

# Elemi algoritmusok

Dr. Ionescu Klára

# 1. Számrendszer

- ◆ Adott egy  $n$  természetes szám ( $1 < n < 2^{31}$ ). Mennyi annak a helyiértékes számrendszernek az alapja, amelyben az  $n$  felírása a lehető legtöbb nulla számjeggyel végződik? A választott számrendszer alap 1-nél nagyobb természetes szám kell legyen. Ha a számvégi nullák száma több számrendszer alap esetén is maximális, akkor ezen számrendszer alapok közül a legkisebbet kell megadni.

## Példák

- ◆  $n = 2$  esetén  $2 = (10)_2$  és bármely  $k > 2$  alapú számrendszerben a  $2$  ábrázolása önmaga – azaz  $(2)_k$  – hiszen  $2 < k = (10)_k$ , a helyes válasz tehát  $2$ ;
- ◆  $n = 4$  esetén a helyes válasz szintén  $2$ ,  $4 = (100)_2 = (11)_3 = (10)_4 = (4)_k$ ,  $(\forall k > 4)$ ;
- ◆  $n = 225$  esetén a helyes válasz  $3$  és itt  $225 = (22100)_3$ .

# Észrevételek és kérdések

- ◆ Ha a  $k$  számrendszer alap  $n$ -nél nagyobb, akkor nincs nulla a felírás végén, hiszen ennek egyetlen számjegye maga az  $n$ .
- ◆ Ha a  $k$  számrendszer alap pont  $n$ , akkor a szám alakja  $(10)_k$ , tehát mindig van olyan felírás, ami egy nullában végződik.
- ◆ Ha  $n$ -t felírjuk a  $k$  alapú számrendszerben, akkor az utolsó számjegy értéke  $n \bmod k$ . Mikor lesz ez nulla?
  - ◆ (válasz: Ha  $n$  osztható  $k$ -val.)
- ◆ Adott  $k$  számrendszer alap esetén, mikor végződik a felírás két nullában?
  - ◆ (válasz: Ha  $n$  osztható  $k^2$ -tel.)
- ◆ Általában hány nullával végződik az  $n$  szám  $k$  alapú számrendszerbe írása?
  - ◆ (válasz: Annyival, amennyi a legnagyobb olyan  $p$  egész hatványkitevő, amire  $n$  osztható  $k^p$ -nel – legrosszabb esetben  $p = 0$ .)

# Számrendszer

## Megoldás

- ◆ Az eredmény  $n$  azon valódi osztója, amelyik a legnagyobb hatványra emelhető úgy, hogy még mindig ossza  $n$ -t – ha pedig több ilyen van, akkor ezek közül a legkisebb.
- ◆ Ez az osztó prímszám?
  - ◆ (válasz: Igen, mert ha összetett lenne, akkor lenne nála kisebb osztó – ennek egyik valódi osztója – mely legalább ugyanarra a hatványra emelhető.)
- ◆ Megkeressük az  $n$  prímtényezős felbontásában azt a legkisebb prímet, amelyhez tartozó hatványkitevő maximális.

**Algorithm megold(n):**

válasz  $\leftarrow$  2; legtöbbNulla  $\leftarrow$  0

**While**  $n \bmod 2 = 0$  **execute**

$n \leftarrow n \text{ DIV } 2$ ; legtöbbNulla  $\leftarrow$  legtöbbNulla + 1

**EndWhile**

**For** osztó  $\leftarrow$  3; osztó \* osztó  $\leq n$ ; osztó  $\leftarrow$  osztó + 2 **execute**

hatvány  $\leftarrow$  0

**While**  $n \bmod \text{osztó} = 0$  **execute**

hatvány  $\leftarrow$  hatvány + 1;  $n \leftarrow n \text{ DIV osztó}$

**EndWhile**

**If** hatvány > legtöbbNulla **then**

legtöbbNulla  $\leftarrow$  hatvány

válasz  $\leftarrow$  osztó

**EndIf**

**EndFor**

**If**  $n \neq 1$  és legtöbbNulla = 0 **then** válasz  $\leftarrow$  n **EndIf**

**return** válasz

**EndAlgorithm**

# Tornyok

- ◆ Legyen megfelelő darabszámú azonos méretű érme, amelyekből tornyok építendők a következő szabályok alapján:



1. a legmagasabb torony magassága  $n$  ( $0 < n \leq 13$ ), a legkisebbnek a magassága 1;
2. a tornyok úgy kerülnek egymás mellé, hogy bármely két, azonos magasságú torony között létezik legalább egy magasabb torony, mint ez a kettő.

