



Belépő a tudás közösségébe

Informatika szakköri segédanyag



Elemi algoritmusok (a programozási tételek után)

Szlávi Péter, Zsakó László

Szerkesztő: Zsakó László, Abonyi-Tóth Andor

A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFO/16-2017-006) című pályázat keretében készült 2019-ben



Eötvös Loránd Tudományegyetem
Informatikai Kar

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

Bár a programozási tételek segítségével minden feladat megoldható, nem mindegy azonban, hogy hogyan alkalmazzuk őket.

Ez a füzet arról szól, hogy hogyan lehet okosan elemi algoritmusokat választani.

1. Kumulatív összegzés

Feladat: Adott egy N elemű számsorozat, adjuk meg a sorozat azon $[a, b]$ intervallumát, ahol az elemek összege maximális! Olyan a és b indexet kell megadnunk, amire az alábbi feltétel teljesül:

$$1 \leq a \leq b \leq N \quad \text{és} \quad \forall p, q (1 \leq p \leq q \leq N): \sum_{i=p}^q X_i \geq \sum_{i=a}^b X_i$$

Az alapmegoldás kiszámolja minden intervallumra a számok összegét, majd ezekre maximumkiválasztást végez:

```

összeg(i, j) :
  S:=0
  Ciklus k=i-től j-ig
    S:=S+X(k)
  Ciklus vége
  összeg:=S
Függvény vége.

Alapmegoldás(MaxÉrt, a, b) :
  MaxÉrt:=-∞
  Ciklus i=1-től N-ig
    Ciklus j=i-től N-ig
      s:=összeg(i, j)
      Ha s>MaxÉrt akkor MaxÉrt:=s; a:=i; b:=j
    Ciklus vége
  Ciklus vége
Eljárás vége.

```

A három ciklus miatt a futási ideje N^3 -bel arányos. Gyakran alkalmazható ötlet, hogy először nem azt számoljuk ki, amit a feladat kér, hanem valami egyszerűbbet, amiből viszont már könnyű megkapni a feladat megoldását.

Számítsuk ki az s vektorba a sorozat első i elemének összegét!

$$\forall i(0 \leq i \leq N): \quad s(i) = \sum_{j=1}^i x_j$$

ami egyszerűbben:

$$\forall i(1 \leq i \leq N) \quad s(i) = s(i-1) + x(i), \quad s(0) = 0$$

Ezután egy intervallum elemei összege így fejezhető ki:

$$\sum_{i=a}^b x_i = s(b) - s(a-1)$$

Az alábbi algoritmus – a kumulatív összegzés módszere – így N^2 -tel arányos lépést tesz meg.

Kumulatív összegzés (Maxért, a, b) :

```

s(0) := 0
Ciklus i=1-től N-ig
    s(i) := s(i-1) + X(i)
Ciklus vége
MaxÉrt := -∞
Ciklus i=1-től N-ig
    Ciklus j=i-től N-ig
        Ha s(j) - s(i-1) > MaxÉrt
            akkor MaxÉrt := s(j) - s(i-1); a := i; b := j
    Ciklus vége
Ciklus vége
Eljárás vége.
    
```

A feladat nemcsak vektorokra, hanem más struktúrákra is általánosítható, például a következőben egy mátrixra:

Feladat: Egy földműves egy téglalap alakú területet szeretne vásárolni egy $N \times M$ -es téglalap alakú földterületen. Tudja minden megvásárolható földdarabról, hogy azt megművelve mennyi lenne a haszna vagy vesztesége. Add meg azt a téglalapot, amelyen a legnagyobb haszon érhető el!

Az elemi megoldásban minden résztéglalapnak ki kellene számolni az összegét, ami 6 ciklus eredményezne, majd ezek maximumát vehetnénk.

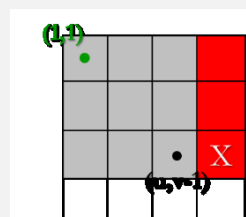
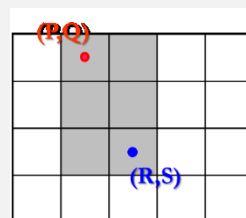
Próbáljunk valami részcélt kitűzni: számoljuk ki az $(1, 1)$ bal felső, (u, v) jobb alsó sarkú téglalapok értékét!

