Računalništvo (fiziki): 2. izpit

30. 6. 2015

Čas reševanja je 150 minut. Doseženih 100 točk šteje za maksimalno oceno. Veliko uspeha!

1. naloga (25 točk)

Matriko v Pythonu predstavimo kot seznam vrstic. Na primer, matriko

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

predstavimo s seznamom `[[1, 2, 3, 4], [5, 6, 7, 8]]`. Sestavite funkcijo `kaca`(n), ki kot argument dobi naravno število n ter sestavi in vrne *kvadratno* matriko, ki vsebuje števila od 1 do n zložena v "kačo". Število 1 naj bo v zgornjem levem kotu matrike, nato pa si sledijo kot je prikazano na primeru $n = 14$:

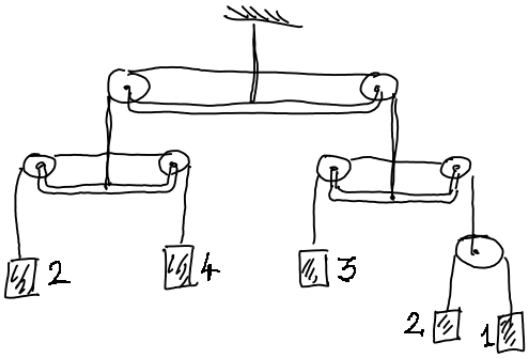
$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 6 & 5 \\ 9 & 10 & 11 & 12 \\ 0 & 0 & 14 & 13 \end{pmatrix}$$

Morebitna "prazna" polja nadomestimo z ničlami.

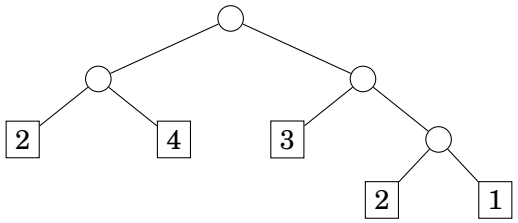
```
>>> kaca(14)
[[1, 2, 3, 4], [8, 7, 6, 5], [9, 10, 11, 12], [0, 0, 14, 13]]
```

2. naloga (25 točk)

Škripčevje, kot ga vidite na spodnji sliki, lahko na naraven način predstavimo z drevesi.



Pripadajoče drevo:



V Pythonu taka drevesa predstavimo z objekti. Škripce predstavimo z razredom `Skripec`, uteži pa z razredom `Utez` (glejte preambulo naloge). Zgornji izraz predstavimo z objektom:

```
>>> s = Skripec(levo=Skripec(levo=Utez(2), desno=Utez(4)),
                desno=Skripec(levo=Utez(3), desno=Skripec(levo=Utez(2),
                                                          desno=Utez(1))))
```

a) (10 točk) Obema razredoma dodajte metodo `miruje(self)`, ki vrne `True`, če bi sistem miroval, če bi ga obesili pod strop. Škripec miruje, če je vsota vseh mas na levi enaka vsoti vseh mas na desni. Cel sistem miruje, če miruje vsak škripec v sistemu. Težo vrvic in škripcev zanemarimo.

```
>>> s.miruje()
False
```

b) (15 točk) Denimo, da lahko maso vsake uteži povečamo za poljubno število. Razredoma dodajte metodo `min_do_mirovanja(self)`, ki izračuna najmanjšo skupno maso, za katero moramo povečati obstoječe uteži, da bo sistem obmiroval. Primer uporabe:

```
>>> s.min_do_mirovanja()
4
```

3. naloga (30 točk)

Mobitel ima tipkovnico, kot jo vidite na skici:

1	2	3
4	5	6
7	8	9
*	0	#

Andrej ima posebne vrste OCD in sicer lahko odtipka številko samo, če je vsaka naslednja števka v številki sosednja prejšnji glede na tipkovnico. To pomeni, da je na številčnici neposredno pod njo, nad njo, levo od nje ali desno od nje. Na primer, števki 8 so sosednje 5, 7, 9 in 0. Sosedi števke 7 sta 4 in 8.

