Programiranje I: 1. izpit

2. svečana leta Gospodovega 2015

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

1. naloga (Skakačev sprehod, 30 točk)

Dana je šahovnica velikosti $n \times n$. Polje v r-ti vrstici in c-tem stolpcu označimo z (r,c), $1 \le r,c \le n$. Zaporedje polj

$$(r_1,c_1),(r_2,c_2),(r_3,c_3),\ldots,(r_k,c_k)$$

opisuje veljaven skakačev sprehod, če velja:

a) $|r_i - r_{i+1}| = 2 \wedge |c_i - c_{i+1}| = 1$ ali $|r_i - r_{i+1}| = 1 \wedge |c_i - c_{i+1}| = 2$ za vse $1 \le i < k$ (tj. skakač dela premike v obliki črke L);
b) vsako polje šahovnice obišče natanko enkrat.

V *Mathematici* sestavite funkcijo `veljavenSprehod[sprehod_, n_]`, ki kot argumenta dobi:

- seznam, ki opisuje sprehod skakača, in
- velikost šahovnice.

Funkcija naj vrne `True`, če je sprehod veljaven, in `False` sicer. Primer:

```
In[1]:= sprehod = {{1, 2}, {2, 4}, {4, 3}, {3, 1}, {2, 3}, {4, 4}};
In[2]:= veljavenSprehod[sprehod, 4]
Out[2]= False
```

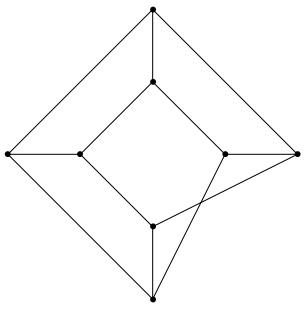
Nasvet 1: Najprej sestavite dve pomožni funkciji. Prva naj preveri pogoj a), druga pa pogoj b).

Nasvet 2: Za katere pare polj (r,c) in $(\tilde{r},\tilde{c})$ velja $|r-\tilde{r}| \cdot |c-\tilde{c}| = 2$?

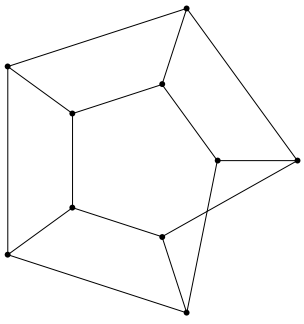
Nasvet 3: Kaj vrne `Sort[{{1, 2}, {2, 4}, {4, 3}, {3, 1}, {2, 3}, {4, 4}}]`?

2. naloga (Möbiusova lestev, 30 točk)

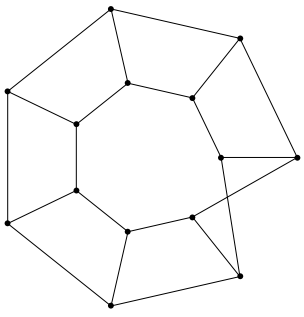
V *Mathematici* sestavite funkcijo `lestev[n_]`, ki nariše Möbiusovo lestev reda n. Slika grafa naj ima vozlišča ekvidistančno razporejena na dveh krožnicah (s polmeroma 1 in 2); na vsaki krožnici naj bo n vozlišč. Vsako "notranje" vozlišče je povezano z ustreznim "zunanjim" vozliščem. Povezave so tudi med zaporednimi vozlišči na notranji oz. zunanji krožnici, razen na enem mestu, kjer se povezavi "prekrižata", kakor lahko vidite na primerih:



