Pálcikák

Legyen n *pálcika (*$5 \le n \le 1000$*), amelyek nem feltétlenül különböző hosszúak (*$1 \le hossz_i \le 100$, $i = 1, 2, ..., n$*). A pálcikákat úgy szeretnénk két csoportba osztani, hogy a két csoportból kialakítható pálcikasorok hosszúságai legyenek minél közelebbiek. Egy pálcikasor hossza egyenlő az őt alkotó pálcikák hosszúságainak összegével.*

Határozzuk meg a két pálcikasor hosszát úgy, hogy a két hosszúság közti különbség legyen minimális.

Példa:

Ha $n = 7$, $hossz = (28, 7, 11, 8, 9, 7, 27)$

az első sor hossza **48 (= 28 + 11 + 9)**,

a másodiké **49 (= 7 + 8 + 7 + 27)**.

Megoldás

- Tehát adott egy ***n*** elemű, nem feltétlenül egész számokat tároló sorozat, amelyet kétfele kell osztanunk úgy, hogy a két sorozat összegének különbsége legyen minimális.
- Kiszámítjuk a sorozat elemeinek összegét, majd ennek az összegnek a felét, mivel a kiszámítandó két összeget ennek a számnak a közelében kell keresnünk.
- A feladat hasonlít az előzőhöz, de itt nem ismerjük az ***S*** értékét.
- Felépítünk egy logikai tömböt, amely az eredeti tömb elemeinek megfelelően azt tárolja, hogy felhasználtuk-e már az illető számot, illetve, hogy felhasználtuk-e már valamely részösszeget, amely ezekkel a számokkal kialakítható. Így ennek a tömbnek a hossza az összes szám összegének hosszával lehetne egyenlő.

Megoldás

- Mivel erre nincs szükség, csak azokat fogjuk tárolni, amelyek nem haladják meg a kiszámított összeg felét.
- Miután ezt megtaláltuk, a másik szám a teljes összeg és ezen „közelítő” fél közötti különbség lesz.

Adósság

- *Egy nehéz anyagi helyzetbe került vállalat azt tervezi, hogy* ***n*** *lépésben talpra áll. Rendre, minden i. lépésben fölvesz egy* ***kölcsön***$_i$ *értékű bankkölcsönt. De ezeket a kölcsönöket vissza kell fizetnie. Az első kölcsön visszafizetése után a vállalat vezetői rájöttek, hogy nem fogják tudni visszafizetni az összes kölcsönt, csak azokat, amelyeket nem egymás után vettek fel.*
- *Határozzuk meg a visszafizethető maximális összeget.*

Példa: *Ha* **n = 6** *és* **kölcsön = (1, 3, 6, 2, 4, 3)** *a visszafizethető maximális összeg:* **11** *vagyis az* **1.**, **3.** *és* **5.** *kölcsönt fizetik vissza.*

Megoldás

- Az ***n*** elemű ***vissza*** tömb ***i.*** elemének értéke az a maximális összeg, amit a feltételek tiszteletben tartása mellett az ***i.*** lépésben vissza tudnak fizetni:

$$vissza_i = kölcsön_i + max\{vissza_j,\ j > i + 1\}$$

- A fenti képletben ***j*** azért „áll meg” ***i + 1***-nél, mivel az eredménybe nem kerülhet be két egymás utáni elem.
- Ahhoz, hogy kiírhassuk a visszafizetett összegeket lépésenként, fölhasználjuk a az ***n*** elemű ***melyek*** tömböt, amelybe elmentjük azoknak az elemeknek az indexeit, amelyek a maximális visszafizethető összeghez vezetnek

Kukorica

*Adott **n** kukoricapalánta és a koordinátáik (a kukoricákat egy egyenes vonal mentén ültették). A gazdának ki kell gyomlálnia bizonyos palántákat abból a célból, hogy bármely két egymás után következő kukoricapalánta között a távolság ne legyen kisebb, mint **x**.*

*Határozzuk meg azt a legkisebb számot, amely azoknak a kukoricapalántáknak a darabszáma, amelyeket a gazdának ki kell gyomlálnia, tudva azt, hogy az első kukoricapalántát nem szabad kigyomlálni. Írjuk ki **a lehetséges legtöbb megmaradt palánta számát, valamint ezeknek sorszámait**.*