Dvomestne številke, ki jih Andrej lahko vtipka so: 08, 12, 14, 21, 23, 25, 32, 36, 41, 45, 47, 52, 54, 56, 58, 63, 65, 69, 74, 78, 85, 87, 89, 80, 96 in 98.

a) (15 točk) Sestavite funkcijo `veljavna(`n`)`, ki kot argument dobi neprazen niz, ki vsebuje le števke od `'0'` do `'9'`, in vrne `True`, če Andrej to številko lahko vtipka, in `False` sicer. Zgled:

```
>>> veljavna('452363')
True

>>> veljavna('125780')
False
```

b) (15 točk) Sestavite funkcijo `st_veljavnih(`k`)`, ki kot argument dobi naravno število $k \geq 1$ in vrne skupno število vseh k-mestnih številk, ki jih Andrej lahko vnese. Številke imajo lahko vodilne ničle. Algoritem mora delovati učinkovito tudi za večje vrednosti k, na primer $k = 100$.

```
>>> st_veljavnih(2)
26

>>> st_veljavnih(3)
76

>> st_veljavnih(100)
252625044012857262477689787555791528111367416 40
```

4. naloga (45 točk)

Vse ovce na Novi Zelandiji so se postavile v vrsto, da bi jih ostrigli. Striglo jih je m strižcev, ki niso vsi enako izkušeni. Strižec i potrebuje za striženje ovce t_i minut. Ko je strižec prost, se k njemu zapodi naslednja ovca. Če je hkrati na voljo več strižcev, se ovca zapodi k tistemu z najmanjšim indeksom i.

Denimo, da je $m = 3$ in $t = (2,3,4)$. Na začetku se 1., 2. in 3. ovca po vrsti zapodijo k 1., 2. in 3. strižcu. Najprej konča 1. strižec in 2 minuti po začetku prične striči 4. ovco. Naslednji konča 2. strižec in sicer 3 minute po začetku in prične striči 5. ovco. Štiri minute po začetku hkrati končata 1. in 3. strižec. Šesta ovca izbere 1. strižca (ker ima manjši indeks), sedma ovca pa 3. strižca in tako dalje.

a) (15 točk) Napišite funkcijo `strizenje_ovac`(n,t), kjer je n število ovac in $t = [t_1,\ldots,t_m]$ seznam časov, ki jih potrebujejo strižci za striženje ene ovce. Funkcija naj vrne seznam n trojic, vsaka za eno ovco, pri čemer je k-ta trojica po vrsti:

- indeks strižca, pri katerem se je strigla k-ta ovca,
- čas pričetka striženja k-te ovce,
- čas konca striženja k-te ovce.

Primer:

```
>>> strizenje_ovac(11, [2, 3, 4])
[(1, 0, 2), (2, 0, 3), (3, 0, 4), (1, 2, 4), (2, 3, 6), (1, 4, 6), (3, 4, 8),
 (1, 6, 8), (2, 6, 9), (1, 8, 10), (3, 8, 12)]
```

Funkcija naj deluje v času $O(nm)$, kjer je m število strižcev.

b) (15 točk) Sestavite funkcijo `obdelane_in_obdelovane`(p,t), ki sprejme naravno število p in seznam t časov striženja. Funkcija naj vrne skupno število ovac, ki so do časa p minut že bile ostrižene ali pa so ob času p minut še v obdelavi. Če ovca ob času p minut ravno pride na vrsto, je v obdelavi.

```
>>> obdelane_in_obdelovane(6, [2, 3, 4])
9
```

Ob času 6 je 1. strižec ravno začel s svojo 4. ovci, drugi je ravno začel s svojo 3. ovco, tretji strižec pa je sredi striženja svoje 2. ovce. Skupno število obdelanih in obdelovanih je torej $4+3+2=9$.

c) (15 točk) Sestavite funkcijo `hitra_napoved`(n,t), ki dobi enake argumente kot `strizenje_ovac`. Funkcija naj vrne čas pričetka striženja n-te ovce.

```
>>> hitra_napoved(11, [2, 3, 4])
8
```

Funkcija naj ima časovno zahtevnost $O\left(m \log\left(\frac{n}{m} \max_i t_i\right)\right)$.

Nasvet 1: Kličite funkcijo iz točke b).

Nasvet 2: Ali znate navzgor oceniti čas, ko bo n-ta ovca prišla na vrsto?