`lestev[4]`



`lestev[5]`



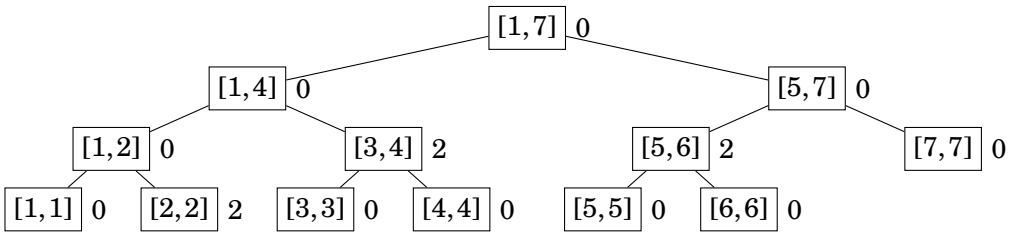
`lestev[7]`

Še formalna definicija Möbiusove lestve M_n:

$$\begin{aligned} V(M_n) &= \{(i,j) \mid 0 \le i < n, 0 \le j \le 1\} \\ E(M_n) &= \{\{(i,0),(i,1)\} \mid 0 \le i < n\} \cup \{\{(i,0),(i+1,0)\} \mid 0 \le i < n-1\} \cup \\ &\quad \{\{(i,1),(i+1,1)\} \mid 0 \le i < n-1\} \cup \{\{(0,0),(n-1,1)\},\{(0,1),(n-1,0)\}\} \end{aligned}$$

3. naloga (Zasnežena cesta, 10 + 10 + 10 točk)

Izdelali bomo razred `Cesta`, ki bo v pomoč snežni službi, ki skrbi za čiščenje zelo dolge ceste. Cesta je razdeljena na kilometrske odseke, ki so oštevilčeni od 1 do n. Predstavimo jo lahko s drevesom, kot ga vidite na sliki (primer za $n = 7$):



Razred `Cesta` (ki je nekakšno dvojiško drevo) je že delno implementiran. Vsako vozlišče predstavlja nek interval odsekov in ima atributa `zacetek` in `konec`. Koren predstavlja celotno cesto in ima `zacetek` = 1 in `konec` = n. Če je `zacetek` $\neq$ `konec`, ima vozlišče še atributa `levo` in `desno`, ki predstavljata levi in desni podinterval. Vsak podinterval pokriva polovico odsekov (oz. je levi za 1 daljši, če število odsekov ni sodo).

Poleg tega ima vsako vozlišče še atribut `sneg`, ki predstavlja količino snega na tem intervalu (števila, ki so zapisana desno od vozlišč na zgornji sliki). Če bi bila cesta v celoti očiščena, bi povsod imeli vrednost 0. Ker je na $[2,6]$ zapadlo 2 cm snega, imamo ponekod vrednost 2. Atribut `sneg` smo povečali samo pri tistih vozliščih, za katere velja, da:

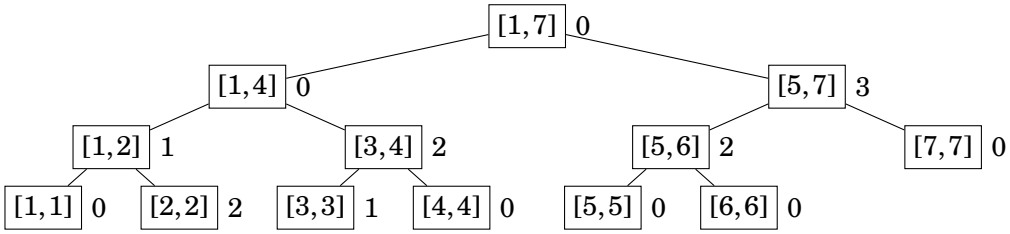
a) so v celoti vsebovana v $[2,6]$ in
b) njihov oče ni v celoti vsebovan v $[2,6]$.

Tako smo atributu `sneg` vozlišča $[3,4]$ prišteli 2, saj $[3,4] \subseteq [2,6]$ in oče$([3,4]) = [1,4] \not\subseteq [2,6]$. Atributu `sneg` vozlišča $[5,5]$ pa nismo prišteli nič, saj oče$([5,5]) = [5,6] \subseteq [2,6]$.

a) (10 točk) Sestavite metodo `zasnezi(self, a, b, k)`, ki strukturo `Cesta` ustrezno spremeni, če na odsekih $[a,b]$ zapade k centimetrov snega. Primer (če `c` ustreza zgornji sliki):

```
>>> c.zasnezi(5, 7, 3)
>>> c.zasnezi(1, 3, 1)
>>> c.seznam()
[1, 3, 3, 2, 5, 5, 3]
```

