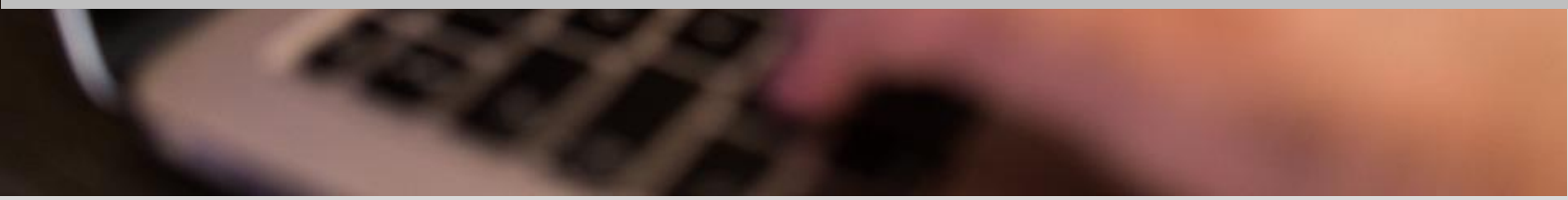
Belépő a tudás közösségébe

Informatika szakköri segédanyag

Mohó algoritmusok 2.

Bende Imre, Heizlerné Bakonyi Viktória, Menyhárt László, Szlávi Péter, Törley Gábor, Zsakó László

Szerkesztő: Abonyi-Tóth Andor, Zsakó László



A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-3.4.4-16-2017-006) című pályázat keretében készült 2018-ban.

Eötvös Loránd Tudományegyetem
Informatikai Kar

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

Többféle feladat megoldási stratégia létezik. Közülük az egyik legegyszerűbb a mohó stratégia, melynek lényege röviden megfogalmazva: minden döntési helyzetben válasszuk azt a döntést, ami pillanatnyilag a legkedvezőbbnek tűnik.

Ez a stratégia persze nem mindig szerencsés, de a feladatok egy viszonylag széles körére alkalmazható.

A stratégia alapján egyértelmű, hogy olyan feladatok esetén merülhet fel egyáltalán az alkalmazása, amikor több döntést kell hozni egymás után (lépésenként), és a döntés jóságát is meg kell fogalmazni valahogy (azaz minimum vagy maximum feltételt fogalmazunk meg).

A mohó stratégia elemei

1. Fogalmazzuk meg az optimalizációs feladatot úgy, hogy választások sorozatával építjük fel a megoldást!
2. Mohó választási tulajdonság: Mutassuk meg, hogy mindig van olyan megoldása az eredeti feladatnak, amely a mohó választással kezdődik!
3. Optimális részprobléma tulajdonság: Bizonyítsuk be, hogy a mohó választással olyan redukált problémát kapunk, amelynek optimális megoldásához hozzávéve a mohó választást, az eredeti probléma megoldását kapjuk!
4. Ha a bizonyítás nem megy, akkor keressünk ellenpéldát, ami megmutatja, hogy a mohó választás nem vezet optimumra!

Benzinkút probléma

Az **A** és **B** közötti útvonalon N benzinkút található. (**A**-ból indulunk, ahol van benzinkút – itt kell tankolnunk!.) Ismerjük az egyes benzinkutak távolságát (és az utolsó benzinkút **B**-től való távolságát), valamint azt, hogy tele tankkal az autónk hány kilométert tud megtenni (K)! Számold ki, hogy minimum hány helyen kell tankolnunk, s mondd is meg, hogy mely benzinkutaknál!

