			Összesen:												15	15	15	20		
21																				
22	Összes költség:	11 670 Ft																		

5. ábra - megoldás

- *A H6:H18 tartomány egy cellájába akkor és csak akkor írunk 1-est, ha az adott iskolát az adott vizsgahelyre osztjuk be, egyébként 0-t.*
- *H6:H18 függvényében a sorösszeg (M6:M20) mindenkor 1, hiszen minden iskolát pontosan egy vizsgahelyre kell beosztanunk.*
- *Mivel a versenyzők összlétszáma és az összes férőhelyek száma megegyezik, minden vizsgahelyet ki kell használnunk. Az egyes iskolákból az egyes vizsgahelyekre jutó versenyzők számát H6:H18 függvényében O6:R18 adja meg, az egyes vizsgaközpontokra jutó versenyzőszám (oszlopösszeg) O20:R20-ban, míg az egyes vizsgaközpontok befogadóképessége O3:R3-ban található.*
- *H6:H18 függvényében adódik minden iskola sorában, T6:T18-ban a számára kijelölt vizsgahely neve is.*
- *H6:H18 függvényében adódik B22-ben a fizetendő költség is.*

Feladatunk tehát: töltsük ki a H6:H18 tartomány celláit 0-kkal és 1-esekkel úgy, hogy a B22 cella értéke minimális legyen, miközben a feltételek mindegyikét betartjuk.

A. Van `NJSZT` *munkafüzet, benne* `beosztás` *munkalap;*
ezen helyesen szerepel az összes importált adat *1+1 pont*

B. A versenyzők létszámát (O6:R18) az iskola-vizsgahely párosítás (H6:K18) függvényében megadó képlet legalább egy vizsgahelyre helyes,

*pl. az A vizsgahelyre pl. O6=KEREKÍTÉS($B6*H6;0);*
az összes vizsgahelyre helyes *1+1 pont*

C. Az iskola útiköltségét az iskola-vizsgahely párosítás (H6:K18) függvényében megadó képlet legalább egy (tényleges) iskola-vizsgahely párosra helyes;
minden iskola-vizsgahely párosra helyes;
a végösszegre is helyes
(ha ez utóbbit közvetlen képlettel számolja, akkor a C részre 3 pontot kap, pl. B22=SZORZATÖSSZEG(C6:F18;O6:R18)) *1+1+1 pont*

D. Az egyes vizsgahelyekre jutó versenyzőszámot mindenütt helyesen adja meg, a példában O20:R20-ban, pl. O20=SZUM(O6:O18) *1 pont*

E. Az iskola-vizsgahely párosításhoz a Solverben minimalizálandó célcellaként az utazási összköltség celláját adja meg, a példában B22-t *1 pont*

F. Az iskola-vizsgahely párosításhoz a Solverben módosuló cellákként az iskolához vizsgahelyet rendelő tartományt adja meg, a példában H6:K18-at *1 pont*

G. Korlátozó feltételként ügyel arra, hogy egyazon iskola versenyzői ugyanazon helyre utaznak, tehát a példában H6:K18-ban csak bináris értékeket enged meg;
soronként pontosan 1 db 1-essel, a példában M6:M18=1 *1+1 pont*

H. Korlátozó feltételként ügyel arra, hogy minden vizsgahelyre éppen annyi versenyző kerüljön, mint az ottani férőhelyek száma, a példában O3:R3=O20:R20 *1 pont*

I. A Solver minden paraméterét helyesen adta meg, helyes a minimális összköltség értéke, a példában B22-ben *3 pont*

J. Legalább egy helyen jó az adott iskola vizsgahelyét megadó függvény;
mindenütt jó, legfeljebb számábrázolási-kerekítési problémák vannak;
számábrázolási-kerekítési problémák sincsenek *1+2+1 pont*

Ez a feladat több dologra is rávilágít. Nem véletlen, hogy az alkalmazói kategória első helyezettje szintén indulhat a válogatóversenyeken, mert tudásban hasonló szinten kell állnia a programozókkal. Itt tárgyi tudásként a solver használata, és értelmezése, mint lineáris

programozási feladat jön elő. Aki életében nem csinált ilyet, nem is fogja tudni nagy valószínűséggel megcsinálni, mert nincs idő rá a versenyen, hogy találgasson. Már a bekapcsolása is érdekes a táblázatkezelő alkalmazásban, mivel külső komponensről van szó. Ugyanakkor pár pontot lehet szerezni az A-B, esetleg C-D értékelési szempontnál.

III.4. Dusza verseny

III.4.1.2009. döntő

Processzor

Az Intal cég új processzort kíván gyártani. A processzor fantázianeve: Intal Pentimum.

A tervek elkészültek, de mielőtt a sorozatgyártásba kezdenének, meg akarnak győződni arról, hogy a Pentimum tervezett utasításkészlete megfelel a felhasználói igényeknek.

A Pentimumot egy olyan alaplapra integrálták, amelynek egy IO-felülete a hozzá csatlakoztatott $400[3] byte kapacitású memória, egy outputja pedig egy képernyő illesztési felület. A képernyő felbontása 10x10 karakter, a képernyő csak szöveges információ megjelenítésére alkalmas.

A csapat feladata, hogy szoftveres támogatást nyújtson a processzor funkcionalitásának és architektúrájának[4] a kipróbálásához, teszteléséhez.

Feladatok:

- ***fordítóprogram készítése:***
  A program bemenete egy szövegfile, amely a Pentimum utasításkészletén implementált[5] programot ír le assembly[6] nyelven.
  A fordítóprogram ezt az assembly programot fordítja le gépi kódra a processzor megadott utasításkészletének megfelelően.
  A program kimenete a gépi kódot tartalmazó szövegfájl.

- ***emulátor készítése:***

[3] A $400 hexadecimális adat (decimálisan: 1024)
[4] **Architektúra:** terv, koncepció
[5] **Implementál:** megvalósít
[6] Az **assembly nyelv** a gépi kód szimbolikus megfelelője, azért jött létre, hogy könnyebben lehessen programozni gépi kód szinten. Gépközeli nyelvnek is szokták nevezni.

Ennek a programnak a segítségével szoftveresen emulálható[7] a processzor működése. Az emulátor bemenete egy byte-kódú[8] (gépi kódú) program. Az emulátor PC-n futtatható, és a PC képernyőjén jeleníti meg a memória és a processzor állapotát.

- ***mintaprogram készítése:***

  A cég mintaprogramot is kíván adni a processzor dokumentációja mellé, ezért egy kitűzött feladatot is meg kell oldani a processzor utasításkészletével.

  A mintaprogram egy olyan assembly kód, amelyet a processzor utasításkészletével kell megírni, a fordítóprogrammal le kell fordítani, majd az így kapott gépi kódú programot kell az emulátoron futtatni.

