Računalništvo (fizika): 1. izpit

31. 1. 2017

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 `stackoverflow.com`, `rosettacode.org` in ostalih 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)

Rekurzivno zaporedje $a_1, a_2, a_3, \ldots$ je podano s praviloma:

$$\begin{aligned} a_1 &= 1 \\ a_n &= 1 \cdot a_1 + 2 \cdot a_2 + 3 \cdot a_3 + \cdots + (n-1) \cdot a_{n-1} \qquad \text{če je } n > 1 \end{aligned}$$

Torej je $a_1 = 1$, $a_2 = 1$, $a_3 = 3$, $a_4 = 12$, $a_5 = 60$, $a_6 = 360$, ...

a) (15 točk) Sestavite funkcijo `a(n)`, ki izračuna `n`-ti člen zaporedja.

b) (10 točk) Za vse točke naj funkcija deluje učinkovito. Na primer, a_{666} mora izračunati v manj kot eni sekundi na vašem računalniku.

2. naloga (25 točk)

Za končni množici A in B definiramo *razdaljo* med njima kot velikost njune simetrične razlike, se pravi, število elementov množice $(A \cup B) \setminus (A \cap B)$.

Podan imamo slovar, ki živali preslika v množice lastnosti:

```
{ 'gad' : { 'strupen', 'vretenčar', 'plazilec', 'hladnokrven' },
  'volk' : { 'toplokrven', 'vretenčar', 'sesalec', 'zver' },
  'lisica' : { 'toplokrven', 'vretenčar', 'sesalec', 'zver' },
  'sršen' : { 'žuželka', 'strupen' },
  'lipan' : { 'vretenčar', 'sladkovoden', 'hladnokrven' }
}
```

Različnost med dvema živalima merimo z razdaljo med množicama njunih lastnosti. Na primer, različnost med volkom in lisico je 0, med lipanom in gadom pa je 3.

Sestavite funkcijo `najbolj_razlicna(d)`, ki sprejme tak slovar živali `d` in vrne par živali (x, y) z maksimalno različnostjo. Če je rešitev več, naj vrne katerokoli par z maksimalno različnostjo.

3. naloga (30 točk)

Dana je matrika 0 in 1 velikosti $m \times n$, na primer:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{matrix}$$

V matriki želimo poiskati velikost največje *ničelne kvadratne podmatrike*, se pravi take podmatrike, ki je kvadratna in ki ima same ničle. V zgornjem primeru je to podmatrika velikost 3×3 z zgornjim levim kotom v $(2,0)$, kar pomeni "vrstica 2, stolpec 0". Še ena taka matrika ima zgornji levi kot $(2,1)$.

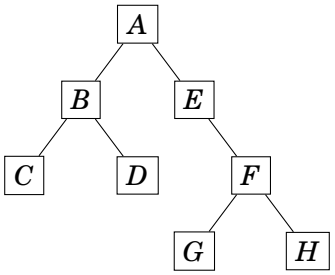
a) (20 točk) Sestavite funkcijo `podmatrika(a)`, ki sprejme matriko `a` in vrne velikost največje ničelne kvadratne podmatrike. Nalogo rešite z dinamičnim programiranjem, da bo delovala v času $O(m \times n)$.

Namig: najprej ugotovite, kako bi določili velikost največje ničelne kvadratne podmatrike z zgornjim levim kotom (i,j), če že veste, kako velike so take podmatrike z zgornjimi levimi koti $(i+1,j)$, $(i,j+1)$ in $(i+1,j+1)$.

b) (10 točk) Sestavite funkcijo `zgornji_levi(a)`, ki vrne položaj (i,j) zgornjega levega kota največje ničelne podmatrike, pri čemer je i vrstica in j stolpec. Seveda štejemo vrstice in stolpce od 0 dalje, kot v Pythonu. Če je več rešitev, naj funkcija vrne katerokoli. Če ni nobene ničelne podmatrike, naj funkcija vrne `None`.

4. naloga (30 točk)

Podatkovna struktura dvojiško drevo je definirana v razredu `Drevo`, ki ga najdete na strežniku. Vsako vozlišče označimo z *enolično* oznako, na primer:



a) (15 točk) *Pot* do vozlišča je zaporedje črk `'L'` in `'D'`, ki pove, kako pridemo od korena do vozlišča. Na primer, v zgornjem drevesu vodi do vozlišča G pot `['D','D','L']`. Razredu `Drevo` dodajte metodo `pot_do(self, x)`, ki vrne pot do vozlišča označenega z `x`. Če se x v drevesu ne pojavi, naj metoda vrne `None`.

b) (15 točk) *Razdalja* med vozliščema, ki sta označeni z x in y, je dolžina najkrajše poti, ki vodi od x do y. V zgornjem drevesu je razdalja med A in D enaka 2, med D in E je 3 in med G in C je 5. Razredu `Drevo` dodajte metodo `razdalja(self,x,y)`, ki vrne razdaljo med vozliščema z oznakama `x` in `y` v drevesu `self`. Če se katera od oznak ne pojavi, naj metoda vrne `None`. Metoda naj deluje v času $O(n)$, kjer je n število vozlišč v drevesu.

Navodilo: predpostaviti smete, da dodajanje in odvzemanje elementov tabele deluje v času $O(1)$.