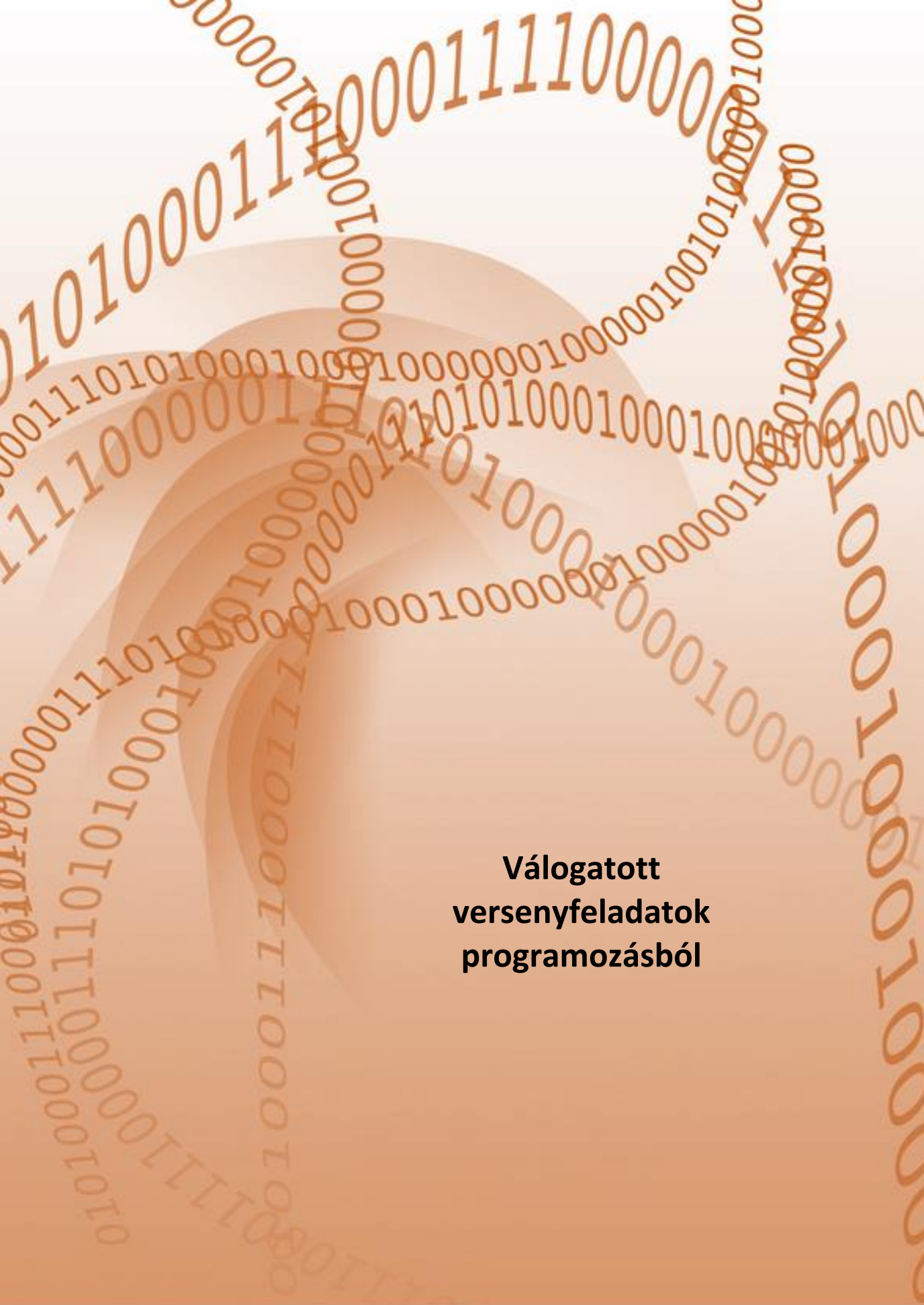


Válogatott versenyfeladatok programozásból

Válogatott versenyfeladatok programozásból

Összeállította: Juhász Tibor

– 2015 –

TARTALOM

A feladatgyűjtemény a Nemes Tihamér OITV és az Informatika OKTV programozás kategóriájának azon feladatait tartalmazza, melyek szerepelnek *Juhász Tibor–Tóth Bertalan: Programozási ismeretek versenyzőknek* (Műszaki Kiadó, 2015) című könyvében, a 3. fejezet (*Válogatott feladatok*) javasolt feladatai között.

A feladatok szövegének gyűjteménye és témakörök szerinti csoportosítása megkönnyíti a felkészülést a fent említett versenyekre. A témakörökbe sorolás a feladatok szövege alapján történt. Így találhatunk a legkönnyebben egymáshoz hasonló feladatokat. Egy-egy témakörön belül a megoldáshoz felhasznált algoritmusok szerint, azon belül pedig évenkénti sorrendben következnek a feladatok.

A besorolás természetesen nem egyértelmű. Egy-egy feladat többféle témakörhöz tartozhat és többféle algoritmussal is megoldható.

A feladatok listáját, továbbá az 1. fordulós feladatok megoldását a *Függelékben* közöljük. A feladatok és az 1. fordulós megoldások forrása:

http://nemes.inf.elte.hu/nemes_archivum.html

A 2. és 3. fordulós feladatok megoldása Visual Basic és C++ nyelven megtalálható a *Programozási ismeretek versenyzőknek* könyv webhelyén:

www.zmgzeg.sulinet.hu/programozas

A C++ programokat Tóth Bertalan készítette.

A felhasznált irodalom jegyzéke és a megoldásokra vonatkozó tudnivalók a fenti könyv bevezetésében olvashatók. A könyv 3. fejezetében (*Válogatott feladatok*) minden egyes feladattípushoz további, részletesen kidolgozott feladatokat találunk. Célszerű először ezekkel megismerkedni, majd utána hozzáfogni az itt következő, hasonló feladatok megoldásához.

Köszönet illeti dr. Zsakó Lászlót, amiért engedélyezte a feladatok szövegének közlését. Hálás vagyok Jákli Aidának a feladatok csoportosításához nyújtott segítségéért, továbbá azért, mert hozzájárult jó néhány megoldásának felhasználásához.

Juhász Tibor

FOTÓZÁS

A *Programozási ismeretek versenyzőknek* (Műszaki Kiadó, 2015) könyvben szereplő feladatok:

OKTV 2004-2f3 Fényképezés
OKTV 2013-2f3 Csoportkép

Első fordulós feladatok

Osztálybuli

Nemes Tihamér OITV 2001. 1. forduló 1. korcsoport 4. feladat

Osztályod két bulit rendezett. A rendezvényre az osztálytársak különböző időpontokban érkeztek, illetve távoztak. Valaki pontosan feljegyezte minden résztvevő tanulóról, hogy az mikor érkezett, illetve távozott. Egy feljegyzés egy [*a*, *b*] számpár, ami azt jelenti, hogy a tanuló az *a* időponttól a *b* időpontig volt jelen a rendezvényen (beleértve az *a* és a *b* időpontokat is). Minden időpont a rendezvény kezdete óta eltelt idő másodpercben mérve.

1. feljegyzés:

```
[600, 1000],[1, 100], [100, 500], [500, 700], [400, 800],
[650, 900], [300, 600], [66, 120]
```

2. feljegyzés:

```
[1, 1000], [4000, 6000], [500, 2113], [2345, 5000],
[601, 1000], [5000, 6000], [2000, 3000], [2345, 3333],
[3001, 4000], [3350, 5010], [4001, 5000], [6300, 9000],
[5056, 5156], [5621, 6000], [3350, 5056], [6000 7000]
```

a) Hányan voltak legtöbben egyszerre jelen a rendezvényen?
b) Adj meg egy olyan időpontot, amikor a legtöbben voltak egyszerre jelen!
c) Legalább hány fényképet kellett volna készíteni ahhoz, hogy mindenki szerepeljen legalább egy fényképen?
d) Adj meg a c) kérdésben szereplő számú időpontot, hogy ha ekkor készült volna egy-egy fénykép, akkor mindenki szerepelne legalább egy fényképen!

Mohó algoritmus

Szemtanúk

Nemes Tihamér OITV 2006. 3. forduló 2. korcsoport 3. feladat

A Rendőrség szemtanúkat keres egy rendezvényen történt gyanús események kivizsgálásához. A rendezvény szervezői feljegyezték minden vendégről, hogy mikor érkezett és mikor távozott. A Rendőrség ki akar választani a lehető legkevesebb számú vendéget úgy, hogy minden gyanús esemény időpontjához legyen olyan kiválasztott vendég, aki jelen volt az esemény időpontjában. Ha egy gyanús esemény az X időpontban történt, akkor az olyan vendég, aki az E időpontban érkezett és a T időpontban távozott szemtanúja volt az eseménynek, ha $E \le X \le T$.

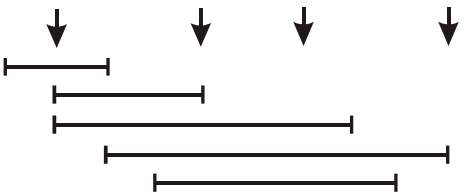
Készíts programot *rkeres* néven, amely megadja, hogy minimálisan hány vendéget kell kiválasztania a Rendőrségnek, hogy minden gyanús esemény időpontjához legyen olyan kiválasztott vendég, aki jelen volt az esemény időpontjában!

Az *rkeres.be* szöveges állomány első sorában a vendégek *M* (1≤*M*≤1000), és a gyanús események *N* (1≤*N*≤300) száma van, egy-egy szóközzel elválasztva. A következő *M* sor mindegyikében két egész szám van, egy vendég *E* érkezési és *T* távozási időpontja (1≤*E*<*T*≤20000). Az állomány *i*+1-edik sorában az *i*-edik vendég adata van. A vendégek adatai érkezési idejük szerint monoton nemcsökkenő sorrendben vannak. Az utolsó sor *N* pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, a gyanús események időpontjait monoton növekvő sorrendben.

Az *rkeres.ki* szöveges állomány első sorába a kiválasztandó vendégek *K* minimális számát kell írni! A második sorba kell kiírni a kiválasztott vendégek sorszámait egy-egy szóközzel elválasztva (tetszőleges sorrendben). Ha nincs megoldás, akkor az első sorba a 0 számot kell írni. Több megoldás esetén bármelyik megadható.

Példa:

```
rkeres.be          rkeres.ki
5 4                2
1 3                1 4
2 5
2 8
3 10
4 9
2 5 7 10
```



Fénykép (2011)

Nemes Tihamér OITV 2011. 3. forduló 2. korcsoport 1. feladat

Egy rendezvényre sok vendéget hívtak meg. Minden vendég előre megadta, hogy mikor érkezik, és mikor távozik. A rendezők fényképen akarják megörökíteni a résztvevőket. A munkára kiválasztott fényképész úgy dolgozik, hogy egy menetben lefényképezi mindazokat, akik a menet *F* kezdete és *F*+*K* vége közötti időintervallumban jelen voltak a rendezvényen. Pontosabban lefényképez minden olyan vendéget, akinek *E* érkezési és *T* távozási idejére teljesül, hogy *E* < *F*+*K*, és *F* ≤ *T*. A fényképészt a menetek száma szerint kell fizetni, tehát az a cél, hogy a lehető legkevesebb menet legyen, de mindenki rajta legyen legalább egy fényképen.

Készíts programot *fenykep* néven, amely megadja, hogy legkevesebb hány menetre van szükség, hogy mindenki rajta legyen legalább egy fényképen, és meg is adja, hogy mikor kezdődjenek a menetek!

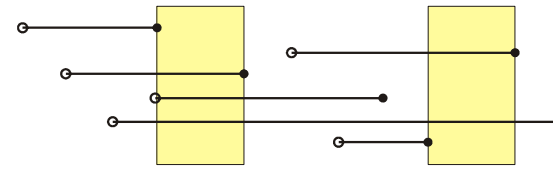
A *fenykep.be* szöveges állomány első sorában két egész szám van egy szóközzel elválasztva, a vendégek *N* száma (1≤*N*≤100000), és a menetek hosszát megadó *K* (2≤*K*≤1000) szám van. A további *N* sor mindegyikében két egész szám van egy szóközzel elválasztva, egy vendég *E* érkezési és *T* távozási ideje (1≤*E*<*T*<20000).

A *fenykep.ki* szöveges állomány első sorába a minimálisan szükséges menetek *M* számát kell írni! A második sor pontosan *M* egész számot tartalmazzon egy-egy szóközzel elválasztva, az egyes menetek kezdő időpontját. Több megoldás esetén bármelyik megadható.

Példa:

```
fenykep.be
6 2
1 4
7 12
2 6
4 9
3 13
8 10
```

```
fenykep.be
2
4 10
```



Párok

Informatika OKTV 2011. 3. forduló 3. korcsoport 1. feladat

Egy rendezvényre sok vendéget hívtak meg. Minden vendég előre megadta, hogy mikor érkezik, és mikor távozik. A rendezők fényképeken akarják megörökíteni a résztvevőket. A rendezőknek két betartandó kikötése van:

1. Minden képen pontosan két vendég legyen rajta.
2. Minden vendég legfeljebb egy képen szerepelhet.

Természetesen két vendég csak akkor szerepelhet azonos képen, ha van olyan F időpont, amikor mindketten jelen vannak. Egy vendég akkor és csak akkor van jelen az F időpontban, ha az E érkezési és T távozási idejére teljesül, hogy $E \leq F$, és $F < T$.

Készíts programot *parok* néven, amely kiszámítja, hogy legjobb esetben hány fénykép készülhet, és megadja, hogy mely párok szerepeljenek egy képen!

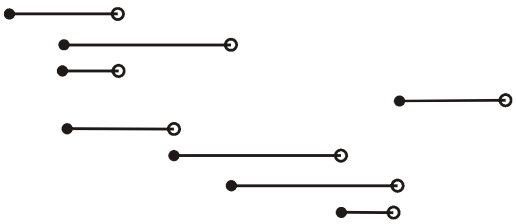
A *parok.be* szöveges állomány első sorában egy egész szám van, a vendégek N száma ($1 \leq N \leq 30000$). A vendégeket a sorszámukkal azonosítjuk 1-től N-ig. A további N sor mindegyikében két egész szám van egy szóközzel elválasztva; egy vendég E érkezési és T távozási ideje ($1 \leq E < T < 20000$). Az állomány i+1-edik sorában van az i sorszámú vendég adata.

A *parok.ki* szöveges állomány első sorába a lehetséges legtöbb készíthető fényképek M számát kell írni! A további M sor mindegyikébe egy számpárt kell írni, egy szóközzel elválasztva, azon két vendég sorszámát (tetszőleges sorrendben), akik egy képen szerepelnek. Több megoldás esetén bármelyik megadható.

Példa:

```
parok.be
8
1 3
2 5
2 3
8 10
2 4
4 7
5 8
7 8
```

```
parok.be
3
3 1
5 2
6 7
```



Fénykép (2014)

Informatika OKTV 2014. 3. forduló 3. korcsoport 2. feladat

Egy rendezvényre vendégek érkeznek. Ismerjük mindenkinek az érkezési és távozási időpontját. A szervező megbízott egy fényképészt, hogy a résztvevőkről csoportképeket készítsen. A fényképész minél hamarabb szeretne végezni, ezért amint jelen van legalább K vendég, akkor közülük pontosan K vendéget lefényképez egy csoportképen, azaz csak abban dönthet, hogy adott időpontban kiket fényképez le. Egy időpontban csak egy fényképet tud készíteni, és minden vendég legfeljebb 1 képen szerepelhet. A vendégek már az érkezési időpontjukban lefényképezhetők és az utolsó lehetőség a lefényképezésükre a távozási időpontjuk.

Készíts programot *fenykep* néven, amely megadja, hogy maximum hány fényképet tud készíteni a fényképész, és megadja, hogy az egyes képeken kik lesznek!

A *fenykep.be* szöveges állomány első sora két egész számot tartalmaz, az első szám a vendégek N száma ($1 \le N \le 100000$), a második szám a K értéke ($1 \le K \le 100$). A következő N sor mindegyikében egy-egy vendég érkezési és távozási időpontja ($1 \le E_i < T_i \le 10000$) van, érkezési időpont szerint nemcsökkenő sorrendben. A vendégeket az 1, …, N számokkal azonosítjuk.

