Tartalomjegyzék

1.. Mérő . . . 2
. Útmutatás . . . 2
2.. Ritmusok . . . 3
. Útmutatás . . . 3
3.. Sziget . . . 3
. Útmutatás . . . 4
4.. Találka . . . 4
. Útmutatás . . . 5
5.. Strázsa . . . 5
. Útmutatás 1. . . . 5
. Útmutatás 2. . . . 6
6.. Robot . . . 6
. Útmutatás . . . 6
7.. Elfogó . . . 7
. Útmutatás . . . 7
8.. Net . . . 7
. Útmutatás . . . 8
9.. Lefed . . . 8
. Útmutatás . . . 9
10.. Azonosító kód . . . 9
. Útmutatás . . . 9
11.. Bányászrobot . . . 9
. Útmutatás . . . 10
12.. Bírálók . . . 10
. Útmutatás . . . 10
13.. Barátok . . . 10
. Útmutatás . . . 11
14.. Képtár . . . 11
. Útmutatás . . . 11
15.. Autó . . . 11
. Útmutatás . . . 12
16.. Minta . . . 12
. Útmutatás . . . 12
17.. Vasút . . . 12
. Útmutatás . . . 13

Bevezető

Az alábbi feladatok mind valamilyen középiskolásoknak szervezett programozási versenyről valók. A versenykörnyezetben egy-egy ilyen feladat megoldására elegendő háromnegyed-egy óra. (A legjobb – középiskolásoknak! – két óra alatt három ilyen feladatot kell megoldaniuk.)

A feladatmegoldás fázisai:

1. Feladat átolvasása, megértése.
2. Absztrakt feladat megfogalmazása.
3. Megoldó algoritmus kidolgozása; esetleg kiválasztása és a feladathoz igazítása, ha ismert algoritmuson alapszik.

   Eddig a pontig kizárólag papír és ceruza kellett.
4. Megoldó program elkészítése.
5. Tesztelés; hibák keresése és kijavítása.

A feladatok nehézsége mindig a 3. pontig terjed. Ezért a feladatkiírások után következő útmutatásokban vázolom a megoldó algoritmust.

Az első feladatot első látásra az algoritmus kidolgozásának kb. 20 perce végeztével 20-30 percembe telt leprogramozni. Az algoritmust megadtam; azt hiszem, így viszonylag jogos mintegy két hét határidővel feladni őket.

Egyszer lemértem, hogy az algoritmus és az implementáció apróságainak ismeretében (azaz egy már korábban megodott feladathoz hasonlót) a megoldó programot öt perc alatt el tudtam készíteni. Ez még egy kezdő programozónak is legfeljebb egy órájába telhet.

Minden feladat helyes megoldása kb. 150-200 soros programot eredményez. Ez a nagyságrend független a kiválasztott feladattól.

Nem elfogadható az a megoldás, amelynek ismertetése során egyértelműen kiderül, hogy nem a beadója készítette el a programot. Ebben az esetben, vagy nem kielégítő félévközi teljesítmény miatt bukásra álló hallgatók mentőövet kapnak az elégségesért egy apró, legfeljebb két képernyőoldalnyi program elkészítése formájában.

A megoldások értékelésénél a futási időt is figyelembe vesszük. Ez a versenyen három évvel ezelőtti gépeken legfeljebb egy másodperc volt, ebben a kurzusban a labor gépein tíz másodperc, kivéve ahol egyéb megszorítások találhatóak. Ezt a követelményt következő megfontolások vezérlik: Ha egy program futási ideje nem túl nagy kitevővel polinomiális a bemenet méretében, akkor a tíz másodperc alatt rendelkezésre álló mintegy 4,3 milliárd utasítás alatt megoldja a feladatot. Ha viszont exponenciális, akkor egy nagyobb tesztesetet a világ végezetéig sem old meg. Röviden: ha tíz másodperc alatt nincs kimenet, akkor egy óra, egy nap vagy egy év után sem lesz, annyit pedig nem vagyunk hajlandóak várni. (Útmutatás: becsüljük meg, mennyi 2^{300}.) A program gyorsasága amúgy is a megoldó algoritmuson múlik, amelyet rendre megadunk.

Megjegyzés: Ezeket a feladatokat lényegesen gyorsabb egyedül megoldani, mint egy alkalmas barátot/barátnőt keríteni aki megírja, majd a kész programot olyan szinten elsajátítani, hogy átcsússzon a beszámolón.

Rácz Balázs

1.. Mérő

Van[1] két kannánk, az első űrtartalma A liter, a másodiké B liter. Ki kell mérnünk N liter vizet a második kannába. Kezdetben mindkét kanna üres. A kannákkal a következő műveleteket végezhetjük:

- TA az első kannát teletöltjük a csapról,
- TB a második kannát teletöltjük a csapról,
- UA az első kannát kiürítjük,
- UB az második kannát kiürítjük,
- AB az első kannából áttöltjük a benne lévő vizet a második kannába úgy, hogy ha mind belefér, akkor mind áttöltjük, egyébként annyit töltünk át, hogy a második kanna tele legyen,
- BA a második kannából áttöltjük a benne lévő vizet az első kannába úgy, hogy ha mind belefér, akkor mind áttöltjük, egyébként annyit töltünk át, hogy az első kanna tele legyen.

Írj programot, amely a két kanna űrtartalma és az előállítandó N mennyiség ismeretében meghatároz egy olyan műveletsort, amelyet sorban végrehajtva a második kannában N liter víz lesz!

A MERO.BE állomány első sorában az előállítandó N vízmennyiség ($1 \leq N \leq 100$), második sorában pedig az első és a második kanna űrtartalma, egy A B egész számpár van, egyetlen szóközzel elválasztva ($1 \leq A, B \leq 100$).

