Felvételi előkészítő
(Bináris keresés és alkalmazásai; Algoritmusok bonyolultsága)

1. Keresztmetszet

Adott két sorozat, amelyeknek elemei *különböző* természetes számok: az ***a*** sorozat elemeinek száma ***n*** ($0 < n \leq 10\ 000$), a ***b*** sorozat elemeinek száma ***m*** ($0 < m \leq 10\ 000$) és *növekvően rendezett*. Határozzuk meg azt a ***c*** sorozatot, amelynek ***k*** ($0 < k \leq 10\ 000$) eleme lesz, és amely a két sorozat minden *közös* elemét egyszer tartalmazza.

Példa: ha $n = 4$, $a = (5, -7,-2, 3)$, $m = 5$ és $b = (-2, 3, 5, 7, 8)$, a ***c*** sorozatnak $k = 3$ eleme van: $c = (5, -2, 3)$.

2. S összegű elemek kiválasztása

Legyen egy ***n*** elemű ($3 \leq n \leq 100\ 000$), különböző természetes számokat tartalmazó sorozat és az ***S*** természetes szám. Válasszunk ki az adott sorozatból *három elemet, amelyeknek az összege* ***S***! Adjunk meg minden megoldást!

3. Négyzetszámok darabszáma

Számoljuk meg egy ***n*** ($1 \leq n \leq 1\,000\,000$) elemű sorozat négyzetszámait! A számok kisebbek, mint 1 000 000.

4. Síkmértan

Két függőleges fal egymástól ***t*** távolságra található. Egy h_1 hosszúságú deszkát az egyik fal alapjától a másik falnak támasztunk. Egy h_2 hosszúságú deszkát a másik fal alapjától az első falnak támasztunk. A két deszka ***m*** magasságban érinti egymást egy pontban, amely valahol a két fal között található. Számítsuk ki ***t***-t h_1, h_2 és ***m*** ismeretében (megengedett hibalehetőség 10^{-5}).

5. Ládák

Költözik a múzeum. A tárgyakat kocka alakú, különböző méretű ládákba csomagolták. Kicsomagoláskor több személy dolgozik egyidőben, és a rendetlenség elkerülése végett, azokba a helyiségekbe, ahol kicsomagolás folyik, felszereltek egy futószalagot, amelyre az üres ládákat helyezik, a nyitott fedelükkel fölfele. A futószalag végéhez egy robotot állítottak, amelynek az a feladata, hogy összeszedje a ládákat és úgy helyezze egyiket a másikba (ha lehet) hogy végül *a ládacsomagok száma a lehető legkisebb legyen*. A robotot egy program irányítja úgy, hogy:

- A ládákat az érkezésük sorrendjében szedi le a futószalagról.
- Az aktuális ládát csak egy nála nagyobb méretű ládába helyezi.
- Ha nincs olyan megkezdett csomag, amelybe elhelyezhető az aktuális láda, akkor ez a láda egy új csomag első ládája lesz.
- Egy megkezdett csomagba csak egyetlen ládát helyez, vagyis nem helyez két ládát egymás mellé, még akkor sem, ha ez egyébként lehetséges volna.
- Egy elhelyezett ládát, többé nem mozgat.
- Egy megkezdett csomagot nem helyez egy másik csomagba még akkor sem, ha ez egyébként lehetséges volna.
- Egyetlen ládát sem hagy figyelmen kívül.

6. Kalitkák

Az állatkertben a papagájok 1-től ***n***-ig számozott kalitkákban 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, ... sorszámúakat). A majomnak megtetszik ez a játék. Ezért újra elindul az elejéről és meglátogat minden harmadik kalitkát (vagyis a 3, 6, 9, ... 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.