A *fenykep.ki* szöveges állomány első sorába a fényképezések maximális F számát kell írni! A következő F sor mindegyike pontosan K különböző egész számot tartalmazzon egy-egy szóközzel elválasztva, azon vendégek sorszámait, akit az adott időpontban a csoportképen lesznek. Több megoldás esetén bármelyik megadható.

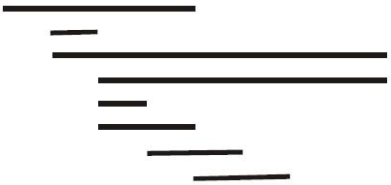
Példa:

`fenykep.be`

```
8 3
1 5
2 3
2 9
3 9
3 4
3 5
4 6
5 7
```

`fenykep.ki`

```
2
2 1 3
5 6 4
```



PAKOLÁS

A *Programozási ismeretek versenyzőknek* (Műszaki Kiadó, 2015) könyvben szereplő feladatok:

OITV 1999-2f2	Kockák
OKTV 1999-2f3	Kockák
OKTV 2000-2f3	Kockavilág

Első fordulós feladatok

Ládapakoló robot

Nemes Tihamér OITV 2000. 1. forduló 1. korcsoport 4. feladat

Egy robot egy raktárban ládákat pakol a polcokra. A polcok egymás mellett vannak. A robot kezdetben a bal oldalinál áll, új ládát csak itt foghat a kezébe, jobbra és balra lépkedhet, s a kezében levő ládát az előtte levő polcra teheti. Minden polcra tetszőleges darabszámú ládát tehet, de kisebbre nagyobbat nem (a ládák mérete 1 és 999 közötti). Üres polc esetén a legfelső láda 1000 méretű.

Az alábbi két algoritmus alapján pakolhat egy ládát valamelyik polcra (a robot az eljárások elején mindig automatikusan a bal oldali polchoz áll):

```
Első:
  Vedd fel a ládát
  Ismételd amíg a láda nagyobb, mint az előtted levő polc
                legfelső ládája és polc előtt állsz
    Lépj egyet jobbra
  Ismétlés vége
  Tedd a polcra a ládát
Eljárás vége.
```

```
Második:
  Vedd fel a ládát
  Ismételd amíg a láda nem nagyobb, mint az előtted levő
                polc legfelső ládája és polc előtt állsz
    Lépj egyet jobbra
  Ismétlés vége
  Lépj egyet balra
  Tedd a polcra a ládát
Eljárás vége.
```

Tegyük fel, hogy a raktárban 3 polc van, s *N* láda érkezik az alábbi méretekkel: 5,8,4,9,1,3,6.

a) Az egyes eljárások alapján melyik ládát hova teszi a robot?
b) Fogalmazd meg, hogy mikor nem tudja a robot elhelyezni a következő ládát!
c) Add meg, hogy a fenti ládasorozatot milyen méretű ládával kell kibővíteni, hogy a feladat ne legyen megoldható!

Kockarakás

Nemes Tihamér OITV 2001. 1. forduló 2. korcsoport 5. feladat

A 7 törpe szabadidejében építőkockákból épít tornyokat. Különböző méretű építőkockákkal rendelkeznek, s a toronyépítésnek egyetlen szabálya van: kisebb kockára nagyobbat nem lehet tenni. Az építkezéshez N kockát használnak, az egyes kockát méretét $K(i)$ jelöli ($1 \le K(i) \le 99$). A kockákat sorban helyezik el, s egy elhelyezett kockát már nem lehet mozgatni. A megoldásban M lesz a tornyok száma, $T(j)$ pedig a j. torony tetején levő kocka mérete.

```
Hapci:
  M:=1; T(1):=K(1)
  Ciklus i=2-től N-ig
    j:=1
    Ciklus amíg j≤M és K(i)>T(j)
      j:=j+1
    Ciklus vége
    Ha j≤M akkor T(j):=K(i) különben M:=M+1; T(M):=K(i)
  Ciklus vége
Eljárás vége.
```

```
Tudor:
  M:=1; T(1):=K(1)
  Ciklus i=2-től N-ig
    mm:=T(1); jj:=1
    Ciklus j=2-től M-ig
      Ha T(j)>mm akkor mm:=T(j); jj:=j
    Ciklus vége
    Ha mm<K(i) akkor M:=M+1; T(M):=K(i) különben T(jj):=K(i)
  Ciklus vége
Eljárás vége.
```

```
Kuka:
  M:=1; T(1):=K(1)
  Ciklus i=2-től N-ig
    Ha T(1)<K(i) akkor M:=M+1; T(M):=K(i) különben T(1):=K(i)
  Ciklus vége
Eljárás vége.
```

a) Milyen elven választják ki a továbbépítendő tornyot az egyes törpék?
b) Állítsd sorrendbe a törpéket aszerint, hogy melyik tudja kevesebb toronyba elhelyezni a kockákat!
c) Milyen kockasorozatra építi mindhárom törpe ugyanazt az egy tornyot ($N>2$)?
d) Adj olyan kockasorozatot ($N=5$), amelyre a törpék mind különböző számú tornyokat építenek!

Kocka (2005-1f2)

Nemes Tihamér OITV 2005. 1. forduló 2. korcsoport 5. feladat

Építőkockából úgy lehet stabil tornyot építeni, hogy kisebb kockára nem lehet nagyobbat, illetve könnyebb kockára nem lehet nehezebbet tenni. Van 10 kockánk, a súlyuk szerint csökkenő sorrendbe rakva, melyek magassága: 10, 7, 6, 4, 11, 3, 8, 14, 5, 9. Meg kell adni a belőlük építhető legmagasabb torony magasságát.

A feladat megoldásához töltsd ki az alábbi táblázatot, amelyben $M(i)$ a legmagasabb olyan torony magassága, ahol az i-edik kocka van legfelül! A kitöltött táblázat alapján add meg a legmagasabb építhető torony magasságát!

M(1)	M(2)	M(3)	M(4)	M(5)	M(6)	M(7)	M(8)	M(9)	M(10)

Kocka (2005-1f3)

Informatika OKTV 2005. 1. forduló 3. korcsoport 5. feladat

Építőkockából úgy lehet stabil tornyot építeni, hogy kisebb kockára nem lehet nagyobbat, illetve könnyebb kockára nem lehet nehezebbet tenni. Van 10 kockánk, a súlyuk szerint csökkenő sorrendbe rakva, melyek magassága: 10, 7, 6, 4, 11, 3, 8, 14, 5, 9. Meg kell adni a belőlük építhető, legtöbb kockából álló torony kockaszámát.

A feladat megoldásához töltsd ki az alábbi táblázatot, amelyben $M(i)$ a legtöbb kockából álló, olyan torony kockaszáma, ahol az i-edik kocka van legfelül! A kitöltött táblázat alapján add meg a legtöbb kockából álló torony kockaszámát!

M(1)	M(2)	M(3)	M(4)	M(5)	M(6)	M(7)	M(8)	M(9)	M(10)

Pakolás (2008-1f2)

Nemes Tihamér OITV 2008. 1. forduló 2. korcsoport 4. feladat

Egymás mellé N darab ládát helyezünk el, különböző méretűeket. Tetszőleges üres láda megfogható és belehelyezhető valamely, tőle balra vagy jobbra elhelyezkedő nagyobb ládába, ha köztük nincs másik láda. Ha például balról jobbra haladva a ládák mérete rendre 1, 3, 2, akkor először a 2 méretűt rakjuk a 3 méretűbe, majd az 1 méretűt beletehetjük; de ha először az 1-est tesszük a 3-asba, akkor a 2-est már nem tehetjük bele. Az a cél, hogy a pakolás végén a ládákat a lehető legkevesebb ládába pakoljuk be.

Példa:

1 3 2 4 5 5 $\Rightarrow$ 3 (azaz például az 5-ösbe tesszük a mellette levő 4-est, azután ebbe beletesszük a most már szomszédos 2-est, majd a 3-asba tesszük az 1-est, a másik 5-ös üresen marad)

Add meg, hogy az alábbi ládasorozatok esetén a ládák hány ládába pakolhatók!

a) 1 2 3 4 5 5 4 3 2 1
b) 1 3 5 2 4 2 3 5 4 1 2 4
c) 2 4 3 5 3 5 3 2 2 3 4 5 4 5
d) 3 2 5 1 4 3 4 2 5

Pakolás (2008-1f3)

Informatika OKTV 2008. 1. forduló 3. korcsoport 4. feladat

Egymás mellé *N* darab ládát helyezünk el, különböző méretűeket. Tetszőleges üres láda megfogható és belehelyezhető valamely, tőle jobbra akárhol elhelyezkedő nagyobb ládába. Ha például balról jobbra haladva a ládák mérete rendre 1, 2, 3, akkor először a 2 méretűt rakjuk a 3 méretűbe, majd az 1 méretűt beletehetjük; de ha először az 1-est tesszük a 2-esbe, akkor azok együtt már nem mozgathatók. Az a cél, hogy a pakolás végén a ládákat a lehető legkevesebb ládába pakoljuk be.

Példa:

3 1 5 4 $\Rightarrow$ 2 (azaz például a 3-as betehető az 5-ösbe, ezután az 1-es az 5-ösben levő 3-asba, tehát marad az 5-ös és a 4-es)

Add meg, hogy az alábbi ládasorozatok esetén a ládák hány ládába pakolhatók!

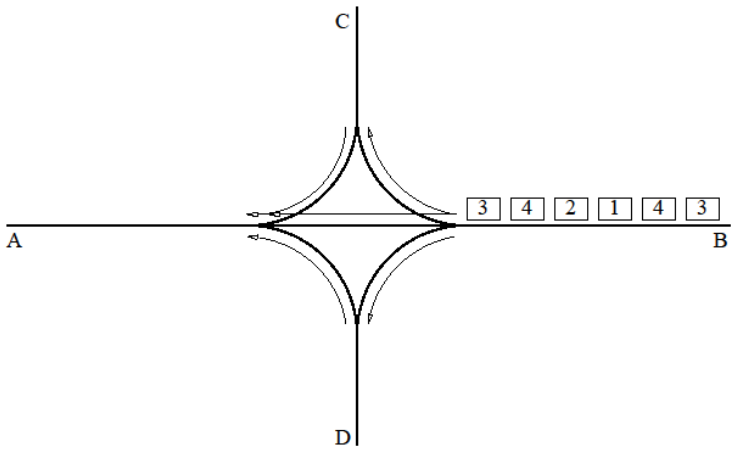
a) 1 2 3 4 5 5 4 3 2 1
b) 1 3 5 2 4 2 3 5 4 1 2 4
c) 2 4 2 4 3 5 3 5 2 3 2 3 4 5 4 5

Összetett feltételek vizsgálata

Vasúti kocsik rendezése

Informatika OKTV 2010. 3. forduló 3. korcsoport 3. feladat

Duplaverem város vasútállomásán sok gondot okoz a vasúti kocsik rendezése. Az állomásról továbbítandó szerelvényeket úgy kell kialakítani, hogy amikor az megérkezik a célállomásra, a szerelvény végéről mindig lekapcsolható legyen az oda 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 át kell rendezni 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 *B* pályaszakaszon vannak. A vasúti váltók működése csak a következő műveleteket teszi lehetővé. Az átrendezendő kocsisorból balról az első kocsit át lehet mozgatni vagy az *A* szakaszba a már ott lévő kocsik mögé, vagy a *C* vagy *D* szakaszba a már ott lévő kocsik elé. Továbbá, a *C* vagy *D* szakaszon lévő első kocsit át lehet mozgatni az A szakaszon kialakítandó rendezett kocsisor végére.



Készíts programot *vasut* néven, amely megállapítja, hogy a beérkező kocsisor rendezhető-e!

A *vasut.be* szöveges állomány első sorában a rendezésre váró szerelvények *N* száma van ($1 \leq N \leq 20$). A következő *N* sor mindegyike egy legfeljebb 1000 kocsit tartalmazó rendezendő szerelvényt ír le, minden sort a 0 szám zárja (ami nem része a bemenetnek). Az utolsó 0 kivételével minden szám értéke 1, 2, 3 vagy 4.

A *vasut.ki* szöveges állományba pontosan *N* sort kell írni! Az *i*-edik sorba az *IGEN* szót kell írni, ha a bemenet *i*+1-edik sorában szereplő kocsisor rendezhető, egyébként pedig a *NEM* szót!

Példa:

```
vasut.be              vasut.ki
2                     IGEN
1 2 3 2 4 0           NEM
2 3 1 4 2 1 0
```

Mohó algoritmus

Pakolás (2004)

Nemes Tihamér OITV 2004. 2. forduló 2. korcsoport 3. feladat

Egy vállalat udvarán egyetlen sorban vannak az elszállításra várakozó üres ládák. Három különböző típusú láda van, jelölje ezeket *A*, *B* és *C*. Minden láda egyik oldalán nyitott kocka alakú. Az *A* típusú láda a legnagyobb és a *C* típusú a legkisebb. Tehát minden *C* típusú láda belerakható *A* típusú és *B* típusú ládába, minden *B* típusú belerakható *A* típusúba és *A* típusúba belerakható *B* típusú, majd ebbe egy *C* típusú. Az a cél, hogy a ládákat úgy pakoljuk össze, hogy a lehető legkevesebb összepakolt láda legyen. A pakolást olyan robot végzi, amely a ládasor felett tud mozogni mindkét irányban, de ládát csak balról jobbra mozgva tud szállítani.

Írj programot *pakol* néven, amely megadja, hogy legkevesebb hány ládába lehet összepakolni a ládasort!

A *pakol.be* szöveges állomány első sorában a ládák *N* ($1 \leq N \leq 10000$) száma van. A második sor pontosan *N* karaktert tartalmaz (szóközök nélkül), a ládasor leírását. Minden karakter vagy '*A*', vagy '*B*' vagy '*C*'.

A *pakol.ki* szöveges állomány első és egyetlen sorába azt a legkisebb *K* számot kell írni, amelyre a bemeneti ládasor összepakolható *K* ládába!

Kiegészítés az eredeti feladathoz: a ládák számának meghatározása mellett a program végezze is el a pakolást!

Példa:

```
pakol.be              pakol.ki
10                    6
ABACACBCCA
```

Pakolás (2008-2f2)

Nemes Tihamér OITV 2008. 2. forduló 2. korcsoport 3. feladat

Egymás mellé *N* darab ládát helyezünk el, különböző méretűeket. Tetszőleges üres láda megfogható és belehelyezhető valamely, tőle balra vagy jobbra elhelyezkedő nagyobb ládába, ha köztük nincs másik láda. Ha például balról jobbra haladva a ládák

mérete rendre 1, 3, 2, akkor először a 2 méretűt rakjuk a 3 méretűbe, majd az 1 méretűt beletehetjük; de ha először az 1-est tesszük a 3-asba, akkor a 2-est már nem tehetjük bele. Az a cél, hogy a pakolás végén a ládákat a lehető legkevesebb ládába pakoljuk be.

Készíts programot *pakol* néven, amely kiszámítja, hogy legkevesebb hány ládába lehet összepakolni a bemeneti ládasort!

A *pakol.be* szöveges állomány első sorában a ládák N ($1 \leq N \leq 10000$) száma van. A második sor pontosan N különböző pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, a ládák méreteit balról jobbra sorrendben. A legnagyobb láda mérete N.

A *pakol.ki* állomány első és egyetlen sorába egy egész számot kell írni, ahány ládába a ládasor bepakolható!

Kiegészítés az eredeti feladathoz: a ládák számának meghatározása mellett a program végezze is el a pakolást!

Példa:

```
pakol.be                pakol.ki
8                       2
3 4 7 2 1 5 8 6
```

Pakolás (2011)

Nemes Tihamér OITV 2011. 2. forduló 2. korcsoport 3. feladat
Informatika OKTV 2011. 2. forduló 3. korcsoport 4. feladat

A két feladat megegyezik egymással.

