Računalništvo (fizika): 1. izpit

29. 1. 2018

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)

Za dano preslikavo f lahko tvorimo zaporedje

$$0, \quad f(0), \quad f(f(0)), \quad f(f(f(0))), \quad \ldots,$$

ki mu pravimo *orbita* preslikave f. Orbita se lahko ujame v periodo (neprazno končno zaporedje), ali pa tudi ne. Če jo doseže, se perioda ponavlja od nekega elementa naprej. Na primer, če je $f(n) = (17n^2 + 8) \bmod 11$, dobimo zaporedje

$$0,8,7,5,4,5,4,5,4,5,\ldots$$

ki je periodično s periodo $[5,4]$.

a) (10 točk) Sestavite *generator* `orbita(f)`, ki sprejme preslikavo `f` in vrača elemente njene periode.

b) (15 točk) Sestavite funkcijo `perioda(f)`, ki sprejme preslikavo `f` in vrne njeno periodo, če ta obstaja. V komentar zapišite, kaj naredi vaša rešitev, če perioda ne obstaja. Primer uporabe:

```
>>> perioda(lambda n : (17 * n * n + 8) % 11)
[5,4]
```

2. naloga (25 točk)

Aljaž se ob čuvanju študentov na pisnem izpitu kratkočasi z aritmetično igrico. Na papir zapiše zaporedje števil, na primer

$$5 \quad 7 \quad 3 \quad 8$$

nato pa vstavlja znake (,) in − tako, da dobi čim večjo vrednost izraza. V zgornjem primeru je to $(5-7)-(3-8) = 3$.

Pomagajte Aljažu in **sestavite funkcijo** `maksi(t)`, ki sprejme tabelo števil `t` in vrne največjo vrednost, ki jo lahko dobimo tako, da vstavljamo oklepaje in minus. Primer uporabe:

```
>>> maksi([5, 7, 3, 8])
3
>>> maksi([29, 1, 2018])
2046
>>> maksi([5, 8])
-3
>>> maksi([5])
5
```

Namig: Sestavite tudi funkcijo `mini(t)`, ki poišče namanjšo možno vrednost. Obe funkciji sta rekurzivni, vsaka od njiju kliče obe. Delujeta tako, da tabelo `t` razdelita na levi in desni del na vse možne načine in naredita ustrezne rekurzivne klice na obeh delih. Izmed vseh tako dobljenih vrednosti vrneta najbolj ugodno.

3. naloga (30 točk)

Posadka vesoljske ladje *USS Voyager* pod poveljstvom kapitanke Janeway potuje po kvadrantu Δ, ki je razdeljen v kvadratno mrežo $n \times n$ sektorjev. Pot se začne zgoraj levo v sektorju $(0,0)$, cilj pa je spodaj desno v sektorju $(n-1,n-1)$. Ladja lahko v eni potezi z nadsvetlobno hitrostjo preskoči k sektorjev navzdol ali k sektorjev desno, pri čemer porabi $3+k$ enot devterija (tri enote devterija se porabijo za aktivacijo warp jedra, preostalo pa za pot).

Ladja je imela bližnje srečanje z Vaadwaursko floto, zato so povsem porabili svoje zaloge devterija. Vsak sektor (i,j) vsebuje zalogo devterija `d[i][j]`, ki je lahko tudi ničelna. Vodja inžinirjev B'Elanna Torres mora izračunati pot, po kateri bodo potovali, da bodo dosegli sektor $(n-1,n-1)$, ali se mu vsaj čim bolj približali. Razdaljo med sektorji merimo kot običajno evklidsko razdaljo, torej sta sektorja (i,j) in (k,ℓ) oddaljena $\sqrt{(i-k)^2+(j-\ell)^2}$.

Seveda začnejo tako, da osvežijo zaloge devterija, ki so na voljo v sektorju $(0,0)$. Zaloge lahko obnovijo v sektorjih, kjer se ustavijo. Zaloge iz sektorjev, ki jih preletijo z nadsvetlobno hitrostjo, niso na voljo.

Sestavite funkcijo `najblizje(d)`, ki sprejme tabelo z opisom zalog devterija. Funkcija naj vrne koordinate sektorja (i,j), ki je čim bližje $(n-1,n-1)$ in do katerega še lahko priletijo. Če je takih sektrojev več, naj vrne koordinate enega izmed njih. Uporabite primeren tip algoritma za reševanje tega problema.

4. naloga (40 točk)

Tudi študentom je med izpitom dolgčas, zato bi radi igrali "4 v vrsto". Igro igrajo na poljih različnih velikosti.

a) (10 točk) Sestavite razred `StiriVVrsto`, ki hrani stanje igre. Konstruktor sprejme števili v in s, ki določata število vrstic v in število stolpcev s. Objekt vsebuje naslednje atribute:

- `polje`: tabela velikosti $s \times v$, v kateri je shranjena trenutna pozicija,
- `na_potezi`: kdo je na potezi, ali `None`, če je igralno polje polno.

Vaša rešitev naj vsebuje ustrezne komentarje, ki pojasnjujejo kodo. Na primer, kako ste označili igralce, kakšne vrednosti so v tabeli, ki predstavlja trenutno pozicijo itd.

b) (10 točk) Sestavite metodo `__str__`, ki vrne niz, ki predstavlja trenutno stanje igralnega polja, na primer za polje velikosti 6×9:

```
|.........|
|.........|
|.....O...|
|....**...|
|...OO**..|
|..*OO*O..|
+---------+
```

c) (10 točk) Sestavite metodo `poteza(s)`, ki sprejme stolpec `s`, v katerega trenutni igralec vrže žeton. Žeton pade navzdol, dokler ne zadane drugega žetona ali doseže dna igralnega polja. Metoda naj vrne `True`, če je poteza uspešno izvedena, in `False`, če je to neveljavna poteza (ker `s` ni veljaven stolpec, ker ni nihče na potezi, ali ker je `s`-ti stolpec že poln). Pazite, da se ustrezno spremeni tudi igralec na potezi.

d) (10 točk) Sestavite še metodo `zmagovalec()`, ki preveri, ali je kateri od igralcev že zmagal. Metoda naj vrne zmagovalnega igralca, ali `None`, če ga ni (ker igre še ni konec ali ker je neodločeno). Štejejo vodoravne, navpične in diagonalne četverice.