Számítsuk ki azt a *legnagyobb toronyszámot* (**tornyokSzáma**), amelyek felépíthetők az adott szabályok alapján és az építkezéshez szükséges *érmék számát* (**érmékSzáma**).

**Példa:** ha  $n = 3$ , **tornyokSzáma** = 7 és **érmékSzáma** = 11.

# Megoldás

- ◆ Rajzolgatunk és rájövünk, hogy ha lerajzoltuk azokat a tornyokat, amelyek megfelelnek egy bizonyos  $n$  értéknek, majd növeljük az  $n$  értékét, hogy lerajzoljuk a beszúrandó tornyokat, a tornyok száma  $2^{n-1}$ -gyel nő.
- ◆ Tehát, ahhoz, hogy megadhassuk a tornyok számát, generáljuk a kettőhatványokat ( $2^{n-1}$ -ig), majd összeadjuk őket. Vigyáznunk kell a hatékonyságra, tehát egy-egy kettőhatvány kiszámítását nem kezdjük mindig előlről, hanem felhasználjuk az előző lépésben kiszámítottak az értékét.
- ◆ A **tornyokSzáma** és **érmékSzáma** értékeket egyetlen ismétlődő struktúrával számítjuk ki:

# Tornyok

**Algorithm** tornyok( $n$ , tornyokSzáma, érmékSzáma):

érmékSzáma  $\leftarrow 0$

tornyokSzáma  $\leftarrow 0$

hatvány  $\leftarrow 1$

**For** magasság =  $n$ , 1, -1 **execute**

tornyokSzáma  $\leftarrow$  tornyokSzáma + hatvány

érmékSzáma  $\leftarrow$  érmékSzáma + hatvány \* magasság

*// minden toronyban „magasság” darab érme van*

hatvány  $\leftarrow$  hatvány \* 2

**EndFor**

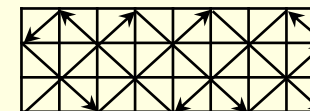
**EndAlgorithm**



# Fénysugár

Legyen egy tükrökből kialakított, téglalap alakú keret.

Egy fénysugár elindul a téglalap bal alsó sarkából,  $45^\circ$  fokos szöget alkotva a téglalap alsó oldalával, és nekiütközik a téglalap felső vagy jobboldali oldalának.



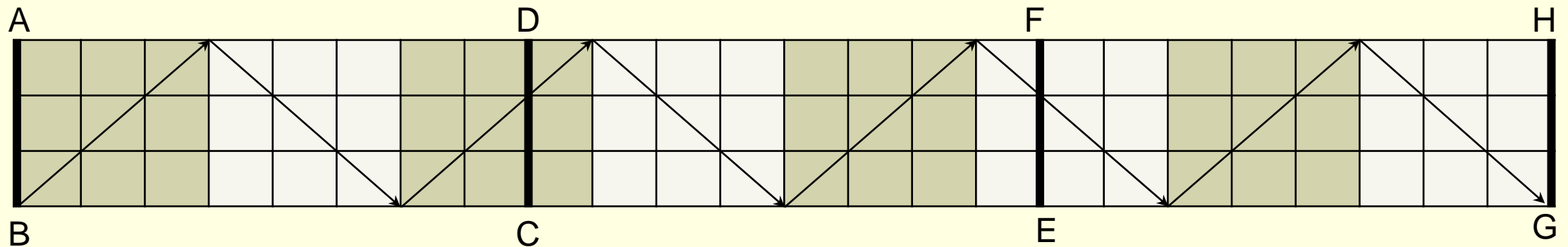
Itt tükröződik (elindul egy másik oldal felé, szintén  $45^\circ$  fokos szöget alkotva azzal az oldallal, amelybe beleütközött). Így folytatja az útját, amíg a keret valamelyik sarkába nem ér.