Egy raktárban egyetlen hosszú sorban ládák vannak. Minden láda kocka alakú, de méretük különböző lehet. A ládák egymásra rakásával akarnak helyet felszabadítani. A biztonsági előírás szerint több ládát is lehet egymásra rakni, de minden ládát csak nálánál nagyobbra lehet helyezni. Továbbá, az i-edik helyen lévő ládát csak akkor lehet rárakni a j-edik helyen lévő torony tetejére, ha az i-edik és j-edik helyek között már nincs láda (j lehet akár kisebb, akár nagyobb, mint i). Minden ládát legfeljebb egyszer lehet mozgatni.

Készíts programot *pakol* néven, amely kiszámítja, hogy legkevesebb hány toronyba lehet a ládákat összepakolni!

A *pakol.be* szöveges állomány első sorában a ládák N ($2 \leq N \leq 30000$) száma van. A második sor pontosan N pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, a ládák méreteit. A második sorban lévő számok mindegyike 1 és 30000 közötti érték.

A *pakol.ki* szöveges állomány első és egyetlen sora egy egész számot tartalmazzon, azt a legkisebb M számot, hogy a bementben megadott ládasor összepakolható M számú toronyba!

Példa:

```
pakol.be                pakol.ki
10                      2
1 2 4 6 7 5 3 2 5 3
```

Párosítás

Informatika OKTV 2012. 2. forduló 3. korcsoport 4. feladat

Egy raktárból *N* boltba kell kiszállítani ládákba csomagolt árut. Minden boltba pontosan két ládát kell vinni. A ládák az előkészítés időrendi sorrendjében egymás mellett egy sorban vannak, mindegyikre ráragasztva annak a boltnak a sorszáma, ahova szállítani kell. A raktárosnak át kell rendezni a ládák sorrendjét, hogy az egy boltba kerülő ládák egymás mellett legyenek. Az átrendezés során egy lépésben két láda helyét cserélheti meg.

Készíts programot *parosit* néven, amely kiszámítja, hogy legkevesebb hány cserével lehet kialakítani a kívánt sorrendet!

A *parosit*.be szöveges állomány első sorában a boltok *N* ($2 \le N \le 20000$) száma van. A második sor pontosan 2*N* pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, az 1, …, *N* számok mindegyike pontosan kétszer fordul elő a sorban.

A *parosit.ki* szöveges állomány első és egyetlen sora egy egész számot tartalmazzon, azt a legkisebb *M* számot, amire a bemenetben megadott ládasor *M* számú cserével a kívánt sorrendbe rakható!

Példák:

```
parosit.be          parosit.ki          Magyarázat:
4                   3                   1 3 2 1 3 4 4 2
1 3 2 1 3 4 4 2                         1 1 2 3 3 4 4 2
                                        1 1 2 3 3 2 4 4
                                        1 1 2 2 3 3 4 4

parosit.be          parosit.ki          Magyarázat:
4                   2                   3 2 1 4 2 3 1 4
3 2 1 4 2 3 1 4                         3 2 1 1 2 3 4 4
                                        2 2 1 1 3 3 4 4
```

Rekurzió, dinamikus programozás

Pakolás (2005)

Nemes Tihamér OITV 2005. 2. forduló 2. korcsoport 3. feladat

Egy konténer raktárban egy sorban tárolják a konténereket, ahol *N* darab konténernek van hely. Mivel a konténerek kiszállítása az igénynek megfelelően történik, ezért egy adott időpontban a raktárban lévő *M* konténer tetszőlegesen helyezkedik el. A raktárosnak időnként át kell rendezni a raktárban lévő konténereket úgy, hogy folyamatosan egymás mellett legyenek. Az átrendezést bizonyos konténerek egyesével történő átrakásával lehet végezni. Egy *i*-edik konténerhelyen lévő konténert a *j*-edik konténerhelyre csak akkor rakhatunk át, ha köztük minden konténerhely üres. Ha az átrendezés során az *i*-edik konténerhelyen lévő konténert a *j*-edik konténerhelyre rakja át, ennek költsége *i*–*j* abszolút értéke. A raktáros az optimális átrendezést keresi, tehát amelyre az összköltség minimális.

Készíts programot *pakol* néven, amely kiszámítja a raktár optimális átrendezésének költségét!

A *pakol.be* szöveges állomány első sora a raktár konténerhelyeinek *N* számát tartalmazza ($2 \le N \le 10000$). A második sor pontosan *N* egész számot tartalmaz egy-egy szó-

közzel elválasztva. A sorban minden szám vagy 0, vagy 1. A raktárban az i-edik helyen akkor és csak akkor van konténer, ha a második sorban az *i*-edik szám 1. Legalább egy konténer van a raktárban.

A *pakol.ki* szöveges állomány első sora egyetlen számot tartalmazzon, az optimális átrendezés költségét! A második sor is egyetlen számot tartalmazzon, a legkisebb konténerhelynek a sorszámát, amely az átrendezés után konténert tartalmaz. Több megoldás esetén bármelyik kiírható.

Példa:

```
pakol.be                    pakol.ki
10                          8
1 0 1 1 0 0 1 1 0 1         2
```

Pakolás (2008-2f3)

Informatika OKTV 2008. 2. forduló 3. korcsoport 4. feladat

Kamionnal kell elszállítani tárgyakat. Ismerjük a kamion kapacitását, tehát azt a súlyt, amelynél több nem rakható a kamionra, és ismerjük az elszállítandó tárgyak súlyát. Az a cél, hogy a kamiont úgy pakoljuk meg tárgyakkal, hogy az összsúly a lehető legnagyobb legyen.

Készíts programot *pakol* néven, amely kiszámítja, hogy mekkora az a legnagyobb összsúly, amit a kamionnal elszállíthatunk! A program adja meg, hogy mely tárgyak kamionra rakásával érhető ez el.

A *pakol.be* szöveges állomány első sorában két egész szám van, a tárgyak *N* száma ($1 \le N \le 100$) és a kamion *K* kapacitása ($1 \le K \le 600$). A második sor pontosan *N* pozitív egész számot tartalmaz egy-egy szóközzel elválasztva. Az *I*-edik szám az *I*-edik tárgy súlya, ami nem nagyobb, mint a kamion *K* kapacitása.

A *pakol.ki* szöveges állomány első sorába a kamionnal elszállítható legnagyobb *S* összsúlyt kell írni! A második sorba az *S* összsúlyt adó pakolásban szereplő tárgyak *M* számát kell írni! A harmadik sorba a kamionra pakolt *M* tárgy sorszámát kell írni tetszőleges sorrendben, egy-egy szóközzel elválasztva! Több megoldás esetén bármelyik megadható.

Példa:

```
pakol.be                    pakol.ki
6 20                        19
18 12 4 7 10 5              3
                            3 6 5
```

Dobozok

Nemes Tihamér OITV 2014. 3. forduló 2. korcsoport 4. feladat

Van *N* darab téglatest alakú dobozunk, mindegyiknek ismerjük a méreteit. A dobozokat egymásba szeretnénk pakolni, hogy minél kevesebb helyet foglaljanak. Egy dobozba olyan másik doboz tehető, amelynek mindhárom mérete kisebb az adott doboz méreteinél, de a dobozok tetszőlegesen forgathatók (azaz pl. egy (7,5,3) méretű dobozba betehető egy (4,2,6) méretű doboz).

Készíts programot *dobozok* néven, amely megadja a legtöbb dobozból álló dobozsorozatot, amelyek egymásba pakolhatók!

A standard bemenet első sorában a dobozok száma ($1<N\leq2000$) van. A következő N sor mindegyike 3 pozitív egész számot tartalmaz (egy-egy szóközzel elválasztva), az egyes dobozok méretét ($1\leq x_i, y_i, z_i\leq10000$).

A standard kimenet első sorába a legtöbb dobozból álló dobozsorozat H elemszámát kell írni, amelyek egymásba pakolhatók! A következő sorba pontosan H számot kell írni, a leghosszabb ilyen dobozsorozatban szereplő dobozok sorszámait! Az első legyen közülük a legnagyobb doboz, s minden dobozt olyan kövessen, amely az adott dobozba belefér! Több megoldás esetén bármelyik megadható.

Példa:

```
bemenet          kimenet
6                3
2 3 8            5 3 2
1 4 2
4 2 7
2 1 3
5 3 9
1 2 3
```

Hasonlítsuk össze a megoldást a leghosszabb növekvő részsorozat kiválasztásának algoritmusával! Oldjuk meg a feladatot gráfbejárással is!

Visszalépéses keresés

Üvegválogatás (2002)

Nemes Tihamér OITV 2002. 2. forduló 2. korcsoport 3. feladat

Egy palackozó üzembe N db ládában érkeznek be az üvegek. Alakjuk szerint K fajta üveget különböztetnek meg. Ismert, hogy az egyes ládákban hány darab üveg van az egyes fajtákból. A palackozáshoz az üvegeket a fajtájuk szerint szét kell válogatni. Minden üvegfajta számára kijelölnek egy ládát (a meglévő N közül), és a többi ládából az adott fajta üveget ebbe a ládába rakják át. A cél az, hogy a lehető legkevesebb üveget kelljen átrakni a válogatás során.

Írj programot *valogat* néven, amely kiszámítja, hogy legkevesebb hány üveget kell átrakni, és ez mely ládák kijelölésével érhető el!

A *valogat.be* állomány első sorában a ládák ($2\leq N\leq10$) és a fajták ($2\leq K\leq N$) száma van. A következő N sor mindegyike egy-egy láda tartalmát írja le. Minden sor pontosan K db nemnegatív egész számot tartalmaz, ahol a J-edik szám a ládában található J-edik üvegfajta darabszáma ($1\leq J\leq K$). (A ládák elég nagyok ahhoz, hogy mindegyikbe tetszőleges számú üveg beleférjen.)

A *valogat.ki* állományba két sort kell írni. Az első sorban a válogatáshoz minimálisan szükséges átrakások száma legyen. A második sor pontosan K számot tartalmazzon egy-egy szóközzel elválasztva, ahol a J-edik szám annak a ládának a sorszáma legyen, amelyiket a J-edik üvegfajta számára kijelöltünk. Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni.

Példa:

```
valogat.be        valogat.ki
5 4               58
1 7 2 6           5 1 3 4
3 1 2 4
3 1 5 6
6 4 7 8
6 7 1 4
```

Hasonlítsuk össze a megoldást a következő feladat (*Üvegválogatás* (2012-2f3) megoldásával!

Üvegválogatás (2012)

Informatika OKTV 2012. 2. forduló 3. korcsoport 3. feladat

Egy üzletben válogatás nélkül gyűjtötték össze az üres üvegeket *N* ládában. Mivel az üvegfajták száma is *N*, ezért az azonos fajtájú üvegeket egy-egy ládában el lehet tárolni. Ezért szét akarják válogatni az üvegeket, hogy minden ládában csak azonos fajtájú legyen, de úgy, hogy a lehető legkevesebb üveget kelljen átrakni más ládába.

Készíts programot *valogat* néven, amely kiszámítja, hogy legkevesebb hány darab üveget kell másik ládába átrakni, és megadja, hogy ehhez az egyes fajtákat melyik ládába kell gyűjteni!

A *valogat.be* szöveges állomány első sorában egy egész szám, a ládák *N* száma ($1 \le N \le 8$) van. A következő *N* sor mindegyike pontosan *N* nemnegatív egész számot tartalmaz egy-egy szóközzel elválasztva. Az *I*+1-edik sorban a *J*-edik szám az *I*-edik ládában lévő *J*-fajta üvegek száma. Minden szám értéke legfeljebb 5000. A ládákat és az üvegfajtákat is az 1, …, *N* számokkal azonosítjuk.

A *valogat.ki* szöveges állomány első sora egy egész számot tartalmazzon, a kívánt szétválogatáshoz szükséges átrakások minimális számát! A második sor pontosan *N* egész számot tartalmazzon (egy-egy szóközzel elválasztva): az *I*-edik szám annak az üvegfajtának a sorszáma legyen, amelyet az *I*-edik ládába rakunk!

Példa:

```
valogat.be          valogat.ki
3                   182
15 30 8             3 2 1
55 80 10
50 60 12
```

Hasonlítsuk össze a megoldást az előző feladat (*Üvegválogatás* (2022-2f2) megoldásával!

Gráfok

Az alábbi feladatokon kívül a dinamikus programozáshoz besorolt

Kockák (1999-2f2, 2f3) és

Dobozok (2014-3f2)

feladatokat célszerű megoldani gráfalgoritmusokkal is.

Konténer rendezés (2003)

Nemes Tihamér OITV 2003. 2. forduló 2. korcsoport 4. feladat

Egy konténer raktárban *N* db konténer van egy sorban tárolva. A konténereket el akarják szállítani, ezért mindegyikre rá van írva, hogy melyik városba kell szállítani. A városokat 1-től 4-ig sorszámozzák. A konténereket át kell rendezni úgy, hogy balról jobbra először az 1-essel, majd a 2-essel, aztán a 3-assal, végül a 4-essel jelölt konténerek álljanak. A raktár majdnem tele van, csak az utolsó konténer után van egy konténer számára szabad hely. A rendezést a konténerek fölött mozgatható daruval végezhetjük, amely egy lépésben kiemel a helyéről egy konténert és átteszi azt a szabad helyre, ezzel az átmozgatott konténer helye lesz szabad.

Írj programot *kontener* néven, amely kiszámítja, hogy legkevesebb hány lépésben lehet rendezni a konténersort! A rendezés végén a szabad helynek a sor végén kell lennie!

A *kontener.be* szöveges állomány első sorában a konténerek *N* száma van ($1 \le N \le 10000$). A második sor N egész számot tartalmaz egy-egy szóközzel elválasztva. Az *i*-edik szám annak a városnak a sorszáma (1 és 4 közötti érték), ahova az *i*-edik konténert szállítani kell.

A *kontener.ki* állomány első és egyetlen sorába a rendezés végrehajtásához minimálisan szükséges lépések számát kell írni!

Példa:

```
kontener.be                    kontener.ki
12                             7
1 2 1 3 3 2 2 4 3 4 1 4
```

Könyvtári pakolás (2012)

Nemes Tihamér OITV 2012. 2. forduló 2. korcsoport 3. feladat

Egy könyvtár polcán egy sorban *N* darab könyv van, de nem a kívánt sorrendben. A könyvtáros minden könyvre ráragasztott egy cetlit, amire ráírta, hogy a helyes sorrendben hányadik helyen kell majd lennie. A kívánt sorrend kialakítását párok cseréjével akarja megvalósítani.

Készíts programot *pakol* néven, amely kiszámítja, hogy legkevesebb hány cserével lehet kialakítani a kívánt sorrendet!

A *pakol*.be szöveges állomány első sorában a könyvek *N* ($2 \le N \le 30000$) száma van. A második sor pontosan *N* különböző pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, az *i*-edik szám értéke az *i*-edik könyv helyes sorrendbeli sorszáma.

A *pakol.ki* szöveges állomány első és egyetlen sora egy egész számot tartalmazzon, azt a legkisebb *M* számot, amire a bemenetben megadott könyvsorozat *M* számú cserével a kívánt sorrendbe rakható!

Példa:

```
pakol.be                       pakol.ki
10                             7
7 10 1 3 2 8 4 9 6 5
```

SZÁLLÍTÁS

A *Programozási ismeretek versenyzőknek* (Műszaki Kiadó, 2015) könyvben szereplő feladatok:

OKTV 1999-2f3 Kamion
OKTV 2008-3f3 Robot
OKTV 2012-1f3 Taxi

Első fordulós feladatok

A *Raktár* (OITV 2013-1f2) feladat a dinamikus programozásra vonatkozó feladatok között található.

Fűrészmalom

Informatika OKTV 2011. 1. forduló 3. korcsoport 5. feladat

A folyó mentén kitermelt fát *N* helyen gyűjtik össze és szállítják a folyón lefelé úsztatva az első gyűjtőhelyre, ahol fűrészmalomban végzik a feldolgozást. A vállalat elhatározta, hogy további fűrészmalmot állít üzembe néhány gyűjtőhelyen. Minden gyűjtőhelyről a fát a folyón lefelé haladva az első fűrészmalomba fogják szállítani. A szállítási költség a megtett távolság és a tömeg szorzata. Ismerjük az egyes gyűjtőhelyek elhelyezkedését (az elsőtől vett távolságot km-ben) és azt, hogy mennyi fa keletkezik évente az egyes gyűjtőhelyen. Kiszámítandó, hogy hova kell telepíteni az új fűrészmalmokat, hogy a szállítási összköltség a lehető legkisebb legyen!