Novo stanje:



b) (10 točk) Napišite metodo `kolicina(self, t)`, ki vrne količino snega na t-tem odseku ceste. Primer (če `c` ustreza zadnjemu stanju ceste):

```
>>> c.kolicina(6)
5
```

c) (10 točk) Sestavite metodo `ocisti(self, t)`, ki strukturo `Cesta` ustrezno spremeni, če plug očisti ves sneg s t-tega odseka ceste. Primer (če `c` ustreza zadnjemu stanju ceste):

```
>>> c.ocisti(6)
>>> c.seznam()
[1, 3, 3, 2, 5, 0, 3]
```

4. naloga (Izpitno obdobje, 15 + 15 točk)

Prihaja izpitno obdobje in študentje so sklenili, da bodo rešili vse stare izpitne naloge pri vseh predmetih. Organizirali so se in si razdelili delo. Seznam $p = [p_1, p_2, \ldots, p_n]$ predstavlja število nalog pri vsakem od n predmetov. V letniku je d dobrih študentov in s slabih študentov. Dober študent je sposoben rešit m nalog na dan, pod pogojem, da *so vse naloge iz istega predmeta*. Slabi študentje lahko rešijo le 1 nalogo na dan.

Vincenc se je vprašal, ali so sposobni rešiti vse naloge v k dneh. Razmišljal je takole: Na voljo imamo dk "dobrih" človek-dni in sk "slabih" človek-dni. Dobro je, če pri vsakem predmetu naredimo paketke po m nalog in jih razdelimo dobrim študentom. Tako lahko naredimo

$$\delta = \sum_{i=1}^{n} \left\lfloor \frac{p_i}{m} \right\rfloor$$

takšnih paketkov, pri čemer nam ostane še

$$p' = [p_1 \bmod m, p_2 \bmod m, \ldots, p_n \bmod m]$$

nalog pri vsakem predmetu.

- Če je $dk < \delta$, ni dovolj dobrih študentov, da bi rešili vse paketke, zato jih rešijo le dk in ostane še
$$\sigma = m(\delta - dk) + \sum_{i=1}^{n} p_i \bmod m$$
nalog, ki jih morajo rešiti slabi študentje. Naloge je možno rešiti v danem času, če je $\sigma \leq sk$.
- Če pa je $dk \geq \delta$, lahko dobri študentje poskrbijo za vse paketke in ostane še
$$\delta' = dk - \delta$$
dobrih človek-dni. Seznam p' smemo brez škode urediti nenaraščajoče (predmete lahko drugače oštevilčimo, pa se zaradi tega problem nič ne spremeni). Preostale naloge pri predmetih $1, 2, \ldots, \delta'$ lahko razdelimo med dobre študente. Slabim študentom ostane še
$$\sigma' = \sum_{i=\delta'+1}^{n} p_i \bmod m$$
nalog. Naloge je možno rešiti v danem času, če je $\sigma' \leq sk$.

a) (15 točk) Sestavite funkcijo `mozno_resiti(p, s, d, m, k)`, ki kot argumente dobi:
- `p` – seznam s številom nalog pri posameznih predmetih;
- `s` – število slabih študentov;
- `d` – število dobrih študentov;
- `m` – število nalog, ki so jih sposobni rešiti dobri študentje v enem dnevu;
- `k` – število dni, ki jih imajo študentje na voljo za reševanje nalog.

Funkcija naj vrne `True`, če lahko študentje rešijo vse naloge v danem času, in `False` sicer. Postopajte tako, kot predlaga Vincenc.

b) (15 točk) Napišite funkcijo `koliko_dni(p, s, d, m)`, ki na učinkovit način izračuna najmanjše število dni, ki jih potrebujejo študentje, da rešijo vse naloge. *Nasveti:*
- Kličite funkcijo `mozno_resiti` iz prejšnje podnaloge (tj. uporabljajte jo kot črno škatlo).
- Študentje lahko naloge zagotovo rešijo v
$$\left\lceil \frac{\sum_{i=1}^{n} p_i}{d + s} \right\rceil$$
dnevih (če bi se organizirali tako, da bi vsak rešil po 1 nalogo na dan).