Számítsuk ki, hogy hányszor változtatja a tükröződés irányát a fénysugár, amíg leáll valamelyik sarokban. A kiindulási pontot nem számítjuk be ebbe a számba. A téglalap hossza és szélessége  $1 < a, b < 10\,000$ .

**Példák** ha  $a = 8$  és  $b = 3$ , akkor  $váltSz = 9$ , ha  $a = 8$  és  $b = 4$ , akkor  $váltSz = 1$ .

# Megoldás

- ◆ Legyen egy ABCD téglalap, amelyet  $1 \times 1$  méretű négyzetecskékre osztottunk.
- ◆ A téglalap hossza  $a = 8$  (négyzetecske), a szélessége  $b = 3$  (négyzetecske).
- ◆ Ha tükrözzük a téglalapot a CD oldal mentén, megkapjuk a CDFE téglalapot.
- ◆ Hasonlóan járunk el a CDFE téglalappal (most az EF oldal mentén tükrözzünk).



Ha a fénysugarat az ABGH téglalapon belül mozgatjuk,  $k - 1$  alkalommal változtatja az irányát, ahol  $k$  azoknak a négyzeteknek a száma, amelyeknek oldalhosszúsága  $b = 3$ , tehát a BG oldal hossza egyenlő  $l_{kkt}(a, b)$ .

# Megoldás

- ◆ Kiszámítjuk a példánk esetében:  $k = \frac{lkkt(a,b)}{b} = \frac{lkkt(8,3)}{3} = \frac{24}{3} = 8$
- ◆ Láttuk az ábrán, hogy az ABGH téglalapon belül a fénysugár 7-szer változtat irányt.
- ◆ Visszatérünk az eredeti ABCD téglalaphoz. Amikor a fénysugár nekiütközik a CD oldalnak, majd az AB oldalnak,  $p - 1$ -szer változtatja meg az irányát, ahol  $p$  a tükrözött (a példában 8 oldalhosszúságú) téglalapok száma.
- ◆  $p = \frac{lkkt(a,b)}{a} = \frac{24}{8} = 3$ . Az ábrán látjuk, hogy a fénysugár kétszer ütközik függőlegesen.

# Megoldás

- ◆ Tehát, az irányváltások összes száma:

$$\mathbf{váltSzám} = (k - 1) + (p - 1) = \frac{lkkt(a,b)}{b} - 1 + \frac{lkkt(a,b)}{a} - 1.$$

- ◆ Tudjuk, hogy  $lkkt(a, b) = \frac{a*b}{lnko(a,b)}$ . Elvégezzük a behelyettesítéseket:

$$\mathbf{váltSzám} = \frac{a*b}{lnko(a,b)*b} - 1 + \frac{a*b}{lnko(a,b)*a} - 1 = \frac{a}{lnko(a,b)} - 1 + \frac{b}{lnko(a,b)} - 1.$$

- ◆ A példánk esetében:  $lnko(a, b) = lnko(8, 3) = 1$ , tehát behelyettesítve:

$$\mathbf{váltSzám} = 8 / 1 - 1 + 3 / 1 - 1 = 9.$$

- ◆ Következik, hogy a fénysugár  $a$  és  $b$  értékeinek függvényében változtatja az irányt, de tulajdonképpen az  $lnko(a, b)$  függvényében.