A Pentimum és architektúrája

A memória mérete $400 byte. Az első $64 byte a 10x10 karaktert megjelenítő képernyő számára fenntartott memória. Minden egyes byte egy cellát jelöl a képernyőn, az ott lévő értékek pedig – a megjelenítés szempontjából – ASCII kódolásúak. A memóriahelyek és a 10x10-es képernyő kapcsolata:

$00	*$01*	*$02*	*$03*	*$04*	*$05*	*$06*	*$07*	*$08*	*$09*
$A	*$B*	*$C*	*$D*						*$13*
$14	*$15*	*$16*							*$1D*
$1E	*$1F*								*$27*
$28									*$31*
$32									*$3B*
$3C									*$45*
$46									*$4F*
$50									*$59*

[7] **Emulál**: utánoz (Az emuláció lényege, hogy az emulált környezethez készült szoftverek és adatok feldolgozását lehetővé teszi az attól eltérő jellemzőkkel rendelkező környezetben is.)

[8] **Byte-kód**: Utasítássorozat, amelyben minden utasítás egy vagy több egymást követő byte-ban található. Az egyes byte-okban szereplő adatokat leggyakrabban kettes vagy tizenhatos számrendszerben írják le.

$5A	*$5B*	*$5C*	*$5D*	*$5E*	*$5F*	*$60*	*$61*	*$62*	*$63*

A memória a $64. byte-tól kezdve az $3FF. byte-ig tetszőlegesen használható. A képernyő mind a $64 byte-ja kezdetben a $20-as byte-tal (szóköz) van feltöltve.

Az emulátorba töltött program byte-jai a memória $64. byte-jától kerülnek be, és a program futtatása is onnan kezdődik.

A Pentimum utasításkészlete nagyon egyszerű. A processzornak nincsenek regiszterei, a műveleteket mindig 8 bites byte-okon végzi.

Az utasításkészlet

A címzés hexadecimálisan történik.

Assembly utasítás	***Leírás***	***Byte-kód***
MOV [ABC] [DEF]	*A DEF memóriahelyen lévő 8 bites érték átmásolása az ABC memóriahelyre.*	***00 0A BC 0D EF*** *(5 byte)*
pl. MOV [A1] [113]	*A $113-as (decimálisan 275-ös) memóriahelyen lévő byte az A1-es (decimálisan 161-es) címre kerül.*	*00 00 A1 01 13*
MOV [ABC] $XY	*Az $XY byte bemásolása az ABC memóriahelyre.*	***01 0A BC XY*** *(4 byte)*
pl. MOV [A] $48	*A 'H' karakter hexakódja az A (decimálisan 10-es) memóriacímre kerül.*	*01 00 0A 48*
ADD [ABC] [DEF]	*Az ABC és DEF memóriahelyen lévő két 8 bites szám összeadása. A 8 bites eredmény az ABC*	***02 0A BC 0D EF*** *(5 byte)*

	memóriahelyre kerül.	
ADD [ABC] $XY	*Az ABC memóriahelyen lévő 8 bites érték ill. az $XY érték összeadása. A 8 bites eredmény az ABC memóriahelyre kerül.*	***03 0A BC XY*** *(4 byte)*
SUB [ABC] [DEF]	*Az ABC memóriahelyen lévő 8 bites számból a DEF memóriahelyen 8 bites szám kivonása. A 8 bites eredmény az ABC memóriahelyre kerül.*	***04 0A BC 0D EF*** *(5 byte)*
SUB [ABC] $XY	*Az ABC memóriahelyen lévő 8 bites értékből az $XY érték kivonása. A 8 bites eredmény az ABC memóriahelyre kerül.*	***05 0A BC XY*** *(4 byte)*
JMP ABC	*A következő utasítás végrehajtása az ABC címtől folytatódik.*	***06 0A BC*** *(3 byte)*
JZ ABC	*A következő utasítás végrehajtása az ABC címtől folytatódik, ha az utolsó MOV, ADD vagy SUB utasítás eredménye 0 volt, egyébként tovább folytatódik a végrehajtás.*	***07 0A BC*** *(3 byte)*
JC ABC	*A következő utasítás végrehajtása az ABC címtől folytatódik, ha az utolsó ADD vagy SUB utasítás eredmény e túl- ill.*	***08 0A BC*** *(3 byte)*

	alulcsordult, egyébként tovább folytatódik a végrehajtás.	
WAIT	*A processzor 100ms-ig várakozik*	***09*** *(1 byte)*
END	*A processzor megáll. (A program véget ér.)*	***0A*** *(1 byte)*

A WAIT utasításon kívüli összes utasítás végrehajtási ideje elhanyagolható (0ms) a WAIT utasítás 100ms-os végrehajtási idejéhez képest.

Az assembly utasítások paraméterei között egy-egy szóköz van. Az utasítások azonosítására szolgáló paraméter minden betűje nagybetű.

Az utasításkészlet táblázatában a byte-kódok egyes byte-jait szóköz választja el. Az emulátor program bemeneteként egy olyan szövegfájlt kell készíteni, amelyben az egyes utasítások összetartozó byte-jai, és az egymást követő utasítások szomszédos byte-jai elválasztójel nélkül követik egymást.

A fordítóprogram

A fordítóprogram bemeneti állománya (egy assembly-forráskódú program) soronként egy-egy utasítást tartalmaz a fenti Pentimum utasításkészlet alapján. A forráskódban lehetőség nyílik kommentezésre: amennyiben egy sor a # karakterrel kezdődik, azt a fordítóprogram figyelmen kívül hagyja.

A fordítóprogram által készített gépi-kódú program byte-kódja szöveges állományba kerül kiírásra úgy, hogy az egymást követő byte-ok között ***semmilyen elválasztójel*** *nincsen. Minden egyes byte-ot hexadecimálisan kell kiírni.*

(Javaslat: A fordítóprogram az alapértelmezett kimenet mellett hozzon létre egy másik kimenő fájlt is. Ebben az egyes utasításokhoz tartozó byte-kódok kerüljenek külön sorba, és az egyes byte-ok a soron belül egy-egy szóközzel legyenek elválasztva. Az utasítások byte-kódjainak elválasztása a fordítóprogram ellenőrzését teszi könnyebbé.)

Ha a fordítóprogram fordítás közben hibát talál, meg kell állnia, és hibaüzenetet kell megjelenítenie.

A lehetséges hibák:

- *szintaktikai hiba az assembly programban (az assembly utasítás nem a megadott szerkezetű),*
- *túl hosszú a program (nem fér be a memóriába).*
- *ha olyan memóriacímre történik hivatkozás, ami nem létezik.*

A hibára vonatkozó figyelmeztetést és a hibás sor sorszámát a képernyőre kell kiíratni. Pl. „Szintaktikai hiba a 15. sorban.", vagy „A program nem fér be a memóriába. Hiba a 40. sorban." (Itt az assembly program sorának száma decimális adatként jelenjen meg!)

A fordítóprogram leállása esetén a kimeneti állomány tartalmának nincs jelentősége. (Nem szükséges azt elkészíteni.)

Az emulátor működése

Az emulátor az elindításakor kér egy byte-kódot (szöveges bemeneti állományt), ennek az adataival azonnal fel is tölti a Pentimum memóriájának a gépi kódú program számára rendelkezésre álló részét.