$X =$ a szürke téglalap értéke + a piros téglalap összege

Az (i, j) bal felső, (u, v) jobb alsó sarkú téglalap értékei összege ezek alapján:

$$E[u, v] - E[u, j-1] - E[i-1, v] + E[i-1, j-1]$$

	...	j-1	j	...	v	...
⋮						
i-1	...	$+E_{i-1,j-1}$	$E_{i-1,j}$	...	$-E_{i-1,v}$	...
i	...	$E_{i,j-1}$	$E_{i,j}$	...	$E_{i,v}$	
⋮						
u	...	$-E_{u,j-1}$	$E_{u,j}$		$E_{u,v}$	
⋮						



Összegzés helyett megszámlálási feladatokra is alkalmazható ugyanez az elv.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Egyéb feladatok
Feladat	31. Legnagyobb hasznú téglalap

A kumulatív összegzés alapgondolata tehát: Tetszőleges részhalmazon számolt érték helyett csak olyan részhalmazokra számoljunk értéket, amelyek kezdete az egész halmaz kezdete, majd a részhalmazon számolt értéket fejezzük ki ezekből!

2. Ablak léptetés

Feladat: Adott egy N elemű számsorozat, adjuk meg azt a pontosan K hosszú részintervallumát, amelyben az értékek összege maximális!

Az elemi megoldás minden K hosszú intervallumra összeget számol, majd veszi ezek maximumát.

```
Alapmegoldás (Maxért, Max) :
  MaxÉrt := -∞
  Ciklus i = 1-től N-K+1-ig
    s := összeg(i, i+K-1)
    Ha s > Maxért akkor MaxÉrt := s; Max := i
  Ciklus vége
Eljárás vége.
```

Ez a megoldás – az előző fejezetbeli összeg függvénnyel együtt – N^2 lépésben adja meg az eredményt.

Könnyen észrevehető azonban, hogy két egymást követő intervallum elemei összege két értékkel tér el, a korábbi legelső elemét le kell vonni, a korábbi összegéből, majd hozzá kell adni az új utolsó elemét:

```
Ablakozás (Maxért, Max) :
  MaxÉrt := összeg(1, K); Max := 1
  Ciklus i = 2-től N-K+1-ig
    s := s - x[i-1] + x[i+K-1]
    Ha s > Maxért akkor MaxÉrt := s; Max := i
  Ciklus vége
Eljárás vége.
```

Ezzel a futási idő a tömb elemszámával lett arányos.

Az ablak persze nem mindig ilyen szabályosan változik, mint ahogy a következő feladatban is látható:

Feladat: Egy országban az elmúlt N napon M földrengés volt, ismerjük az egyes földrengések napsorszámát, időpont szerint növekvő sorrendben. Az is lehet, hogy egy napon több földrengés volt, ekkor a napsorszám ismétlődik. Meg kell adni annak a K napos időszaknak az első napját, amelyen belül a lehető legtöbb földrengés volt!

Az M elemű F vektorban kapjuk az egyes földrengések időpontját. A naiv megoldás az F vektor minden elemétől kezdődően leszámolja, hogy K napon belül hány földrengés volt.

```
Alapmegoldás (MaxÉrt, Max) :
  MaxÉrt:=1; Max:=1; F[M+1]:=N+K
  Ciklus i=1-től M-1-ig
    j:=i
    Ciklus amíg j≤M és F[j+1]-F[i]<K
      j:=j+1
    Ciklus vége
    Ha j-i+1>MaxÉrt akkor MaxÉrt:=j-i+1; Max:=i
  Ciklus vége
Eljárás vége.
```

A két ciklus miatt a futási idő itt is az elemszám négyzetével arányos.

A hatékony megoldás ötlete egy ablak bevezetése, amit a sorozaton balról jobbra csúsztatunk úgy, hogy mindig egy K hosszúságú időtartamot fed le:

```
Ablakozás (MaxÉrt, Max) :
  Kezdet:=1; Vég:=1; F[M+1]:=N+K
  Ciklus amíg Vég≤M és F[Vég+1]-F[Kezdet]<K
    Vég:=Vég+1
  Ciklus vége
  MaxÉrt:=Vég-Kezdet+1; Max:=Kezdet
  Ciklus Kezdet=2-től M-K+1-ig
    Ciklus amíg Vég≤M és F[Vég+1]-F[Kezdet]<K
      Vég:=Vég+1
    Ciklus vége
    Ha Vég-Kezdet+1>MaxÉrt akkor MaxÉrt:=Vég-Kezdet+1; Max:=kezdet
  Ciklus vége
Eljárás vége.
```

Ebben a megoldásban a Kezdet is 1-től növekszik egyesével, valamint a Vég is, bár utóbbi lehet, hogy más ütemben, mint a Kezdet. Ezek alapján a futási idő az elemek számával arányos.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Földrengések