Példa

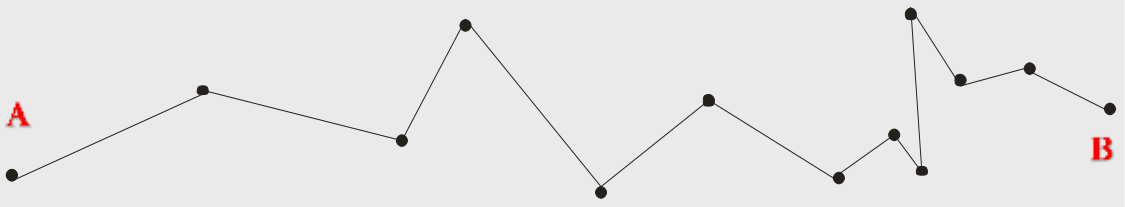
Bemenet

N=12
K=350
Távolságok: 280, 260, 60, 230, 100, 130, 70, 40, 120, 80, 60, 100

Eredmény

Tankolások száma: 6

Lehetséges helyük: 1, 2, 4, 6, 9, 12



Ötlet: Ha a kutak közötti távolság értékek helyett az A ponttól való távolságokat használjuk, akkor a szükséges benzinmennyiségeket egyszerűbben kezelhetjük. Ne felejtsük el, hogy A-ban tankolnunk kell, bővítsük a Táv sorozatot egy kiinduló 0 értékkel!

Probléma: a benzinkutak halmazának egy B_1, …,B_k részhalmaza, ahol Táv(B_{i+1})-Táv(B_i)≤K és Táv(N)-Táv(B_k)≤K. Egy N elemű halmaznak azonban sajnos 2^N darab részhalmaza van, amit meg kellene vizsgálni.

Ami nem kérdés: a kezdőpontban (azaz az 1. benzinkútnál) tankolni kell, azaz B_1=1.

Ami szintén nem kérdéses: a célpontban már nem kell tankolni!

Megoldás:

Mohó választás: Mindig a lehető legkésőbb tankoljunk, azaz B_2 legyen az a benzinkút, amelyre Táv(B_2)-Táv(B_1)≤K, de Táv(B_2+1)-Táv(B_1)>K.

Bizonyítás: Ha korábban (valamely j sorszámú benzinkútnál) tankolnánk, akkor 2 lehetőség van:

- Táv(B3)-Táv(j)≤K → ugyanolyan számú tankolással célba érhetünk.
- Táv(B3)-Táv(j)>K → másodszor is hamarabb kell tankolnunk, ekkor a megoldás a további tankolási helyektől függ,
- …

Azaz ha nem mohó módon választunk, akkor a tankolások száma vagy nem változik, vagy nagyobb lesz!

```
Benzinkút(N,Táv,db,B):
  db:=1; B(db):=1
  Ciklus i=1-től N-ig
    Ha Táv(i+1)-Táv(B(db))>K akkor db:=db+1; B(db):=i
  Ciklus vége
Eljárás vége.
```

Megjegyzés: Ha `Táv(N)-Táv(B(db))>K`, akkor nincs megoldás.

Minta kódok

C++	cpp/1_Tankolas/feladat.cpp
C#	cs/1_Tankolas/feladat.cs
Java	java/1_Tankolas/feladat.java
Pascal	pas/1_Tankolas/feladat.pas
Python	py/1_Tankolas/feladat.py

Staféta

Az olimpiai lángot egy kiindulási városból a cél városba kell eljuttatni. A két város távolsága K kilométer. A szervezők meghirdették, hogy olyan futók jelentkezését várják, akik pontosan H kilométert futnak az olimpiai lánggal. Sok futó jelentkezett, mindegyik megadta, hogy hányadik kilométertől vállalja a futást. A szervezők ki akarják választani a jelentkezők közül a lehető legkevesebb futót, akik végig viszik a lángot.

Ha egy futó az x kilométertől fut, akkor minden olyan futó át tudja venni tőle a lángot, aki olyan z kilométertől vállalja a futást, hogy $z \leq x+H$.

A kiindulási városból biztosan indulni kell egy futónak. A megoldás a futók egy olyan $F_1, \ldots, F_k$ részhalmaza, amikor minden futó a lehető legkésőbb adja át a lángot a következő futónak.

Példa

Bemenet

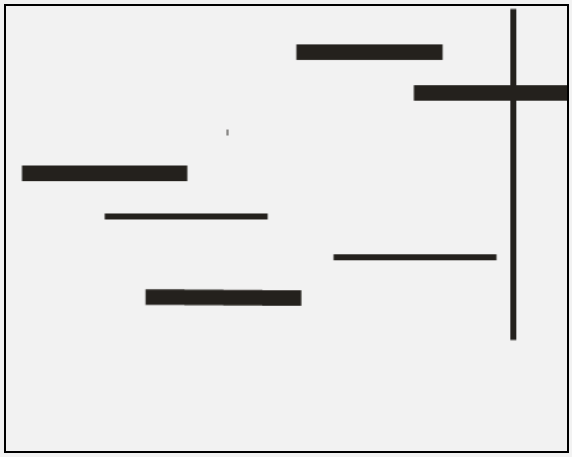
K=30

Futók száma=7, egy futó 10 kilométert futhat.