A MERO.KI állományba a megoldás műveletsorát (soronként egy-egy műveletet) kell írni. Ha nincs megoldás, akkor az állomány első és egyetlen sora a NINCS szöveg legyen.

Példa:

MERO.BE

```
2
3 7
```

MERO.KI

```
TA
AB
TA
AB
TA
AB
UB
AB
```

. Útmutatás

Használjunk szélességi keresést (l. Kombinatorika és gráfelmélet I.).

2.. Ritmusok

Egy[2] vers ritmikai elemzéséhez fel kell ismerni a ritmusmintákat (daktilus, spondeus, jambus, ...). A ritmusminták hosszú és rövid szótagok speciális sorrendjéből állnak, melyet a MINTA.BE állományban találsz. Ennek minden egyes sora egy ritmusmintát ír le, a hosszú szótagokat kötőjel (-), a rövideket pedig pont (.) karakterrel jelölve. A ritmusmintákat a sorszámuk azonosítja.

Készíts programot, amely egy vers hosszú-rövid szótageloszlásához meghatározza a sorokhoz tartozó ritmusmintákat!

A RITMUS.BE állomány sorai a MINTA.BE állományhoz hasonló sorokat tartalmaznak, az i. sorban annyi jel van, ahány szótagból áll a vers i. sora. A RITMUS.KI állományba soronként a felismert ritmusminták sorszámait kell írni, egy-egy szóközzel elválasztva. Ha többféle ritmusminta is illeszthető egy sorra, akkor közülük tetszőleges megadható. Ha a bemenő állomány i. sorára nem illeszthető ritmusminta, akkor az eredmény I. sorába a NINCS szöveg kerüljön. Példa:

[1] IOIv '98, 3. feladat
[2] IOIv '98, 1. feladat

MINTA.BE
-..
..-
-.
.-
RITMUS.BE
-..-....-
-...–.
-.-..-.-
.–
RITMUS.KI
1 1 2
1 4 3 vagy 3 2 3
3 3 4 4
NINCS

. Útmutatás

1. . Készítsünk egy irányított gráfot, melynek pontjai a verssor szótagjai közötti elválasztók, él vezet két elválasztó között, ha a közöttük levő szótagok egy érvényes mintát alkotnak. Keressünk (mondjuk legrövidebb) utat a sor eleje és vége között.

Útmutatás 2.. Inkrementálisan, szótagról szótagra fogunk mintákkal való lefedést keresni az egyes verssorokhoz. Algoritmusunk a következő megfigyelésen alapul: egy verssor $1..i$. szótagjai pontosan akkor fedhetőek le úgy, hogy az utolsó minta a k számú minta legyen (melynek hossza l), ha

1. A k. minta illeszkedik a sor $i - l + 1..i$. szótagjaira, és
2. A sor $1..i - l$. szótagjai (valahogyan) lefedhetőek mintákkal.

A sor $1..i$. szótagjai lefedhetőek mintákkal, ha

1. $i = 0$, vagy
2. van olyan k minta, amelyre az előző megfigyelés feltételei teljesülnek.

Ezek látszólag egymásra utalnak, de mivel minden minta legalább 1 szótag hosszúságú, ezért ezek az értékek kiszámíthatóak.

Egy verssor feldolgozásához tehát töltsük ki előlről haladva a következő tömböt: A_i legyen 0, ha nincs lefedése az $1..i$ szótagoknak, és legyen k, ha az $1..i$ szótagok valamely lefedésében az utolsó minta a k.

Ha a tömb utolsó eleme 0, akkor nincs jó lefedés, ha nem nulla, akkor a tömbből könnyen rekonstruálható egy lefedés.

3.. Sziget

Egy[3], a tengeren levő sziget sorsát nagy mértékben befolyásolja a tengerszint változása. A tengerszint emelkedése esetén a sziget területe csökken, a tengerszint csökkenése esetén pedig növekszik. A vízszint emelkedésekor az alacsonyan fekvő területeket elönti a tenger, csökkenésekor pedig a magasabban fekvő területek szárazra kerülhetnek, ha a víz le tud folyni róluk. (A víz mozgásakor minden négyzet 4 oldalszomszédját kell figyelembe venni.)

A szigetet tartalmazó területet kis négyzetekre osztjuk, megadva minden négyzeten a talaj eredeti tengerszint feletti magasságát (ez a szám negatív ott, ahol jelenleg tenger van – ilyenkor a tenger mélységét jelenti). Feltehetjük, hogy a szélső négyzetek a legnagyobb tengerszint csökkenés esetén is vízzel fedettek maradnak.

Készíts programot, amely a terület magasságadatai és a tengerszint változása alapján kiszámítja a sziget területét (a vízzel nem fedett négyzetek számát)!

A SZIGET.BE állomány első sorában a terület mérete (N sor, M oszlop, $1 \leq N \leq 100$,$1 \leq M \leq 100$) és a tengerszint változások száma ($1 \leq K \leq 100$) van, egy szóközzel elválasztva. A következő N sor mindegyikében M egész szám van, az egyes területek eredeti tengerszint feletti magassága, szintén egy-egy szóközzel elválasztva. Az utolsó K sor egy-egy pozitív vagy negatív egész számot tartalmaz, amely a tengerszint növekedését vagy csökkenését írja le. A SZIGET.KI állomány $K + 1$ sorába egy-egy egész számot kell írni. Az első sor a sziget területét (azaz a vízzel nem fedett négyzetek számát) tartalmazza a kiinduló állapotban. A további K sorban az egyes tengerszintváltozások utáni sziget terület értéke legyen!