Az emulátor működése ezután az alábbi parancsokkal vezérelhető:

- ***„L fájlnév" (Load):** az emulátor betölti a megadott filenévhez tartozó file-ban lévő programot, és inicializálja magát;*
- ***„R" (Run):** az emulátor megállás nélkül folytatja a program futtatását;*
  - *a futtatás tetszőleges billentyű lenyomásával félbeszakítható, az emulátor a továbbiakban az említett utasításokkal vezérelhető;*
- ***„S" (Step):** az emulátor végrehajtja a soron következő utasítást;*
- ***„M 123" (Memory):** a megadott memórián lévő értéket megjeleníti. A memóriacímet hexadecimálisan kell megadni.*
- ***„Q" (Quit):** kilépés az emulátorból.*

Az emulátor futtatás közben három ok miatt állhat le:

- *END utasításhoz ér,*
- *lefagy*
  - *ha az emulátor a memória végéhez ér,*
  - *ha az emulátor olyan utasítást kísérel meg végrehajtani, ami nincs az utasításkészletben,*
- *a futtatását felhasználói közreműködéssel félbeszakítják.*

Az R (Run) és S (Step) utasítások csak akkor hajthatók végre, ha a processzor nem fagyott le, vagy az emulátor program nem ért el egy END utasítást. Leállás után az emulátor csak az L (Load) utasítással hozható alaphelyzetbe.

Az emulátornak külön kérés nélkül mutatnia kell a következő információkat:

- *az aktuális 10x10-es képernyő tartalmát,*
- *a végrehajtás alatt álló vagy az éppen végrehajtott utasítás byte-kódját és memóriacímét.*

Feltételezhetjük, hogy a bemeneti byte-kód nem tartalmaz szintaktikai hibát.

Példa

Assembly forráskód:

példa (Hello World):

```
# Print „HELLO”
MOV [0] $48
MOV [1] $45
MOV [2] $4C
MOV [3] $4C
MOV [4] $4F
# Print „WORLD”
MOV [A] $57
MOV [B] $4F
MOV [C] $52
MOV [D] $4C
MOV [E] $44
END
# End of program
```

A lefordított szöveges állományú gépi byte-kód:

A teszteléshez:

```
01 00 00 48
01 00 01 45
01 00 02 4C
01 00 03 4C
01 00 04 4F
01 00 0A 57
01 00 0B 4F
01 00 0C 52
01 00 0D 4C
01 00 0E 44
```

0A

Az emulátor bemenetként megkapja az egy sorban megjelenített byte-kódot:

```
0100004801000145010002 4C0100034C0100044F01000A5701000B4F01000C
5201000D4C01000E440A
```

A felhasználóbarát felületen az emulátor vezérlési lehetőségeinek rövid leírása is megjelenik.

Az emulátor felhasználói felületét tekintve nem elvárás a grafikus megjelenítés. Az emulátor programban a karakteres és a grafikus megjelenítés azonos értékűnek számít.

A mintaprogram

A processzor mellé adott assembly-mintaprogramnak a következőt kell megvalósítania.

A képernyő 0. sorában a „Dusza” feliratot kell képújságszerűen görgetni. A képernyő 0. sora először üres, majd minden eltelt 500ms után megjelenik egy újabb karakter úgy, hogy az eddig megjelent szöveg egy karakternyit balra mozdul. A következő táblázat mutatja a 0. sor tartalmát az idő függvényében:

Idő/oszlop	*0.*	*1.*	*2.*	*3.*	*4.*	*5.*	*6.*	*7.*	*8.*	*9.*
0ms										
500ms										*D*
1000ms									*D*	*u*
1500ms								*D*	*u*	*s*
2000ms							*D*	*u*	*s*	*z*
2500ms						*D*	*u*	*s*	*z*	*a*
3000ms					*D*	*u*	*s*	*z*	*a*	
3500ms				*D*	*u*	*s*	*z*	*a*		
4000ms			*D*	*u*	*s*	*z*	*a*			
4500ms		*D*	*u*	*s*	*z*	*a*				
5000ms	*D*	*u*	*s*	*z*	*a*					
5500ms	*u*	*s*	*z*	*a*						
6000ms	*s*	*z*	*a*							
6500ms	*z*	*a*								
7000ms	*a*									
7500ms										

A 7500. ms egyben a képújság újraindulása is: a 8000. ms-ban ugyanaz folytatódik, mint ami az 500. ms-ban elkezdődött.

A képújság görgetése közben a képernyő (a Pentimum képernyőjének) bal alsó sarkában jelenjen meg egy másodperc számláló óra is, amely a program elindítása után eltelt másodperceket számolja. (Ez az óra nem az emulátornak, hanem az elkészítendő assembly-programnak a része.)

Az elkészített assembly-forráskódnak fordíthatónak kell lennie a fordítóval, majd az elkészült byte-kódot az emulátornak képesnek kell lennie futtatni úgy, hogy a PC-képernyőjén nyomon követhető legyen a megírt assembly-program működése (látszódjon a képújság és az óra is).

Beadandó:

- *A fordítóprogram forráskódja és a lefordított állomány*
- *Az emulátor forráskódja és a lefordított állomány*
- *A fejlesztői dokumentáció (A megadott sablon alapján elektronikusan kell elkészíteni.)*
- *A mintaprogram assembly kódja (Elektronikusan kell elkészíteni.)*

A bemutatáshoz külön szemléltető anyag készítése (pl. PowerPoint prezentáció) nem kötelező, de ha készül ilyen, azt is be kell adni!

A munka szóbeli bemutatása:

Szempontok, ajánlott vázlat:

- *A feladat előkészítésének bemutatása, a feladatok szétosztásának elvei*
- *Az elkészített fordítóprogram és emulátor bemutatása*
  - *a felhasználó számára*

    A programok használatát kell bemutatni a megadott tesztfájlok használatával.

    (A programok írásakor a teszteléshez használt assembly kódok és byte-kódok mellett a zsűri által a bemutatón rendelkezésre bocsátott további tesztekkel is le kell futtatni a fordítót és az emulátort.)
  - *a fejlesztő számára*

    A programok szerkezetét kell bemutatni.
- *a mintaprogram bemutatása*

A csapatverseny döntő feladata egyrészt egy elég komplex, de azért mégis érthető és elvégezhető feladat 3 főnek való elosztott megoldását tűzte ki célul úgy, hogy közben a szoftverfejlesztésben is szükséges elvárások irányába is elmozdul.

A dokumentumkészítés (terv és leírások), a programbemutatás mind ezt szolgálja. A gyerekek bizonyítékot szolgáltattak arra, hogy tudnak, ha nem is 100%-ban, de nagyrészt együtt

dolgozni, automatikusan szétosztódtak a szerepek, és nem csak azért mert három, jól elválló részfeladatot kaptak, hanem mert logikusan is szét tudták osztani a munkaelemeket.

Az idővel is versenyeztek, aki hamarabb kész volt, az besegített a másiknak tesztelésben, dokumentálásban, stb.

Nagyon hangos verseny, a csapatok (belül) a verseny hevében szinte egymást túlkiabálva igazoltak egy-egy állítást, ellenőriztek teszteseteket, algoritmusokat, a programok működését.

Természetesen van olyan csapat, ahol a domináns szereplő köré csak holdudvarként jelent meg 1-2 további versenyző, de az eredményben ez meg is látszott, a feladat szinte kivitelezhetetlen 1 főnek.