Kezdetek:

17
24
13
0
5
19
7

Szemléltetése



Ötlet:

Ha sorba rendezzük a futókat az indulási hely szerint, akkor a feladat megoldása azonos a benzinkutas feladat megoldásával. Mivel a sorszámok rendezés közben elvesznének, ezért az S vektorban tároljuk a futók eredeti sorszámait!

Megoldás:

```
Staféta(N,E,H,db,B):
  Rendezés(N,E,S)
  db:=1; B(db):=S(1)
  Ciklus i=2-től N-1-ig
    Ha E(i+1)>E(B(db))+H akkor db:=db+1; B(db):=S(i)
  Ciklus vége
Eljárás vége.
```

Megjegyzés: Ha `E(N)>E(B(db))+K`, akkor nincs megoldás.

*Futási idő: O(N*log(N))*

Ötlet: Ha eleve rendezve kapjuk az adatokat, akkor a megoldás egyszerűbb

```
Staféta(N,E,H,db,B):
  db:=1; B(db):=S1
  Ciklus i=2-től N-1-ig
    Ha E(i+1)>E(B(db))+H akkor db:=db+1; B(db):=i
  Ciklus vége
Eljárás vége.
```

Futási idő: O(N)

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Mohó algoritmusok
Feladat	33. Olimpiai láng ***

Staféta

Az olimpiai lángot egy kiindulási városból a cél városba kell eljuttatni. A két város távolsága K kilométer. Sok futó jelentkezett, mindegyikről tudjuk, hogy hányadik kilométertől hányadik kilométerig vállalja a futást. A szervezők ki akarják választani a jelentkezők közül a lehető legkevesebb futót, akik végig viszik a lángot.

Ha egy futó az x kilométertől az y kilométerig vállalja a futást, akkor minden olyan futó át tudja venni tőle a lángot, aki olyan z kilométertől vállalja a futást, hogy $x \leq z \leq y$.

Példa

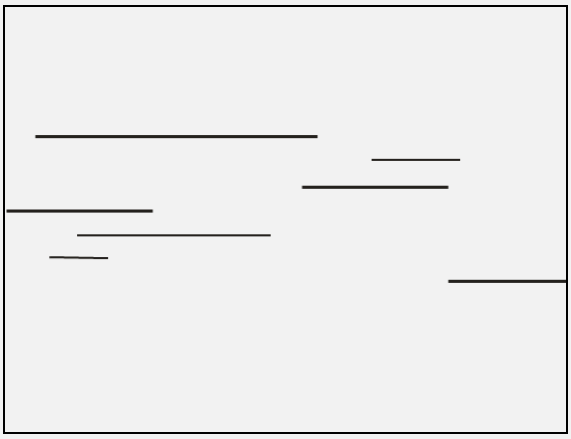
Bemenet

K=40

Futók száma=7

Kezdet	Vég
2	21
25	35
20	34
0	10
5	18
3	7
34	40

Szemléltetése



A kiindulási városból biztosan indulni kell egy futónak. A megoldás a futók egy olyan F_1, …,F_k részhalmaza, amikor minden futó annak adja át a lángot, aki a lehető legtovább tudja vinni.

Az i-edik futó `E(i)` kilométertől `V(i)` kilométerig vállalja a láng vitelét.

Ötlet:

Rendezzük sorba a futókat az indulási hely szerint! Mivel a sorszámok rendezés közben elvesznének, ezért az S vektorban tároljuk a futók eredeti sorszámait!

Az utoljára kiválasztott futó érkezési helyéig válasszuk ki azt a futót, aki a legmesszebb vinné a lángot! Ha a következő futó már később indul, mint az aktuális futó befejezné a futást, akkor a legmesszebb menőnek kell átadnia a lángot.

Megoldás:

```
Staféta(N,E,V,db,B):
  Rendezés(N,E,V,S)
  db:=1; B(db):=S(1); lm:=1
  Ciklus i=2-től N-1-ig
    Ha V(i)>V(lm) akkor lm:=i
    Ha E(i+1)>V(B(db)) akkor db:=db+1; B(db):=S(lm)
  Ciklus vége
Eljárás vége.
```

Megjegyzés: Ha `E(N)>V(B(db))`, akkor nincs megoldás.