**Algorithm** fény sugar(a, b):

$d \leftarrow \text{Inko}(a, b)$

**If**  $d = a$  **then**

$\text{váltSz} \leftarrow b \text{ DIV } d - 1$

**else**

**If**  $d = b$  **then**

$\text{váltSz} \leftarrow a \text{ DIV } d - 1$

**else**

$\text{váltSz} \leftarrow b \text{ DIV } d + a \text{ DIV } d - 2$

**EndIf**

**EndIf**

**return**  $\text{váltSz}$

**EndAlgorithm**

# Banánok

---

- ◆ Egy hajótörés után a tengerészeknek sikerült a mentőcsónakokkal megmenekülni.
- ◆ Minden csónakban volt három tengerész és minden csónak más-más szigeten vetődött partra.
- ◆ Élelem után kellett nézniük és minden  $i$ . szigeten a tengerészek összegyűjtöttek  $b_i$  banánt, amelyeket elraktározták a csónakjukba, mivel úgy döntöttek, hogy csak másnap osztják szét egymás között.
- ◆ Bármely csónakban legtöbb  $k$  banán fér el.

# Banánok

---

- ◆ Az éjszaka folyamán, az egyik szigeten, az egyik tengerész felkelt és szétosztotta a csónakban levő banánokat három részre, minden halomba ugyanannyi banánt tett, de maradt még egy banán, amit nem tehetett egyik halomba sem, így ezt megette.
- ◆ Ezután, az egyik részt elrejtette, a másik két részt visszatette a csónakba és lefeküdt aludni.
- ◆ Reggelig, minden tengerész, mindegyik szigeten elvégezte ugyanezt a titkos beavatkozást (elosztotta a banánokat három egyenlő részre és megette az egyetlen megmaradt banánt).
- ◆ Reggel, mindegyik szigeten, a megmaradt banánokat három egyenlő (nem üres) részre osztották és megint maradt egy banán, amit a tengerészek egy majomnak adtak. Igen, mindegyik szigeten élt egy-egy majom!☺

# Banánok – követelmények

- ◆ Ha az egyik szigeten, az éjszaka beállta előtt, a tengerészek összegyűjtöttek **241** banánt, hány banán maradt egy-egy halomban az osztozkodás végén (ezen a szigeten)?
- ◆ Ha az egyik szigeten, az osztozkodás végén minden halom **15** banánt tartalmazott, és egy másik szigeten minden halom **31** banánt, hány banánt gyűjtöttek összesen a két szigeten?
- ◆ Számítsátok ki adott  **$k$**  számra azt a lehetséges legnagyobb  **$b_{max}$**  értéket, amely azoknak a banánoknak a száma, amelyeket a tengerészek egy bizonyos szigeten összegyűjtöttek. Keressetek képletet, amely kiszámítja  **$b_{max}$**  legnagyobb értékét  **$k$**  függvényében. ( **$b_{max}$** ,  **$k$**  – természetes számok,  **$1 \leq b_{max} \leq k$** ,  **$100 \leq k \leq 10\,000\,000$** ).

**Példa:** ha  **$k = 200$** , akkor  **$b_{max} = 160$** .



# Banánok

- ◆ Számítsátok ki (adott  $k$  számra) az összes szigeten összegyűjtött maximális banánszámok összegét (**összegMax**).
- ◆ Minden szigeten a tengerészek különböző  $b_i$  számú banánt gyűjtöttek ( $b_i$  – természetes szám,  $i = 1, 2, \dots, sz$ ,  $1 \leq b_i \leq k$ ,  $100 \leq k \leq 10\,000\,000$ ).
- ◆  $sz$  – a szigetek száma, természetes szám,  $2 \leq sz \leq 10$ .
- ◆ **összegMax** – az összes szigeten összegyűjtött maximális banánszámok összege.
- ◆ A feladatnak mindig lesz megoldása

## Példa:

- ◆ Ha  $k = 400$ ,  $sz = 3$ , akkor a három szigeten rendre **322**, **241** és **160** banánt gyűjtöttek.
- ◆ Az összes szigeten összegyűjtött maximális banánszámok összege **összegMax = 322 + 241 + 160 = 723**.

# Megoldás

## Jelölések:

- $n$  az összegyűjtött banánok száma.
- $n = 3p + 1$  (eredetileg összegyűjtött banánok,  $p$  ismeretlen)
- $2p = 3t + 1$  (az első éjszaka után maradt banánok,  $t$  ismeretlen)
- $2t = 3q + 1$  (a második éjszaka után,  $q$  ismeretlen)
- $2q = 3r + 1$  (a harmadik éjszaka után,  $r$  ismeretlen)
- reggel  $3 * r + 1$  banánt osztanak szét, (egy végleges halomban  $r$  banán van)
- ◆ Behelyettesítések után:  $r = (8 * n - 65) / 81 (*)$  (az osztzkodás reggelén található banánok száma.
- ◆ Következik, hogy **81**-esével haladunk ahhoz, hogy végeredményhez jussunk.