A táblázat első sora a gyűjtőhely sorszámát, a második sor az 1. gyűjtőhelytől vett távolságot, a harmadik sor a gyűjtőhelyen keletkező fa tömegét tartalmazza.

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
0	2	3	6	7	20	22	34	35	44	57	66	88	100
1	2	3	44	5	6	33	18	9	10	11	2	13	44

Hová kell telepíteni
a) további 1 fűrészmalmot;
b) további 2 fűrészmalmot;
c) további 3 fűrészmalmot;
d) további 4 fűrészmalmot?

Mohó algoritmus

Lift

Nemes Tihamér OITV 2000. 2. forduló 2. korcsoport 4. feladat
Informatika OKTV 2000. 2. forduló 3. korcsoport 1. feladat

A két feladat megegyezik egymással. 2000-ben egy átlagos PC-n a 3. korcsoport nagy tesztjének adatai nem fértek be egyszerre a memóriába, így a feldolgozást beolvasás közben kellett elvégezni.

Egy sokemeletes házban szokatlan módon üzemeltetik a liftet. A lift az első szintről indult és mindig felmegy a legfelső szintre, majd visszatér az első szintre. Menet

közben megáll minden olyan szinten, amelyik úti célja valamelyik liftben tartózkodó utasnak. Hasonlóan, olyan szinten is megáll, ahonnan utazni szándékozik valaki az aktuális irányban, feltéve, hogy még befér a liftbe (figyelembe véve az adott szinten kiszállókat).

Készíts programot *lift* néven, amely kiszámítja, hogy legkevesebb hány menet (1 menet = egyszer felmegy, majd lejön) szükséges ahhoz, hogy minden várakozó embert elszállítson a lift.

A *lift.be* bemeneti állomány első sorában két szám van, N és K. N az épület szintjeinek ($2 \le N \le 500$) száma, K ($1 \le K \le 20$) pedig a lift kapacitása. A további N sor tartalmazza az egyes szinteken várakozó emberek adatait. Az állomány i-edik sorában azoknak a szinteknek a sorszáma van felsorolva, ahová az i–1-edik szintről utazni akarnak. A felsorolást minden sorban egy 0 szám zárja. Minden sorban legfeljebb 500 szám lehet, tetszőleges sorrendben.

A *lift.ki* állomány egyetlen sort tartalmazzon, a legkevesebb menetek számát, amely az emberek elszállításához szükséges!

Példa:

```
lift.be            lift.ki
6 2                3
2 3 2 0
1 3 0
1 2 0
2 5 0
3 6 2 0
1 2 3 0
```

Hasonlítsuk össze a megoldást a *Taxi* (2012-1f3) feladat megoldásával!

Délkert

Nemes Tihamér OITV 2008. 3. forduló 2. korcsoport 3. feladat

A DélKert szövetkezetben N termelő termel gyümölcsöt, amit a szövetkezet két hűtőházban gyűjt össze. Az i-edik termelő a leszedett gyümölcsöt az A hűtőházba a_i, a B hűtőházba b_i idő alatt tudja beszállítani. A hűtőházak kapacitása korlátozott, az A hűtőház N_1, a B pedig N_2 termelőtől tud gyümölcsöt fogadni. Az a cél, hogy minden termelőtől a lehető leghamarabb hűtőházba kerüljön a leszedett gyümölcs.

Készíts programot *delkert* néven, amely kiszámítja, hogy az egyes termelőknek melyik hűtőházba kell szállítnia a leszedett gyümölcsöt, hogy az összes termelőtől a gyümölcs a lehető legkorábban hűtőházba kerüljön!

A *delkert.be* szöveges állomány első sorában a termelők N ($1 \le N \le 10000$) száma, az A hűtőház N_1, és a B hűtőház N_2 kapacitása van ($N_1+N_2 \ge N$). A második és a harmadok sor pontosan N pozitív egész számot tartalmaz (egy-egy szóközzel elválasztva). A második sorban az i-edik szám azt adja meg, hogy az i-edik termelő mennyi idő alatt tudja a gyümölcsöt az A hűtőházba szállítani. A harmadik sorban az i-edik szám azt adja meg, hogy az i-edik termelő mennyi idő alatt tudja a gyümölcsöt az B hűtőházba szállítani. A második és a harmadik sorban minden szám értéke legfeljebb 1000 lehet.

A *delkert.ki* szöveges állományba három sort kell írni! Az első sorba azt a legkisebb K számot kell írni, amelyre teljesül, hogy minden termelő legfeljebb K idő alatt el tudja juttatni a gyümölcsét valamelyik hűtőházba, ha alkalmas beosztás szerint szállí-

tanak! A második sorba azoknak a termelőknek a sorszámát kell kiírni (egy-egy szóközzel elválasztva), akik az *A*, a harmadik sorba pedig azokét, akik a *B* hűtőházba szállítják a gyümölcsöt! A sorokba a számokat tetszőleges sorrendben ki lehet írni. Ha több megoldás is van, bármelyik megadható.

Példa:

```
delkert.be                    delkert.ki
10 4 7                        6
2 8 9 2 3 2 4 3 6 5           4 5 6 8
6 3 2 7 6 9 3 8 5 2           1 2 3 7 9 10
```

Dinamikus programozás

Raktár (2013)

Nemes Tihamér OITV 2013. 1. forduló 2. korcsoport 4. feladat

Egy áruházlánc két raktárból látja el a hozzá tartozó áruházakat egy adott termékkel. Az *A* raktárban *M*, a *B* raktárban *N* darab termék van, az áruházak száma *M*+*N*. Minden áruházba egy darab terméket kell szállítani, és ismerjük minden áruházra az *A*, illetve a *B* raktárból történő szállítás költségét (*A*(*i*), illetve *B*(*j*)).

a) Add meg az alábbi adatokra, hogy melyik raktárból kell szállítani az egyes áruházakba, hogy a szállítás összköltsége minimális legyen! Add meg a minimális költséget is!
   N=3, M=3
   A=1 2 3 4 3 2
   B=1 3 2 7 3 1
b) Definiáld rekurzívan azt a táblázatot (*Költség*(*I*, *J*)), amely annak minimális költségét tartalmazza, hogy az első *I*+*J* áruházba kell szállítani árut, *I* darabot az első, *J* darabot a második raktárból!

Robot (2014)

Informatika OKTV 2014. 3. forduló 3. korcsoport 4. feladat

Egy négyzetrácsos elrendezésű raktárban robot alkalmazásával dolgoznak. A raktár egy mezőjét a négyzetrácsos elrendezésben a mező (*x*, *y*) koordinátájával azonosítják, ahol *x* a sor, *y* pedig az oszlop koordinátája, a sorokat alulról felfelé, az oszlopokat balról jobbra sorszámozzuk, a bal alsó mező koordinátái (1, 1). A robot egy lépésben a szomszédos mezőre léphet vagy felfelé, vagy jobbra. A raktár *N* mezőjén van tárolva áru. A robotnak az (1, 1) mezőről indulva, legfeljebb *L* lépés megtételével olyan útvonalon kell haladnia, amelyen a lehető legtöbb árut tartalmazó mező van.

Készíts programot *robot* néven, amely megoldja a feladatot!

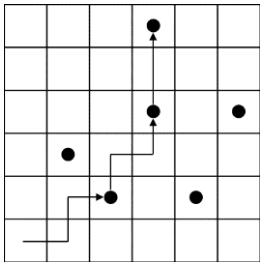
A *robot.be* szöveges állomány első sorában két egész szám van, az árut tartalmazó mezők *N* száma ($1 \le N \le 10000$) és a robot által megtehető lépések *L* maximuma ($1 \le L \le 2000000$). A további *N* sor mindegyikében két pozitív egész szám van (egy szóközzel elválasztva), egy olyan mező koordinátái, ahol áru van. Minden koordináta értéke legfeljebb 2000000. A mezőket az 1, ..., *N* számokkal is azonosítjuk.

A *robot.ki* szöveges állomány első sorába azt a legnagyobb *M* számot kell írni, ahány árut tartalmazó mezőn áthaladhat a robot. A második sorba pontosan *M* számot kell írni egy-egy szóközzel elválasztva, a bejárás sorrendjében a mezők bemenetbeli sorszámait. Több megoldás esetén bármelyik megadható.

Példa:

```
robot.be
6 8
3 2
2 3
5 2
4 4
6 4
4 6
```

```
robot.ki
3
1 4 6
```



A megoldásban a leghosszabb növekvő részsorozat kiválasztásának egy hatékony algoritmusát használtuk fel. Oldjuk meg a feladatot dinamikus programozással is!

Gráfok

Raktár (2012)

Nemes Tihamér OITV 2012. 3. forduló 2. korcsoport 4. feladat

Egy megye településeiről tudjuk, hogy bármely településről bármelyik másikra pontosan egy útvonalon lehet eljutni. Egy vállalat az összes településen nyit termelő üzemet. Tudjuk, hogy melyik településen van a raktár, ahova az egyes üzemek szállítanák az árut. Ismerjük, hogy melyik településen mennyi terméket gyárthatnak. Az egyes utakon egy nap maximum *M* mennyiségű termék szállítható.

Készíts programot *raktar* néven, amely kiszámítja, hogy mekkorára kell építeni a raktárt, hogy a lehető legtöbb árut befogadja!

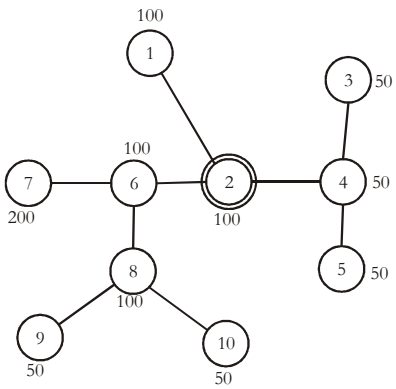
A *raktar.be* szöveges állomány első sorában a települések *N* száma ($1 \le N \le 1000$), az utakon szállítható termékek *M* maximális száma ($1 \le M \le 100\,000$) és a raktáros település *R* sorszáma ($1 \le R \le N$) van. A második sor pontosan *N* egész számot tartalmaz (egy-egy szóközzel elválasztva), az *i*. szám az *i*. településen gyártott áru mennyisége ($1 \le mennyiség \le 1000$). A következő *N*–1 sorban két egész szám van (egy-egy szóközzel elválasztva): két település sorszám, amelyek között út van ($1 \le A \ne B \le N$).

A *raktar.ki* szöveges állomány egyetlen sorába a maximális raktárméretet kell írni, ami az üzemekből szállított áruval egy nap megtölthető.

Példa:

```
raktar.be
10 200 2
100 100 50 50 50 100 200 100 50 50
1 2
3 4
4 5
4 2
2 6
6 7
6 8
8 9
10 8
```

```
raktar.ki
550
```



Hasonlítsuk össze a megoldást az *Üzletek* (2012-3f3) feladat megoldásával!

Szállítás (2012)

Informatika OKTV 2012. 3. forduló 3. korcsoport 1. feladat

Az ország *N* városa között különböző teherbírású utak vannak. Két város között árut szeretnénk szállítani a lehető legnagyobb kapacitású teherautóval olyan útvonalon, ahol az autó tehersúlya nem nagyobb, mint az egyes utak teherbírása.

Készíts programot *szallitas* néven, amely adott *A* és *B* városra megadja, hogy maximum mekkora tehersúlyú teherautó közlekedhet közöttük és merre kell menni!

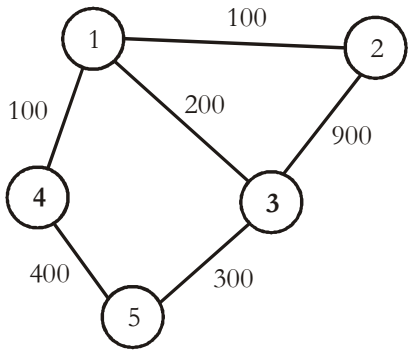
A *szallitas.be* állomány első sorában a városok száma ($1 \le N \le 100$), a köztük levő utak száma ($1 \le M \le 10000$), a kezdő és a cél város sorszáma ($1 \le A \ne B \le N$) van, egy-egy szóközzel elválasztva. A következő *M* sor mindegyikében egy-egy út leírása található: azon két város sorszáma ($1 \le sorszám \le N$), amelyek között a kétirányú út vezet, valamint az út teherbírása ($1 \le tehebírás \le 1000$).

A *szallitas.ki* szöveges állományba 2 sort kell írni. Az elsőbe a maximális tehersúly kerüljön, a másodikba pedig az oda vezető úton levő városok sorszáma, egy-egy szóközzel elválasztva, az útvonal sorrendjében (azaz az első sorszám biztosan *A*, az utolsó sorszám biztosan *B* legyen)! Több megoldás esetén bármelyik megadható.

Példa:

```
szallitas.be
5 6 3 4
2 1 100
1 4 100
3 1 200
3 5 300
2 3 900
4 5 400
```

```
szallitas.ki
300
3 5 4
```



Üzletek

Informatika OKTV 2012. 3. forduló 3. korcsoport 4. feladat

Egy megye településeiről tudjuk, hogy bármely településről bármelyik másikra pontosan egy útvonalon lehet eljutni. Egy üzlethálózat minél több településen szeretne üzletet nyitni. Tudjuk, hogy melyik településen van a raktár, ahonnan az egyes üzletekbe szállítaná az árut. Ismerjük, hogy melyik településen mekkora hasznot fog hozni az üzlet, valamint hogy melyik út használatáért mekkora úthasználati díjat kell fizetni (ez nem függ attól, hogy hány településről viszik ezen az úton az árut). Az összhaszon az üzletekben termelt haszonból levonva az úthasználati díjakat.

Készíts programot *uzlet* néven, amely kiszámítja, hogy mekkora az elérhető legnagyobb haszon és ehhez mely településeken kell üzletet nyitni!

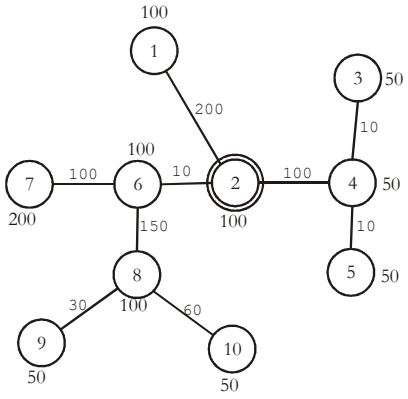
Az *uzlet.be* szöveges állomány első sorában a települések *N* száma ($1 \le N \le 10000$) és a raktáros település *R* sorszáma ($1 \le R \le N$) van. A második sor pontosan *N* egész számot tartalmaz (egy-egy szóközzel elválasztva), az *i*-edik szám az *i*. településen elérhető hasznot. A következő *N*–1 sorban három egész szám van (egy-egy szóközzel elválasztva): két település sorszáma, amelyek között kétirányú út van, és az érte fizetendő úthasználati díj.

Az *uzlet.ki* szöveges állomány első sorába az elérhető legnagyobb hasznot kell írni! A második sorba a megnyitandó üzletek *M* számát kell írni, a harmadikba *M* sorszámot, egy-egy szóközzel elválasztva: azon települések sorszámát, ahol üzletet nyitunk! A számok sorrendje közömbös. Több megoldás esetén bármelyik megadható.

Példa:

```
uzlet.be
10 2
100 100 50 50 50 100 200 100 50 50
1 2 200
3 4 10
4 5 10
4 2 100
2 6 10
6 7 100
6 8 150
8 9 30
10 8 60

uzlet.ki
320
6
2 3 4 5 6 7
```



Hasonlítsuk össze a megoldást a *Raktár* (2012-3f2) feladat megoldásával!

Újság

Nemes Tihamér OITV 2013. 2. forduló 2. korcsoport 2. feladat

Egy folyóirat terjesztő cég vasúton szállítja minden nap az újságokat a megfelelő címekre. Az újságot egy központi helyen nyomtatják, vonatra rakják és elküldik. A vasúti csomópontokban átrakják a megfelelő irányokba továbbinduló szerelvényekre. Ismerjük minden vasúti csomópontra, hogy közvetlenül honnan kapja az újságcsomagot.

