Računalništvo (Fizika): 3. izpit

31. avgust 2012

Čas reševanja: 120 minut. Doseženih 100 točk šteje za maksimalno oceno. Vsako nalogo rešujte v posebni .py datoteki. Na koncu po elektronski pošti pošljete .zip datoteko z vsemi .py datotekami. Zahtevane utemeljitve oz. odgovore napišete kot komentarje v .py datoteke.

Naloga 1 [25]

Sestavi razred `Matrika`, ki predstavlja celoštevilske matrike oblike $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ in ima spodaj naštete metode.

(a) Konstruktor `__init__(self, a, b, c, d)`.

(b) Metodo `__mul__(self, m)`, ki vrne produkt matrik `self` in `m`.

(c) Metodo `__repr__(self)`, ki vrne tekstovno predstavitev matrike v obliki, kot je prikazana na primeru:

```
|   1   66|
| 123   -3|
```

V tekstovni predstavitvi naj bodo stolpci desno poravnani, med stolpcema je en presledek prostora, vsako število pa ima za izpis prostor, ki je enak dolžini zapisa po absolutni vrednosti največjega števila v matriki povečani za ena (v konkretnem primeru je to $4 = 3 + 1$).

(d) Metodo `__pow__(self, n)`, ki za dano matriko `self` in pozitivno celo število `n` vrne `n`-to potenco matrike. Potenco računaj učinkovito s pomočjo formul $M^{2k} = (M^k)^2$ in $M^{2k+1} = M \cdot (M^k)^2$, ki za n-to potenco potrebuje kvečjemu $O(\log n)$ množenj.

Naloga 2 [25 točk]

Dvojiško drevo je podano z razredom `Drevo`, ki v vozliščih vsebuje cela števila. Dodatno vsako vozlišče opremimo še s številom elementov v drevesu k in vrednostjo maksimalnega elementa v drevesu max. Drevo je uravnoteženo, kar pomeni, da se števili elemenov v obeh poddrevesih ne razlikujeta za več kot 1 ter da sta levo in desno drevo uravnoteženi. Prazno drevo je vedno uravnoteženo.

(a) Sestavi metodo `dodaj(self, x)`, ki v obstoječe drevo doda nov element z vrednostjo x kot list, pri čemer poskrbi za pravilne popravke atributov k in max ter da je drevo uravnoteženo. Kakšna je časovna zahtevnost metode v odvisnosti od števila elementov v drevesu n?

(b) Napiši metodo `manjKot(self, m)`, ki čim bolj učinkovito izračuna, koliko elementov v drevesu je strogo manjših od m.

(obrni list)

Naloga 3 [25 + 10 točk]

Dano je urejeno zaporedje celih števil $a = [a_0, a_1, \ldots, a_{n-1}]$. Element zaporedja je *večinski*, če se v zaporedju pojavi več kot $n/2$-krat. Primer: večinski element zaporedja $[1, 2, 5, 5, 5, 5, 5, 5, 10, 30]$ je 5.

(a) Sestavi funkcijo `vecinskiUrejeno(a)`, ki vrne večinski element zaporedja ali `None`, če ta ne obstaja. Pri tem uporabi kar se da majhno število primerjav elementov zaporedja in utemelji, zakaj je to število primerjav minimalno.

(b) Predpostavimo, da zaporedje ni nujno urejeno. Kakšno je najboljša časovna zahtevnost algoritma, ki najprej zaporedje uredi potem pa uporabi rešitev iz naloge (a)? Utemelji!

(c) **(Dodatna +10 točk)** S pomočjo strategije deli in vladaj lahko sestavimo algoritem, ki brez urejanja tabele poišče večinski element. Zaporedje najprej razdelimo na približno dva enaka dela. Nato rekurzivno poiščemo večinska elementa v vsakem izmed delov. Ugotovi, kako lahko informacijo o obstoju in vrednosti večinskega elementa v obeh polovicah zaporedja uporabimo, da sestavimo algoritem, katerega časovna zahtevnost ne presega $O(n \log n)$. Algoritem zapiši v funkciji `vecinski(a)`, ki vrne večinski element ali `None`, če ta ne obstaja. Utemelji časovno zahtevnost!

Naloga 4 [25 točk]

Dano je zaporedje celih števil $[a_0, a_1, \ldots, a_{n-1}]$. Zanima nas dolžina najdaljšega (strogo) naraščajočega podzaporedja (NNP). Na primer, eden od NNP-jev zaporedja $[3, 7, 3, 11, 15, 11, 12, 6]$ je $[3, 7, 11, 12]$. Naj bo $d(i)$ dolžina najdaljšega naraščajočega podzaporedja, ki se začne z a_i. Potem velja naslednja rekurzivna formula:

$$d(i) = 1 + \max\{d(j) | j > i \text{ in } a_j > a_i\}.$$

Pri tem upoštevamo, da je maksimum prazne množice enak 0.

(a) Sestavi funkcijo `dolzine(a)`, ki vrne tabelo

$$[d(0), d(1), \ldots, d(n-1)].$$

Funkcija naj bo učinkovita tudi za večje tabele. Oceni časovno zahtevnost svoje funkcije (odgovor zapiši kot komentar v kodi).

(b) Sestavi funkcijo `nnp(a)`, ki vrne enega od NNP-jev za zaporedje a. Najprej izračunaj tabelo $J = [j_0, j_1, \ldots, j_{n-1}]$, kjer j_k predstavlja enega izmed tistih j v formuli za $d(i)$ pri katerem dosežemo maksimum oz. je j_k enak `None`, če tak j ne obstaja. Vemo, da se NNP pri definiciji $d(i)$ začne z a_i in ima dolžino $1 + d(j_i)$.