Példa:

```
SZIGET.BE
8 10 3
-5 -5 -5 -5 -5 -5 -5 -5 -5 -5
-5 -5 +1 +2 +1 +1 +1 +1 -5 -5
```

[3] IOIv '98, 5. feladat

```
-5 +1 +1 +9 +9 +9 +1 +1 +1 -5
-5 +1 +1 +9 +4 +4 +9 +1 +1 -5
-5 +1 +1 +9 +4 +4 +9 +1 +1 -5
-5 -5 +1 +4 +9 +9 +4 +4 +1 -5
-5 +0 +0 +1 +1 +1 +1 +1 +1 -5
-5 -5 -5 -5 -5 -5 -5 -5 -5 -5
5
8
-5
```

Egy lapos partú sziget, amelyen egy magas, gyűrű alakú hegygerinc zár körbe egy közepes magasságú völgyet. (A +0 magasságú pont kezdetben még szárazföld.)

A tengerszint háromszor változik, kétszer nő, majd egyszer csökken.

```
SZIGET.KI
45
13
0
9
```

Magyarázat:

5 méter növekedés elönti a partot, de a hegyekkel körülzárt terület szárazon marad.
Újabb 8 méter növekedés után a teljes sziget elmerül.
5 méter csökkenés után a hegyek már kilátszanak, de a belső völgyből nem tud lefolyni a víz.

. Útmutatás

Áradásnál és apadásnál is elég egyetlen szélső pontból elindulva valamilyen gráfbejárási módszerrel (pl. mélységi kereséssel) vizsgálni a víz számára elérhető pontokat. Ügyeljünk rá, nehogy végtelen ciklusba essen a program (használjunk valamilyen színezést mint a gráfelméletből tanult algoritmusok).

Áradáskor: pontosan azokat a pontokat járjuk be, amelyeket eláraszt a víz. Egy pontot eláraszt a víz, ha szélső pont vagy valamelyik szomszédját elárasztja és magassága kisebb, mint az új vízszint.

Apadáskor: szintén egy szélső pontból indulva vizsgáljuk a domborzatot, és a bejárásnál mindig tartsuk nyilván, hogy az éppen vizsgált pontból milyen magasságig tud lefolyni a víz.

4.. Találka

A[4] római birodalomban lovasfutárok vitték a postát egyik helyről a másikra. A Róma-Aquincum útvonalon a kezdő és a végállomást is beleértve N fogadó épült, ahol a futárok megállhattak, egyenletes távolságra egymástól.

Az útvonalon M futár teljesít szolgálatot, egy napon belül egy irányban megtéve adott távolságot, s másnap ugyanannyit visszafelé.

Készíts programot, amely megadja azokat a futárpárokat, akik szolgálat közben valahol találkozhatnak egymással.

A TALALKA.BE állomány első sorában az állomások N ($1 \leq N \leq 1000$), és a futárok M ($1 \leq M \leq 1000$) száma van, egyetlen szóközzel elválasztva. A következő M sorban az egyes futárok szolgálati adatai vannak: a kezdő- és a végállomásuk sorszáma, valamint a szolgálatuk kezdő- és végideje (0 és 23 között egész számok), egy-egy szóközzel elválasztva.

A TALALKA.KI állomány első sorába azon futárpárok K számát kell írni, akik találkozhatnak szolgálat teljesítés közben, a következő K sorba pedig az egyes párok sorszámait. (Ha az egyik végállomása azonos a másik kezdőállomásával, s az egyik érkezési ideje megegyezik a másik indulási idejével, akkor szolgálati időben még éppen találkozhatnak.)

Példa:

```
TALALKA.BE
100 5
1 10 5 14
5 20 7 14
22 18 13 14
6 9 12 18
9 10 18 19
TALALKA.KI
2
2 3
4 5
```

[4] IOIv '98, 7. feladat

. Útmutatás

Minden párra határozzuk meg, hogy találkozhatnak-e vagy sem.

5.. Strázsa

Egy[5] iskola több napig tartó rendezvényén a tanulók folyamatos ügyeletet tartanak. A rendezvény kezdete az 1., vége az N-edik időegység. Minden tanuló megad egy ügyeleti idő-intervallumot (egy $[a, b)$ számpárt), amikor vállalná az ügyeletet. Az ügyeleti beosztást úgy kell elkészíteni, hogy az egymást váltó tanulók át tudják adni egymásnak a szolgálatot, azaz ha az A tanulót ($[a_1, a_2)$ intervallum) a B tanuló ($[b_1, b_2)$ intervallum) követi a szolgálatban, akkor $b_1 \leq a_2$ teljesüljön.

Írj programot, amely kiszámítja, hogy minimálisan hány tanulóval lehet a kapuügyeletet megoldani a rendezvény teljes idejére a fent leírt feltételnek megfelelően!

A STRAZSA.BE állomány első sora a rendezvény időtartamát (N időegység) tartalmazza ($1 \leq N \leq 10000$). Az állomány ezt követő minden sorában egy `a b` számpár van, egy szóközzel elválasztva ($1 \leq a \leq b \leq N$): egy tanuló által vállalt ügyeleti idő-intervallum. A bemeneti állomány utolsó sora a 0 0 számpárt tartalmazza.

A STRAZSA.KI állományba egyetlen számot kell írni: a lehető legkevesebb tanulót igénylő beosztásban alkalmazott tanulók számát. Ha a feladatnak nincs megoldása (azaz nem lehet a rendezvény teljes időtartamára a jelentkezésekből megfelelő beosztást kialakítani), akkor az állományba 0-t kell írni.

Példa:

```
STRAZSA.BE
10
1 4
2 3
3 7
2 5
4 9
5 10
0 0
STRAZSA.KI
3
```

. Útmutatás 1.

Absztrakt feladat: keressünk az adott intervallumokból minimális darabszámú lefedő intervallumhalmazt.