Hasonló feladat: Adjuk meg egy pozitív számokból álló N hosszúságú sorozat egy olyan részintervallumát, ahol a számok összege pontosan K ! Ha tetszőleges $[Kezdet, Vég]$ intervallum S összege kisebb, mint K , akkor $Vég$ növelhető. Ha nagyobb, mint K , akkor $Kezdet$ -et kell növelni. Ha pontosan K , akkor találtunk egy jó intervallumot.

```
Ablakozás (van, Kezdet, Vég) :
  Kezdet:=1; Vég:=1; S:=x[1]
  Ciklus amíg Vég<N és S≠K
    Ha S<K akkor Vég:=Vég+1; S:=S+x[Vég]
    különben ha S>K akkor S:=S-x[Kezdet]; Kezdet:=Kezdet+1
  Ciklus vége
  van:=S=K
Eljárás vége.
```

Egy ablakot mátrixban is mozgathatunk, csak ott a mozgatás is több irányba történhet.

Feladat: Egy $N \times M$ -es téglalap alakú területen határozzuk meg azt a $K \times K$ -as négyzetet, amelyikben a legtöbb hegycsúcs van! Hegycsúcsnak nevezzük azokat a pontokat, amelyek magasabbak 4 közvetlen szomszédjuknál.

Az alapmegoldásban minden $K \times K$ -as négyzet alakú területre kiszámoljuk a csúcsok számát, és vesszük ezek maximumát. A futási idő a méret negyedik hatványával lesz arányos. Egy $K \times K$ méretű négyzet csúcsai számából az eggyel jobbra tolt ugyanekkora négyzetben levő csúcsok száma könnyen számolható: balról kimarad egy K magasságú oszlop, jobbról pedig bejön egy K magasságú oszlop. Ha már egy, ezeknél a négyzeteknél egy sorral feljebb levő négyzeteket már számoltunk, akkor a kimaradó és a belépő oszlop értékei már majdnem ki vannak számolva, tudjuk a belépő oszlopnál eggyel feljebb levő oszlop csúcsai számát és ugyanezt a kimaradóra is.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Hegycsúcsok

A módszer lényege: Ha egy sorozat részintervallumain kell kiszámolni valamilyen értéket, akkor a részintervallumot, mint egy ablakot csúsztatassuk végig a sorozat elemein (akár úgy is, hogy az ablak két széle különböző ütemben mozog)!

3. Darabszám helyett változás

Feladat: Adott N intervallum, kezdő- és végpontjaik 1 és M közötti számok. Adjunk meg egy értéket, amely a legtöbb intervallumban benne van!

A legegyszerűbb megoldási ötlet: minden 1 és M közötti i értékre számoljuk meg, hogy hány intervallumban szerepel, majd vegyük ezen értékek maximumát!

```
Alapmegoldás (Maxért, Max) :
  Maxért := -1
  Ciklus i = 1-től M-ig
    db := 0
    Ciklus j = 1-től N-ig
      Ha  $i \geq \text{Kezdet}[i]$  és  $i \leq \text{Vég}[i]$  akkor  $\text{db} := \text{db} + 1$ 
    Ciklus vége
    Ha  $\text{db} > \text{Maxért}$  akkor  $\text{Max} := i$ ;  $\text{Maxért} := \text{db}$ 
  Ciklus vége
Eljárás vége.
```

A megoldásból jól látszik, hogy az algoritmus futási ideje $N \cdot M$ -mel arányos.

Másik megoldási ötlet: A darabszámokat az intervallumokból számoljuk.

```
Legtöbb (Maxért, Max) :
  Darab := (0, ..., 0)
  Ciklus i = 1-től N-ig
    Ciklus j = Kezdet[i]-től Vég[i]-ig
      Darab[j] := Darab[j] + 1
    Ciklus vége
  Ciklus vége
  Max := 1
  Ciklus i = 2-től M-ig
    Ha  $\text{Darab}[i] > \text{Darab}[\text{Max}]$  akkor  $\text{Max} := i$ 
  Ciklus vége
Eljárás vége.
```

Ennek a megoldásnak a lépésszáma az intervallumok összhosszúságával arányos. Ez az előző változatnál csak akkor nem jobb, ha minden intervallum 1-től M -ig tart.