Írj programot *ujsag* néven, amely megadja:

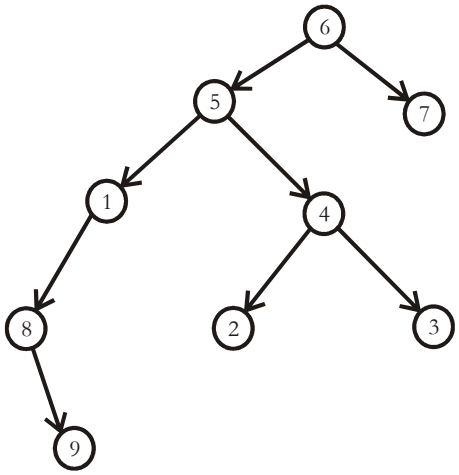
a) Az adott *A* csomópontból hány helyre visznek még tovább újságot?
b) Adott *A* csomópontba küldendő újságokat hányszor kell átrakni másik vonatra?
c) Adott *A* és *B* csomópontba küldendő újságokat legtovább melyik csomópontig vihetik együtt?

Az *ujsag.be* szöveges állomány első sorában a csomópontok száma ($1 \le N \le 1000$), valamint két csomópont sorszáma van ($1 \le A, B \le N$), egy-egy szóközzel elválasztva. A következő $N-1$ sor mindegyikében két csomópont *I* és *J* sorszáma van ($1 \le I \ne J \le N$), ami azt jelenti, hogy az *I*-edik csomópontba a *J*-edik csomópontból szállítják az újságokat.

Az *ujsag.ki* szöveges állomány első sorába az a), a másodikba a b), a harmadikba pedig a c) részfeladat eredményét kell írni!

Példa:

```
ujsag.be        ujsag.ki
9 1 3           2     a 8.-ba és a 9.-be
1 5             1     az 5.-ben
2 4             5     az 5.-ig
3 4
5 6
7 6
4 5
9 8
8 1
```



ÜTEMEZÉS

A *Programozási ismeretek versenyzőknek* (Műszaki Kiadó, 2015) könyvben szereplő feladatok:

OITV 2001-3f2	Fazekas
OITV 2002-2f2	Ütemezés
OITV 2004-3f2	Ütemezés
OITV 2010-2f2	Ütemezés
OITV 2010-3f2	Gépkölcsönzés
OKTV 2009-2f3	Munka

Néhány alábbi feladat az intervallum-feladatokhoz is besorolható.

Első fordulós feladatok

Fazekas (2006)

Nemes Tihamér OITV 2006. 1. forduló 2. korcsoport 4. feladat

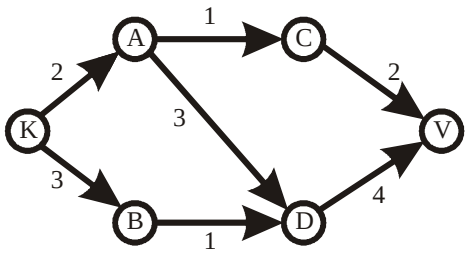
Egy fazekas műhelyében sorban várakoznak a kiégetésre váró tárgyak. Minden tárgyról tudjuk, hogy mennyi az a legkevesebb idő, ami a kiégetéséhez kell. Az égetésre váró tárgyakat az érkezésük sorrendjében kell kiégetni. Egyszerre több tárgyat is rakhatunk a kemencébe, azonban legfeljebb annyit, amennyi a kemence *K* kapacitása. Az égetési idő egy menetben mindig a kemencébe rakott tárgyak minimális égetési idejének a maximuma kell legyen.

Jelöljük *Opt*(*i*)-vel az első *i* tárgy kiégetéséhez szükséges minimális időt!

a) Add meg, hogy 10,8,20,25,30,12,40 égetési idők és *K*=3 esetén hogyan néz ki az *Opt* vektor!
b) Adj képletet *Opt*(*i*) kiszámítására tetszőleges *K* esetén, *Idő*(*i*)-vel jelöld az *i*-edik tárgy kiégetéséhez szükséges időt!

Ütemezés (2008-1f2)

Nemes Tihamér OITV 2008.
1. forduló 2. korcsoport 1. feladat

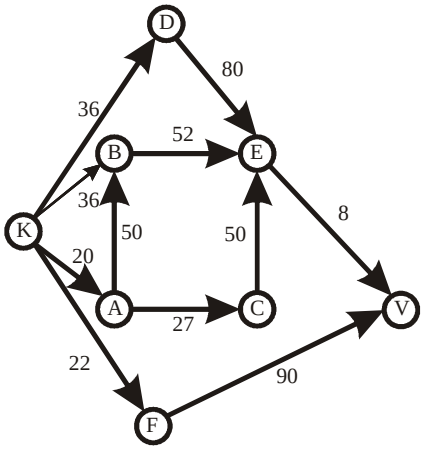


A mellékelt ábra egy feladat részfeladatainak ütemezését mutatja. Az ábrán *K* betű jelöli a megoldás kezdetét, *V* pedig a végét. Az egyes részfeladatokat az ábécé betűivel jelöljük. Megadjuk minden részfeladatra, hogy valamely előző részfeladat után mennyi idővel lehet elvégezni (például az ábrán az *A* részfeladat a *K* kezdés után 2 időegységgel kezdhető, a *C* pedig az *A* elkezdése után 1 időegységgel). Add meg a mellékelt ábra alapján, hogy

a) mikor kezdhető el az egyes részfeladatok legkorábban (a K a 0. időpontban kezdődik)!
b) melyek azok a részfeladatok, amelyek később kezdése nem befolyásolja a *V* részfeladat lehető legkorábbi elkezdését, és mennyivel lehet őket később kezdeni!

Ütemezés (2008-1f3)

Informatika OKTV 2008.
1. forduló 3. korcsoport 1. feladat



A mellékelt ábra egy feladat részfeladatainak ütemezését mutatja. Az ábrán *K* betű jelöli a megoldás kezdetét, *V* pedig a végét. Az egyes részfeladatokat az ábécé betűivel jelöljük. Megadjuk minden részfeladatra, hogy valamely előző részfeladat után mennyi idővel lehet elvégezni (például az ábrán az *A* részfeladat a *K* kezdés után 20 időegységgel kezdhető, a *C* pedig az *A* elkezdése után 27 időegységgel). Add meg a mellékelt ábra alapján, hogy

a) mikor kezdhetők el az egyes részfeladatok legkorábban (a *K* a 0. időpontban kezdődik)!
b) melyek azok a részfeladatok, amelyek később kezdése nem befolyásolja a *V* részfeladat lehető legkorábbi elkezdését, és mennyivel lehet őket később kezdeni!

Pakolás

Informatika OKTV 2009. 1. forduló 3. korcsoport 4. feladat

Egy fazekas műhelyében sorban várakoznak az kiégetésre váró tárgyak. Minden tárgyról tudjuk, hogy mennyi az a legkevesebb idő, ami a kiégetéséhez kell (*Idő*(*i*)). Az égetésre váró tárgyakat az érkezésük sorrendjében kell kiégetni. Egyszerre több tárgyat is rakhatunk a kemencébe, azonban legfeljebb annyit, amennyi a kemence adott kapacitása (*K*). Az égetési idő egy menetben mindig a kemencébe rakott tárgyak minimális égetési idejének a maximuma kell legyen.

Példa:
7 tárgy, 3 fér be, idők: 10,8,20,25,30,12,40
Égetési idő: 75
Egy lehetséges égetési sorrend: 1–2, 3–4, 5–6–7.

a) Add meg, hogy az alábbi adatok alapján mennyi lesz a minimális égetési idő és adj is meg egy égetési sorrendet!
A1: 5 tárgy, 2 fér be, idők: 15,25,15,25,15
A2: 5 tárgy, 2 fér be, idők: 20,25,15,25,20
A3: 7 tárgy, 3 fér be, idők: 15,10,20,15,30,25,20
b) Fogalmazd meg képlettel (*Opt*(*i*)=…), hogy mennyi az első *i* tárgy kiégetésének lehető legkisebb összideje!

Munka (2012)

Nemes Tihamér OITV 2012. 1. forduló 2. korcsoport 3. feladat

Egy vállalkozó megrendeléseket fogad. Minden megrendelt munkát pontosan egy nap alatt tud elvégezni. Minden megrendelés tartalmazza, hogy az adott munkát mikorra kell elvégezni, és mekkora hasznot eredményez, ha a munkát határidőig elvégzi a vállalkozó. A vállalkozó a megrendelések közül ki akarja választani azokat, amelyeket el tud végezni határidőre és a lehető legtöbb hasznot adják.

Add meg, hogy az alábbi megrendelések esetén mennyi lehet a haszon és ehhez mely munkákat kell elvégezni!

a) N=8, munkák: (2,1),(2,3),(2,8),(3,7),(3,10),(4,5),(4,11),(5,4)
   Magyarázat: határidő a 2. nap, összeg 1 forint, ...,
   határidő az 5. nap, összeg 4 forint.
b) N=7, munkák: (5,1),(5,2),(5,3),(5,4),(5,5),(5,6),(5,7)
c) N=7, munkák: (2,1),(3,2),(3,3),(4,4),(4,5),(5,6),(5,7)
d) N=7, munkák: (2,9),(3,2),(3,4),(4,3),(4,6),(5,5),(5,7)
e) N=9, munkák: (2,9),(3,9),(3,9),(4,3),(4,6),(4,9),(5,5),(5,7),(5,9)

Munkavállalás

Informatika OKTV 2013. 1. forduló 3. korcsoport 5. feladat

Egy vállalkozó 1 napos munkákat vállal. Ismerjük mindegyik munka (M darab) határidejét ($1 \le H(i) \le N$). Egy nap csak 1 munkát végezhet. Add meg, hogy az M munka közül N napra maximum hány munkát vállalhat el és mely napokra melyikeket! A feladat megoldására két algoritmust készítettünk.

1. megoldás:

```
Kiválogatás(N,M,H,Db,Nap)
  DB:=0; Nap():=(0,…,0)
  Ciklus i=1-től M-ig
    Ciklus amíg H(i)>0 és Nap(H(i))>0
      H(i):=H(i)-1                                  {*}
    Ciklus vége
    Ha H(i)>0 akkor Db:=Db+1; Nap(H(i)):=i
  Ciklus vége
Eljárás vége.
```

2. megoldás:

```
Kiválogatás(N,M,H,Db,Nap)
  S():=(1,2,…,M)
  Ciklus i=1-től M-1-ig
    min:=i
    Ciklus j=i+1-től M-ig
      Ha H(j)<H(min) akkor min:=j                   {**}
    Ciklus vége
    Csere(H(i),H(min)); Csere(S(i),S(min))
  Ciklus vége
  DB:=1; Nap(Db):=S(1)
  Ciklus i=2-től N-ig
    Ha Db<H(i) akkor Db:=Db+1; Nap(Db):=S(i)       {***}
  Ciklus vége
Eljárás vége.
```

a) Fogalmazd meg, hogy mi és hol szerepel a *Nap* tömbben az első, illetve a második változat esetén!
b) Az elvállalható munkákat a határidejükhöz képest mikorra osztja be a két változat?
c) Hogyan függ a *-gal jelölt sorok végrehajtási száma *N*-től, illetve *M*-től? (A kérdésre ilyen jellegű válaszok adhatók: N, $N \cdot M$, N^2+M^3, …)
d) Az első algoritmus *-os sorhoz tartozó végrehajtási száma milyen *N* és *M* értékekre kisebb, mint a második algoritmus *-os sorokhoz tartozó végrehajtási száma?

Mohó algoritmus

Ütemezés (2002-2f3)

Informatika OKTV 2002. 2. forduló 3. korcsoport 3. feladat

Mekk Elek ezermester népszerű vállalkozó, sokan keresik fel megrendelésekkel. Minden megrendelt munkának ismeri a kezdési és a befejezési idejét. A mester a következő évre szóló megrendelések közül a lehető legtöbbet akarja elvállalni, de egyszerre csak egy munkán tud dolgozni.

Írj programot *utemez* néven, amely meghatározza a munkák egy lehető legnagyobb elemszámú részhalmazát úgy, hogy az összes kiválasztott munka elvégezhető legyen.

Az *utemez.be* állomány első sora a megrendelések *N* számát ($1 \le N \le 10000$) tartalmazza. A következő *N* sor mindegyike két pozitív egész számot tartalmaz, a megrendelt munka *K* kezdési, illetve *B* befejezési idejét ($1 \le K \le B \le 365$), tehát a *J*-edik munkát az állomány *J*+1-edik sora írja le.

Az *utemez.ki* állomány első sorában a kiválasztott munkák *M* száma legyen. A második sorba *M* számot, a kiválasztott munkák sorszámát kell írni egy-egy szóközzel elválasztva, tetszőleges sorrendben. Ha több megoldás is van, közülük egy tetszőlegeset kell kiírni.

Példa:

```
utemez.be          utemez.ki
6                  3
2 3                6 3 4
2 4
5 7
3 4
2 2
1 2
```

Málna

Informatika OKTV 2007. 3. forduló 3. korcsoport 2. feladat

Egy gyümölcslevet gyártó üzem naponta *K* kg málnát tud feldolgozni. *N* termelő ajánlotta fel a málnáját, az is lehet, hogy eltérő árakon. Mivel a málna romlandó, ezért az árusok naponta egységesen *F* forintot engednek az árból. Az üzem a lehető leghamarabb fel akarja dolgozni a beérkezett málnát, de ezen belül azért arra törekszik, hogy a lehető legkevesebb pénzt kelljen fizetnie a termelőknek. Feltehető, hogy az ár az engedményekkel sem megy le 0 forintra.

Készíts programot *malna* néven, amely megadja, hogy az üzem melyik napon melyik termelőtől vegyen málnát, hogy a kiadása a lehető legkisebb legyen!

A *malna.be* szöveges állomány első sorában az üzem napi kapacitása ($100 \le K \le 1000$), a termelők száma ($1 \le N \le 5000$), valamint a napi engedmény ($1 \le F \le 100$) szerepel. A következő *N* sor mindegyikében két egész szám van egy szóközzel elválasztva, az adott termelő által felajánlott málna mennyisége ($1 \le mennyiség \le 200$), illetve kilónkénti ára ($500 \le ár \le 1500$).

A *malna.ki* szöveges állomány első sorába az összes málna feldolgozásához szükséges napok *M* számát és az üzem által kifizetett teljes vételárat kell írni! A következő *M* sor mindegyikében egy-egy nap adatai szerepeljenek, egy-egy szóközzel elválasztva: azon termelők sorszáma és a tőlük vásárolt málnamennyiség, akiktől azon a napon vesz málnát az üzem!

Példa:

```
malna.be          malna.ki
100 5 100         3 144000
40 800            3 20 4 70 2 10
120 600           2 100
20 500            2 10 5 30 1 40
70 500
30 700
```

Ütemezés (2010-2f3)

Informatika OKTV 2010. 2. forduló 3. korcsoport 4. feladat

Egy vállalkozó alkatrészek gyártásával foglalkozik. Minden alkatrészen kétféle műveletet kell elvégeznie, *A* és *B* műveletet. Mindkét művelet elvégzésére egy-egy munkagépe van, amelyek egymástól függetlenül tudnak dolgozni. Minden alkatrészen először az *A* műveletet kell elvégezni, majd ezután lehet elvégezni a *B* műveletet (bármikor, nem feltétlenül folyamatosan). Minden legyártandó alkatrészre ismert, hogy mennyi időt igényel az *A*,valamint a *B* művelet elvégzése.

Készíts programot *utemez* néven, amely kiszámítja, hogy legkevesebb mennyi idő alatt lehet legyártani az összes alkatrészt!

Az *utemez*.be szöveges állomány első sorában az alkatrészek *N* ($2 \le N \le 2000$) száma van. Az alkatrészeket az 1, …, *N* számokkal azonosítjuk. A második és a harmadik sor pontosan *N* egész számot tartalmaz egy-egy szóközzel elválasztva, a legyártandó alkatrészeken elvégzendő *A*, illetve *B* műveletek idejét. A második sorban az *i*-edik szám az *i*-edik alkatrészen végzendő *A* művelet ideje. A harmadik sorban az *i*-edik szám pedig az *i*-edik alkatrészen végzendő *B* művelet ideje. A második és harmadik sorban lévő számok mindegyike 1 és 50 közötti érték.