Nagyon hasznos, hogy 8-13 korosztály is van, ahol a kicsik a nagyokkal együtt dolgozva lesik el a fortélyokat, technikákat.

IV. Összefoglalás

Akkor értem el a célomat e dolgozattal, ha sikerült a versenyek „útvesztőjében” iránymutatást adni, egy picit segíteni a választást, mert minden versenyen mindenkit elindítani igen fáradtságos feladat a felkészítő tanárnak és a diáknak egyaránt.

Az, hogy melyik verseny, milyen stílusú, inkább algoritmizáló vagy inkább alkalmazás specifikus, egyéni vagy csapat nem biztos, hogy elhanyagolható szempont, amikor a diák és/vagy a felkészítő tanár választ, javasol. A diák tudását igazítani kell ugyan a követelményekhez, de egy adott pillanatban olyan tudáshalmazzal rendelkeznek, amihez kell választani. (Pl. 9. évfolyam év eleje, közepe)

Az, hogy ennyi fajta verseny van, és még az említetteken kívül jó pár létezik, azt példázza, hogy a gyerekek igenis bevonhatók – bár egyre nehezebben, – az egymással való összemérésbe. Az NJSZT nem véletlenül tűzte ki céljául, hogy egy újabb versenytípust is szeretne bevezetni, alkalmazás-fejlesztő kategóriában, mert látja, hogy az eddig egysíkúan, „programozóként” mért tudás az idő előre haladásával komplexebb, más területeket is, vagy csak más területeket lefedő mérést kíván.

Végül sok sikert kívánok minden hozzám hasonló (a pedagógus pályán kezdőnek nevezhető, 3-5 év tapasztalattal rendelkező) tanár kollégámnak, aki magánszorgalomból, társadalmi munkában, vagy valami csekély ellentételezésért végez tehetséggondozás, felkészítést.

Konfucius:
„tanulni csak gondolkodva érdemes,
gondolkodni tanulás nélkül veszélyes…”

V. Köszönetnyilvánítás

Szeretném köszönetemet kifejezni témavezetőmnek, *Dr. Papp Zoltán Lajosnak*, aki mindig türelemmel és segítőkészen igazította munkámat, feszített időrendje ellenére.

Nem csak témavezetőként, hanem versenyeztető, feladat-kitaláló, zsűriző tanárként, oktatóként, példaképként, később már tanár kollégaként öntötte belém azt a hitet, hogy igenis kell a gyermekeket serkenteni a versengésre, örülni az eredményeknek, még a jelenlegi, nem természettudomány-barát középiskolai oktatási rendszerünk mellett is.

Végül *Családomnak* is köszönöm a türelmet, akik ismételten elviselték dolgozatírás alatti magánzárkámat, amíg e munka megszületett.

VI. Irodalom és hivatkozásjegyzék

1. **Abonyi-Tóth, Andor, Szlávi, Péter és Zsakó, László.** Az informatika oktatás téveszméi. [Online] 2009.. http://people.inf.elte.hu/szlavi/InfoOkt/Eloadasok/Teveszmek.pdf.

2. Borsod-Abaúj-Zemplén megyei középiskolai számítástechnika verseny. [Online] 2009. http://www.kossuth-saujhely.sulinet.hu/Magyar/Szvers/index.htm.

3. Logo Országos SZámítástechnikai Tanulmányi Verseny. [Online] NJSzT. http://logo.inf.elte.hu.

4. Nemes Tihamér Országos Középiskolai Tanulmányi Verseny . [Online] http://nemes.inf.elte.hu.

5. Országos Középiskolai Tanulmányi Verseny. [Online] OKM. http://www.okm.gov.hu.

6. Informatikai Diákolimpiák Válogató Versenye. [Online] http://valogatok.inf.elte.hu.

7. Dusza Árpád Emlékverseny. [Online] http://www.isze.hu.

8. Közép-Európai Informatikai Diákolimpia . [Online] http://ceoi.inf.elte.hu.

9. Nemzetközi Informatikai Diákolimpia . [Online] http://ioi.inf.elte.hu.

10. Nemzetközi Informatikai Diákolimpia. [Online] http://tehetseg.inf.elte.hu/ioi/index.html.

11. Neumann János Számítógép-tudományi Társaság. [Online] http://www.njszt.hu.

12. **Cormen, Thomas H.** *Új algoritmusok.* 2003. 978 963 9193 90 1 .

VII. Függelék

VII.1. IOI tanmenet

VII.1.1. Célja

Elsősorban iránymutatást adni arra nézve, hogy milyen feladatok alkalmasak a Nemzetközi Diákolimpia versenyre. E dokumentum alapján értékelik a beadott javaslatokat, hogy melyikből legyen valódi feladat.

A másik nem kevésbé kisebb cél, hogy segítse a szervezőket és tanárokat a diákok felkészítésében. Útbaigazítást ad a matematikai és számítástechnikai témák és fogalmak rendszerében, melyek milyen szinten szükségesek a sikeres diákolimpiai részvételhez.

A tanmenet meghatározza, hogy mely fogalmaknak, témakörök jelenhetnek meg a feladatokban és/vagy a megoldásokban.

Öt kategóriát állítottak fel:

♥	Korlátlanul felhasználható, mert alapértelmezett tudást jelent (pl. egész szám)
∆	Tisztában kell lenni a témával, de a feladatban további iránymutatásokat, specializációt kell alkalmazni.
⊖	Feladat leírásra nem használható, de a megoldásban, modell-alkotásban segíthet
Nem szükséges	Sem a feladatokban, sem a megoldásban jelen pillanatban nincs szükség
Kizárt	Nem téma

VII.1.2. Matematika

VII.1.2.1. Aritmetika és geometria

♥ Egész számok, műveletek (beleértve a hatványozást), összehasonlítás
♥ Tulajdonságait egészek (pozitív, negatív, páros, páratlan, oszthatóság, prímek)
♥ Törtek, százalékok
♥ Pont, vektor, Descartes-koordináták (a 2D-s egész négyzetrácson)
∆ Euklideszi távolság, Pitagorasz-tétel
♥ A vonalszakasz, keresztezés tulajdonságai
∆ Szögek
♥ Háromszög, négyzet, négyzet, kör
♥ Poligon (vertex, oldalsó / szélső, egyszerű, domború, belső / külső) terület,
Kizárt: Valós és komplex számok, általános kúpszeletek (parabolák, hiperbolák, ellipszis), trigonometriai függvények

VII.1.2.2. Diszkrét struktúrák (DS)

DS1. Funkciók, kapcsolatok, és halmazok

△ Függvények (szürjekciók, injekciók, inverzek, összetétel)
△ Relációk (reflexivitás, szimmetria, tranzitivitás, egyenértékűség relációk, teljes / lineáris rendezési relációk, lexikográfiai sorrend)
♥ Halmazok (Venn-diagramok, komplemens, Descartes-termékek, energia készletek)
△ Galambdúc elv
Kizárt: Számosság és megszámlálhatóság