Példa: Ha ***n* = 5**, ***x* = 3** és a koordináták: **(1, 3, 4, 6, 9)**, akkor az eredmény: **(1, 3, 5)**

Megoldás

A feladatnak optimális belső szerkezete van:

- Legyen a ***maradt*** segédsorozat hossza ***n***.
- A $\boldsymbol{maradt_i}$ elem értéke = az i. palántától a végéig a megőrzött palánták száma

Rekurzív összefüggések

- $\boldsymbol{maradt_n = 1}$,
- $\boldsymbol{maradt_i = 1 + max\{maradt_k \mid i < k \text{ és } táv(k, i) \geq x\}}$, ahol $\boldsymbol{táv(k, i)}$ az ***i***-edik és a ***k***-adik kukorica közötti távolság.
- A $\boldsymbol{maradt_i}$ értékek kiszámítása közben a ***k*** változik ***i+1*** és ***n*** között.
- Észrevétel: ezek a határok túl nagyok; elégséges, ha a ***k*** nem haladja meg a ***maradt*** sorozatban annak az elemnek az indexét, amely a $\boldsymbol{maradt_{i+1}}$ elem maximumát adta.

Rekurzív összefüggések

- Az ***előző*** sorozatban megőrizzük a $\boldsymbol{maradt_i}$ elemek maximumát eredményező elemek indexeit.
- Az eredmény összerakását és kiírását rekurzívan oldjuk meg.

Tulajdonság

A $\boldsymbol{maradt_i}$ számok ($\boldsymbol{i = 1, \ldots, n}$) csökkenő sorozatot alkotnak.

Bizonyítás

- A tulajdonságot a lehetetlenre való visszavezetéssel bizonyítjuk be.
- Feltételezzük, hogy létezik két elem, amelyre

$$maradt_i < maradt_j \text{ ha } j > i.$$

Bizonyítás

- Ha $a_j - a_i \geq x$, akkor ez a feltételezés hamis a $maradt_i$ elemek kiszámítási módja miatt (az a sorozat a koordináták sorozata).
- Ha $a_j - a_i < x$, akkor legyen k a $maradt_j$ maximumát generáló elem indexe, vagyis $maradt_j = 1 + maradt_k$.
- Mivel $j > i$, kapjuk, hogy $k > i$ és $a_k - a_i \geq x$.
- Tehát $maradt_i$ egyenlő lesz legalább $1 + maradt_k = maradt_j$-vel
- $\Rightarrow$ $maradt_i \geq maradt_j$.
- Ellentmondáshoz jutottunk, és így a tulajdonság be van bizonyítva.

Következmény

- Mivel az első kukoricát nem gyomláljuk ki, az eredményt a ***$maradt_1$***-ben őrizhetjük meg.
- Ha kigyomlálnánk az első kukoricát, nem biztos, hogy optimális eredményt kapunk:

Példa: Ha ***$n = 3$***, ***$x = 4$*** és koordináták = **(2, 4, 6)**

- Ha kigyomláljuk az első kukoricát (koordinátája **2**), akkor az optimális eredményben csak a **4**-es (vagy **6**-os) marad, míg ha a másodikat gyomláljuk ki (**4**-es koordináta) akkor az eredmény optimális és az **1**-es és **3**-as kukoricákból áll.

Algoritmus Kukorica(n, x, a, előző):
  Minden $k = 1, n$ **végezd el:**
    $előző_k \leftarrow 0$
  vége(minden)
  $maradt_n \leftarrow 1$ { $imax$ = *az utolsó elem indexe, amely a* }
  $imax \leftarrow n$ { *$maradt_i$ maximumához vezet* }
  Minden $i = n - 1, 1$ **végezd el:**
    $maradt_i \leftarrow 1$
    Amíg $a_{imax-1} - a_i \geq x$ **végezd el:**
      $imax \leftarrow imax - 1$
    vége(amíg)
    Ha $a_{imax} - a_i \geq x$ **akkor**
      $maradt_i \leftarrow maradt_{imax} + 1$
      $előző_i \leftarrow imax$
    vége(ha)
  vége(minden)
  térítsd $maradt_1$
Vége(algoritmus)

Eredmény visszafejtése

Algoritmus Visszafejt(előző, p):
  Ki: p
  Ha $\text{előző}_p \neq 0$ **akkor**
    $\text{Visszafejt}(\text{előző}_p)$
  vége(ha)
Vége(algoritmus)