Jelöljük az ügyeleti intervallumok halmazát J-vel. Először rendezzük sorba a vállalt ügyeleteket a záróidőpont szerint. Eldobhatjuk azokat a jelentkezéseket, amelyek kezdő és záróidőpontja azonos (ezek nulla hosszú intervallumnak felelnek meg). Eldobhatjuk még azokat az $[a, b)$ jelentkezéseket is, amelyekhez van olyen $[c, b)$ jelentkezés, hogy $c < a$. (Biz.: ha egy optimális beosztásban szerepel az $[a, b)$ intervallum, akkor azt kicserélhetjük a $[c, b)$ intervallumra és még mindig optimális lesz a beosztás.) Eldobhatjuk ugyanezen megfigyelésnek a duálisából adódó feltételnek nem megfelelő intervallumokat is, valamint azokat az intervallumokat is, amelyeket teljesen lefed egy másik intervallum.

Számítsuk ki tehát $1 \leq i \leq N$-re a $K_i = \min\{a : \exists b : [a, b) \in J \wedge a < i \leq b\}$ Üres halmaz minimuma végtelen; azonban bizonyítsuk be, hogy ha valamely K_i végtelen, akkor egyáltalán nincs jó beosztás.

Jelölje T_i az $[1, i)$ intervallum őrzéséhez minimálisan szükséges tanulók számát. $T_1 := 0$.

Igazoljuk a következő egyenlőséget: $i = 1, 2, \ldots N$ esetén $T_i = 1 + T_{K_i}$. Írjuk ki a kimeneti fájlba T_N-et.

. Útmutatás 2.

Bizonyítsuk be, hogy a fenti eljárás a következő (mohó) stratégiát követi: Először kiválasztja azt a leghosszabb intervallumot, amelynek záróidőpontja az esemény befejezésének ideje. Majd kiválasztja azt a legkisebb kezdetű intervallumot, amelynek vége legalább az előző kiválasztott intervallum kezdete... Egyszerűbb-e ezt megvalósítani? Megtakarítjuk-e ezzel a K látszólag bonyolult kiszámítását?

6.. Robot

Egy[6] automatizált üzemben a gyártó berendezések egy nagy gépcsarnokban négyzetrácsos elrendezésben vannak. A műszak végeztével egy robot gyűjti össze a legyártott alkatrészeket, amely a berendezések felett a rácsszerkezet által meghatározott pályán mozoghat, csak jobbra, lefelé vagy felfelé tud mozogni. Egy lépés megtételéhez 1 időegységre (másodperc) van szüksége, egyszerre tetszőleges számú alkatrész tud szállítani. A robot induláskor a rácsszerkezet bal felső sarkában van, és a jobb alsó sarokba kell eljuttatnia a legyártott alkatrészeket.

Írj programot, amely kiszámítja, hogy minimálisan hány időegység alatt tudja egy robot begyűjteni az összes legyártott alkatrészt!

[5] IOIv '98 2. feladat
[6] IOIv '98 7. feladat

A feladat megoldásához rendelkezésre áll az üzemcsarnok térképe, amely azt írja le, hogy a rácsszerkezet mely helyein találhatók a gyártó gépek.

A ROBOT.BE állomány első sora a rácsszerkezet méretét megadó M és N értékeket tartalmazza egy szóközzel elválasztva ($1 \leq M, N \leq 1000$). M a rácsszerkezet sorainak, N pedig az oszlopainak a száma. Az állomány ezt követő soraiban a gyártó gépek sor- és oszlopkoordinátái vannak: minden sorban egy a b ($1 \leq a \leq M, 1 \leq b \leq N$) számpár, egyetlen szóközzel elválasztva. A bemeneti állományt a 0 0 számpárt tartalmazó sor zárja. A bal felső sarok koordinátái $(1, 1)$, a jobb alsóé pedig (M, N).

A feladat megoldását adó egyetlen számot a ROBOT.KI állomány első sorába kell írni.

Példa:

```
ROBOT.BE
6 5
1 3
2 1
4 1
3 3
5 3
6 5
0 0
ROBOT.KI
15
```

. Útmutatás

Mivel a ropbot nem tud balra menni, ezért ha egyszer lahagyott egy oszlopot, akkor oda soha vissza nem térhet. Tehát egy oszlop elhagyása előtt minden ott gyártott alkatrészt fel kell szednie. Ekkor csupán annyi választási lehetősége van a pályák köztt, hogy egy adott oszlopban a legfelsőtől a legalsó felé szedi össze az alkatrészeket, vagy a legalsótól a legfelső felé.

Megfeledkezhetünk azokról az oszlopokról, amelyekben nincs alkatrész, hiszen ezeken csak átgyalogolni kell. A számukat persze jegyezzük meg, mert áthaladni rajtuk időbe telik.

Készítsünk egy súlyozott irányított gráfot a következőképpen: Minden i. oszlophoz tartozzon négy pont: MinBe_i, MaxBe_i, MinKi_i, MaxKi_i. A Be jelű pontokba az előző oszlop Ki nevű pontjaiból vezet él, melynek súlya a két pont távolsága (amennyi idő alatt a robot megteszi az utat). A Be jelű pontokból az azonos oszlop ellentétes Ki jelű pontjaiba vezet él (MaxBe $\longrightarrow$ MinKi, MinBe $\longrightarrow$ MaxKi), melynek súlya szintén a két pont távolsága. A Ki jelű pontokból pedig a következő oszlop Be jelű pontjaiba vezet él...

Készítsünk még egy-egy pontot a bal felső és jobb alsó saroknak, kössük őket össze értelmeszerűen az első oszlop MinBe illetve az utolsó oszlop MaxKi pontjaival.

Mutassuk meg, hogy minden optimális összegyűjtéshez van egy út a készített gráfban a kezdő és végpont között, melynek összsúlya pont az optimális idő! Számítsuk ki az optimális időt úgy, hogy miniális összsúlyú utat keresünk a tekintett gráfban. (Figyelem: ez irányított, körmentes gráf.)