A fejezet ötlete most következik: felesleges minden i értékre előre kiszámolni, hogy hány intervallumnak eleme. Elég csupán annyit tudni, hogy az i hányval több vagy kevesebb intervallumnak eleme, mint az $i-1$ volt.

```
Legtöbb (Maxért, Max) :
  Darab := (0, ..., 0)
  Ciklus i = 1-től N-ig
    Darab[Kezdet[i]] := Darab[Kezdet[i]] + 1
    Darab[Vég[i+1]] := Darab[Vég[i+1]] - 1
  Ciklus vége
  Max := 1; db := Darab[1]; Maxért := db
  Ciklus i = 2-től M-ig
    db := db + Darab[i]
    Ha  $\text{db} > \text{Maxért}$  akkor  $\text{Max} := i$ ;  $\text{Maxért} := \text{db}$ 
  Ciklus vége
Eljárás vége.
```

A futási idő az első megoldás $N \cdot M$ értéke helyett $N + M$ -mel lesz arányos.

Feladat: Ismerjük egy vállalkozó havi bevételeit az elmúlt N hónapra. Tudjuk, hogy kezdetben hány százalék adót kellett fizetnie erre, az adóhivatal azonban havonta –akár többször is– megváltoztathatja ezt a százalékot, növelheti is valamennyivel, csökkentheti is valamennyivel! Az aktuális havi százalékkal kell az a havi bevétel adóját kiszámolni. Számítsuk ki a vállalkozó által az N hónap során fizetendő adót!

Az alapmegoldásban első lépésként minden adóváltozás után minden hónapra kiszámolhatnánk, hogy mennyi az arra a hónapra az adószázalék, majd ezután számolnánk a fizetendő adót. Ebben a feladatban viszont éppen az adóváltozásokat ismerjük.

M alkalommal történt adóváltozás, tudjuk mindegyiknél a hónapsorszámot és az adószázalék változását.

Adózás (A) :

Haviszázalék:=(0,...,0)

Ciklus $i=1$ -től M -ig

Haviszázalék[hónap[i]]:=Haviszázalék[hónap[i]]+Változás[i]

Ciklus vége

Százalék:=Kezdőszázalék; Adó:=0

Ciklus $i=1$ -től N -ig

Százalék:=Százalék+Haviszázalék[i]

Adó:=Adó+bevétel[i]*Százalék

Ciklus vége

Eljárás vége.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Mozgó adók

A módszer lényege: Ha egy sorozat egyes értékeit sokáig tart kiszámolni, akkor célszerű lehet e helyett csak azt számolni, hogy a sorozat egyes elemei hogyan változnak az előzőhöz képest, majd a sorozat elemeit a változásokból kiszámolni.

4. Intervallum szűkítés

Feladat: Adott egy növekvő számsorozat. Adjuk meg két elemét, amelyek összege pontosan Z !

A naiv megoldás nagyon egyszerű, minden számpárra megnézzük, hogy az összegük Z -e.

```
Alapmegoldás (van, a, b) :
  i:=1; j:=2
  Ciklus amíg i<N és x[i]+x[j]≠Z
    Ha j<N akkor j:=j+1 különben i:=i+1; j:=i+1
  Ciklus vége
  van:=(i<N)
  Ha van akkor a:=i; b:=j
Eljárás vége.
```

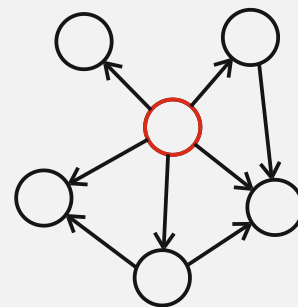
A megoldás rossz esetben minden számpárt megnéz, azaz futási ideje $N*(N-1)/2$ -vel arányos.

Fogalmazzuk meg a feladatot kicsit általánosabban: Keressük az $[a, b]$ intervallum két olyan számát, amelyek összege pontosan Z !

Ha $x[a] + x[b] < Z$, akkor az $x[a]$ biztosan túl kicsi, a növelhető. Ha $x[a] + x[b] > Z$, akkor $x[b]$ biztosan túl nagy, b csökkenthető. Ha pedig $x[a] + x[b] = Z$, akkor megtaláltuk a megoldást: a-t és b-t. Ha közben $a=b$ lesz, akkor nincs megoldás.

```
Összeg (van, a, b) :
  a:=1; b:=N
  Ciklus amíg a<b és x[a]+x[b]≠Z
    Ha x[a]+x[b]<Z akkor a:=a+1 különben b:=b-1
  Ciklus vége
  van:=(a<b)
Eljárás vége.
```

Hasonló, bonyolultabbnak tűnő **feladat** a gráfok témaköréből: Adjunk meg egy gráfban egy superforrást (ha van)! A superforrás olyan pontja a gráfnak, amelyből minden más pontba megy él, bele pedig sehonnan. A gráfra egyetlen műveletünk van: meg tudjuk kérdezni, hogy van-e él A-ból B-be.



