Drugi izpitni rok pri predmetu Programiranje 1
12. februar 2024

Dopolnite in oddajte datoteke `Prva.java`, `Druga.java`, `Tretja.java` in `Cetrta.java`. Testirate jih lahko takole:

(1) `tj.exe Prva.java . .` (2) `tj.exe` (3) `tj.exe` (4) `tj.exe`

Vsa števila, ki nastopajo v navodilih in testnih primerih tega izpita, so cela.

(1) V prvi vrstici vhoda sta podani praštevili $p \in [2, 10^3]$ in $q \in [p+1, 10^3]$ ter število $n \in [1, 10^3]$, nato pa sledi zaporedje n števil z intervala $[1, 10^5]$. Napišite program, ki za vsako število iz zaporedja izpiše `DA`, če se ga dá zapisati kot $p^a q^b$ za nek par $a \geqslant 0$, $b \geqslant 0$, in `NE`, če to ni mogoče. Vsako besedo izpišite v svoji vrstici.

Primer 1 (vhod/izhod):

```
2 3 7
30
27
1
5
8
48
18
```

```
NE
DA
DA
NE
DA
DA
DA
```

(2) Geodeti bi radi izmerili nadmorske višine vseh točk na mreži velikosti $h \times w$ ($h, w \in [1, 50]$). Vedo, da je nadmorska višina točke (0, 0) enaka 0, za vsako od ostalih točk pa lahko izmerijo le razliko do ene od njenih štirih sosed. Dopolnite metodo

`public static int[][] visine(int[][] rel, int[][] smer)`,

ki vrne tabelo velikosti $h \times w$, ki podaja nadmorske višine vseh točk, če za vsak $i \in [0, h-1]$ in $j \in [0, w-1]$ (razen za $(i, j) = (0, 0)$) velja, da točka (i, j) leži za `rel[`i`][`j`]` enot višje od svoje leve sosede (če je `smer[`i`][`j`]` $= 1$), od svoje zgornje sosede (če je `smer[`i`][`j`]` $= 2$), od svoje desne sosede (če je `smer[`i`][`j`]` $= 3$) oziroma od svoje spodnje sosede (če je `smer[`i`][`j`]` $= 4$). Velja tudi `rel[0][0]` $=$ `smer[0][0]` $= 0$. Pri vseh testnih primerih je vse nadmorske višine mogoče enolično izračunati na podlagi že izračunanih nadmorskih višin (krožnih sklicev ni).

V 50% testnih primerov je `smer[`i`][`j`]` povsod bodisi 1 ali 2 (razen seveda za $(i, j) = (0, 0)$, kjer je 0).

Primer 1 (`rel`/`smer`/*rezultat*):

```
{{ 0,  5, -3, 4},
 {-2,  9, -4, 6},
 {-3, -1,  7, 3}}
```

```
{{0, 1, 1, 4},
 {4, 3, 2, 1},
 {3, 2, 3, 2}}
```

```
{{0, 5,  2, 8},
 {1, 7, -2, 4},
 {3, 6, 14, 7}}
```

Točka (0, 1) (vrstica 0, stolpec 1) leži 5 enot nad svojo levo sosedo (točko (0, 0)), torej na višini 5. Točka (0, 2) leži -3 enote nad svojo levo sosedo, torej na višini 2. Točka (1, 2) leži -4 enote nad svojo zgornjo sosedo (točko (0, 2)), torej na višini -2. Točka (1, 1) leži 9 enot nad svojo desno sosedo (točko (1, 2)), torej na višini 7. In tako naprej ...

(3) Na pravokotniku velikosti `stVrstic` × `stStolpcev` bomo zgradili mesto. Mesto bo imelo stanovanjske in poslovne stavbe. Vsaka stavba bo zavzemala parcelo velikosti 1×1. Za vsako stanovanjsko stavbo vemo, koliko stanovanj bo vsebovala.

V razredu `Mesto` dopolnite konstruktor

`public Mesto(int stVrstic, int stStolpcev)`,

ki inicializira objekt tako, da predstavlja prazno (nepozidano) mesto s podano velikostjo. Dopolnite tudi sledeče metode:

- `public void postavi(int vrstica, int stolpec, Stavba stavba)`

  Na parcelo na podanih koordinatah (indeks vrstice, indeks stolpca) v mestu `this` postavimo podano stavbo. Kazalec `stavba` kaže na objekt tipa `Stanovanjska` (stanovanjska stavba) ali `Poslovna` (poslovna stavba). Če na podani parceli že stoji kakšna stavba, jo porušimo in na njenem mestu zgradimo podano stavbo.

  V 50% testnih primerov ni nobenih rušitev (na vsaki parceli zidamo kvečjemu enkrat).

- [32%] `public int steviloStanovanj()`

  Vrne skupno število stanovanj v mestu `this`.

- [34%] `public int koliko(Stavba stavba)`

  Vrne število stavb v mestu `this`, ki pripadajo istemu tipu kot objekt, na katerega kaže kazalec `stavba`.

- [34%] `public boolean poslovnaCetrt(int vrZac, int stZac, int vrKon, int stKon)`

  Vrne `true` natanko v primeru, če poslovna stavba stoji na vsaki od (`vrKon` − `vrZac` + 1)(`stKon` − `stZac` + 1) parcel v pravokotniku z zgornjim levim ogliščem na koordinatah (`vrZac`, `stZac`) in spodnjim desnim ogliščem na koordinatah (`vrKon`, `stKon`).

(4) Zaporedje $\langle a_0, a_1, a_2, \ldots \rangle$ naj bo *alternirajoče monotono*, če je $a_0 < a_2 < a_4 < \ldots$ in hkrati $a_1 > a_3 > a_5 > \ldots$ (Zaporedje dolžine 2 ali manj je vedno alternirajoče monotono.) Dopolnite sledeči metodi:

- [50%] `public static <T> boolean alternirajoceMonotono(List<T> zaporedje, Comparator<T> prim)`

  Vrne `true` natanko v primeru, če je podano zaporedje glede na podani primerjalnik alternirajoče monotono.

- [50%] `public static Supplier<Integer> generator(int a, int b)`

  Vrne generator neskončnega alternirajoče monotonega zaporedja $\langle a,\ a-b,\ a+b,\ a-2b,\ a+2b,\ a-3b,\ a+3b,\ \ldots \rangle$. V vseh testnih primerih je $b > 0$.

  Vmesnik `Supplier<T>` vsebuje abstraktno metodo `T get()`.