DS2. Alapvető logika

♥ Propozicionális logika
♥ Logikai összefüggések (beleértve az alapvető tulajdonságokat)
♥ Igazságtáblák
♥ Elsőrendű logika
♥ Univerzális és egzisztenciális számszerűsítés
⊖ Modus ponens és modus tollens
Nem szükséges: Érvényesség, normál formák
Kizárt: Predikátum logika korlátai

DS3. Bizonyítási technikák

△ Fogalmak: implikáció, ellentét, inverz, kontrapozíció, tagadás, ellentmondás
⊖ Közvetlen bizonyítások, bizonyítás ellenpéldával, kontrapozícióval, ellentmondással
⊖ Matematikai indukció
⊖ Teljes indukció
♥ Rekurzív matematikai fogalmak (beleértve a kölcsönösen rekurzív definícióját
Nem szükséges: Formális bizonyítás struktúrája
Kizárt: Jól-rendezettség

DS4.A számolás alapjai

♥ Számolási paraméterek (összeg és a szorzási szabály, a befogadás-kizárás elve, számtani és geometriai progresszió, Fibonacci-számok)
△ galambdúc elv (korlátok eléréséhez)
△ permutációk és kombinációk (alap definíciók)
△ Faktoriális függvény, binomiális együttható
⊖ Pascal azonosságai, Binomiális tétel
Kizárt: Ismétlődő kapcsolatok megoldása

DS5. Gráfok és fák

△ Fák (és az alapvető tulajdonságaik)
△ Irányítatlan gráfok (fok, útvonal, kör, összekapcsoltság, Euler / Hamilton út/kör, kézfogás lemma)
△ Irányított gráfok (bejövő/kimenő fokok, irányított út/kör, Euler / Hamilton út/kör)
△ Feszítőfák
△ Bejárási stratégiák (csomópont sorrend meghatározása rendezett fáknál)
△ Színezett fák szél vagy csomópont címkékkel, súlyok, színek
△ Többszörös-gráfok, gráfok hurok-éllel
Nem szükséges: Síkba rajzolható gráf, Páros gráfok, hipergráfok

DS6. Diszkrét valószínűség
Kizárt

VII.1.2.3. Egyéb matematikai területek

Nem szükséges: Polinomok, mátrixok és műveleteik, 3 dimenziós geometria
Kizárt: (lineáris) algebra, analízis, valószínűség-számítás, statisztika

VII.1.3. Számítástechnika

VII.1.3.1. A programozás alapjai (PF)

PF1. Alapvető programozási szerkezetek (absztrakt gépek)
♥ Egy magas szintű nyelv alapvető szintaxisa és szemantikája
♥ Változók, típusok, kifejezések, és a hozzárendelések
♥ Egyszerű I/O
♥ Feltételes és iteratív ellenőrzési struktúrák
♥ Funkciók és paraméterátadás
⊖ Strukturált alábontás (dekompozíció)

PF2. Algoritmusok és probléma-megoldás
⊖ Probléma-megoldási stratégiák (értelmezés-tervezés-munkavégzés-ellenőrzés, vonatkozóság elkülönítése, általánosítás, specializáció, eset megkülönböztetés, visszafelé dolgozás)
⊖ Az algoritmusok szerepe a probléma-megoldási folyamatban
⊖ Algoritmus implementációs stratégiák
⊖ Nyomkövetési stratégiák
△ Az algoritmusok fogalma és tulajdonságai (korrektség, hatékonyság)

PF3. Alapvető adatstruktúrák
♥ Egyszerű típusok (logikai, előjeles / előjel nélküli egész szám, karakter)
♥ Tömbök (beleértve a többdimenziós tömböket)
♥ Rekordok
♥ Szövegek és szövegfeldolgozás
△ Statikus és verem helyfoglalás (elemi automatikus memória-kezelés)
△ Mutatók (lineáris és elágazó)
△ Mutatók statikus memóriakezelési stratégiái
△ Verem és sor megvalósítási stratégiái
△ Gráfok és fák megvalósítási stratégiái
△ A helyes adatszerkezet megválasztásának stratégiái
△ Absztrakt adattípusok, prioritás sor, dinamikus halmaz, dinamikus térkép
Nem szükséges: Adatreprezentáció a memóriában, Kupac-foglalás, futásidejű memória-kezelés, mutatók és hivatkozások
Kizárt: lebegőpontos számok, hasító táblák megvalósítási stratégiái

PF4.Rekurzió

♥ A rekurzió fogalma
♥ Rekurzív matematikai függvények
♥ Egyszerű rekurzív eljárások
⊖ oszd meg és uralkodj stratégia
⊖ Rekurzív visszalépés

Nem szükséges: rekurzió megvalósítás

PF5. Eseményvezérelt programozás

Nem szükséges

VII.1.3.2. Algoritmusok és komplexitása (AL)

AL1. Alapvető algoritmus elemzés

∆ Algoritmus specifikáció, előfeltételek, utófeltételek, helyesség, invariánsok
⊖ aszimptotikus komplexitás elemzés (nem hivatalosan ha lehetséges)
⊖ Nagy O jelölés (ordo)
⊖ Standard összetettség osztályok (állandó, logaritmikus, lineáris, *O (N log N)*, négyzetes, köbös, exponenciális)
⊖ az idő és a tér egymással ellentétes hatása az algoritmusokban

Nem szükséges: A legjobb, az átlagos és a legrosszabb esetek viselkedése közti különbségek azonosítása, kis o, Omega, és théta jelölés, empirikus teljesítménymérés

Kizárt: Átlagos komplexitás határainak aszimptotikus elemzése, rekurzív algoritmusok elemzése ismétléses relációkkal

AL2. Algoritmus stratégiák

⊖ Egyszerű ciklus tervezési stratégia
⊖ „Nyers erő” algoritmus (részletes keresés)
⊖ Mohó algoritmusok (helyesség megértése) -
⊖ oszd meg és uralkodj, (hatékonyság megértése)
⊖ Visszalépés (rekurzív és nem rekurzív)
⊖ Elágazás és ugrás (helyesség és hatékonyság megértése)
⊖ Mintaillesztés és szöveg-kezelő algoritmusok (hatékonyság és helyesség megértése)
⊖ Dinamikus programozás
⊖ Diszkrét közelítő algoritmusok

Kizárt: Heurisztikák numerikus közelítési algoritmusok

AL3. Alapvető számító algoritmusok

⊖ Egyszerű numerikus algoritmusok, egészek bevonásával (Euklideszi algoritmus, radix átalakítás, prímteszt O(√N) próba osztás, Eratoszthenészi szita, faktorizáció, hatékony hatványozás)
⊖ Egyszerű műveletek tetszőleges pontosságú egészek(összeadás, kivonás, egyszerű szorzás)
⊖ Egyszerű tömb manipuláció (feltöltés, eltolás, forgatás, megfordítás, átméretezés, minimum / maximum, előtag összegek, hisztogram, vödörrendezés)
⊖ Szekvenciális feldolgozás, szekvenciális és bináris keresési algoritmusok
⊖ Keresés eliminációval, "lejtőn" keresés
⊖ Másodfokú rendezési algoritmusok (kiválasztás, beillesztés)

