

Megoldott feladatok

Ionescu Klára

clara@cs.ubbcluj.ro

1. Balra át

- Szeptember 15-én az iskola igazgatója felkéri az első osztályos gyermekeket, hogy forduljanak arccal feléje, majd abból a célból, hogy felvezethesse őket az osztálytermekbe, kiadja a parancsot: „Balra át!” A gyermekek bizonytalanok. Van, aki balra fordul, van, aki jobbra. Aki szemben találja magát a mellette állóval, azt hiszi, hogy rosszul fordult és egy időegység alatt egyszer sarkon fordul. Adott egy bizonyos kezdeti konfiguráció, amely a „Balra át!” parancs után alakult ki. Állapítsátok meg, **hány időegység** alatt „nyugszik meg” a sor. (Legtöbb 255 gyerekkel dolgozhatunk.)

Példa: ha $n = 5$, a parancs utáni konfiguráció: **bjbjb**, ahol **b** balra fordult gyereket, **j** jobbra fordult gyereket jelent, az eredmény: **2**

Elemzés

- Amikor két gyermek egymással szemben találja magát, a két gyermeket egy **jb** karakterpáros ábrázolja.
- Az algoritmusban ezt a karakterpárost felülírjuk **bj**-vel.
- Ekkor előfordulhat, hogy a **j**-vel ábrázolt gyermek mellett egy **b**-vel ábrázolt gyermek áll, de most már nem szabad felülírni az értékeiket, mivel a szövegben ki van kötve, hogy egy időegység alatt egy gyermek csak egyszer fordul meg.
- Ezt két féleképpen is megoldhatjuk:
 - Készítünk egy „másolatot” a sorozatról, amelyen végigmegyünk, keressük a **jb** párokat de a módosításokat az eredeti sorozaton végezzük; végül a másolatra rámásoljuk az eredetit.
 - A másolat használata elkerülhető, ha abban az esetben, amikor két gyermek állapota megváltozik, a ciklusváltozót 2-vel növeljük

A legfontosabb észrevétel

- Az algoritmus hasonlít a buborékrendezéshez!
- Különbségek:
 - Nem szükséges felcserélni a két elemet, felülírjuk ezeket.
 - Vagy készítünk ideiglenes másolatot a gyermekek sorozatáról, vagy minden felülírás után kettőt lépünk előre; ebben az esetben **Amíg** struktúrával dolgozunk a **Minden** helyett.
- Az időt csak akkor növeljük, ha volt fordulás, mivel az utolsó bejárást csak ellenőrzés céljából végezzük.

```
Algoritmus balraÁt(n, gy):           // Négyzetes bonyolultság
    időEgység ← 0                     // Bemenet: az n elemű gy karakterlánc
    Ismételd                           // Kimenet: időEgység
        i ← 1
        fordult ← hamis
        Amíg i ≤ n végezd el:
            Ha gyi = 'j' és gyi+1 = 'b' akkor
                gyi ← 'b'
                gyi+1 ← 'j'
                i ← i + 2
                fordult = igaz
            különben
                i ← i + 1
        vége(ha)
    vége(amíg)
    Ha fordult akkor időEgység ← időEgység + 1
    vége(ha)
    ameddig nem fordult
        térítsd időEgység
Vége(algoritmus)
```

Algoritmus balraÁt (n, gy):

// lineáris algoritmus

időEgység $\leftarrow 0$; idő $\leftarrow 0$; lépés $\leftarrow 0$

Minden $i = 1, n$ végezd el:

Ha $gy_i = 'j'$ akkor

idő \leftarrow idő + 1

Ha lépés > 0 akkor 1

lépés \leftarrow lépés - 1

vége(ha)

vége(ha)

Ha $gy_i = 'b'$ akkor

Ha idő > 0 akkor

időEgység \leftarrow idő + lépés

lépés \leftarrow lépés + 1

vége(ha)

vége(ha)

vége(minden)

térítsd időEgység

Vége(algoritmus)

2. Az átlagos várakozási idő minimalizálása

Egy ügyvédi irodába egyszerre érkezik n személy, akiknek az intéznivalóit az ügyvéd ismeri, és így azt is tudja, hogy egy-egy személlyel hány percet fog eltölteni.

*Állapítsuk meg azt a **sorrendet**, amelyben fogadnia kellene a személyeket ahhoz, hogy az **átlagos várakozási idő minimális legyen**.*

Megoldás

Az átlagos várakozási idő az n személy várakozási idejének számtani középarányosa,
⇒ **az átlagos várakozási idő csökkentése a várakozási idők összegének csökkentését jelenti.**

Példa

Legyen $n = 3$ és a tárgyalási idők:

$t_1 = 60$, $t_2 = 10$ és $t_3 = 30$.

- az első személy nem kell várakozzon,
- a második személy annyit várakozik, amennyi ideig tárgyal az első,
- a harmadik addig várakozik ameddig befejeződik a két előtte levővel való beszélgetés.

Ha a fogadás sorrendje változik, a várakozási idők összege is változik.

