Računalništvo (fizika): 3. izpit

28. avgust 2019

Naloge rešujte na strežniku Tomo, `https://www.projekt-tomo.si`.

Brskanje po spletni učilnici, nalogah na strežniku Tomo, video zapiskih predavanj in *uradni* dokumentaciji za Python *je dovoljeno*. Brskanje po spletnih straneh, ki vsebujejo rešitve nalog in nasvete o tem, kako se rešuje naloge, *ni dovoljeno*. Komunikacija s komerkoli, razen z asistentom, *ni dovoljena*.

Čas reševanja je 180 minut. Doseženih 100 točk šteje za maksimalno oceno. Veliko uspeha!

1. naloga (25 točk)

V matriki

$$A = \begin{bmatrix} a_{0,0} & \dots & a_{0,m} \\ a_{1,0} & \dots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,0} & \dots & a_{n,m} \end{bmatrix}$$

je element $a_{i,j}$ *lokalni minimum*, če so vsi njegovi štirje sosedi $a_{i-1,j}$, $a_{i+1,j}$, $a_{i,j-1}$, $a_{i,j+1}$ strogo večji od $a_{i,j}$. Če je $a_{i,j}$ na robu matrike, upoštevamo samo tiste sosede, ki jih ima.

Sestavite funkcijo `prestej`, ki sprejme matriko in vrne *število* lokalnih minimumov v dani matriki. Primer:

```
>>> prestej([
...     [-10, -14,   4,   6],
...     [ -7,   0, -13, -13],
...     [  6,   9, -11,  13]
... ])
1
```

2. naloga (30 točk)

Dana je tabela števil $[a_0, a_1, \ldots, a_n]$ ter preslikavi ℓ in d, ki slikata indekse v indekse:

- $\ell : i \mapsto 2 \cdot i + 1$,
- $d : i \mapsto 2 \cdot i + 2$.

Začenši pri indeksu 0, lahko do vsakega elementa dostopamo enolično z uporabo preslikav ℓ in d. Na primer, a_{18} je element z indeksom $d(d(\ell(\ell(0))))$. Če pogledamo z drugega zornega kota: od elementa a_i vodita dve povezavi, ena do $a_{\ell(i)}$ in druga do $a_{d(i)}$, Če je $\ell(i)$ ali $d(i)$ večji od n, potem seveda taka povezava ne obstaja.

a) (5 točk) Katero znano podatkovno strukturo smo predstavili s tabelo in preslikavama ℓ in d? Pojasnite, kakšno vlogo igrajo elementi tabele in kakšno preslikavi ℓ in d?

b) (10 točk) *Pot* v tabeli je zaporedje indeksov, ki se začne z 0, vsak naslednji indeks pa dobimo iz prejšnjega z uporabo ene od preslikav ℓ in d. Pot končamo, ko ne moremo več nadaljevati, ker obe funkciji ℓ in d prekoračita dolžino tabele.

Sestavite funkcijo `najvecja_vsota`, ki sprejme tabelo a in poišče tako pot $i_0, i_1, \ldots, i_k$, da je vsota

$$a_{i_0} + a_{i_1} + \cdots + a_{i_k}$$

čim večja. Funkcija naj vrne največjo možno vsoto. Primer:

```
>>> najvecja_vsota([25, -49, 74, 70])
99
```

c) (15 točk) Sedaj bomo posplošili korake in namesto funkcij ℓ in d dopustili poljubne dane funkcije $k_0, \ldots, k_m$. Le-te slikajo indeks v strogo večji indeks. Sestavite funkcijo `najvecja_vsota2`, ki poleg tabele a prejme še seznam funkcij $k_0, \ldots, k_m$, ki predstavljajo korake. Primer:

```
>>> najvecja_vsota2(
...     [-5, 0, 2, -12, -4, -2, -5, 4, 5, 14],
...     [lambda i: 2 * i + 1, lambda i: 3 * i + 1, lambda i: 4 * i + 1]
... )
5
```

3. naloga (25 točk)

Andrej ima vsak dan veliko obveznosti, ki jih zapisuje v slovar. Ključ je opis obveznosti, vrednost pa pove, kdaj se obveznost začne in konča. Čas je določen do minute natančno in je predstavljen z urejenim parom (h, m), kjer so h ure in m minute. Na primer:

```
{ 'Predavanja - Racunalnistvo (FIZ)': ((8,15), (10,0)),
  'Predavanja - Logika in mnozice': ((10,15), (12,0)),
  'Ustni izpiti': ((12,0), (16,0)),
  'Malica': ((13,00), (13,30)) }
```

Opazimo, da se obveznosti lahko tudi prekrivajo.

Sestavite generator `obveznosti`, ki prejme slovar obveznosti in generira urejene pare, ki povedo, kaj Andrej počne vsako minuto. Na primer, za zgornji slovar bi dobili pare

```
((8,15), {'Predavanja - Racunalnistvo (FIZ)'})
((8,16), {'Predavanja - Racunalnistvo (FIZ)'})
...
((9,59), {'Predavanja - Racunalnistvo (FIZ)'})
((10,0), {'Predavanja - Racunalnistvo (FIZ)'})
((10,01), {})
...
((15,59), {'Ustni izpiti'})
((16,0), {'Ustni izpiti'})
```

Prva komponenta para je čas, druga komponenta pa množica vseh dogodkov, ki potekajo ob tem času. Množica dogodkov je lahko tudi prazna.

4. naloga (30 točk)

B-drevo je podatkovna struktura, ki posplošuje običajna iskalna drevesa. Definirana je takole:

1. *list* je B-drevo, ki vsebuje podatek p.
2. *vozlišče* je B-drevo, ki sestoji iz seznama

$$B_0, p_0, B_1, p_1, B_2, \ldots, B_{n-2}, p_{n-1}, B_n$$

v katerem so prepletena B-poddrevesa B_i in podatki p_j. Seznam mora vsebovati vsaj en podatek, mora pa še veljati, da so vsi podatki v poddrevesu B_i po velikosti med p_{i-1} in p_i. Za B_0 in B_n seveda ta predpis razumemo smiselno, torej sta omejeni samo z ene strani.

List je torej vozlišče, ki sestoji iz seznama `[p]`. Primer B-drevesa vam bo asistent narisal na tablo.

a) (10 točk) Definiraje razred `BDrevo`, s katerim predstavimo B-drevo. Implementirajte ustrezni konstruktor in metodo `__repr__`.

b) (10 točk) V B-drevesu lahko učinkovito poiščemo podatek, podobno kot v običajnem iskalnem drevesu. Sestavite metodo `vsebuje(self,p)`, ki vrne `True`, če drevo `self` vsebuje podatek `p`, sicer naj vrne `False`.

c) (10 točk) Sestavite metodo `elementi(self)`, ki vrne seznam podatkov, ki so shranjeni v drevesu `self`, v naraščajočem vrstnem redu.