Az *utemez.ki* szöveges állomány első sora egy egész számot tartalmazzon, azt a legkisebb *T* időt, amely alatt a két gép le tudja gyártani az összes alkatrészt! A második sor az alkatrészek sorszámát tartalmazza abban a sorrendben, ahogy azokon az *A* műveletet kell elvégezni. A második sor az alkatrészek sorszámát tartalmazza abban a sorrendben, ahogy azokon az *B* műveletet kell elvégezni. Több megoldás esetén bármelyik megadható.

Példa:

```
utemez.be          utemez.ki
3                  16
8 1 6              2 3 1
1 6 3              2 3 1
```

Hasonlítsuk össze a megoldást az előző feladat (*Ütemezés*, 2010-2f2) megoldásával!

Gépek (2014)

Informatika OKTV 2014. 2. forduló 3. korcsoport 4. feladat

Egy vállalkozó a következő *N* napra megrendeléseket fogad. Minden munkát egy nap alatt tud elvégezni. *M* megrendelés érkezett, minden megrendelésben szerepel, hogy az igényelt munkát milyen határidőig kell elvégezni. A vállalkozónak ki kell számítani, hogy legkevesebb hány gépre van szükség, hogy minden igényelt munkát határidőre el tudjon végezni.

Készíts programot *gepek* néven, amely kiszámítja, hogy legkevesebb hány gép kell ahhoz, hogy minden megrendelt munkát határidőre el tudjon végezni a vállalkozó! A program adja is meg, hogy ez egyes megrendelést melyik napon, melyik gépen végezze el a vállalkozó!

A *gepek.be* szöveges állomány első sora két egész számot tartalmaz, a munkanapok *N* számát ($1 \le N \le 10000$), és a megrendelések *M* számát ($1 \le M \le 100000$). A második sor pontosan *M* egész számot tartalmaz egy-egy szóközzel elválasztva. Az *i*-edik szám az *i*-edik megrendelés határideje. Minden *h* határidőre teljesül, hogy $1 \le h \le N$.

A *gepek.ki* szöveges állományba *M*+1 sort kell írni! Az első sor a minimálisan szükséges gépek *G* számát tartalmazza! A további *M* sor tartalmazza az igényelt munkák beosztását az igények sorszáma szerinti sorrendben! Minden sor két számot tartalmazzon, egy szóközzel elválasztva, az első szám annak a napnak a sorszáma legyen, amelyik napon a munkát elvégzik, a második szám pedig annak a gépnek a sorszáma legyen, amelyiken a munkát elvégzik! Több megoldás esetén bármelyik megadható.

Példa:

```
gepek.be                gepek.ki
10 8                    2
3 2 3 2 4 5 6 2         2 2
                        1 1
                        3 1
                        2 1
                        3 2
                        4 1
                        4 2
                        1 2
```

Rekurzió, dinamikus programozás

Kemence (2001)

Informatika OKTV 2001. 3. forduló 3. korcsoport 2. feladat

A porcelángyár égetőkemencéjéhez futószalagon érkeznek az égetésre váró tárgyak. Minden tárgynak ismert a súlya és a kiégetéséhez minimálisan szükséges idő. Az égetésre váró tárgyakat az érkezésük sorrendjében kell kiégetni. Egyszerre több tárgyat is rakhatunk a kemencébe, az összsúlyuk azonban nem haladhatja meg a kemence kapacitását. Az égetési idő egy menetben mindig a kemencébe rakott tárgyak minimális égetési idejének a maximuma kell legyen.

Készíts olyan programot *kemence* néven, amely kiszámítja, hogy legkevesebb mennyi idő kell az összes tárgy kiégetéséhez, továbbá megadja azt is, hogy ezen idő eléréséhez mely tárgyakat kell egy-egy menetben a kemencében együtt égetni.

A *kemence.be* állomány első sora két egész számot tartalmaz; a tárgyak N ($1 \le N \le 10000$) számát és a kemence K ($1 \le K \le 1000$) kapacitását. A következő N sor mindegyike két egész számot tartalmaz; egy tárgy S ($1 \le S \le 1000$) súlyát és E ($1 \le E \le 100$) minimális égetési idejét.

A *kemence.ki* állomány első sorába az összes tárgy kiégetéséhez minimálisan szükséges időt kell írni. A következő sorok mindegyikébe két egész számot, I-t és J-t kell írni egy szóközzel elválasztva, I az első, J pedig az utolsó tárgy sorszáma, amelyek egyszerre kerülnek a kemencébe.

Példa:

```
kemence.be          kemence.ki
6 10                46
3 10                1 1
2 12                2 3
7 20                4 6
5 15
3 11
2 16
```

Hasonlítsuk össze a megoldást az előző feladat (*Fazekas*, 2001-3f2) megoldásával!

Kemence (2011)

Informatika OKTV 2011. 3. forduló 3. korcsoport 3. feladat

Cserépkorsók kiégetésére szakosodott vállalkozó egy kemencét üzemeltet. Az égetésre érkező korsókat az érkezés sorrendjében kell a kemencében kiégetni. Egy menetben legfeljebb K korsó rakható a kemencébe. Minden korsóra adott a minimális és maximális égetési idő percben kifejezve. Továbbá, minden korsóra adott egy H határidő, ami azt jelenti, hogy a munka megkezdésétől számítva, a H időpontig el kell készülnie a kiégetésének. Figyelembe kell venni, hogy egy menet előkészítése 1 percet vesz igénybe!

Készíts programot *kemence* néven, amely kiszámítja, hogy a követelmények betartásával legkevesebb mennyi idő alatt lehet kiégetni az összes korsót és meg is ad egy helyes beosztást!

A *kemence.be* szöveges állomány első sorában két egész szám van, a korsók N száma ($1 \le N \le 10000$), és a kemence K kapacitása ($1 \le K \le 100$). A következő N sor mindegyike három egész számot tartalmaz egy-egy szóközzel elválasztva. Az első szám *min*, a minimális, a második szám *max* a maximális égetési idő (percben megadva) $1 \le min \le max \le 1000$). A sorban a harmadik szám a határidő értéke percben megadva. A korsókat a sorszámukkal azonosítjuk 1-től N-ig az érkezésük sorrendjében.

A *kemence.ki* szöveges állomány első sorába két egész számot kell írni egy szóközzel elválasztva, az összes korsó kiégetéséhez szükséges legkisebb időt, és az égetési menetek M számát. A következő M sor mindegyike két egész számot tartalmazzon egy szóközzel elválasztva, *u v*, ami azt jelenti, hogy ebben a menetben az u, $u+1$, …, v sorszámú korsók kerülnek a kemencébe ($1 \le u \le v \le N$, $v \le u+K-1$)! Több megoldás esetén bármelyik megadható. A feladat minden tesztesetre megoldható.

Példa:

```
kemence.be          kemence.ki
4 3                 9 3
1 2 4               1 2
2 3 3               3 3
3 4 8               4 4
1 2 9
```

Gépek (2013)

Nemes Tihamér OITV 2013. 3. forduló 2. korcsoport 5. feladat

Egy vállalkozó alkatrészek gyártásával foglalkozik. K különböző fajta alkatrészt tud gyártani két gépén. Mindkét gépe képes legyártani mind a K fajtát, de az egyes fajták legyártása a két gépen különböző ideig tart. Egy időpontban csak az egyik gép dolgozhat a nyersanyag-ellátás miatt. A beérkezett igényeket az érkezés sorrendjében kell kielégítenie. Menet közben átválthat a másik gépre, de az időt igényel.

Készíts programot *gepek* néven, amely kiszámítja, hogy a legkevesebb mennyi idő alatt lehet legyártani az összes igényelt alkatrészt!

A *gepek.be* szöveges állomány első sorában négy egész szám van egy-egy szóközzel elválasztva. Az első szám az alkatrészfajták K száma ($1 \le K \le 100$), a második a legyártandó alkatrészek N száma ($1 \le N \le 1000$). A harmadik szám azt adja meg, hogy mennyi időt igényel az átállás az A gépről a B gépre, a negyedik szám pedig azt, hogy mennyi időt igényel az átállás a B gépről az A gépre. A második sor pontosan K pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, az i-edik szám értéke az i-fajta alkatrész legyártásának ideje az A gépen. A harmadik sor is pontosan K pozitív egész számot tartalmaz egy-egy szóközzel elválasztva, az i-edik szám értéke az i-fajta alkatrész legyártásának ideje a B gépen. A negyedik sor tartalmazza a legyártandó alkatrész-fajtákat, N számot egy-egy szóközzel elválasztva. A fajtákat az 1, …, K számokkal azonosítjuk.

A *gepek.ki* állomány egyetlen sorába azt a legkisebb időt kell írni, ami alatt a megadott sorrendben legyártható mind az N alkatrész!

Példa:

```
gepek.be              gepek.ki
3 7 4 3               17
1 3 9
7 1 1
1 2 3 2 1 2 3
```

Fazekas (2014)

Informatika OKTV 2014. 2. forduló 3. korcsoport 5. feladat

Korondi János fazekasmester két kemencében égeti ki a termékeit. A kiégetésre váró tárgyak az elkészülés sorrendjében várakoznak a kiégetésre. Minden tárgyra rá van írva, hogy legkevesebb hány percig kell égetni a kemencében. Mindkét kemencébe egyidejűleg legfeljebb *K* darab tárgy rakható kiégetésre. Minden tárgyat az elkészülés sorrendjében betesz valamelyik kemencébe, mindkettőbe legalább egy tárgyat kell rakni egy menetben és a következő menet csak akkor kezdődik, ha mindkét kemence befejezte az égetést. Ha valamelyik kemencében több tárgyat éget egy menetben, akkor az égetési idő az abba a kemencébe tett tárgyak egyedi égetési idejének a maximuma. Mivel a kemencék energiafogyasztása az égetési idő függvénye, ezért a fazekasnak arra kell törekednie, hogy az összesített égetési idő a lehető legkevesebb legyen.

Írj programot *fazekas* néven, amely kiszámítja, hogy legkevesebb mennyi összesített égetési idő alatt lehet kiégetni az összes tárgyat!

A *fazekas.be* szöveges állomány első sora két egész számot tartalmaz, a kiégetésre váró tárgyak *N* számát ($2 \le N \le 1000$) és a két kemence egyedi *K* kapacitását ($2 \le K \le 20$), azaz egyidejűleg legfeljebb *K* tárgy rakható mindkét kemencébe. A második sor pontosan *N* egész számot tartalmaz (egy-egy szóközzel elválasztva), az *i*-edik szám az *i*-edik tárgy minimális égetési ideje. Minden égetési idő legfeljebb 20000.

A *fazekas.ki* szöveges állomány első sora egy egész számot tartalmazzon, a legkevesebb összesített égetési időt! A következő *N* sor mindegyike két egész számot tartalmazzon egy szóközzel elválasztva! Az *i*-edik sorban az első szám annak a menetnek a sorszáma legyen, amelyikben az *i*-edik tárgyat kemencébe rakjuk, a második szám pedig 1, ha az első, 2, ha a második kemencébe rakjuk égetni! A kiírást a menetek sorrendjében kell megadni! Több megoldás esetén bármelyik megadható.

Példa:

```
fazekas.be            fazekas.ki
8 2                   22
1 7 4 9 2 9 1 2       1 1
                      1 2
                      1 2
                      2 1
                      2 2
                      2 1
                      3 1
                      3 2
```

Gráfok

Terv

Informatika OKTV 1998. 3. forduló 3. korcsoport 1. feladat

Egy nagyszabású építkezéshez részletes tervet készítettek, amelyben többek között azt is megadják, hogy egy-egy munka hány napig tart. A munkák sorrendje nem tetszőleges. A tervben a sorrendet *a b* feltételpárok írják elő. Az *a b* feltételpár azt jelenti, hogy a *b* munkát csak az *a* munka befejezése után lehet elkezdeni. Az építkezés az 1. napon kezdődik. *N* munka esetén a munkákat az 1, ..., *N* számokkal azonosítjuk.

Írj programot *terv* néven az alábbi részfeladatok megoldására!

a) Számítsd ki, hogy a terv végrehajtásához minimálisan hány nap szükséges!
b) Add meg az egyes munkák legkorábbi kezdési idejét a számuk sorrendjében! (Ez az a legkorábbi időpont, amikor az adott munkát el lehet kezdeni, mert az összes korábbi, ugyancsak a lehető legkorábban elkezdett munka már befejeződött.)
c) Add meg a munkák legkésőbbi kezdési idejét a számuk sorrendjében! (Ez az a legkésőbbi időpont, amikor az adott munkát el kell kezdeni ahhoz, hogy ne késleltesse az építkezés befejezését.)
d) Adj meg egy ún. kritikus utat! Kritikus útnak nevezzük olyan munkák egy sorozatát, amelyeket az adott sorrendben lehet elvégezni, és
   - a munkák összideje megegyezik a terv végrehajtásához minimálisan szükséges idővel, továbbá
   - minden egyes munka legkorábbi és legkésőbbi kezdési ideje ugyanaz.
e) Add meg, hogy minimum hány vállalkozóval kell szerződést kötni a munkák elvégzésére, ha egy vállalkozó egy időben csak egy munkán tud dolgozni, és ha minden munka a lehető legkorábban kezdődik!
f) Add meg, hogy minimálisan mennyi az összes állási idő! (Az állási idő abból adódik, hogy egy vállalkozó két, időben egymást követő munka elvégzése között várakozásra kényszerülhet, ha a soron következő munkát még nem kezdheti el.) Feltesszük, hogy minden munka a lehető legkorábban kezdődik, és a kivitelezők a lehető legkevesebb vállalkozót alkalmazzák.

A *terv.be* állomány első sora a munkák *N* ($1 \le N \le 100$) számát tartalmazza. A második sorban *N* db pozitív egész szám van, ahol az *i*-edik szám a sorban az *i*-edik munka végrehajtásához szükséges napok száma (≤ 100). A harmadik sorban a feltételpárok *M* ($0 \le M \le 1000$) száma van. A következő *M* sor mindegyike egy-egy *a b* számpárt tartalmaz ($1 \le a, b \le N$); ami azt jelenti, hogy a *b* munkát csak az *a* munka befejezése után lehet elkezdeni.

A *terv.ki* szöveges állományba és a képernyőre hat sorba kell írni az a)–f) részfeladatok megoldását. Ha valamelyik részfeladat megoldása hiányzik, akkor a megfelelő sor legyen üres! A b), c) és d) részfeladatok esetén (2., 3. és 4. sor) a számsorozatok elemeit egy-egy szóköz válassza el.

Példa:

```
terv.be                terv.ki
5                      6
1 2 1 3 2              1 2 2 4 4
5                      1 2 3 4 5
1 2                    1 2 4
2 5                    2
1 3                    1
3 4
2 4
```

INTERVALLUM FELADATOK

A *Programozási ismeretek versenyzőknek* (Műszaki Kiadó, 2015) könyvben szereplő feladatok:

OITV 2001-2f2	Licit
OITV 2014-2f2	Mozi
OKTV 2005-3f3	Bérlet
OKTV 2009-3f3	Koncert

Néhány további feladat az ütemezési feladatokhoz is besorolható.

Első fordulós feladatok

Újság

Nemes Tihamér OITV 2005. 1. forduló 2. korcsoport 4. feladat
Informatika OKTV 2005. 1. forduló 3. korcsoport 4. feladat

A két feladat megegyezik egymással.

Egy újság minden számában egy 1 oldalas hirdetést jelentet meg. Hetenként vesznek fel hirdetési igényeket. Minden igénylő megadja, hogy mennyit fizet a hirdetéséért, ha adott sorszámú napig megjelenik az újságban. Úgy kell kiválasztani az egyes napokra a hirdetéseket, hogy az újságnak a lehető legnagyobb bevétele legyen.

