Prvi izpitni rok pri predmetu Programiranje 1
27. januar 2020

Oddajte datoteke `Prva.java`, `Druga.java`, `Tretja.java` in `Cetrta.java`. Svoje rešitve lahko testirate s programom `tj.exe` in množico javnih testnih vhodov in pripadajočih izhodov (naloga 1) oziroma testnih razredov in pripadajočih izhodov (naloge 2–4).

(1) Škatle, označene z zaporednimi številkami, po vrsti zložimo v kupe. Prvi kup vsebuje prvih a škatel, vsak naslednji pa b škatel več kot prejšnji kup. Na primer, če je $a = 3$ in $b = 2$, potem prvi kup vsebuje tri škatle (njihove številke so 1–3), drugi pet škatel (številke 4–8), tretji sedem škatel (9–15), četrti devet škatel (16–24) itd.

Program `Prva` dopolnite tako, da bo prebral cela števila $a \in [1,\ 10^3]$, $b \in [0,\ 10^3]$ in $k \in [1,\ 10^6]$ (ločena so s presledkom) in izpisal zaporedno številko kupa, v katerem se nahaja škatla s številko k.

Javni testni primer 6 (vhod/izhod):

```
3 2 24
```

```
4
```

V prvih 50% testnih primerov velja $b = 0$.

(2) Na tekaškem tekmovanju je m tekačev odteklo po n krogov. Tabela `t` velikosti $m \times n$ vsebuje podatke za posamezne kroge: element v i-ti vrstici in j-tem stolpcu pove, koliko časa je i-ti tekač porabil od začetka do konca svojega j-tega kroga (neodvisno od prejšnjih krogov).

Razred `Druga` dopolnite s sledečima metodama:

- `public static int najCas(int[][] t, int krog)` [50%]:

  Vrne najboljši rezultat (torej najmanjši čas) v krogu z indeksom `krog` (prvi krog ima indeks 0, drugi 1 itd.).

- `public static int[][] kumulativa(int[][] t)` [50%]:

  Vrne novo tabelo velikosti $m \times n$, v kateri element v i-ti vrstici in j-tem stolpcu podaja skupni čas, ki ga je i-ti tekač porabil od začetka teka do konca svojega j-tega kroga. (Zadnji stolpec vrnjene tabele bo potemtakem podajal končne rezultate tekmovanja.)

(3) Zaposleni v podjetju se delijo na *delavce* in *vodje*. Vsak zaposleni ima plačo in kvečjemu enega neposredno nadrejenega. Nadrejeni je lahko samo vodja.

```
public class Tretja {
    public static class Zaposleni {
        private int placa;
        private Vodja nadrejeni;
    }
    public static class Delavec extends Zaposleni { }
    public static class Vodja extends Zaposleni { }
}
```

Napišite metode, navedene v nadaljevanju. Po potrebi lahko v razrede dodate še več metod.

- `public int placaNadrejenega()` v razredu `Zaposleni` [34%]:

  Vrne plačo vodje, ki je neposredno nadrejen zaposlenemu `this`, oziroma -1, če zaposleni `this` nima nadrejenega.

- `public Vodja vrhovni()` v razredu `Vodja` [32%]:

  Vrne tistega neposredno ali posredno nadrejenega vodji `this`, ki nad seboj nima nikogar. (Če vodja `this` nima nadrejenega, naj metoda vrne kar `this`.)

- `public static int steviloAnomalij(Zaposleni[] zaposleni)` v razredu `Zaposleni` [34%]:

  Vrne število vseh parov delavec-vodja iz podane tabele, pri katerih ima delavec večjo plačo od vodje. (Ni nujno, da je vodja nadrejen delavcu; pregledati morate vse pare (D, V), kjer je D nek delavec, V pa nek vodja iz podane tabele.)

(4) Razred `Cetrta` vsebuje dva statična notranja razreda. Objekt razreda `Celica` hrani koordinati neke celice v neki dvodimenzionalni tabeli, objekt razreda `Ovojnik` pa hrani kvadratno tabelo tipa `boolean[][]` z liho dolžino stranice.

```
public static class Celica {
    private int vrstica;    // vrstična koordinata (indeks vrstice)
    private int stolpec;    // stolpčna koordinata (indeks stolpca)
}

public static class Ovojnik {
    private boolean[][] tabela;  // kvadratna tabela z liho dolžino stranice
}
```

Razreda dopolnite takole:

- [30%] Za objekte razreda `Celica` definirajte naravno urejenost, in sicer tako, da bodo celice v naravnem vrstnem redu urejene po naraščajočih vrstičnih koordinatah, tiste z enako vrstično koordinato pa po naraščajočih stolpčnih koordinatah.

- [40%] V razredu `Ovojnik` napišite metodo

  `public NavigableSet<Celica> enice()`,

  ki vrne množico koordinat tistih celic tabele `this.tabela`, ki vsebujejo vrednosti `true`. Množica naj bo urejena po naraščajoči manhattanski oddaljenosti celic od sredinske celice tabele, celice z enako manhattansko razdaljo do sredinske celice tabele pa naj bodo naravno urejene. Manhattanska razdalja med celicama (v_1, s_1) in (v_2, s_2) se izračuna kot $|v_1 - v_2| + |s_1 - s_2|$.

  Na primer, pri tabeli `{{F, F, T}, {T, T, F}, {T, T, F}}` (`T` = `true`, `F` = `false`) naj se vrnjena množica prične s celico (1, 1) (razdalja do sredinske celice znaša 0), nato pa naj sledita celici (1, 0) in (2, 1) (razdalja = 1) in zatem še celici (0, 2) in (2, 0) (razdalja = 2).

- [30%] Naredite vse, kar je treba, da bo koda

  ```
  Cetrta.Ovojnik ovojnik = new Cetrta.Ovojnik(...);
  for (Cetrta.Celica celica: ovojnik) {
      System.out.println(celica);
  }
  ```

  izpisala koordinate celic z vrednostmi `true` v naravnem vrstnem redu.