Računalništvo (fiziki): 4. izpit

21. 9. 2015

Naloge rešujte na strežniku Tomo (`https://tomo.fmf.uni-lj.si/`). V sistem se prijavite s svojim uporabniškim računom na omrežju FMF.
Čas reševanja je 150 minut. Doseženih 100 točk šteje za maksimalno oceno. Veliko uspeha!

1. naloga (Zavetišče, 25 točk)

Skrbnik zavetišča za živali potrebuje informacijski sistem, ki mu bo v pomoč pri vodenju evidence najdenih živali in iskanju primernih posvojiteljev. Sestavili boste dve funkciji, ki bosta sestavni del tega kompleksnega sistema.

a) (10 točk) V zavetišču so psi in mačke, bolj eksotičnih živalskih vrst pa nimajo. Predstavimo jih lahko s seznamom parov, npr.

```
zivali = [
    ('Šarki', 'p'), ('Tom', 'm'), ('Suki', 'm'), ('Taček', 'p'), ('Nyan', 'm'),
    ('Rex', 'p'), ('Čupko', 'p'), ('Snoopy', 'p'), ('Tiger', 'm')
]
```

Prva komponenta je ime živali, druga pa je niz, ki pove za katero vrsto gre (`'p'`, če gre za psa, in `'m'`, če gre za mačko). Sestavite funkcijo `macke_psi`(l), ki kot argumenta dobi seznam l, kot je opisan zgoraj. Funkcija naj vrne dva seznama. V prvem naj bodo imena psov, v drugem pa imena mačk. Vrstni red imen naj bo enak, kot v seznamu l. Zgled:

```
>>> macke_psi(zivali)
(['Šarki', 'Taček', 'Rex', 'Čupko', 'Snoopy'],
 ['Tom', 'Suki', 'Nyan', 'Tiger'])
```

b) (15 točk) Sestavite funkcijo `dodeli`(l,v), ki kot argument dobi seznam živali l, kot je opisan zgoraj, in seznam nizov v, ki predstavljajo vloge posvojiteljev. Niz `'p'` pomeni, da posvojitelj želi psa in `'m'` da želi mačko. Funkcija naj vrne seznam, ki bo enake dolžine kot v. Na i-tem mestu naj bo ime živali, ki jo bo dobil i-ti posvojitelj oz. `None`, če želje ni mogoče izpolniti. Razdelitev naj bo takšna, da prej pridejo na vrsto živali, ki so na začetku seznama l. Zgled:

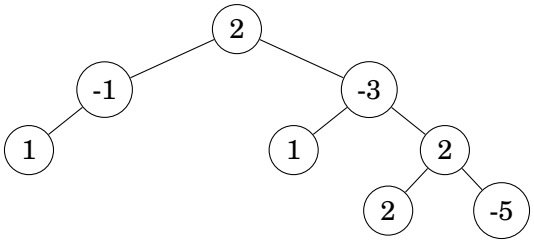
```
>>> dodeli(zivali, ['m', 'm', 'p', 'p', 'p', 'm', 'm', 'm', 'p'])
['Tom', 'Suki', 'Šarki', 'Taček', 'Rex', 'Nyan', 'Tiger', None, 'Čupko']
```

2. naloga (Dvojiško drevo, 30 točk)

Podatkovna struktura `Drevo` predstavlja dvojiško drevo, ki ima v vsakem vozlišču shranjeno eno celo število. Naslednji izraz v Pythonu ustvari drevo, ki je prikazano na spodnji sliki:

```
>>> s = Drevo(2, levo=Drevo(-1, levo=Drevo(1)), desno=Drevo(-3, levo=Drevo(1),
        desno=Drevo(2, levo=Drevo(2), desno=Drevo(-5))))
```

Pripadajoča slika:



Delno implementiran razred `Drevo` najdete na strežniku. Vsako vozlišče ima atribut `prazno`. Če je njegova vrednost `True`, predstavlja prazno poddrevo in nima drugih atributov. Če pa drevo ni prazno, ima še atribute `vsebina`, `levo` in `desno`. Dodali bomo še nekaj novih metod.

a) (10 točk) Razredu `Drevo` dodajte metodo `max_vsota(self)`, ki vrne največjo možno vsoto, ki jo lahko dobimo kot seštevek vrednosti na poti od korena do nekega lista v drevesu. Za prazno drevo naj metoda vrne 0. Zgledi:

```
>>> s.max_vsota()
3
```

b) (10 točk) Razredu `Drevo` dodajte metodo `max_pot(self)`, ki vrne seznam sestavljen iz znakov `'l'` in `'d'` (npr. `['l', 'd', 'l', 'l', 'd']`), ki pove, kako v drevesu iz korena pridemo do tistega lista, ki maksimizira vsoto vrednosti na tej poti. Znak `'l'` pomeni, da se moramo spustiti v levo poddrevo, znak `'d'` pa, da moramo v desno poddrevo. Primer:

```
>>> s.max_pot()
['d', 'd', 'l']
```

Če metodo pokličemo na praznem drevesu, naj vrne `None`. Če je možnih več rešitev, naj metoda vrne katerokoli.

c) (10 točk) Razredu `Drevo` dodajte metodo `najdi_pot(self, s)`, ki v drevesu poišče takšno pot od korena do nekega vozlišča (ki ni nujno list), da bo vsota vrednosti na tej poti enaka s. Primer:

```
>>> s.najdi_pot(0)
['d', 'l']
>>> d = Drevo(2, levo=Drevo(-1), desno=Drevo(-1))
>>> d.najdi_pot(2)
[]
>>> print(d.najdi_pot(0))
None
```

Če metodo pokličemo na praznem drevesu ali če iskana pot ne obstaja, naj vrne `None`. Če obstaja več rešitev, naj metoda vrne katerokoli.