A következő számpár sorozatokban a sorszámozott számpárok első tagja mindig a hirdetés legutolsó lehetséges megjelenési napja, a második pedig az érte fizetett összeg. Add meg mindegyikre, hogy mennyi belőlük az újság lehető legnagyobb bevétele, s az egyes napokon a felsorolás sorrendjében hányadik hirdetés jelenjen meg!

a) 1:(6,1000), 2:(3,200), 3:(6,1200), 4:(6,800), 5:(5,500), 6:(2,600), 7:(2,300), 8:(1,400), 9:(5,700), 10: (1,400)
b) 1:(2,1500), 2:(2,1200), 3:(2,1000), 4:(4,500), 5:(4,600), 6:(4,700), 7:(7,1000), 8:(7,800), 9:(7,200), 10:(7,100)
c) 1:(5,200), 2:(3,300), 3:(7,300), 4:(1,400), 5:(3,400), 6:(7,500), 7:(1,600), 8:(5,800), 9:(3,800), 10:(5,800)

Példa:

```
1:(7,1000), 2:(2,500), 3:(2,400), 4:(1,300), 5:(4,100)
```

esetén az 1,2,3,5 sorszámú hirdetések adják a legnagyobb bevételt, 2000 forintot, s egy lehetséges hirdetés elosztás a 7 napra: (3,2,-,5,-,-,1), azaz a hét első napján a 3. igénylő hirdetése jelenik meg, a hét utolsó napján pedig az 1. igénylőé. De sok más jó elosztás is van, pl. a 3 és a 2 felcserélhető, az 5. eggyel előbbre hozható, a 7-est is előre lehet hozni valamelyik üres helyre, …

Mohó algoritmus

Megrendelés

Informatika OKTV 2003. 2. forduló 3. korcsoport 2. feladat

Egy rendezvényt olyan teremben tartanak, ahol *M* db ülőhely van. Az ülőhelyek 1-től *M*-ig sorszámozottak. A rendezvény szervezője megrendeléseket fogad. Minden megrendelés egy *A B* számpárt tartalmaz, ami azt jelenti, hogy a megrendelő olyan ülőhelyet szeretne kapni, amelynek *S* sorszáma *A* és *B* közé esik ($A \le S \le B$).

Írj programot *kioszt* néven, amely kiszámítja, hogy a szervező a megrendelések alapján a legjobb esetben hány megrendelést tud kielégíteni és meg is ad egy olyan jegykiosztást, amely kielégíti a megrendeléseket!

A *kioszt.be* szöveges állomány első sorában két egész szám van, *M* és *N*. *M* az ülőhelyek száma ($1 \le M \le 1000$), *N* ($1 \le N \le 1000$) pedig a megrendelések száma. A következő *N* sor mindegyike két egész számot *A*, *B* ($1 \le A \le B \le M$) tartalmaz egy szóközzel elválasztva. Az állomány *i*+1-edik sorában lévő megrendelés sorszáma *i*.

A *kioszt.ki* szöveges állomány első sorába a legtöbb kielégíthető megrendelés *K* számát kell írni! A további *K* sor tartalmazza a jegykiosztást, minden sor két egész számot tartalmazzon egy szóközzel elválasztva. Az első szám egy megrendelés sorszáma, a második pedig azon ülőhely sorszáma, amelyet a megrendelő kap. A kiosztás kiírása tetszőleges sorrendben lehet. Ha több megoldás van, akkor egy tetszőlegeset ki lehet írni.

Példa:

```
kioszt.be          kioszt.ki
10 6               4
3 3                5 1
2 2                2 2
2 3                1 3
1 3                6 4
1 2
2 4
```

Zenekar

Nemes Tihamér OITV 2005. 3. forduló 2. korcsoport 4. feladat

Egy népszerű zenekar a következő évre vonatkozó fellépéseit tervezi. Sok meghívása van fellépésre, ezek közül kell a zenekarnak választani, hogy melyeket fogadja el. Minden fellépés pontosan egy napot foglal el. Minden beérkezett meghívási igény egy (*e*, *u*) szám-párral adott, ami azt jelenti, hogy az igénylő azt szeretné, hogy a zenekar olyan *k* sorszámú napon tartson nála koncertet, hogy $e \le k \le u$. A zenekarnak az a célja, hogy a lehető legtöbb fellépése legyen.

Készíts programot *zenekar* néven, amely kiszámítja, hogy mely meghívásokat fogadja el, hogy a következő évben a lehető legtöbb fellépése legyen, és a programod adjon is meg egy beosztást!

A *zenekar.be* állomány első sorában a meghívások *N* száma ($1 \le N \le 1000$) van. A meghívásokat az 1, …, *N* számokkal azonosítjuk. A következő *N* sor mindegyike két egész számot tartalmaz egy szóközzel elválasztva; *e u* ($1 \le e \le u \le 365$). Az *i*-edik meghívás az állomány *i*+1-edik sorában van.

A *zenekar.ki* szöveges állomány első sora egy egész számot tartalmazzon, a vállalható legtöbb fellépések *M* számát! A következő *M* sor mindegyike két egész számot tartalmazzon, egy szóközzel elválasztva! Az első szám egy elfogadott meghívás sorszáma legyen, a második pedig annak a napnak a sorszáma, amelyik napon teljesíti a zenekar a fellépést! (Több megoldás esetén bármelyik megadható.)

Példa:

```
zenekar.be          zenekar.ki
6                   5
2 4                 4 1
1 4                 2 2
3 5                 3 3
1 3                 5 4
3 5                 6 5
2 5
```

Hasonlítsuk össze a megoldást a következő feladat (*Mozi*, 2014-2f2) megoldásával!

Dinamikus programozás

Járműkölcsönzés

Informatika OKTV 1998. 2. forduló 3. korcsoport 3. feladat

Egy közlekedési vállalat járműveket ad bérbe. Egyik különleges szállítójárművére nagy az igény, ezért a megrendeléseket a következő évre előre összegyűjtik. A megrendelést az *A B* számpárral adják meg, ami azt jelenti, hogy a megrendelő a járművet az év *A*-adik napjától a *B*-edik napjáig akarja bérbe venni. A vállalat a járművet a lehető legjobban akarja hasznosítani, ezért olyan megrendeléseket fogad el, amelyeket teljes egészükben úgy tud teljesíteni, hogy a jármű a lehető legtöbb napra ki legyen adva.

Írj programot *jarmu* néven, amely kiszámítja, hogy a megrendelések alapján a jármű a legjobb esetben hány napra lesz bérbe adva.

A *jarmu.be* állomány tartalmazza a megrendeléseket. Az első sorban a megrendelések *N* száma van ($1 \le N \le 1000$). A következő *N* sor mindegyikében egy-egy megrendelés, .azaz egy-egy *A B* számpár van, egyetlen szóközzel elválasztva ($1 \le A \le B \le 365$).

A megoldást a *jarmu.ki* állományba kell kiírni.

Példa:

```
jarmu.be            jarmu.ki
9                   69
1 23
6 37
24 43
46 71
12 54
44 66
13 25
22 33
5 10
```

Hasonlítsuk össze a megoldást a *Bérlet* (2005-3f3) feladat megoldásával!

Licit (2001-2f3)

Informatika OKTV 2001. 2. forduló 3. korcsoport 2. feladat

Nevesincs sziget polgármestere elhatározta, hogy értékesíti a sziget tengerpartját. A partot egyforma méretű parcellákra osztotta.

A polgármester úgy döntött, hogy a parcellákat nyilvános pályázat keretében adja el, azaz egy adott határidőig minden érdeklődő lezárt borítékban leadhatja ajánlatát. Egy pályázó csak egy ajánlatot nyújthat be, amelyben meg kell adnia, hogy melyik parcellától melyik parcelláig terjedő részt kívánja megvenni, és mennyiért.

A pályázat sikeres volt, a határidő lejártáig *N* pályázat érkezett. Ezek közül ki kell választani azokat az ajánlatokat, amelyek a legtöbb bevételt eredményezik, s persze úgy, hogy egyetlen parcellát sem ítélnek oda egynél több pályázónak. Egy-egy pályázó vagy az összes kért parcellát megkapja, vagy egyet sem kap meg. Előfordulhat, hogy a maximális bevétel eléréséhez nem kell eladni az összes parcellát.

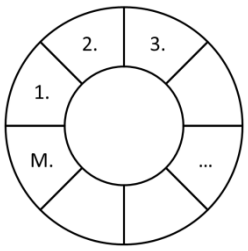
Írj programot *licit* néven, amely megadja a választ a polgármester problémájára!

A *licit.be* állomány első sorában a pályázatok *N* száma ($1 \le N \le 100$) és a parcellák *M* száma ($1 \le M \le 100$) található, egy szóközzel elválasztva. A következő *N* sor az egyes pályázók adatait tartalmazza, a pályázókat ez a sorrend azonosítja. Mindegyik sorban 3 szám van egy-egy szóközzel elválasztva: *A B FT*, ami azt jelenti, hogy a pályázó az *A* sorszámú parcellától ($1 \le A \le M$) a *B* sorszámú parcelláig ($A \le B \le M$) terjedő részért *FT* forintot fizetne ($1000 \le FT \le 1000000$). Ha $B < A$, akkor a pályázó *B*-től *M*-ig és 1-től *A*-ig szeretné megvásárolni a parcellákat.

A *licit.ki* állomány első sorába az elérhető legnagyobb bevételt kell írni. A második sorba a nyertes pályázók sorszámai kerüljenek egy-egy szóközzel elválasztva. Ha több megoldás is van, csak egyet kell kiírni (bármelyiket).

Példa:

```
licit.be            licit.ki
5 6                 14000
1 5 10000           2 4 5
2 3 5000
4 5 5000
4 4 6000
6 1 3000
```



Hasonlítsuk össze a megoldást az előző feladat (*Licit*, 2001-2f2) megoldásával!

Jegy

Nemes Tihamér OITV 2007. 3. forduló 2. korcsoport 3. feladat

Egy jegyiroda nagyszabású koncertre árul jegyeket. Összesen *M* ülőhelyre lehet jegyeket igényelni, pontosabban minden igénylő két egymás melletti jegyet igényelhet, és meg kell adnia, hogy ezért mennyit fizetne. A jegyiroda a beérkezett igények közül ki akarja választani a legtöbb bevételt eredményező igényeket, amelyeket természetesen ki is tud elégíteni.

Készíts programot *jegy* néven, amely kiszámítja az elérhető legnagyobb jegybevételt és meg is adja, hogy melyik igények kielégítése esetén érhető el a legnagyobb bevétel!

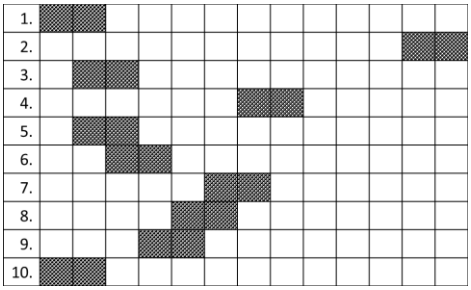
A *jegy.be* szöveges állomány első sora két egész számot tartalmaz, az ülőhelyek számát ($1 \le M \le 6000$), és az igények számát ($1 \le N \le 30000$). A további N sor mindegyike egy igényt leíró két egész számot tartalmaz. Az első szám az igényelt két ülőhely első székének s sorszáma ($1 \le s < M$), a második szám az az f pénzösszeg ($1 \le f < 500$), amit az igénylő a két jegyért fizetne. Az igényeket a sorszámukkal azonosítjuk, az állomány i+1-edik sorában van az i-edik igény.

A *jegy.ki* szöveges állomány első sorába az elérhető legnagyobb bevételt kell írni! A második sorba azoknak az igényeknek a sorszáma kerüljön (tetszőleges sorrendben), amelyek esetén a bevétel a legnagyobb lesz! Több megoldás esetén bármelyik megadható.

Példa:

```
jegy.be
20 10
1 21
12 23
2 12
7 43
2 32
3 24
6 12
5 33
4 32
1 12
```

```
jegy.ki
144
2 4 8 6 1
```



Munka

Nemes Tihamér OITV 2012. 3. forduló 2. korcsoport 5. feladat

Egy vállalkozó N munkaajánlatot kapott, de egyszerre csak egy munkát tud végezni. Minden ajánlatban szerepel, hogy a munkát mikor kellene elkezdeni, meddig tartana és a munka elvégzéséért mennyi fizetést kapna.

Készíts programot *munka* néven, amely kiszámítja, hogy a vállalkozó maximum mennyit kereshet!

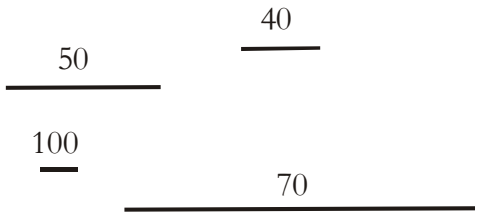
A *munka.be* szöveges állomány első sorában egy egész szám van, a munkák száma ($1 \le N \le 1000$). A következő N sor mindegyike három egész számot tartalmaz (egy-egy szóközzel elválasztva): az igényelt munka kezdő A és befejező B napjának sorszáma ($1 \le A \le B \le 100000$), valamint pénzértéke ($1 \le$ *érték* ≤ 10000).

A *munka.ki* állomány egyetlen sorába az elérhető legnagyobb jövedelem összegét kell írni!

Példa:

```
munka.be
4
30 40 40
0 20 50
5 10 100
15 60 70
```

```
munka.ki
170
```



Koncert (2014)

Informatika OKTV 2014. 3. forduló 3. korcsoport 3. feladat

A nagy érdeklődéssel várt koncertre jegyet lehet igényelni, egy igény *H*, *K* számpár, ami azt jelenti, hogy az igénylő az első *H* ülőhelyből kér *K* egymás melletti ülőhelyet. Mivel minden hely ára azonos, ezért a szervező célja, hogy a lehető legtöbb ülőhelyet adja el úgy, hogy az eladott jegy az igénylőnek megfelelő legyen.

Készíts programot *koncert* néven, amely megadja, hogy mekkora az elérhető legnagyobb bevétel, és mely igények teljesítése adja az elérhető legnagyobb bevételt!

A *koncert.be* szöveges állomány első sorában két egész szám van, az ülőhelyek *M* száma ($1 \le M \le 1000$), és az igények *N* száma ($1 \le N \le 2000$). A következő *N* sor mindegyike két *H K* ($K \le H \le M$) egész számot tartalmaz, egy igényt. Az ülőhelyeket az 1, …, *M*, az igénylőket az 1, …, *N* számokkal azonosítjuk.

A *koncert.ki* szöveges állomány első sorába két egész számot kell írni, az első szám a legtöbb eladható ülőhely száma, a második szám pedig a kielégített igények *S* száma legyen. A következő *S* sor mindegyike két egész számot tartalmazzon, az első szám egy kielégített igénylő sorszáma, a második pedig az első ülőhely sorszáma, amit az igénylő kap! Több megoldás esetén bármelyik megadható.

Példa:

```
koncert.be          koncert.ki
10 7                9 4
2 1                 7 6
3 2                 6 4
8 2                 2 2
8 5                 1 1
7 3
6 2
9 4
```

Gráfok

Ültetés

Informatika OKTV 1999. 3. forduló 3. korcsoport 2. feladat

Az iskola színháztermében *N* számú ülőhely van. A következő előadásra *M* tanuló kérhet jegyet, és mindegyik meghívott tanuló két hely sorszámát megadhatja, mint az általa előnyben részesítettet.

Írj programot *ultet* néven, amely kiszámítja, hogy legjobb esetben hány tanuló kaphat olyan jegyet, amely az igénylésének megfelel. A program azt is megadja, hogy mely tanulók kapják az igénylésüknek megfelelő helyeket. Kiszámítandó továbbá, hogy lehet-e az igényeket úgy kielégíteni, hogy a kiosztott helyek összefüggő tartományt alkossanak.

Az *ultet.be* állomány első sorában az ülőhelyek száma ($1 \le N \le 200$), második sorában a tanulók száma ($1 \le M \le 250$) van. A következő *M* sor mindegyike két különböző ülőhely sorszámot (*A B*) tartalmaz egy szóközzel elválasztva ($1 \le A, B \le N$). Az állomány *i*+2-edik sora az *i*-edik tanuló kívánságát tartalmazza.