7.. Elfogó

A[7] városi rendőrség egy körözött bűnöző elfogását tervezi, aki autóval folyamatosan közlekedik a város utcáin. Minden utca egyirányú, bármely két útkereszteződést egy irányban legfeljebb egy utca köt össze, továbbá a városban lehet zsákutca is, azaz olyan útkereszteződés, amelyből nem vezet kifelé egyirányú utca.

Ha van olyan útkereszteződés, amelyben a bűnözőnek előbb-utóbb fel tűnnie, akkor az elfogásához elegendő a rendőröket egy ilyen helyre összpontosítani.

Írj programot, amely meghatározza az összes olyan útkereszteződést, ahol a bűnözőnek előbb-utóbb fel kell bukkannia, bárhonnan induljon is, feltéve, hogy folyamatosan közlekedik (zsákutcába nem megy be).

A ELFOGO.BE állomány első sora az útkereszteződések N számát tartalmazza ($2 \leq N \leq 200$). Az útkereszteződéseket az $1, \ldots, N$ természetes számokkal azonosítjuk. Az állomány második sorában az egyirányú utcák száma (M) van. Az állomány ezt követő M sora az egyirányú utcákat adja meg. Az egyes sorokban az A-ból B-be vezető utcát az `A B` számpár írja le, egyetlen szóközzel elválasztva ($1 \leq A, B \leq N$).

A ELFOGO.KI állomány egyetlen sorába az összes olyan útkereszteződés azonosító számát kell írni (egy-egy szóközzel elválasztva), amelyen a bűnözőnek előbb-utóbb fel kell bukkannia. Ha nincs ilyen útkereszteződés, akkor a sor tartalma a egyetlen 0 szám legyen.

Példa:

```
ELFOGO.BE
6
8
1 2
```

[7] IOIv '98 9.feladat

```
2 3
3 4
4 5
1 6
2 5
6 3
6 4
ELFOGO.KI
1 5
```

. Útmutatás

Absztrakt feladat: határozzuk meg az összes kör metszetét.

A gráf egy pontja benne van az összes kör metszetében, ha a belőle indított mélységi bejárás szerint osztályozva az éleket vissza-él csak a tekintett gyökércsúcsba vezet.

8.. Net

ByteLand[8] SuliNet hálózata olyan gerinchálózatra épül, amelynek pontjai a városok. Minden városban elhelyeztek egy szolgáltató számítógépet, amely a régiót hivatott ellátni. Bizonyos városokat kétirányú átvitelt biztosító átviteli vonalak kötnek össze. Ha az A és B várost követlenül összeköti átviteli vonal, akkor azt mondjuk, hogy A és B szomszédosak a hálózatban. (A városokat az $1..N$ természetes számokkal azonosítjuk.)

A hálózat biztonságos működése érdekében minden szolgáltatóhoz (A városhoz) ki kell jelölni pontosan egy szomszédos másodlagos szolgáltatót (B várost) úgy, hogy ha A-hoz a B város van kijelölve másodlagosnak, akkor B-hez nem lehet A-t kijelölni. Egy város több más városnak is lehet a másodlagos szolgáltatója. Továbbá, a másodlagos szolgáltató hozzárendelésnek olyannak kell lenni, hogy legyen legalább egy olyan K város, amely a másodlagos szolgáltató kapcsolat mentén bármely másik városból elérhető. Azaz, létezzen olyan K város, hogy bármely A városhoz legyen olyan $A_1 \longrightarrow A_2 \longrightarrow \ldots \longrightarrow A_u$ sorozat, hogy $A_1 = A$, $A_u = K$ és az A_i városhoz rendelt másodlagos szolgáltató A_{i+1} ($i = 1, \ldots, u-1$).

Írj programot, amely meghatároz egy másodlagos szolgáltató hozzárendelést, azaz minden városhoz megad pontosan egy szomszédot, amely teljesíti a fenti feltételeket.

A NET.BE állomány első sorában a városok száma ($3 \leq N \leq 250$), második sorában a szomszédos városok M száma van. A következő M sor mindegyike szomszédos városok `A B` sorszámpárját tartalmazza, egyetlen szóközzel elválasztva ($1 \leq A, B \leq N$).

A NET.KI állomány az egyetlen sorába N számot kell írni, egy-egy szóközzel elválasztva, amely megad egy másodlagos szolgáltató hozzárendelést: az i. szám az i városhoz rendelt másodlagos szolgáltató azonosítója legyen! Ha feladatnak nincs megoldása, akkor a sor tartalma a 0 szám legyen!

Példa:

```
NET.BE
8
10
1 2
2 3
3 4
4 5
2 4
5 6
5 2
6 7
7 8
8 6
4 7
NET.KI
2 3 4 5 6 7 8 6
```

. Útmutatás

Szélességi vagy mélységi fa irányításváltásával kapunk egy megfelelő hozzárendelést, ha összefüggő a gráf. Ha nem összefüggő, akkor nincs.

[8] IOIv '98 10. feladat

9.. Lefed

Egész[9] számok $[a, b]$ intervalluma azon x egész számok halmaza, amelyekre teljesülnek az $a \leq x \leq b$ egyenlőtlenségek. Az $[a, b]$ intervallum hossza az intervallum elemeinek száma, azaz $b - a + 1$. Azt mondjuk, hogy egész számok intervallumainak egy H halmaza lefedi az $[1, N]$ intervallumot, ha az intervallum minden x-eleméhez van olyan intervallum H-ban, amelynek x eleme. Egy lefedés költsége a lefedéshez használt intervallumok hosszainak összege.

Írj programot, amely kiszámítja, hogy adott $[1, N]$ lefedendő intervallum és lefedéshez használható intervallumok egy H halmaza esetén mekkora a minimális lefedés költsége, ha létezik lefedés.