*Futási idő: O(N*log(N))*

Ötlet: Ha eleve indulási hely szerint rendezve kapjuk az adatokat, akkor a megoldás egyszerűbb:

```
Staféta(N,E,V,db,B):
  db:=1; B(db):=1; lm:=1
  Ciklus i=2-től N-1-ig
    Ha V(i)>V(lm) akkor lm:=i
    Ha E(i+1)>V(B(db)) akkor db:=db+1; B(db):=lm
  Ciklus vége
Eljárás vége.
```

Futási idő: O(N)*

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Mohó algoritmusok
Feladat	34. Olimpiai staféta

Raktár

Egy raktárban egyetlen hosszú sorban ládák vannak. Minden láda kocka alakú, de méretük különböző lehet. A ládák egy-másra rakásával akarnak helyet felszabadítani. A biztonsági előírás szerint több ládát is lehet egymásra rakni, de minden ládát csak nála nagyobbra lehet helyezni. 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 pedig nagyobb, mint i). Minden ládát legfeljebb egyszer lehet mozgatni.

Ötlet:

A mohó stratégia: az első ládára, amire már feltétlenül pakolni kell, a lehető legtöbb ládát pakoljuk fel úgy, hogy tőle balra ne maradjon pakolandó láda!

Haladjunk balról jobbra, amíg a láda méret növekszik. Ezek biztosan rátehetők arra, ameddig elértünk, de a tőle jobbra csökkenő sorrendben levők is (hacsak nincs két egyforma a két oldalon).

Példa:

1 3 **5** 4 2 | 6 **8** 7 | **6** 5 3 | **4**

Azaz a ládák 4 toronyba pakolhatók, az egyik lehetséges pakolást a bemenet elemei szegélyezése mutatja.

Megoldás:

```
Pakol(m):
  m:=0; i:=1; a[n+1]:=0
  Ciklus
    b:=i
    Ciklus amíg i<n és a[i]<a[i+1]    {amíg növekszik a méret}
      i:=i+1
    Ciklus vége
    bb:=i-1; j:=i+1; t:=a[i]
    Ciklus amíg b≤bb
      Ha a[j]<t és a[bb]<a[j] akkor t:=a[j]; j:=j+1
                              különben t:=a[bb]; bb:=bb-1
    Ciklus vége
    … {maradtak jobbra}
    Ciklus amíg j≤n és t>a[j]
      t:=a[j]; j:=j+1
    Ciklus vége
    m:=m+1; i:=j
  amíg i≤n
  Ciklus vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Mohó algoritmusok
Feladat	21. Ládapakolás ***

Lift

Egy N emeletes házban szokatlan módon üzemeltetik a liftet. A lift az első szintről indul és mindig felmegy a legfelső szintre, majd visszatér az első szintre. Menet közben megáll minden olyan szinten, amelyik úti célja valamelyik liftben tartózkodó utasnak. Olyan szinten is megáll, ahonnan utazni szándékozik valaki az aktuális irányban, feltéve, hogy még befér a liftbe (figyelembe véve az adott szinten kiszállókat). A liftben egyszerre K ember lehet.

Legkevesebb hány menet (egyszer felmegy, majd lejön) szükséges ahhoz, hogy minden várakozó embert elszállítson a lift?

Megoldás:

Legyen `E(i,j)` az i. szintről utazó j. ember célemelete `(E,i,j)=0` az utolsó után)! Ezek alapján ki tudjuk számolni azt is, hogy az i. emeleten felfelé menő liftből hányan szállnának ki `(CF(i))`, illetve a lefelé menő liftből hányan szállnának ki `(CL(i))`.

```
Lift:
  CF():=(0,…,0); CL():=(0,…,0)
  Ciklus i=1-től N-ig
    j:=1
    Ciklus amíg E(i,j)>0
      Ha E(i,j)>i akkor CF(E(i,j)):=CF(E(i,j))+1
      Ha E(i,j)<i akkor CL(E(i,j)):=CL(E(i,j))+1
      j:=j+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Ezekből azt is kiszámolhatjuk, hogy az i. emeletről hányan mennének felfelé (`F(i)`), illetve lefelé (`L(i)`).