Követelmények:

B.1. Hány kalitka marad nyitva az utolsó bejárás után, ha ***n*** = 10? (2 pont)

B.2. Mely sorszámú kalitkák maradnak nyitva az utolsó bejárás után, ha ***n*** = 10? (2 pont)

B.3. Összesen hányszor látogatja meg a majom a ***k*** sorszámú kalitkát (1 ≤ ***k*** ≤ ***n***) az ***n*** bejárás során? Indokoljátok meg a választ. (4 pont)

B.4. Mi annak szükséges és elégséges feltétele, hogy a ***k*** sorszámú kalitka (1 ≤ ***k*** ≤ ***n***) nyitva maradjon az ***n*** kalitka utolsó bejárása után? Indokoljátok meg a választ. (4 pont)

B.5. Hány kalitka marad nyitva az ***n*** kalitka utolsó bejárása után? Indokoljátok meg a választ. (4 pont)

B.6. Írjatok algoritmust, amely kiszámítja az utolsó bejárás után *nyitva maradt kalitkák számát* (***nyitvaSz***). Az algoritmus bemeneti paramétere a kalitkák ***n*** (1 ≤ ***n*** ≤ 10 000) száma, kimeneti paramétere a ***nyitvaSz*** szám. (14 pont)

1. *Példa:* ha ***n*** = 5, akkor ***nyitvaSz*** = 2 (nyitva marad az 1-es és a 4-es sorszámú kalitka).

2. *Példa:* ha ***n*** = 12, akkor ***nyitvaSz*** = 3.

7. Egészítsétek ki

Adott az ***n*** elemű (3 ≤ ***n*** ≤ 100), növekvően rendezett ***x*** sorozat, amely 30 000-nél kisebb különböző természetes számokat tartalmaz. A `legközelebbi(x, bal, jobb, ér)` algoritmus meghatározza az ***x*** sorozat legnagyobb értékű elemének pozícióját, amely a ***bal*** és ***jobb*** pozíciók között helyezkedik el (1 ≤ ***bal*** < ***jobb*** ≤ ***n***) és, amelynek az értéke kisebb, vagy egyenlő ***ér***-rel. Ha nem létezik ilyen elem, a `legközelebbi(x, bal, jobb, ér)` algoritmus 0-t térít vissza.

A `modulusz(a)` algoritmus az ***a*** egész szám abszolút-értékét téríti vissza.

A `számol(n, x, adottSz)` algoritmus meghatározza azt az elemét az ***x*** sorozatnak, amely a legközelebb áll ***adottSz***-hoz. Ha két elem azonosan közel van ***adottSz*** értékéhez, az algoritmus a nagyobb számot határozza meg.

Legyen ***n*** = 5, ***x*** = (5, 9, 11, 15, 99) és ***adottSz*** = 12. Állapítsátok meg melyik kifejezéssel helyettesíthető a „…” a `legközelebbi(x, bal, jobb, ér)` algoritmusban, ahhoz, hogy a `számol(n, x, adottSz)` algoritmus 11-et térítsen vissza.

```
Algoritmus legközelebbi(x, bal, jobb, ér)
  Ha ér > x[jobb] akkor térítsd jobb
  vége(ha)
  Ha ér < x[bal] akkor térítsd bal - 1
  vége(ha)
  közép ← (bal + jobb) DIV 2
  Ha ... akkor
     térítsd közép - 1
  különben
     Ha ér < x[közép] akkor
        térítsd legközelebbi(x, bal, közép - 1, ér)
     különben
        térítsd legközelebbi(x, közép + 1, jobb, ér)
     vége(ha)
  vége(ha)
Vége(algoritmus)
```

```
Algoritmus számol(n, x, adottSz)
   i ← legközelebbi(x, 1, n, adottSz)
   Ha i = 0 akkor térítsd x[i + 1]
   különben
      Ha modulusz(x[i]- adottSz) < modulusz(x[i + 1] - adottSz)
        akkor
          térítsd x[i]
      különben térítsd x[i + 1]
      vége(ha)
   vége(ha)
Vége(algoritmus)
```

A. `x[közép - 1] <= ér` **és** `ér < x[közép]`

B. `x[közép - 1] <= ér` **vagy** `ér < x[közép]`

C. `x[közép - 1] < ér` **és** `ér <= x[közép]`

D. `x[közép] <= ér` **és** `ér < x[közép - 1]`