A naiv megoldás minden pontra megnézné, hogy teljesül-e a fenti feltétel, ennek a futási ideje a pontszám négyzetével arányos. Ügyesebb megoldást kapunk intervallum szűkítéssel.

Egy A és egy B pontról egyetlen vanél? (A, B) művelettel el tudjuk dönteni, hogy melyikük biztosan nem superforrás. Ha $\text{vanél?}(A, B) = \text{igaz}$, akkor B biztosan nem superforrás (mert vezet bele él. Ha hamis, akkor A biztosan nem superforrás, mert belőle nem megy mindenhova él.

Így az intervallum szűkítéses megoldás:

```
Szuperforrás (van, a) :
  a:=1; b:=n
  Ciklus amíg a<b
    Ha vanél?(a,b) akkor b:=b-1 különben a:=a+1
  Ciklus vége
  i:=1; Ha i=a akkor i:=i+1
  Ciklus amíg i≤N és vanél?(a,i) és nem vanél?(i,a)
    i:=i+1; Ha i=a akkor i:=i+1
  Ciklus vége
  van:=i>N
Eljárás vége.
```

A műveletigény mindkét ciklusra a pontok számával arányos.

Feladat: Egy kétszemélyes játékban páros számú gyöngy kupacokat tartalmazó sorozatból indulunk ki. Egy játékos a sorozat egyik széléről elveheti a szélső kupacot, majd a másik játékos következik. Az nyer, akinek a végén több gyöngye lesz. Írj nyerő stratégiát a feladatra!

Ha a kupacok száma páros, akkor a kezdő játékos dönthet, hogy páros vagy páratlan sorszámú kupacot vesz el először. Az ellenfélnek ezután a két szélén vagy csak páros, vagy csak páratlan sorszámú kupacok maradnak, azaz ő ebben nem dönthet. (Célszerűen az ellenfél mindig a nagyobbik szélsőt választja, remélve, hogy a társa hibázik.) Amint valamelyik szélről elvesz egy kupacot, a kezdő játékos megint szabadon dönthet, hogy páros vagy páratlan sorszámú kupacot vesz el. ha a gyöngyöket előre leszámolják, akkor lehet tudni, hogy a páros vagy a páratlan számú kupacok választása-e a nyerő stratégia.

```
Játék:
  páratlandb:=0; párosdb:=0; páratlan:=igaz
  Ciklus i=1-től N-ig
    Ha páratlan akkor páratlandb:=páratlandb+x[i]
    különben párosdb:=párosdb+x[i]
  páratlan:=nem páratlan
  Ciklus vége
  páratlankell:=páratlandb>párosdb
  első:=1; utolsó:=N
  Ciklus amíg első<utolsó
    Ha páratlankell akkor
      Ha első mod 2=1 akkor Lépésem(első); első:=első+1
      különben Lépésem(utolsó); utolsó:=utolsó-1
    különben
      Ha első mod 2=0 akkor Lépésem(első); első:=első+1
      különben Lépésem(utolsó); utolsó:=utolsó-1
    Ellenféllép(mi)
    Ha mi=első akkor első:=első+1 különben utolsó:=utolsó-1
  Ciklus vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Interaktív feladatok, kétszemélyes játékok
Feladat	Gyöngy kupacok

A módszer lényege: Ha egy intervallumban kell megadnunk egy (kettő vagy több) elemet és az intervallum két széléről könnyű eldönteni, hogy szóba jöhetnek-e a megoldásban vagy nem, akkor érdemes a teljes intervallumból kiindulni, majd azt szűkíteni a megoldás megtalálásáig.

5. Intervallum kezdetek megőrzése

Feladat: Adott egy N elemű sorozat, amely 1 és M közötti értékeket tartalmaz. Adjuk meg azt az értéket, amely két előfordulása a lehető legközelebb van egymáshoz!

A naiv megoldásban minden lehetséges értékhez megkeressük az őt követő legközelebbi, vele megegyező értéket:

```
Alapmegoldás (van, a) :
  minért := N
  Ciklus i = 1-től N-1-ig
    j := i + 1
    Ciklus amíg j ≤ N és x[i] ≠ x[j]
      j := j + 1
    Ciklus vége
    Ha j ≤ N és j - i < minért akkor minért := j - i; min := i
  Ciklus vége
  van := (minért < N)
  Ha van akkor a := x[min]
Eljárás vége.
```

A futási idő legrosszabb esetben (amikor minden érték különböző) elemszám négyzettel arányos.

Ha megőriznénk minden értékhez, hogy mikor volt utoljára (ut tömb), akkor a megoldás sokkal gyorsabb lehet.