# Megoldás

- ◆ Egész számok halmazán dolgozunk, tehát  $r$  páratlan szám és  $r$  legkisebb lehetséges értéke  $7$ ,  $n$  legkisebb lehetséges értéke  $79$ , mert:
  - ha  $r = 1$ ,  $2 * q = 4$ ,  $2 * t = 7$  – lehetetlen,
  - ha  $r = 3$ ,  $2 * q = 10$ ,  $2 * t = 16$ ,  $2 * p = 25$  – lehetetlen
  - ha  $r = 5$ ,  $2 * q = 16$ ,  $2 * t = 25$  – lehetetlen
  - ha  $r = 7$ ,  $2 * q = 22$ ,  $2 * t = 34$ ,  $2 * p = 52$ ,  $n = 79$
- ◆ Erre az eredményre jutunk a következőképpen is:
- ◆ Az előbbi  $(*)$  reláció alapján  $\Rightarrow n = (81 * r + 65) / 8$   $(**)$ .
- ◆ Felírható így is:  $n = (80 * r + 64) / 8 + (r + 1) / 8$ .
- ◆ Következik, hogy  $r + 1$  a  $8$  többszöröse, így  $r$  legkisebb lehetséges értéke  $7$ .

# Megoldás

◆ Válaszok a kérdésekre:

1. (\*)  $\Rightarrow r = (8 * n - 65) / 81$ . Behelyettesítve  $n$ -et:  $(241 * 8 - 65) / 81 = 23$ .

2. (\*\*)  $\Rightarrow n = (81 * r + 65) / 8$ . Behelyettesítve a két szigeten összegyűjtött banánok számát:  $(81 * 15 + 65) / 8 + (81 * 31 + 65) / 8 = 482$  banán.

Algorithm a képletekre támaszkodva:

◆ megkeressük azt a 79-nél nagyobb számot,

◆ 81-esével haladva, amelyre:  $n\_max = 79 + d * 81$ ,

◆ ahol  $n\_max$ -ot helyettesítjük  $k$ -val (lehetséges legtöbb banán), és meghatározzuk  $d$ -t:  $d = (k - 79) / 81$ .

◆ Innen következik, hogy:  $bmax = 79 + [(k - 79) / 81] * 81$ .

# Megoldás

---

**Algorithm banánok\_1(k):**

**return 79 + (k - 79) DIV 81 \* 81**

**EndAlgorithm**

Az algoritmus dolgozhat egy „okos” ismétlő struktúrával (felhasználva a (\*) relációt):

**Algorithm banánok\_2(k):**

**While (8 \* k - 65) MOD 81 ≠ 0 execute**

**k = k - 1**

**EndWhile**

**return k**

**EndAlgorithm**

# Megoldás

- ◆ Fel kell dolgoznunk minden sziget esetében az eseményeket.
- ◆ Az algoritmusnak tartalmaznia kell egy ismétlődő struktúrát a szigetek feldolgozására és ki kell számolnia a különböző szigetekeken összegyűjtött banánok számát, amit összegeznie kell.
- ◆ Vigyáznunk kell arra is, hogy az egyes szigetekeken begyűjtött banánok összege különböző legyen:

```
Algorithm banánokSzigetek_1(k, sz):  
    összegMax = 0  
    For i = 1, sz execute  
        ni = banánok_2(k)  
        összegMax = összegMax + ni  
        k = ni - 1  
    EndFor  
    return összegMax  
Vége(Algorithm)
```