A LEFED.BE szöveges állomány első sorában a lefedendő intervallum végpontja ($1 \leq N \leq 10000$), második sorában pedig a lefedéshez használható intervallumok M száma van ($1 \leq M \leq 1000$). Az állomány ezt követő M sorának mindegyike a lefedéshez használható $[a, b]$ intervallumok végpontjait tartalmazza, egyetlen szóközzel elválasztva ($1 \leq a \leq b \leq N$).

A LEFED.KI állomány egyetlen sorába a minimális lefedési költséget kell írni. Ha a feladatnak nincs megoldása, akkor ez a szám 0 legyen.

Példa:

```
LEFED.BE
10
7
1 3
1 4
7 10
5 7
2 6
3 7
4 8
LEFED.KI
11
```

. Útmutatás

Vegyük észre a hasonlóságot a Strázsa feladattal. Módosítsuk az ott található algoritmust. (Figyelem: nem lehet olyan sok intervallumot eldobni, mint ott!) A legfontosabb, hogy a $T_i = \ldots$ helyére mit kell írni.

10.. Azonosító kód

A[10] processzorgyártó cégek megállapodtak abban, hogy milyen rendszert alkalmaznak az általuk gyártott processzorok azonosítására. Minden cég kap egy betűkészletet, és ezekből kell az azonosító kódot képeznie, úgy, hogy minden betű meghatározott számszor szerepeljen az azonosítóban. Például egy cég azt kapta, hogy minden azonosítója pontosan 3 db 'a' betűt, 2 db 'b' betűt és 1 db 'c' betűt tartalmazhat.

Készíts programot, amely adott kódra meghatározza a lexikografikus (ábécé szerinti) sorrendben rákövetkező szabályos kódot, ha van rákövetkező!

A KOD.BE állomány első sorában a kódok száma ($1 \leq N \leq 100$) van. A további N sorban az egyes kódok találhatók. Minden kód legfeljebb 100 betűből állhat, csak az angol ábécé kis betűit tartalmazhatja.

A KOD.KI állományba N sort kell írni. Az i-edik sorba a bemeneti állomány $i+1$-edik sorában lévő kód rákövetkezőjét kell írni, ha nincs rákövetkezője, akkor a NINCS szót.

Példa:

```
KOD.BE
2
abaacb
cbbaaa
KOD.KI
ababac
NINCS
```

. Útmutatás

Ez egy egyszerű feladat. Használjuk józan paraszti eszünket, ha van. Ha nincs, kérjünk segítséget.

[9] IOIv '98 11. feladat
[10] IOIv '99 1. feladat

11.. Bányászrobot

Egy[11] robotot helyezünk egy labirintusba, megjelölve a kezdő- és a célpozícióját. A robot el kell vezetni a célig, feltéve, hogy minden lépés 1 időegységbe kerül. A robot a labirintus járatain közlekedhet vízszintes vagy függőleges irányban a 4 szomszéd hely valamelyikére lépve, valamint képes a labirintus falait is kibontani. Egy egységnyi fal kibontása a robotnak F időegységbe kerül.

Készíts programot, amely megadja, hogy minimum mennyi idő múlva érhet a robot a kezdőpozícióból a célpozícióba!

A BANYASZ.BE állomány első sorában a négyzetrács magassága és szélessége ($1 \leq N, M \leq 100$), valamint a fal kibontásához szükséges idő ($1 \leq F \leq 100$) van, egy-egy szóközzel elválasztva. A következő N sor mindegyike pontosan M karaktert tartalmaz, az üres helyeken szóközt, a foglalt helyeken `*`-ot, a robot kezdőpozícióján `K` betűt, célpozícióján pedig `C` betűt.

A BANYASZ.KI állományba egyetlen sort kell írni, azt a minimális időtartamot, ami alatt a robot a kezdőhelyről a célpozícióba érhet.

Példa:

```
BANYASZ.BE
4 6 5
******
*K   *
******
*  C**
BANYASZ.KI
8
```

. Útmutatás

Módosítsuk a szélességi keresést a következőképpen: ne színeket tároljunk el az egyes pontok látogatottságát megőrzendő, hanem tároljuk el minden pontra az eddig talált legrövidebb, a kezdőpozícióból oda vezető út hosszát. (Ha még nem volt ilyen út, akkor ez végtelen.) Amikor kiveszünk egy elemet a sorból, akkor tudjuk, mi az eddig talált oda vezető legrövidebb út, így a rajta át a szomszédaihoz vezető utak hosszát is tudjuk. Ha ez rövidebb, mint a szomszédhoz eddig talált legrövidebb út, akkor új utat találtunk a szomszédhoz és újra beteszük a sor végére.

Ez nagyon nem optimális futási idejű algoritmus, azonban könnyű megvalósítani. Ha egy kicsit ügyesebbek vagyunk, akkor a sorból mindig a legkisebb eddig talált legrövidebb úttal rendelkező pontot vesszük ki (prioritásos sort alkalmazunk). Megmutatható, hogy ekkor minden pont pontosan egyszer kerül ki a sorból, így csak egyszer kell feldolgozni a belőle kivezető éleket. Nézzünk utána ennek a módszernek egy algoritmuselmélet könyvben (Dijkstra algoritmusának is nevezik.)

12.. Bírálók

N[12] könyvet 2 bírálónak (A és B) adnak ki, hogy mindegyikről bírálatot készítsenek. Az i. könyv elolvasása az A bírálónak A_i, a B bírálónak pedig B_i ideig tart. A két bíráló egyszerre ugyanazt a könyvet nem olvashatja, és egyszerre mindegyikük legfeljebb 1 könyvet olvashat. Egy könyv olvasását bármelyikük abbahagyhatja, s egy későbbi időpontban onnan folytathatja. A könyvek között az első K fontos könyv. Ezekre teljesülni kell, hogy bármelyik csak akkor olvasható, ha mindkét bíráló elolvasta már az összes, nála kisebb sorszámú könyvet.