```
Legközelebbi (van, a) :
  minért := N; ut := (-N, ..., -N)
  Ciklus i = 1-től N-ig
    Ha i - ut[x[i]] < minért akkor minért := i - ut[x[i]]; min := ut[x[i]]
    ut[i] := i
  Ciklus vége
  van := (minért < N)
  Ha van akkor a := x[min]
Eljárás vége.
```

Ez az algoritmus biztosan elemszámmal arányos lépést tesz meg.

Feladat: N napon keresztül mértük a déli hőmérsékletet. Különösen hideg napoknak nevezzük azokat, ahol a hőmérséklet kisebb volt -20 foknál, különösen melegnek pedig azokat, ahol nagyobb volt 30 foknál. Adjuk meg, hogy különösen hideg vagy különösen meleg folytonos időszakokból volt-e hosszabb!

A különösen meleg intervallumokat legfeljebb 30 fokos napok határolják, a különösen hidegeket pedig legalább -20 fokosak. A határokból kell megőriznünk a legutolsót.

```
Különöseket(van, s):
    maxhért:=-1; maxmért:=-1; uth:=0; utm:=0
    Ciklus i=1-től N-ig
        Ha x[i]<-20 akkor Ha i-uth>maxhért akkor maxhért:=i-uth
                                utm:=i
        különben ha x[i]>30 akkor Ha i-ut>maxmért akkor maxmért:=i-utm
                                uth:=i
        különben utm:=i; uth:=i
    Ciklus vége
    van:=maxhért+maxmért≠-2
    Ha van akkor Ha maxhért<maxmért akkor s:="meleg"
                    különben ha maxhért<maxmért akkor s:="hideg"
                    különben s:="egyforma"
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Különösen szélsőséges napok

A módszer lényege: Ha adott típusú elemek által határolt intervallumokból kell valamit kiszámolnunk, akkor mindegyik típushoz elég tárolni a típus utolsó előfordulását, majd egy új elemnél meg kell néznünk, hogy az a típus mikor fordult elő utoljára.

6. Párok indexeléssel

Feladat: Egy rendezvényen N vendég vesz részt, érkezési sorrendben ismerjük mindegyik érkezési és távozási idejét, mindkettő 1 és M közötti egész szám. Sem érkezni, sem távozni nem akart két vendég egyszerre. Adjuk meg a vendégeket távozási idő szerinti sorrendben!

Alapváltozatban ez egy egyszerű rendezési feladat, amiben adatpárokat kell rendezni a pár második tagjai szerint. Ez azonban elemi algoritmusokkal elemszám négyzettel arányos lépésben történik, gyors rendezések esetén pedig $N \cdot \log(N)$ lépésben.

Az ügyes változatban vegyünk fel egy *Kezd* tömböt, ahol *Kezd[i]* az *i* időpontban távozó vendég érkezési ideje.

Rendezés:

```

Kezd := (0, ..., 0)
Ciklus i=1-től N-ig
    Kezd[távozik[i]] := érkezik[i]
Ciklus vége
Ciklus i=1-től M-ig
    Ha Kezd[i] > 0 akkor Ki: Kezd[i], i
Ciklus vége
Eljárás vége.

```

Jól látszik, hogy a futási idő $N+M$ -mel arányos lett.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Rendezvény

Feladat: Az olimpiai staféta az olimpiai lángot N városon keresztül viszi. Az első városból induló futó bármely városig vállalja a futást. M további városban jelentkezett egy-egy futó (a városok sorrendjében), aki megmondta, hogy meddig vinné a lángot. Minden futó csak annak adhatja át a lángot, aki abból a városból indul, ameddig ő viszi a lángot. Adjuk meg, hogy az első futó melyik városban adja át a lángot, hogy a lehető legtöbben vehessenek részt a stafétában!