- ⊖ Partícionálás, statisztikák rendezése ismételt partícionálással, Gyorsrendezés
- ⊖ O(N log N) legrosszabb eset, rendezési algoritmusok (kupacrendezés, összefésülő rendezés)
- ⊖ Bináris kupac adatstruktúra
- ⊖ bináris keresőfák
- ⊖ Fenwick fák
- ⊖ Gráf reprezentációk (szomszédsági lista szomszédsági mátrix)
- ⊖ Gráfok mélységi és szélességi bejárása, összefüggőség megállapítása
- ⊖ Legrövidebb út algoritmusok (Dijkstra, Bellman-Ford, Floyd-Warshall)
- ⊖ Tranzitív lezárás (Floyd-algoritmus)
- ⊖ Minimális feszítőfa (Prim Jarnık és Kruskal algoritmusok)
- ⊖ Topológiai rendezés
- ⊖ Euler út/kör létezést kereső algoritmusok

Nem szükséges: Hasító táblák (beleértve az ütközés elkerülésére vonatkozó stratégiákat)

Kizárt: Egyszerű numerikus algoritmusok lebegőpontos aritmetika bevonásával, „Maximum flow" algoritmus, Párosság egyeztető algoritmusok, erősen összefüggő komponensek határozottan irányított gráfokban

AL4. Elosztott algoritmusok

Kizárt

AL5. Alap számításelmélet

Nem szükséges: Véges állapot-gép, környezetfüggetlen nyelvtanok

Kizárt: Engedelmes és makacs problémák, nem-kiszámítható funkciók, a megállás problémája, a nem kiszámíthatóság következményei

AL6 P és NP osztályok Összetettsége

Kizárt

AL7. Automaták és nyelvtanok

Kizárt: Véges automata, Reguláris kifejezések

AL8. Fejlett algoritmus elemzés

- ⊖ Kombinatorikus játékelméletlapjai,Minimax algoritmus optimális játék-játszáshoz

Nem szükséges: On-line és off-line algoritmusok, Kombinatorikus optimalizálás

Kizárt: Amortizációs elemzés, véletlenített algoritmusok, alfa-béta vágás, Sprague-Grundy elmélete

AL9. Kriptográfiai algoritmusok

Kizárt

AL10. Geometriai algoritmusok (a 2D-s hálózatok, azaz az egész (x, y)-koordináták)

- ⊖ Line szegmens tulajdonságok, elágazások
- ⊖ Pont helye, egyszerű sokszög
- ⊖ Konvex test megállapítás algoritmusok
- ⊖ „line sweeping" módszer

AL11. Párhuzamos algoritmusok

Kizárt

VII.1.3.3. Egyéb számítástechnikai területek

Az alábbi területek mind
Kizárt.

AR. Architektúra és szervezet
Digitális rendszerek, assembly, utasítás, csővezetékelés, gyorsítótár memóriák.

OS. Operációs rendszerek
Operációs rendszerek tervezése, beleértve: ütemezés, memória-kezelés, biztonság, fájl-rendszerek, a valós idejű és a beágyazott rendszerek, a hibatűrés, stb.

NC. Hálózat-központú számítástechnika

PL. Programozási Nyelvek
Programozási nyelvek elemzése és tervezése, osztályozás, virtuális gépek, fordítás, objektum-orientáltság, funkcionális programozás, típus-rendszerek, szemantika, nyelvi és tervezési jelek. A magas szintű programozási nyelvek alapjai.

HC. Ember-gép kapcsolat
Felhasználói felületek tervezése, grafikus eszközökkel

GV. Grafika és vizuális számítástechnika

IS. Intelligens Rendszerek

IM. Információ-kezelés

SP. Társadalmi és szakmai kérdések

CN. Számítástudományi területek

VII.1.4. Software tervezés (Software Engineering, SE)

SE1. Szoftvertervezés
⊖ alapvető tervezési koncepciók és alapelvek
⊖ Tervezési minták
⊖ Strukturált tervezés
Nem szükséges: Szoftverarchitektúra, újrahasznosításra tervezés
Kizárt: Objektum-orientált analízis és tervezés, Komponens-szintű tervezés

SE2. API-k használata
⊖ API (Application Programming Interface) programozás
Nem szükséges: Példa alapú programozás, Hibakeresés az API környezetben
Kizárt: Osztály böngészők és a kapcsolódó eszközök, Bevezetés a komponens alapú programozásba

SE3. Szoftver eszközök és környezetek
⊖ programozó környezetek, beleértve az IDE (Integrated Development En
Nem szükséges: Tesztelési eszközök, Konfiguráció-menedzsment eszközök
Kizárt: Követelmény elemzést és tervezést modellező eszközök, Eszközök integrációs mechanizmusok

SE4. Szoftvertervezési folyamatok
⊖ Szoftvertervezési életciklus-és folyamat-modellek
Kizárt: Folyamat értékelési modellek, Szoftvertervezési metrikák

SE5. Szoftver követelmények és specifikáció
⊖ Funkcionális és nem funkcionális követelmények
⊖ A formális specifikáció technikáinak alapfogalmai
Nem szükséges: prototipizálás
Kizárt: Követelmények összegyűjtése, követelmény elemzés modellezési technikái

SE6. Szoftver validálás
⊖ A tesztelés alapjai, ideértve a teszt terv létrehozását és teszt eset generálást
⊖ Fekete-doboz és fehér-doboz tesztelés technikák
⊖ Egység, az integrációs, validációs, és a rendszer szintű tesztelés
⊖ Vizsgálatok
Nem szükséges: Validálás tervezés
Kizárt: Objektum-orientált tesztelés

SE7. Szoftver evolúció
Nem szükséges: Szoftver-karbantartási, karbantartható szoftver jellemzői, visszafejtés, Örökölt rendszerek, Szoftver újrafelhasználás

SE8. Szoftver projekt menedzsment
⊖ Project ütemezés (különösen időkezelés)
⊖ Kockázatelemzés
⊖ Szoftver konfiguráció-menedzsment jellemezze
Nem szükséges: Szoftver minőségbiztosítás
Kizárt: Csapatkezelés, Szoftver mérési és becslési technikák, projektkezelési eszközök

SE9. Komponens-alapú számítógépes
Kizárt

SE10. Formális módszerek
⊖ Formális módszerek fogalmai
Nem szükséges: Formális ellenőrzés
Kizárt: Formális specifikációs nyelvek Futtatható és nem futtatható specifikációk -

SE11. Szoftver megbízhatóság
Kizárt

SE12. Speciális rendszerek fejlesztése
Kizárt

VII.1.5. Számítógépes ismeretek

⊖ Az operációs rendszer és a számítógép alapvető szerkezetét ismerni és érteni kell. (processzor, memória, I/O). Szabványos, grafikus felhasználói felület, az operációs rendszer használata. A fájlkezelés hasznos (mappa létrehozása, másolás és mozgó fájlok).