Sorrend	A várakozási idők összege
1 2 3	$v_i = 0 + 60 + (60 + 10) = 130$
1 3 2	$v_i = 0 + 60 + (60 + 30) = 150$
2 3 1	$v_i = 0 + 10 + (10 + 30) = 50$
2 1 3	$v_i = 0 + 10 + (10 + 60) = 80$

Megjegyzés

Magától értetődik, hogy ha létezik több egyenlő fogadási idő, nem számít milyen sorrendben kerülnek sorra az egyenlő tárgyalási időt igénylő személyek.

⇒ A feladatnak több megoldása lesz.

Következtetés

- Ahhoz, hogy minimalizáljuk az átlagos várakozási időt, **minimalizálnunk kell a várakozási idők összegét.**
- Egy személy addig várakozik, amíg az összes előtte fogadott személlyel tárgyal az ügyvéd.

Az eredmény tehát a személyek sorszámainak egy olyan permutációja, amelynek megfelelően az ügyvéd minden lépésben a legkevesebb időt igénylő személyt fogadja:

Megoldás

- Előbb inicializáljuk az M halmazt az $1, 2, \dots, n$ értékekkel (a személyek indexei) és növekvő sorrendbe rendezzük az időket.
- Megfelelően módosítjuk az M halmaz elemeit is.
- A rendezés után: $M = (k_1, k_2, \dots, k_n)$ és $t_1 \leq t_2 \leq \dots \leq t_n$.
- A kiírást az M halmazban található indexpermutáció alapján végezzük.

Algoritmus Sorrend(n, t, M, vi_min):

// Bemeneti adatok: az n elemű t sorozat

// Kimeneti adatok: az n elemű M sorozat és

// vi_min (minimális összvárakozási idő)

Minden $i = 1, n$ végezd el:

$M_i \leftarrow i$ *// személyek indexei*

vége(minden)

Növekvő_sorrendbe_rendezés(n, t, M)

$vi_min \leftarrow 0$ *// minimális összvárakozási idő*

$vi \leftarrow 0$ *// egy-egy személy várakozási ideje*

Minden $i = 1, n-1$ végezd el:

$vi \leftarrow vi + t_i$

$vi_min \leftarrow vi_min + vi$

vége(minden)

Vége(algoritmus)

3. Autó bérbeadás

Egy szállítási vállalat autókat kölcsönöz. Egy bizonyos jármű iránt igen nagy az érdeklődés, ezért az igényeket egy évre előre jegyzik. Az igényt két számmal jelöljük, amelyek az év azon napjainak sorszámait jelölik, amellyel kezdődően, illetve végződően igénylik az illető autót.

Állapítsuk meg a bérbeadást úgy, hogy a lehető legtöbb személyt szolgáljuk ki.

Adott a személyek száma n és az igényelt intervallumok $(a_i, b_i, i = 1, 2, \dots, n)$.

Példa

$n = 10$

a_i 1 12 5 12 13 40 30 22 70 19

b_i 23 20 10 29 25 66 35 33 100 65

Az igényléseket növekvő sorrendbe rendezzük az időintervallum második eleme szerint:

a_i 5 12 1 13 12 22 30 19 40 70

b_i 10 20 23 25 29 33 35 65 66 100

Bérbeadási lehetőségek:

Bérbeadás	1	2	3	4	5
5 személy	[5,10]	[12,20]	[22,33]	[40,66]	[70,100]
4 személy	[1,23]	[30,35]	[40,66]	[70,100]	
5 személy	[5,10]	[13,25]	[30,35]	[40,66]	[70,100]

Algoritmus autóKölcsönzés(n , a , b , maxIgényekSzáma, M):

// Bemeneti adatok: az n elemű a és b sorozat

// Kimeneti adatok: az n elemű M sorozat és

// maxIgényekSzáma: a legtöbb személy, akiknek bérbe adható az autó

Növekvő_sorrendbe_rendezés(n , a , b):

$M_1 \leftarrow 1$

maxIgényekSzáma $\leftarrow 1$

Minden $i = 2, n$ **végezd el:**

Ha $a_i > b[M_{\text{maxIgényekSzáma}}]$ **akkor**

maxIgényekSzáma \leftarrow maxIgényekSzáma + 1

$M_{\text{maxIgényekSzáma}} \leftarrow i$

vége(ha)

vége(minden)

Vége(algoritmus)

4. k-dik legkisebb

Adott egy n elemű tömb, mely valós számokat tartalmaz és egy $1 \leq k \leq n$ természetes szám. Határozzuk meg a tömb k . legkisebb elemét, anélkül, hogy rendezzük azt!

Megoldás

- Jusson eszünkbe a gyorsrendezés sajátos tulajdonsága: a tömb felosztásának eredményeképpen megszületik két részsorozat: az egyik a strázsaelemnél csak kisebb elemeket, a másik csak nagyobbakat tartalmaz.
- Ezt kihasználhatjuk ahhoz, hogy megoldjuk a feladatot: amikor a strázsaelem indexe = k , megtaláltuk a k . legkisebb elemet!

Algoritmus gyorsRendezés(bal, jobb, a, k):

Ha $bal < jobb$ akkor

$m = \text{strázsa}(bal, jobb, a)$

Ha $k < m$ akkor

térítsd gyorsRendezés(bal, m, a, k)

különben

Ha $k > m$ akkor

térítsd gyorsRendezés($m + 1, jobb, a, k$)

különben

térítsd m // most $m = k$

vége(ha)

vége(ha)

vége(ha)

Vége(algoritmus)