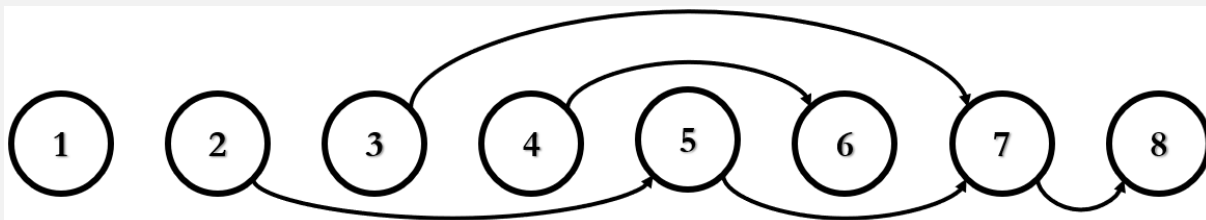
Tudjuk, hogy minden városból maximum egy futó indul, így megadhatjuk a párosítást! A *Meddig* tömb i . eleme tartalmazza azt, hogy az i . városból induló futó melyik városban adná át a lángot a következőnek! Ennek segítségével minden városból induló futóra kiszámolhatjuk, hogy ezzel a módszerrel vele kezdve eljutunk-e az N . városig és addig hány futót kell alkalmaznunk. Figyeljünk arra, hogy ha valaki egy ilyen stafétában már szerepelt nem első helyen, az már nem lehet megoldás első lángot kapóként!

```

Maximum(max) :
  Meddig:=(0,...,0); darab:=(0,...,0); max:=1
  Ciklus i=1-től M-ig
    Meddig[honnan[i]]:=hova[i]
  Ciklus vége
  Ciklus i=1-től M-ig
    Ha darab[honnan[i]]=0 akkor
      j:=honnan[i]; db:=1
      Ciklus amíg Meddig[j]<N és Meddig[Meddig[j]]>0
        db:=db+1; j:=Meddig[j]; darab[j]:=-1
      Ciklus vége
      Ha Meddig[j]=N akkor darab[honnan[i]]:=db
        Ha db>darab[max] akkor max:=honnan[i]
    Elágazás vége
  Ciklus vége
Eljárás vége.

```

A megoldásban a második M lépéses ciklusban van egy belső ciklus. Minden lehetséges kezdőpontra egyetelmű a lánc, azaz K kezdőpont esetén legfeljebb $K \cdot (M - K + 1)$ lépést tehetünk meg.



Megjegyzések:

- A belső ciklust rekurzívan is megírhatnák, és akkor a rekurzió memorizálás módszerével a staféta közbülső pontjairól is tárolhatnánk, hogy belőlük milyen hosszú lánc vezet az N. városig. Ekkor, ha egy már kiszámolt láncba érünk egy új kezdőpontból, akkor a láncot nem kellene tovább követnünk, a tárolt értékből azonnal tudnánk a lánc hosszát.
- Ha a lánc nem adható tovább, akkor az algoritmus 1-et ad eredményül

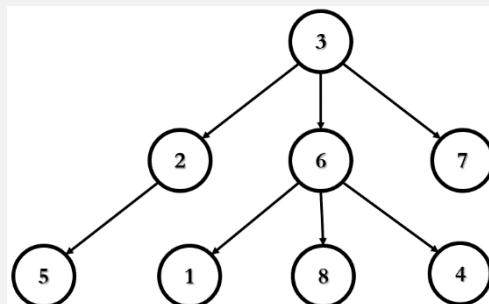
A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Olimpiai staféta

Feladat: Egy N tagú baráti társaságban egy emberrel közöltünk egy hírt, aki azt továbbadta néhány másik embernek, azok szintén továbbadták olyanoknak, akik még nem kapták meg. Mindez addig tartott, amíg a hírt mindenki meg nem kapta. Adjuk meg mindenkiről, hogy a hírt hány közvetítőn keresztül kapta meg!

Egy ember hírt továbbadni akárhány másiktól is tud, hírt kapni viszont mindenki pontosan egytől kap, azaz ez az információ használható párosításra.

$N-1$ híradás történt, mindegyikről tudjuk, hogy ki adta a hírt (adta tömb) és ki kapta a hírt (kapta tömb).



```

Hírek(Darab) :
  Pár:=(0,...,0)
  Ciklus i=1-től N-1-ig
    Pár[kapta[i]]:=adta[i]
  Ciklus vége
  Darab:=(0,...,0)
  Ciklus i=1-től N-ig
    Darab[i]:=Számol(i)
  Ciklus vége
Eljárás vége.

```

A megszámlálásban használjuk a rekurzió memorizálással módszert. Ha egy emberre már egyszer kiszámoltuk, hogy hány emberen keresztül kapta meg a hírt, akkor ne számoljuk újra!

Mivel a Darab vektor N elemű, így a megoldás futási ideje az elemek számával arányos.

```

Számol(i) :
  Ha Darab[i]>0 vagy Pár[i]=0 akkor Számol:=Darab[i]
  különben Számol:=Számol(Pár[i])+1
Függvény vége.