3. naloga (Frnikole, 25 točk)

Janezek se igra s frnikolami različnih barv. Barve označimo s števili $1, 2, \ldots, k$. Janezek frnikole zloži v vrsto, ki jo predstavimo s seznamom barv. Na primer

$$[5,2,3,1,1,2,4,3,4,5]$$

predstavlja vrstico desetih frnikol, kjer je prva barve 5, naslednja barve 2 itd.

Janezek je vraževeren in verjame, da je razpored *srečen*, če za vsako barvo $1 \leq i < k$ velja: za zadnjo frnikolo barve i se v preostanku vrste pojavi vsaj ena frnikola barve $i+1$. Zgornji razpored je srečen, razpored

$$[5,2,3,1,1,4,2,4,3,5]$$

pa ne, saj za zadnjo frnikolo barve 3 ni nobene frnikole barve 4.

a) (10 točk) Sestavite funkcijo `sreca`(k,l), ki kot argument dobi število barv k in seznam l, ki predstavlja razpored frnikol. Funkcija naj vrne `True`, če je seznam srečen in `False` sicer. Predpostavite lahko, da l vsebuje vse barve. Zgled:

```
>>> sreca(5, [5, 2, 3, 1, 1, 2, 4, 3, 4, 5])
True
>>> sreca(5, [5, 2, 3, 1, 1, 4, 2, 4, 3, 5])
False
```

Opomba: Vse točke dobite samo za učinkovito rešitev, ki deluje s časovno zahtevnostjo $O(n)$, kjer je n dolžina seznama l.

b) (15 točk) Janezek je preštel svoje frnikole in ugotovil, da ima $c_i > 0$ frnikol barve i, za vsak $1 \leq i \leq k$. Zanima ga, koliko različnih srečnih razporedov lahko sestavi, tako da bo porabil vse frnikole. Sestavite funkcijo `st_srecnih`(c), ki kot argument dobi $c = [c_1, \ldots, c_k]$ in vrne število vseh srečnih razporedov. Funkcija naj ima časovno zahtevnost največ $O(k \cdot \max_i c_i)$.

```
>>> st_srecnih([2, 2])
3
>>> st_srecnih([2, 2, 2, 2, 2])
945
```

Komentar: V primeru, ko je $c = [2,2]$, obstajajo trije srečni razporedi in sicer $[1,1,2,2]$, $[1,2,1,2]$ in $[2,1,1,2]$.

4. naloga (Mafija, 30 točk)

Mafija je priljubljena družabna igra za več igralcev. Natanko en od udeležencev mora skrbeti za pravilen potek igre in ne sodeluje v sami igri – je torej *nadzornik*. Vse ostalim bomo rekli običajni igralci. (Ostala pravila igre za namen te naloge niso pomembna.)

a) (10 točk) Na zabavi se je zbralo n ljudi, ki se bodo igrali. Dan je seznam $p = [p_0, p_1, \ldots, p_{n-1}]$, kjer je p_i število iger, pri katerih bo i-ta oseba sodelovala kot *nadzornik*, vsi ostali pa bodo običajni igralci. Napišite funkcijo `stevilo_iger(`p`)`, ki kot argument dobi zgoraj opisani seznam in sestavi ter vrne seznam dolžine n, kjer je i-ti element število iger, v katerih bo i-ta oseba sodelovala kot običajni igralec. Zgled:

```
>>> stevilo_iger([2, 3, 1, 0, 5])
[9, 8, 10, 11, 6]
```

Opomba: Za vse točke mora vaša rešitev delovati učinkovito s časovno zahtevnostjo $O(n)$.

b) (10 točk) Vloga nadzornika ni preveč zabavna, zato hoče i-ta oseba *vsaj* a_i iger odigrati kot običajni igralec. Naj bo $a = [a_0, a_1, \ldots, a_{n-1}]$. Sestavite funkcijo `mozno_igrati(`$a, t$`)`, ki vrne `True`, če je možno odigrati t iger tako, da bo i-ta oseba sodelovala kot običajni igralec vsaj a_i-krat. V nasprotnem primeru naj funkcija vrne `False`. Zgled:

```
>>> mozno_igrati([9, 8, 10, 11, 6], 13)
True
>>> mozno_igrati([9, 8, 10, 11, 6], 10)
False
```

Časovna zahtevnost funkcije naj bo $O(n)$.

c) (10 točk) Sestavite funkcijo `min_iger(`a`)`, ki kot argument dobi seznam a, kot je opisan v prejšnji podnalogi. Funkcija naj vrne najmanjše število iger, ki jih morajo odigrati, da bo i-ta oseba vsaj a_i-krat sodelovala kot običajni igralec. Predpostavimo, da sodelujeta vsaj dve osebi. Zgled:

```
>>> min_iger([9, 8, 10, 11, 6])
11
```

Časovna zahtevnost funkcije naj bo $O(n \cdot \log(\max\{a_0, \ldots, a_{n-1}\}))$.

Namig: Ali lahko vedno odigrajo $2 \cdot \max\{a_0, \ldots, a_{n-1}\}$ iger tako, da bo i-ta oseba vsaj a_i-krat v vlogi običajnega igralca?

Zanimivost: Igro Mafija je leta 1986 izumil Dmitrij Davidov, ki je takrat študiral psihologijo na moskovski univerzi. Leta 1997 je Andrew Plotkin izdelal Volkodlake, kar je pravzaprav še ena različica igre Mafija.