Készíts programot, amely megadja, hogy a két bíráló mikor végezhet a lehető leghamarabb!

A BIRALO.BE állomány első sorában a könyvek száma ($1 \leq N \leq 1000$) és a fontos könyvek száma ($1 \leq K \leq N$) van. A következő N sor A és B számára az egyes könyvek elolvasásához szükséges időt tartalmazza, egy szóközzel elválasztva ($1 \leq A_i, B_i \leq 100$).

A BIRALO.KI állományba azt az időtartamot kell írni, ami alatt a két bíráló a leghamarabb végezhet.

Példa:

```
BIRALO.BE
3 2
5 5
4 1
3 4
BIRALO.KI
15
```

. Útmutatás

Vonjuk össze a fontos könyveket egyetlen könyvvé. (Bbe hogy lehet!) A továbbiakban tehát feltesszük, hogy egyetlen fontos könyv van. Igaz-e a következő állítás: a keresett idő $t = \max\{\max_{1 \leq i \leq n}(A_i + B_i), \sum_i A_i, \sum_i B_i\}$

[11] IOIv '99 3. feladat
[12] IOIv '99 4. feladat

13.. Barátok

Egy[13] osztályba N tanuló jár. Mindenkinek vannak barátai. Egyikük születésnapjára kap egy könyvet, amelyről úgy gondolja, hogy mindenkinek el kellene olvasnia. Ezért egy N elemű láncot kell szervezni, ahol mindenki a barátai közül valakinek adja tovább a könyvet, amely a végén visszajut a tulajdonosához.

Készíts programot (BARAT.PAS vagy BARAT.C), amely megadja a láncot!

A BARAT.BE állomány első sorában a tanulók száma és a barátok minimális száma ($1 \leq N \leq 100, \frac{2}{3}N \leq K < N$) van egy-egy szóközzel elválasztva. A következő N sor mindegyike legalább K számot tartalmaz, egy-egy szóközzel elválasztva, az egyes tanulók barátainak sorszámát.

A BARAT.KI állomány első sorába a lánc tagjait kell írni, feltéve, hogy az 1-es sorszámútól indult el a könyv, egy-egy szóközzel elválasztva.

Példa:

```
BARAT.BE
5 3
3 4 5
4 3 5
1 2 4 5
1 3 2
1 2 3
BARAT.KI
1 5 3 2 4
```

. Útmutatás

A feladat Hamilton-köt keresését írja elő. Általános esetben még csak a létezés eldöntése is nagyon nehéz. A megadott gráf azonban nagyon sűrű, lényegesen több éle van, mint egy tanult elégséges feltétel által megköveteltnek. Ebben az esetben könnyű Hamilton-kört találni.

(Ha nagyon nem megy, hogy hogyan kell: keressünk egy viszonylag hosszú kört, majd szúrjuk bele egyenként a megmaradt pontokat. A kritérium szerint ez nagyjából menni fog.)

14.. Képtár

A[14] piripócsi kastély termeiben egy képkiállítást rendeztek be. N terem falain helyeztek el képeket. Egy bejárási útvonalat kell tervezni, amely az 1. teremből indul és az összes termen végighalad úgy, hogy a látogatók az összes képet pontosan egyszer láthassák, majd az 1. teremben ér véget. Ha egy teremből egy szomszéd terembe átmegyünk, akkor csak a jobb oldalon, a bejárat és a kijárat közötti falon levő képeket látja a látogató.

Készíts programot, amely az 1. teremből kiindulva megadja a bejárási útvonalat!

A KEPTAR.BE állomány első sorában a termek száma ($1 \leq N \leq 1000$) van. A következő N sor az egyes termekkel szomszédos termek sorszámát tartalmazza (legalább 1, legfeljebb 20), az óramutató járásával ellentétes sorrendben. Minden sorban annyi szám van, egy-egy szóközzel elválasztva, ahány teremmel szomszédos az illető terem.

A KEPTAR.KI állományba egyetlen sort kell írni, a bejárás sorrendjében a bejárás során érintett termek sorszámát.

Példa:

```
KEPTAR.BE
5
2 3 4 5
1 3
1 4 2
1 3
1
KEPTAR.KI
1 2 3 4 3 2 1 5 1
```

. Útmutatás

Milyen gráfbejárást kellene alkalmazni?

15.. Autó

Egy[15] autót helyezünk egy labirintusba, megjelölve a kezdő- és a célpozícióját. Az autót el kell vezetni a célig, feltéve, hogy sebessége legfeljebb K lehet, s minden lépés után a sebessége 1-gyel változhat. Az autó egy időegység alatt csak

[13] IOIv '99 5. feladat
[14] IOIv '99 7. feladat
[15] IOIv '99 8. feladat

vízszintesen vagy csak függőlegesen mozoghat, az S sebességű autó pontosan S távolságra lép. Az autó a kezdőpozícióban 0 sebességű (a lépése előtt 1 sebességűre vált) és a végpozícióba érve 1 sebességűnek kell lennie.

Készíts programot, amely megadja, hogy minimum mennyi idő múlva érhet az autó a kezdőpozícióból a célpozícióba!

Az AUTO.BE állomány első sorában a négyzetrács magassága és szélessége ($1 \leq N, M \leq 100$), valamint az autók maximális sebessége ($1 \leq K \leq 10$) van, egy-egy szóközzel elválasztva. A következő N sor mindegyike pontosan M karaktert tartalmaz, az üres helyeken szóközt, a foglalt helyeken `*`-ot, az autó kezdőpozícióján `K` betűt, célpozícióján pedig `C` betűt.

Az AUTO.KI állományba egyetlen sort kell írni, azt a minimális időtartamot, ami alatt az autó a kezdőhelyről a célpozícióba érhet. Ha az autó nem érhet el a célpozícióba, akkor az állományba -1-et kell írni.