## Ha a két követelményt összefűzzük:

**Algorithm** banánok\_4(k, b, nb):

n = 1

nb = 0

**While** n <= k **execute**

**If** (8 \* n - 65) MOD 81 = 0 **then**

        nb = nb + 1

        b[nb] = n

**EndIf**

n = n + 1

**EndWhile**

**return** b[nb]

**EndAlgorithm**

**Algorithm** mindenSziget\_2(k, sz):

nb = 0

banánok\_4(k, b, nb)

s = 0

**For** i = nb, i ≥ 1 **AND** sz > 0, -1) **execute**

    s = s + b[i]

    sz = sz - 1

**EndFor**

**return** s

**EndAlgorithm**

# Kalitkák

- ◆ Az állatkertben a papagájok 1-től  $n$ -ig számozott kalitkákbán élnek ( $1 \leq n \leq 10\,000$ ).
- ◆ Egy adott pillanatban egy játékos majom kinyit minden kalitkát. Megijed a következményektől, visszatér az első kalitkához és bezár minden második kalitkát (így bezárja a  $2, 4, 6, \dots$  sorszámúakat).
- ◆ A majomnak megtetszik ez a játék. Ezért újra elindul az elejéről és meglátogat minden harmadik kalitkát (vagyis a  $3, 6, 9, \dots$  sorszámúakat) és bezárja a kalitkát, ha az nyitva van, illetve kinyitja, ha azt zárva találja. A negyedik bejáráskor meglátogat minden negyedik kalitkát, és hasonlóan jár el (megváltoztatva a meglátogatott kalitka állapotát).
- ◆ A majom megismétli a játékot, míg az utolsó bejáráskor (az  $n$ . bejárás) bezárja az  $n$ . kalitkát, ha ez nyitva van vagy kinyitja, ha zárva van



# Megoldás

- ◆ A majom egy bejárás alkalmával, akkor és csakis akkor látogat meg egy kalitkát, *ha a kalitka sorszáma a bejárás sorszámanak többszöröse.*
- ◆ Következik, hogy *egy adott kalitka meglátogatásának darabszáma egyenlő a kalitka sorszáma különböző osztóinak darabszámával.*
- ◆ A kalitka akkor és csakis akkor marad nyitva, *ha a sorszámanak páratlan darabszámú osztója van.*
- ◆ A feladat megoldása visszavezethető azoknak a kalitkáknak a megszámlálására, amelyeknek *sorszáma négyzetszám.*
- ◆ Egy  $n$  természetes szám négyzetszám, ha  $\sqrt{n} * \sqrt{n} = n$ .
- ◆ *Az  $n$ -nél kisebb (vagy egyenlő) négyzetszámok darabszáma egyenlő  $\lfloor \sqrt{n} \rfloor$ -nel.*
- ◆ Ez az érték meghatározható a *bináris keresés* módszerével, amikor keressük a  $\lfloor \sqrt{n} \rfloor$ -et az  $1, 2, \dots, n/2$  rendezett számsorozatban.

**Algorithm kalitkák(n):**

**bal** = 1

**jobb** = **n DIV 2**

**While** **bal** < **jobb** **execute**

**k** = (**bal** + **jobb**) **DIV 2**

**If** **k \* k** >= **n** **then**

**jobb** = **k**

**else**

**bal** = **k + 1**

**EndIf**

**EndWhile**

**If** **bal \* bal** <= **n** **then**

**return bal**

**else**

**return bal - 1**

**EndIf**

**EndAlgorithm**

**Algorithm kalitkák(n):**

**nyitvaSz** = 0

**For** **i** = 1, (**i \* i** <= **n**) **execute**

**nyitvaSz** = **nyitvaSz + 1**

**EndFor**

**return nyitvaSz**

**EndAlgorithm**

# Maximális szorzat (felvételi, 2016)

- ◆ Adott az  $n$  elemű ( $3 \leq n \leq 10\,000$ ), egész számokat tároló  $x$  sorozat, ahol az elemek értéke nagyobb, mint  $-30\,000$  és kisebb, mint  $30\,000$ .
- ◆ Írjatok alprogramot, amely meghatároz *három* számot az  $x$  sorozatból, amelyeknek a szorzata *maximális*.
- ◆ Az alprogram bemeneti paraméterei  $n$  és  $x$ , kimeneti paraméterei az  $a$ ,  $b$  és  $c$  számok, amelyek az  $x$  sorozat elemei és rendelkeznek a kért tulajdonsággal. Ha a feladatnak több megoldása van, csak egyet kell megadnotok.