A mohó megoldás lényege: mindig a lehető legtöbb ember legyen a liftben!

$$\text{Menet} = \max_{i=1..n}\left(\max\begin{Bmatrix} (F(i)-1) & \text{div} & K+1 \\ (L(i)-1) & \text{div} & K+1 \end{Bmatrix}\right)$$

```
Lift:
  Menet:=0; F(0):=0
  Ciklus i=1-től N-ig
    F(i):=F(i-1)-CF(i); j:=1
    Ciklus amíg E(i,j)>0
      Ha E(i,j)>i akkor F(i):=F(i)+1
      j:=j+1
    Ciklus vége
    At:=(F(i)-1) div K+1
    Ha At>Menet akkor Menet:=At
  Ciklus vége
  L(N+1):=0
  Ciklus i=N-től 1-ig -1-esével
    L(i):=L(i+1)-CL(i); j:=1
    Ciklus amíg E(i,j)>0
      Ha E(i,j)<i akkor L(i):=L(i)+1
      j:=j+1
    Ciklus vége
    At:=(L(i)-1) div K+1
    Ha At>Menet akkor Menet:=At
  Ciklus vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Mohó algoritmusok
Feladat	26. Lift-menetek ***

Robotok

Egy üzemben a gyártást automatizálták. A szerszámgépek egy nagy gépcsarnokban négyzetrács mentén vannak elhelyezve. A műszak végén robotok gyűjtik össze a szerszámgépek gyártotta alkatrészeket. A robotok négyzetrács alakú pályán mozognak a szerszámgépek fölötti térben. A négyzetrács bal felső sarkából, az `(1,1)` pontból indulnak, és a jobb alsó sarokba viszik el az alkatrészeket. A robotokat úgy tervezték, hogy csak „jobbra” és „lefelé” haladhatnak.

Minimálisan hány robotot kell elindítani az összes alkatrész begyűjtéséhez?

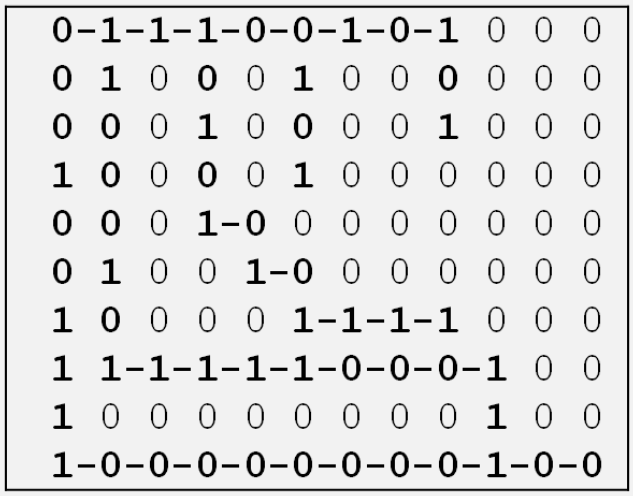
Ötlet:

Keressük meg az aktuális oszlopban a legalsó 1-est!

Idáig egy robotnak biztos le kell jönnie, a korábbi oszlopokból a leglejjebb, de ennél a pontnál feljebb levő átjöhet ezt az 1-est is felszedni, s kereshetünk tovább az oszlopban felfelé.

Ha újabb 1-est találunk, akkor meg kell nézni, hogy az előző oszlopokból jöhet-e át ide valamelyik robot, … és így tovább.

Legyen `U(i)=1`, ha az i-edik sorra kell robot!



Megoldás:

```
Robotok:
  U():=(1,0,…,0)
  Ciklus j=1-től N-ig
    i:=M+1; U(0):=1
    Ciklus amíg i>0
      Ciklus
        i:=i-1
      amíg i>0 és T(i,j)≠1
      Ciklus vége
      Ha i>0 akkor ii:=i
                   Ciklus amíg U(ii)=0
                     ii:=ii-1
                   Ciklus vége
                   U(ii):=0; U(i):=1; i:=ii
    Ciklus vége
  Ciklus vége
  Megold:=0
  Ciklus i=1-től M-ig
    Megold:=Megold+U(i)
  Ciklus vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Mohó algoritmusok
Feladat	3. Alkatrészt gyűjtő robot **