Példa:

```
AUTO.BE
5 10 3
**********
*K       *
****** ***
C
***   ****
AUTO.KI
7
```

. Útmutatás

Használjunk súlyozott legrövidebb utakat kereső algoritmust. (Lásd Bányászrobot feladat.)

16.. Minta

Szövegelemzés[16] során gyakran kíváncsiak bizonyos szókapcsolatok előfordulására. Erre példa ez a feladat is. Adott az S szöveg, továbbá két, A és B szó. Azt szeretnénk megtudni, hogy az S szövegben hány olyan szó van, amely AWB alakú, ahol W tetszőleges (esetleg üres) szó.

Írj programot (MINTA.PAS vagy MINTA.C), amely meghatározza, hogy az S szövegben hányszor fordul elő AWB alakú szó, továbbá meghatározza a leghosszabb ilyen szó előfordulásának kezdő pozícióját és hosszát.

A MINTA.BE szöveges állomány két sort tartalmaz. Az első sorban az A, a másodikban a B szó található. A szavak legfeljebb 100 karakterből állnak.

A SZOVEG.BE szöveges állomány egyetlen sorában van a szöveg, amelyben keresni kell. Az A és B szavak, valamint a szöveg csak az angol ábécé kis betűit tartalmazhatják. A SZOVEG.BE állományban legfeljebb 1000000 karakter van.

A MINTA.KI szöveges állomány első sorába a szövegben található AWB alakú szavak számát kell írni, ha nincs ilyen, akkor 0-át. Ha az első sor tartalma nem 0, akkor a második sor két számot tartalmazzon, a leghosszabb AWB alakú szó előfordulásának kezdő pozícióját és a szó hosszát.

A minták előfordulásainak számításakor megengedett az átfedés is. A szöveg első betűjének pozíciója 1.

Példa:

```
MINTA.BE
ab
ba
SZOVEG.BE
xyaababababaaabaab
MINTA.KI
6
4 11
```

. Útmutatás

A feladat megoldásához nem szükséges az egész szöveget a memóriában eltárolni.

A feladat tulajdonképpen a következő szám kiszámítását kéri: a B minta minden előfurdulásának kezdőpozíciójára adjuk össze, hogy az előtt hány A minta fejeződött be.

Megoldásunkat teszteljük nagy (1 millió betű) méretű bemenetkre is (mind véletlen, mind extremális bemenetre.) Ha nem fér bele a futási időkorlátba, akkor kérjünk segítséget felsőbbévesektől hatékony mintaillesztő algoritmusra.

17.. Vasút

Veremsor[17] város vasútállomásán nagy gondot okoz a szerelvények rendezése. Az állomásról továbbítandó szerelvényeket úgy kell kialakítani, hogy amikor az megérkezik a célállomásra, a szerelvény végéről mindig lekapcsolható legyen az oda

[16] IOIv '99 9. feladat
[17] IOIv '99 10. feladat

továbbított kocsisor. Minden továbbítandó szerelvény négy állomást érint, ezért a rendezés előtt minden kocsit megjelölnek az 1, 2, 3 vagy 4 számokkal. A szerelvény kocsijait rendezzük át úgy, hogy a szerelvény elején legyenek az 1-essel, aztán a 2-essel, majd a 3-assal, végül a 4-essel megjelöltek. Kezdetben a kocsik az ábrán látható E-F pályaszakaszon vannak.

A vasúti váltók működése csak a következő műveleteket teszi lehetővé. Az átrendezendő kocsisorból balról az első kocsit át lehet mozgatni vagy a B-E szakaszba a már ott lévő kocsik mögé, vagy a C-D szakaszba a már ott lévő kocsik elé. A B-E szakaszban lévo első kocsi átmozdítható és hozzáilleszthető az A-B szakaszon kialakítandó kocsisor végére. Hasonlóan, a C-D szakasz első (tehát az utolsónak oda érkezett) kocsija átmozdítható és hozzáilleszthető az A-B szakaszon kialakítandó kocsisor végére.

```
                                D|
                                 |
                                 |
                                C|
                        |       / \      ^
                        |      /   \     |
                        v     /     \    |
                             /       \
                            /         \
                           /           \
    <-----                /  <----     \        <-----
---------------------+-------------+-------------------
 A                    B             E                  F
```

Írj programot, amely eldönti, hogy a kívánt kocsisorrend kialakítható-e az alkalmazható műveletekkel.

A VASUT.BE állomány első sora a kocsik ($4 \leq N \leq 2000$) számát tartalmazza. A második sorban a rendezendő szerelvények száma áll ($1 \leq M \leq 100$). A következő M sor mindegyike egy átrendezendő szerelvényt ír le, N db 1 és 4 közötti egész számot tartalmaz egy-egy szóközzel elválasztva.

A VASUT.KI állomány M sort tartalmaz, soronként az IGEN vagy NEM szöveget, annak megfelelően, hogy a kívánt kocsisorrend kialakítható-e a megfelelő bemeneti kocsisorrendből.

Példa:

```
VASUT.BE
8
3
4 4 1 2 3 4 2 3
4 3 2 1 4 3 2 1
4 1 2 3 1 2 3 4
VASUT.KI
IGEN
NEM
IGEN
```

. Útmutatás

Ez egy szép és elég nehéz feladat, különösen mert kevés útmutatást adok hozzá. Annak ajánlom, aki nagyon jó jegyet szeretne.

- Meddig kell feldolgozni a szerelvényt?
- Próbáljunk meg valamilyen (minél egyszerűbb) állapotleírást adni az első i vagon feldolgozása utáni lehetséges helyzetekről (amelyek a jövő szempontjából ekvivalensek).
- Írjuk fel az állapotátmenetgráfot.