Nem szükséges: Számológép,

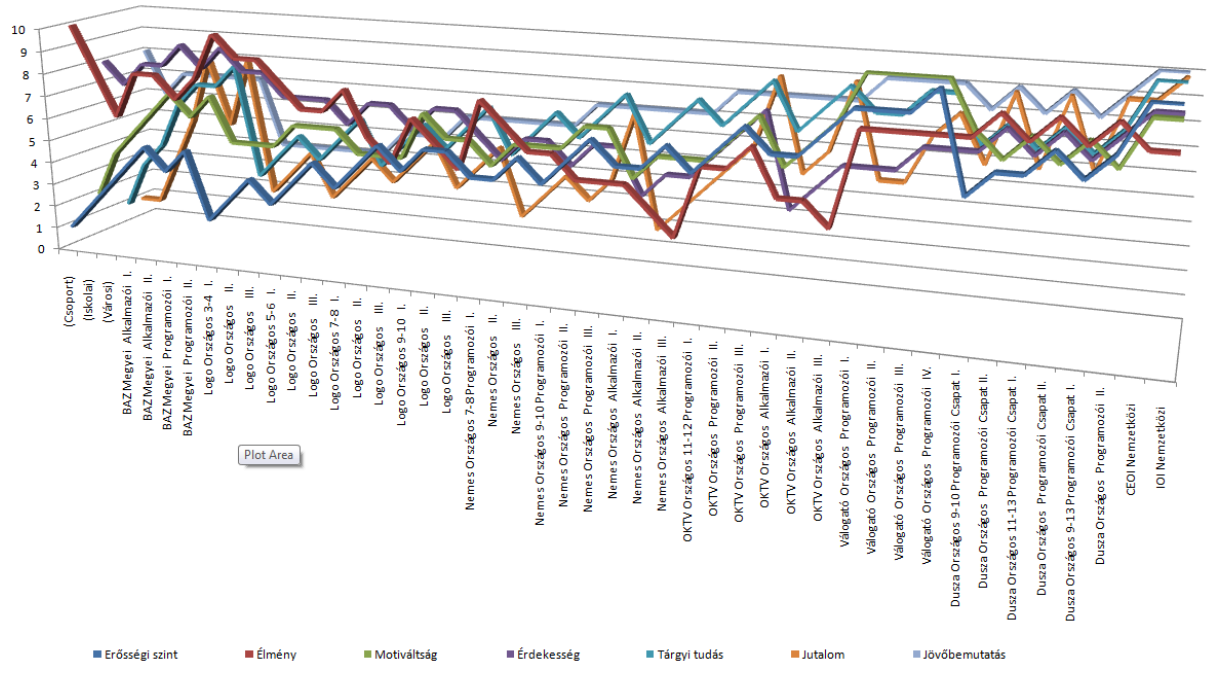
Kizárt: szövegszerkesztők, táblázatkezelő, adatbázis-kezelő rendszer, levelező kliens, grafikai segédprogramok (rajzolás, festés)

VII.2. Táblázatok

VII.2.1. Versenyek besorolása

Szint (terület)	Verseny neve	Korcsoport	Kategória	Típus (egyéni)	Forduló	Erősségi szint	Élmény	Motiváltság	Érdekesség	Tárgyi tudás	Jutalom	Jövőbemutatás
(Csoport)						1	10	2	8	1	1	8
(Iskolai)						2	8	4	7	3	1	6
(Városi)						3	6	5	8	4	3	7
Megyei	BAZ	-	Alkalmazói		I.	4	8	6	8	6	5	7
					II.	5	8	7	9	7	8	7
			Programozói		I.	4	7	6	8	7	5	7
					II.	5	8	7	9	8	8	7
Országos	Logo	3-4			I.	2	10	5	8	3	2	4
					II.	3	9	5	8	4	3	4
					III.	4	9	5	7	5	4	4
		5-6			I.	3	8	6	7	4	2	4
					II.	4	7	6	7	5	3	4
					III.	5	7	6	6	6	4	4
		7-8			I.	4	8	5	7	4	3	5
					II.	5	6	5	7	5	4	5
					III.	6	5	5	6	6	5	5
		9-10			I.	5	7	7	7	5	3	6
					II.	6	6	6	7	6	4	6
					III.	6	5	6	6	7	5	6
	Nemes	7-8	Programozói		I.	5	8	5	5	5	2	6
					II.	5	7	6	6	6	3	6
					III.	6	6	6	6	7	4	6
		9-10	Programozói		I.	5	6	6	5	6	3	7
					II.	6	5	7	6	7	4	7
					III.	7	5	7	6	8	7	7
			Alkalmazói		I.	6	5	5	4	6	2	7
					II.	6	4	6	5	7	3	7
					III.	7	3	6	5	8	4	7

Szint (terület)	Verseny neve	Korcsoport	Kategória	Típus (egyéni)	Forduló	Erősségi szint	Élmény	Motiváltság	Érdekesség	Tárgyi tudás	Jutalom	Jövőbemutatás
Országos	OKTV	11-12	Programozói		I.	6	6	6	6	7	5	8
					II.	7	6	7	7	8	6	8
					III.	8	7	8	8	9	9	8
			Alkalmazói		I.	7	5	6	4	7	5	8
					II.	7	5	7	5	8	6	8
					III.	8	4	8	6	9	9	8
	Válogató		Programozói		I.	9	8	10	6	8	5	9
					II.	9	8	10	6	8	5	9
					III.	9	8	10	7	9	7	9
					IV.	10	8	10	7	9	8	9
	Dusza	9-10	Programozói	Csapat	I.	6	8	8	7	7	6	8
				Csapat	II.	7	9	7	8	8	9	9
		11-13		Csapat	I.	7	8	8	7	7	6	8
				Csapat	II.	8	9	7	8	8	9	9
		9-13		Csapat	I.	7	8	8	7	7	6	8
					II.	8	9	7	8	8	9	9
Nemzetközi	CEOI					10	8	9	9	10	9	10
	IOI					10	8	9	9	10	10	10



6. ábra - grafikon

VII.3. Bemeneti adatok

VII.3.1. radioado.txt

Város	X	Y	kW	Kossuth	Petőfi	Bartók	Danubius	Hungária
Budapest	162	93	78.0	67.40	94.8	105.3	103.3	100.8
Csávoly	172	204	6.3		89.4			96.7
Debrecen	312	85	1.0		89.0	106.6	101.1	
Győr	91	76	7.5		93.1	106.8	101.4	87.6
Kabhegy	87	130	65.6	72.98	93.9	105.0	100.5	107.2
Kékes	221	62	30.2	71.21	102.7	90.7	104.7	95.5
Kiskőrös	180	168	1.7		95.1	105.9		88.4
Komádi	306	133	40.0	66.14	96.7	105.1	101.6	103.0
Miskolc	262	42	1.4	66.80	102.3	107.5	98.3	97.1
Nagykanizsa	49	181	12.0	71.03	94.3	104.7		90.2
Pécs	118	211	50.0	71.81	103.7	107.6	105.5	95.9
Sopron	27	76	10.0	72.86	99.5	107.9	102.0	96.8
Szeged	229	196	5.1		104.6	105.7	94.9	90.3
Szentes	236	165	32.4	66.29	98.8	107.3		100.4
Tokaj	297	38	50.0	71.33	92.7	105.5	103.5	97.5
Ózd	141	168	3.0		90.3	106.9		101.5
Vasvár	38	130	7.5		98.2	106.9		91.6