**Példa:** ha  $n = 10$  és  $a = (3, -5, 0, 5, 2, -1, 0, 1, 6, 8)$ , a három szám:

$a = 5$ ,  $b = 6$ ,  $c = 8$ .

# Elemzés

- ◆ Több megközelítés is lehetséges (különböző bonyolultságú algoritmussal)
- 1. **Naiv algoritmus:** generálunk minden lehetséges indexhármast, kiszámítjuk a megfelelő elemek szorzatát és megőrizzük a szorzatok közül a legnagyobbat
  - ◆ Ha az értékek a megengedett felső határ közelében találhatók, a három szám szorzata túlcsordulhat
  - ◆ Bonyolultság:  $O(n^3)$
- 2. **Rendezéssel:** a sorozat rendezhető **lineáris rendezéssel**, majd feldolgozzuk a három legnagyobb és két legkisebb értéket
  - ◆ Nehézséget okoz a tény, hogy a számok értéke nagyobb, mint -30 000 és kisebb, mint 30 000
  - ◆ Bonyolultság:  $O(n)$
- 3. **Rendezéssel:** a sorozat rendezhető **buborékrendezéssel**
  - ◆ Bonyolultság:  $O(n^2)$

# Elemzés

## 4. *Lineáris algoritmussal* (Ez a jó megoldás!!!):

- ◆ Meghatározzuk a tömb maximumát (**max1**)
- ◆ Meghatározzuk a tömb második maximumát (**max2**)
- ◆ Meghatározzuk a tömb harmadik maximumát (**max3**)
- ◆ Meghatározzuk a tömb minimumát (**min1**)
- ◆ Meghatározzuk a tömb második minimumát (**min2**)
- ◆ Feldolgozzuk ezt az öt értéket
- ◆ Bonyolultság:  $5 * O(n) + O(1) = O(n)$

# Az öt érték feldolgozása

...

**If** max1 > 0 és min1 \* min2 > max2 \* max3 **then**

    a ← max1

    b ← min1

    c ← min2

**else**

    a ← max1

    b ← max2

    c ← max3

**EndIf**

...

# Azonos számjegyek

Adott két természetes szám:  $a$  és  $b$ ,  $1 \leq a \leq 1\,000\,000$  és  $1 \leq b \leq 1\,000\,000$ .

Határozzuk meg azt a  $k$  elemű  $x$  sorozatot, ( $k$  – természetes szám,  $0 \leq k \leq 1000$ ), amely minden olyan természetes számot tárol, amelyek az  $[a, b]$  intervallumhoz tartoznak és azonos számjegyekből állnak. Ha ilyen szám nem létezik,  $k$  értéke  $0$  lesz.

## 1. Példa:

Ha  $a = 8$  és  $b = 120$ , akkor  $k = 12$  és  $x = (8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 111)$ .

## 2. Példa:

Ha  $a = 590$  és  $b = 623$ , akkor  $k = 0$  és az  $x$  sorozat üres.

# Megoldás

---

A hatékony algoritmus nem dolgozik fölöslegesen, hanem számolásokkal generálja a kért sorozat elemeit.

A számok generálása az egyszámjegyű számokkal kezdődik és/vagy a 11, 111, 1111 stb. számok többszöröseivel folytatódik)



**Algorithm** azonosSzamjegyek(a, b, k, x):

    k = 0; i = a

**While** i < 10 és i <= b **execute**

        k = k + 1; x[k] = i; i = i + 1

**EndWhile**

    akt = 11; leptek = 11

**Amíg** akt <= b **execute**

**If** akt >= a **then**

            k = k + 1; x[k] = akt

**EndIf**

        akt = akt + leptek

**If** akt > 9 \* leptek **then**

            akt = leptek \* 10 + 1; leptek = akt

**EndIf**

**EndWhile**

**EndAlgorithm**