```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Hírterjedés

A módszer lényege: Ha minden elemhez egy párt tudunk hozzárendelni és a feladat megoldásához a párokon való végig haladásra van szükség, akkor érdemes az elemekhez a párjukat indexként tárolni, mert így könnyen végig nézhetjük a megfelelő elemeket.

7. Összefésülés

Egy klasszikus feladattal kezdjük ezt a módszert:

Feladat: Adott két rendezett sorozatként ábrázolt halmaz, egyesítsük őket egy rendezett sorozatként ábrázolt halmazzá!

A naiv megoldás az unió programozási tétel, amelyben az első halmazt (N elemű x tömb) lemásoljuk, majd a második halmaz (M elemű y tömb) azon elemeit vesszük hozzá az eredményhez, amelyek nem szerepeltek az első halmazban. Az algoritmus futási ideje a két halmaz elemszáma szorzatától függ.

```
Egyesítés(K, z) :
  Ciklus i=1-től N-ig
    z[i]:=x[i]
  Ciklus vége
  K:=N
  Ciklus j=1-től M-ig
    i:=1
    Ciklus amíg i≤N és x[i]≠y[j]
      j:=j+1
    Ciklus vége
    Ha i>N akkor K:=K+1; z[k]:=y[j]
  Ciklus vége
Eljárás vége.
```

Az erre a feladatra írt összefuttatás algoritmusából indul ki, hogy az eredmény halmaz első eleme csak valamelyik halmaz első eleme lehet. Ezt tegyük be az eredménybe és vegyük ki abból a halmazból, majd ezt folytassuk, amíg mindkét halmaz ki nem ürül! Figyelni kell persze arra, hogy ha az egyik halmaz kiürült, akkor a másikat már csak le kell másolni!

A futási időt ezzel $N \cdot M$ -ről $N+M$ -re csökkentetjük.

```
Egyesítés(K, z) :
  i:=1; j:=1; K:=0
  Ciklus amíg i≤N és j≤M
    K:=K+1
    Elágazás
      x[i]<y[j] esetén z[K]:=x[i]; i:=i+1
      x[i]=y[j] esetén z[K]:=x[i]; i:=i+1; j:=j+1
      x[i]>y[j] esetén z[K]:=y[j]; j:=j+1
    Elágazás vége
  Ciklus vége
  Ciklus amíg i≤N
    K:=K+1; z[K]:=x[i]; i:=i+1
  Ciklus vége
  Ciklus amíg j≤M
    K:=K+1; z[K]:=y[j]; j:=j+1
  Ciklus vége
Eljárás vége.
```

Megjegyzés: ha a két halmaz végére tennénk egy-egy egyforma, de maximális értékű elemet, akkor a két utolsó ciklust megszüntethetnénk.

Feladat: Ismerjük egy iskola tanulói egy pontversenyen elért eredményeit, tanulók szerint ábécé-sorrendben (N elemű inév és pont tömb). Feltesszük, hogy az iskolában nincs két azonos nevű diák. Megkaptuk egy osztály névsorát is (M elemű onév tömb), ki kellene számolni az osztály összpontszámát!

A naiv megoldásban kikereshetjük az osztály tanulóit az iskolanévsorból, majd összeadhatjuk a pontszámaikat:

```
Alapmegoldás(össz) :
    össz:=0
    Ciklus i=1-től M-ig
        j:=1
        Ciklus amíg onév[i]≠inév[j]
            j:=j+1
        Ciklus vége
        össz:=össz+pont[j]
    Eljárás vége.
```

A belső ciklus futási ideje az iskola létszámával arányos, azaz az össz futási idő $N \cdot M$ -től függ.

Itt is alkalmazhatjuk az összefésülés elvét, a közös neveknél mindkét névsorban lépünk, egyébként csak az iskolanévsorban. A megoldásunk N lépéses lesz:

```
Összpontszám(össz) :
    össz:=0; i:=1; j:=1
    Ciklus amíg i≤N és j≤M
        Ha onév[j]=inév[i] akkor össz:=össz+Pont[i]; j:=j+1; i:=i+1
        különben i:=i+1
    Ciklus vége
    Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Tételek összeépítése
Feladat	Iskolai pontverseny

A módszer lényege: Ha kettő (vagy több) sorozat ugyanolyan szempont szerint rendezett, és a feldolgozásukhoz meg kell találni elemek párpait, akkor érdemes a két sorozatban egymáshoz ütemezetten haladni, mindig abban továbblépni, amelybelinek nem lesz már párja a másikban.