VII.3.2. balaton.txt

```
  66 158
aabahaaaaaaababbababaabaababababababcbaaaabaababbabcbbcddeee
ddcgggfegffefeffehhggfdddefffeefeffffeeeffeeeeefdeffffdedfgg
```

VII.3.3. dunatisz.txt

```
152 231
ccddddddddcccbaabcbcbbbbcbbaabcdcccccbccdcdcccccccbccbccdcdcb
cdcdcccbaaabbbcbcdcccccdddcdcccbcbcbcccdcccbcccccbbcdddccccbc
ccbcbbcbbcbbcbcccdccddcdccbcbccbcbcbcccccccccccdcddeeeeffffe
```

```
dcddeeeedefgfffeeeeeeeeeedeefeefeefeeedeeeeeeedeeeefdefefde
eeeeeedededdddeedccdededdcdcdddcdddeeeed
 226 208
bcbcbbcbababcccdccddcbdcccbcdcdccccddcbcbccbccccdcdeddccdedc
dccccbaabbabcdfedbbcbabaabaabddfeedccccccdcbbbaaabcccbbbaahb
aabbccdeccccddedccaaabcbbabccbabaabababcbbccbdbabcbaaaaabaabbb
ccccbabdcbbcbcdccbabcbccccbbhabccbhghaaaabaabbbccddcbcbbbaha
baaahaababbaabbbaabahbcacbbbababccbcbbbcbahahggfagfhgggggega
aghgfhhhhfgghahhhhaahbhbabbbaaahbbabaaaaa
 161 133
dddcdcccccdccbcbbccccccbcbccbbbabbcccc
 169  82
cdbcccbcdccdded
```

VII.3.4. magyaro.txt

```
129 242
abbbabaabbbccbbabcbahahahabbbabbahbaabbaabccccbaahghaaaaaabb
bbbhaabbbbbbbdbbbaaahbbaahhhhaahababababhaaahhbaaahabahhaabb
abaahaahabbcbahahhhhbbabbbbbcdcbbbaaaahaaaahbahhabbcbbaabccb
cccccbaahbccbccdedcbccbbccccbbahabcbccbaabcbcdcccccccbbabbbd
ccbabcbcccddbbbbcbabbcbcccbcbccbcbccccbcbcbaaabcbcccdcdccbcb
bbbbbaabccbccbcbbcbbbbhahabcbccbahahhaaabaabbccbaabcccbbabbb
bdddccbccccdcddeeeeefeffdeedccccddcdedcddfeeededdcccdcddedcc
dedefgfffedeeeeefeefefeffdeeedddcdcdcdcdcddfdedcddedefeeefee
efeeffeeffeddeeedeeeeeedeeeeeeeeefeeeeeeeefegggffgffgfgggfgf
gffgfgfeedffeeefgfffffeeffeffdcddededdedeededfeeffeffggggfgg
gfedeefdeefeefeeeeeeeedfeeedfeeeeeefefgfgfefgghfgggghhhhhfee
edefgfffeeeeeeeeeedeefeefeefeeedeeeeeeedeeeefdefefdeeeeeeed
ededdddeedccdededdcdcdddcdddeeeedfeeffefhfgggggfeefefhahhgf
ggggghgfeeefeeeedddffeeeddcdeeeddeefefgfeffeefghaahaahhaaaah
hggghgggffgggffefefeeeeefggfghhgggffgffgghghhahfgfghhfefhhg
fdfdeedefedffgfffffffgghbaaaaaahhggfgggghghghhhggfghhhhghhhg
ghghhahahhghhhhhaaaahhghggghhghhhahaahahhhahhgggghhhhhggghgh
hhhaaaaaaaahaaaahhhghgghaahhaahahghabbahabaahhaahbbaahabhaha
haaabah
```

VII.3.5. utak.txt

```
167  94       1-es ut
fegfeefeeeedddededeeeeefeeeeeeedddcddedddedeedeeeddedeedeeee
eeeeeeedeeeeeeffefefeeeeddddeddeddddeddddeddcdddeddd
 167  94       2-es ut
ccbcbccbcbcbcccccccccccdcddbdccbcccbbccdcdcdddedefed
 167  94       3-as ut
bbcbbbbabaaabaabaaaaaaabaabaababababaaaababababbbbaaaaahhaaa
aabaaaaabbbaaahhaaababababbbaabbbbbababbbcbccccdccdccdccbaaabb
cbccbcbbcbbcbbabbbcbccbaacdcccbcbbbcc
 167  94       4-es ut
aaaahhahaahghahhahahhhhhhahahhhhahahaahaahahaaaaaaaaaaahaaaa
ahhhaaaababbaaaaaaabababcbaaaaaabaaaaaaaaaaabababbabbabaaaabba
baababababbbbabbababbababababccbccccccccccccbcccccbcccccccccc
bccbcbabbbcbcbaababbccbbcbcccbcccbbccbcbcccbcbcc
 167  94       5-os ut
ahghhhghghghhghahhhhhhhhhghghghghhhghhghhhgghghhghghhghghhgh
hhghhgggggghghghghghghghgggghghgghghghgghgghghhagffefefffff
 167  94       6-os ut
fgggggfffefefgfgfgggggggfgfgggggghgghghghgggfgggghggggfggggg
hgfgfgggfgggggffggggfgfgfgggggggffefggfefeefefffgffgfffggfgfg
```

```
ffffeefeffeefeeeeeeefeeeeeeeeeeeeeeeeeeefeeeefefgffefeeee
 167  94       7-es ut
fegfeefeefffgfffgfffffffeeffefefffffffffffffffffgfffffffe
ffeeeeefefeffefefefefeeefefeefffefeeefeeeefffgfgfgfffffefee
efffeeeeeeeeeefefeeeeg
 131 123       8-as ut
cdddeeeeeefeeedefeffffffeffeefgfeddddeeededeedeeeefeeeefeeeed
eeefeeeeeeeeeeeefeeeeeeeeeeeeeefgffedeedefeffffefeeffeeeeede
eeeeee
 268  41      37-es ut
aahaahabaaaabbabbbbaaaaaaaabcbcbbcbbabbcbbbbbbbbcdb
```

VII.3.6. varosok.txt

```
118 216  Pecs
abhddbaaaddcegfedegggah
 233 203  Szeged
cccccdcefegfhabh
 163 102  Budapest
abaabhhacbbaahabdcbabhhcdccededdceefdccccffddeggfdgghfeddce
ghhgghfahgffgfg
 315  92  Debrecen
abedbaaeedccefdffhhgaa
  85  81  Gyor
aacdbaabdcccedggfgggff
 263  45  Miskolc
abbhcdcdcefefdcdeghfhahhbh
```

VII.3.7. varosxy.txt

```
Debrecen          313  89
Nyíregyháza       317  52
Miskolc           264  42
Kazincbarcika     256  29
Ózd               237  32
Sárospatak        307  22
Sátoraljaújhely   312  13
Szerencs          287  35
Encs              282  22
Edelény           261  25
Mezőkövesd        253  67
```