Az *ultet.ki* állomány első sorába azon tanulók *T* számát kell írni, ahányan a legjobb esetben megkaphatják a két kívánságuknak megfelelő ülőhely valamelyikét. A második sorba egy legjobb kiosztás szerinti ültetést kell írni, tehát *T* számpárt, amelynek első tagja egy tanuló sorszáma, második tagja pedig azt az általa kívánt ülőhelysorszámot tartalmazza, amit a tanuló kap a legjobb kiosztás szerint. A második sorban a számokat egy-egy szóköz válassza el. A harmadik sorba az *IGEN* szót kell írni, ha van olyan legjobb kiosztás, amely szerint nincs üresen maradt szék a foglalt helyek között, egyébként pedig a *NEM* szót.

Példa:

ultet.be	ultet.ki
10	8
10	1 1 2 2 7 3 3 4 4 5 5 6 6 7 8 10
1 4	IGEN (lásd a megjegyzést a feladat szövege után)
2 4	
4 6	
6 5	
6 7	
7 5	
3 5	
8 10	
1 4	
4 7	

Megjegyzés: a feladat eredeti szövegében hibás az összefüggő tartományra vonatkozó *NEM* válasz. Egy lehetséges jegykiosztás például:

9 1 2 2 7 3 1 4 4 5 5 6 6 7 8 8

Ebben az esetben a kiosztott helyek összefüggő tartományt alkotnak 1-től 8-ig.

Ha kioszthatók a helyek úgy, hogy összefüggő tartományt alkossanak, akkor a program ezt a megoldást határozza meg!

TEREMBEOSZTÁS

A *Programozási ismeretek versenyzőknek* (Műszaki Kiadó, 2015) könyvben szereplő feladatok:

OITV 2001-3f2	Terem
OKTV 2007-2f3	Rendezvény

Mohó algoritmus

Kemping

Nemes Tihamér OITV 1998. 2. forduló 2. korcsoport 3. feladat

A Napsugár Kemping *K* faházat üzemeltet az év 365 napján. Minden vendég ugyanannyi időre, pontosan *M* napra foglalhat le egy-egy faházat. A kemping a teljes évre előre összegyűjtötte az igényeket. Minden vendég igényként annak a napnak a sorszámát adta meg, amelytől kezdve (*M* napra) le akar foglalni egy faházat.

Írj programot *kemping* néven, amely kiszámítja, hogy a kemping maximálisan hány vendéget fogadhat.

A kemping.be állomány első sora a faházak *K* ($1 \le K \le 100$) számát tartalmazza, második sora pedig az *M* ($1 \le M \le 14$) számot. A harmadik sorban az igények *N* ($1 \le N \le 1000$) száma, a további *N* sor mindegyikében pedig egy-egy igény van. Az eredményt a *kemping.ki* állományba kell írni.

Példa:

kemping.be	kemping.ki
2	5
7	
8	
1	
10	
2	
11	
1	
3	
4	
18	

Rendezvény (2009)

Nemes Tihamér OITV 2009. 3. forduló 2. korcsoport 3. feladat

Egy rendezvény szervezőinek *N* előadást kell elhelyezni termekben. Minden előadáshoz adott annak *A* kezdési és *B* befejezési időpontja.

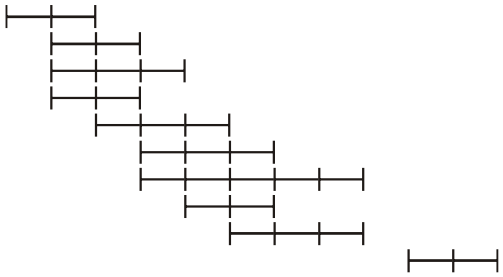
Készíts programot *termek* néven, amely kiszámítja, hogy legkevesebb hány terem szükséges ahhoz, hogy minden előadást meg lehessen tartani!

A *termek.be* szöveges állomány első sorában az előadások *N* száma van ($1 \le N \le 200$). Az előadásokat az 1, …, *N* számokkal azonosítjuk. A további *N* sor mindegyike két egész számot tartalmaz egy szóközzel elválasztva, az első szám az előadás kezdő ideje, a második a befejezési ideje. Az időpontok értékei nem nagyobbak, mint 500. Az előadások a kezdő időpontjuk szerint nem csökkenő sorrendben vannak.

A *termek.ki* szöveges állomány első sorába az előadások beosztásához minimálisan szükséges termek *T* számát kell írni. A további *T* sor azt adja meg, hogy az előadásokat mely termekbe osztottuk be. Az állomány *i*+1-edik sorában azoknak az előadásoknak a sorszámai vannak felsorolva időrendi sorrendben, amelyeket az *i*-edik terembe osztottunk be. Több megoldás esetén bármelyik megadható.

Példa:

```
termek.be          termek.ki
10                 4
1 3                1 5 9
2 4                2 6
2 5                3 8
2 4                4 7 10
3 6
4 7
4 9
5 7
6 9
10 12
```



Hasonlítsuk össze a megoldást a *Terem* (2001-3f2) feladat megoldásával! Miben térnek el egymástól (!)?

FÜGGELÉK

AZ 1. FORDULÓS FELADATOK MEGOLDÁSA

A megoldásokat a témakörök, azokon belül pedi az évek szerinti sorrendben közöljük. A megoldások forrása:

http://nemes.inf.elte.hu/nemes_archivum.html

Fotózás

Osztálybuli (2001-1f1)

a) 1. 4
   2. 6
b) 1. 500 vagy 600 vagy [650, 700]
   2. 5000
c) 1. 3
   2. 7
d) 1. A három időpontnak rendre az alábbi zárt intervallumokban kell lenni:
   [66, 99] [300, 499] [650, 700] vagy
   [66, 99] [500, 500] [650, 900] vagy
   [100, 100] [300, 499] [650, 700] vagy
   [100, 100] [500, 600] [650, 900]
   2. A hét időpontnak (növekvő sorrendben) rendre az alábbi zárt intervallumokban kell lenni:
   1. szám: [601, 1000]
   2. és 3. szám: [2000, 3000] és [3001, 3333] vagy [2345,3000] és [3001,4000]
   4. szám: [4001, 5000]
   5. szám: [5056, 5156]
   6. és 7. szám: [5621, 5999] és [6300, 7000] vagy [6000, 6000] és [6300, 9000]

Pakolás

Ládapakoló robot (2000-1f1, 4. feladat)

a) Első: 1. polc: 5, 4, 1
   2. polc: 8, 3
   3. polc: 9, 6
   Második: 1. polc: 9, 6
   2. polc: 8, 3
   3. polc: 5, 4, 1
b) Ha az összes polcon nála kisebb van legfelül.
c) 6-nál nagyobb legyen.

Kockarakás (2001-1f2, 5. feladat)

a) Hapci: Az új kockát arra a toronyra teszi, amire rátehető és amelynek a tetején lévő kocka a legkisebb, ha nincs ilyen, akkor új toronyba.
Tudor: Az új kockát a legnagyobb felső kockájú toronyra teszi, ha lehet, ha nem lehet, akkor pedig új toronyba.
Kuka: Az új kockát vagy az első toronyra teszi, vagy új tornyot kezd vele.
b) Hapci < Tudor
Tudor < Kuka
c) Ha a kockák mérete monoton csökkenő.
d) Például 1 3 5 2 4 esetén Kuka 5 tornyot épít, Tudor négyet, Hapci pedig hármat.
Hapci < Tudor, ha a legnagyobb nem az első tornyon van, amire egy kocka tehető.
Tudor < Kuka, ha Tudor tud az elsőtől különböző oszlopra is tenni kockát.

Kocka (2005-1f2, 5. feladat)

M(1)	M(2)	M(3)	M(4)	M(5)	M(6)	M(7)	M(8)	M(9)	M(10)
10	17	23	27	11	30	19	14	28	23

A legmagasabb építhető torony magassága: 30.

Kocka (2005-1f3, 5. feladat)

M(1)	M(2)	M(3)	M(4)	M(5)	M(6)	M(7)	M(8)	M(9)	M(10)
1	2	3	4	1	5	2	1	4	2

A legtöbb kockából álló torony kockaszáma: 5.

Pakolás (2008-1f2, 4. feladat)

a) 2 b) 4 c) 5 d) 5

Pakolás (2008-1f3, 4. feladat)

a) 6 b) 4 c) 4

Ütemezés

Fazekas (2006-1f2, 4. feladat)

a) *Opt* = (10, 10, 20, 35, 40, 50, 75)

b) Opt(1)=Idő(1)

$$Opt(i) = \text{MIN}\left\{\begin{matrix} Opt(i-1) + Idő(i), \\ \text{MIN}_{j \geq i-K+1}\{Opt(j-1) + \text{MAX}_{k=j-\text{től } i-\text{ig}}\{Idő(k)\}\} \end{matrix}\right\}$$

Ütemezés (2008-1f2, 1. feladat)

a) A=2, B=3, C=3, D=5, V=9
b) B, 1 időegységgel, C, 4 időegységgel

Ütemezés (2008-1f3, 1. feladat)

a) A=20, B=70, C=47, D=36, E=122, F=22, V=130
b) C, 25 időegységgel, D, 6 időegységgel, F, 18 időegységgel

Pakolás (2009-1f3, 4. feladat)

A1: égetési idő: 65, egy lehetséges sorrend: 1-2, 3-4, 5
(jó még: 1-2, 3, 4-5; 1, 2-3, 4-5)
A2: égetési idő: 65, a lehetséges sorrend: 1-2, 3, 4-5
A3: égetési idő: 65, egy lehetséges sorrend: 1-2, 3-4, 5-6-7
(jó még: 1, 2-3-4, 5-6-7; 1-2-3, 4, 5-6-7)

Opt(0) = 0 vagy *Opt*(1) = *Idő*(1)

$$Opt(i) = \text{MIN}_{j \geq i-K+1}\{Opt(j-1) + \text{MAX}_{k=j\text{-től } i\text{-ig}}\{Idő(k)\}\}$$

Munka (2012-1f2, 3. feladat)

a) haszon=40, munka sorszámok: 3, 4, 5, 7, 8
b) haszon=25, munka sorszámok: 3, 4, 5, 6, 7
c) haszon=25, munka sorszámok: 3, 4, 5, 6, 7
d) haszon=31, munka sorszámok: 1, 3, 5, 6, 7
e) haszon=45, munka sorszámok: 1, 2, 3, 6, 9

Munkavállalás (2013-1f3, 5. feladat)

a) 1. megoldás: *Nap*(*j*)=*i*, ha a *j*. napon az *i*. munkát kell elvégezni, vagy
2. megoldás: *Nap*(*j*)=*i*, ha a *j*. napon az *i*. munkát kell elvégezni, *DB*+1-től 0
b) 1. megoldás: a legutolsó napra, amikor még elvégezhető
2. megoldás: a legelső napra, amikor elvégezhető
(az első *DB* napra teszi a munkákat)
c) {*}: $M \cdot N$ {**}: M^2 {***}: N
d) $N \leq M/2$ esetén kisebb az első algoritmus lépésszáma.

Szállítás

Fűrészmalom (2011-1f3, 5. feladat)

a) 13
b) 7, 13
c) 6, 8, 13
d) 6, 8, 13, 14

Taxi (2012-1f3, 3. feladat)

a) Az utasok száma: 7, utasok: 2, 3, 4, 5, 6, 7, 8
b) Az utasok száma: 5, utasok: 2, 3, 4, 5, 6
c) Az utasok száma: 8, utasok: 1, 4, 5, 7, 9, 10, 11, 12
d) Az utasok száma: 10, utasok: 1, 3, 4, 5, 7, 8, 9, 10, 11, 12

Raktár (2013-1f2, 4. feladat)

a) AABABB vagy BABAAB
minimális költség: 13

b) *Költség*(0, 0) = 0

Költség(*i*, 0) = *Költség*(*i*–1, 0) + *A*(*i*)

Költség(0, *j*) = *Költség*(0, *j*–1) + *B*(*j*)

Költség(*i*, *j*) = MIN{*Költség*(*i*–1, *j*) + *A*(*i*+*j*), *Költség*(*i*, *j*–1) + *B*(*i*+*j*)}

Intervallum-feladatok

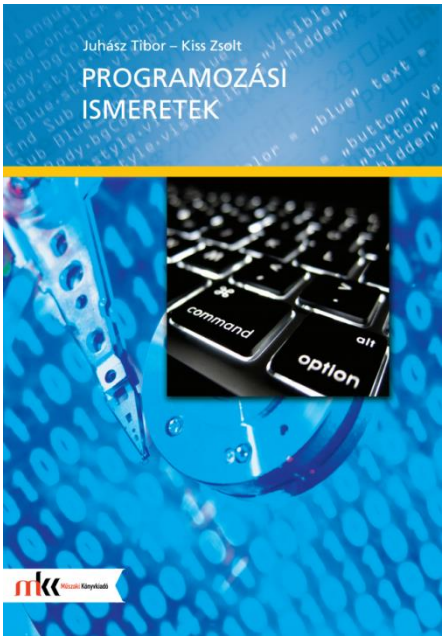
Újság (2005-1f2, 4. feladat; 1f3, 4. feladat)

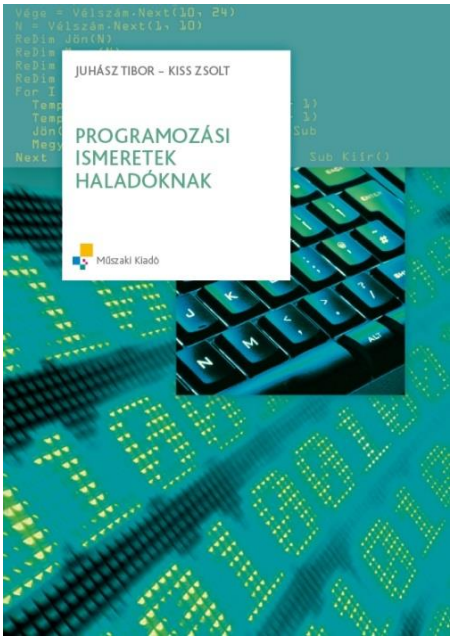
a) 1: (6, 1000), 2: (3, 200), 3: (6, 1200), 4: (6, 800), 5: (5, 500), 6: (2, 600), 7: (2, 300), 8: (1, 400), 9: (5, 700), 10: (1, 400)

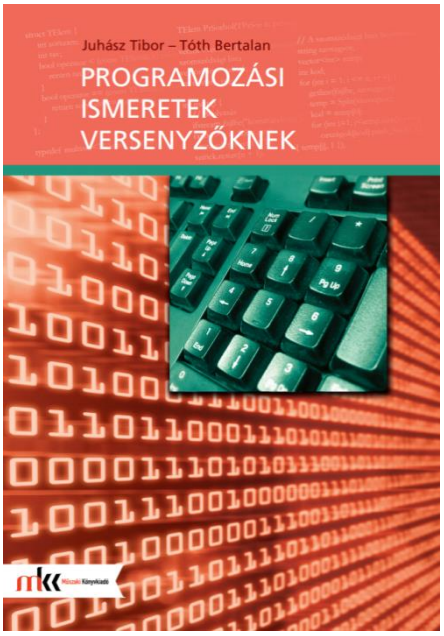
b) 1: (2, 1500), 2: (2, 1200), 3: (2, 1000), 4: (4, 500), 5: (4, 600), 6: (4, 700), 7: (7, 1000), 8: (7, 800), 9: (7, 200), 10: (7, 100)

c) 1: (5, 200), 2: (3, 300), 3: (7, 300), 4: (1, 400), 5: (3, 400), 6: (7, 500), 7: (1, 600), 8: (5, 800), 9: (3, 800), 10: (5, 800)

A FELADATOK JEGYZÉKE







Juhász Tibor – Kiss Zsolt:
Programozási ismeretek
(Műszaki Kiadó, 2011, 2015)

Juhász Tibor – Kiss Zsolt:
Programozási ismeretek haladóknak
(Műszaki Kiadó, 2012)

Juhász Tibor – Tóth Bertalan:
Programozási ismeretek versenyzőknek
(Műszaki Kiadó, 2015)

www.zmgzeg.sulinet.hu/programozas