Algoritmi in podatkovne strukture 1 2018/2019

Seminarska naloga 2

Rok za oddajo programske kode prek učilnice je **sobota**, **12. 1. 2019.**

Zagovori seminarske naloge bodo potekali v terminu vaj v tednu **14. 1. – 18. 1. 2019**.

Navodila

Oddana programska rešitev bo avtomatsko testirana, zato je potrebno strogo upoštevati naslednja navodila:

- Uporabite programski jezik java (program naj bo skladen z različico JDK 1.8).
- Rešitev posamezne naloge mora biti v eni sami datoteki. Torej, za pet nalog morate oddati pet datotek. Datoteke naj bodo poimenovane po vzorcu NalogaX.java, kjer X označuje številko naloge.
- Uporaba zunanjih knjižnic ni dovoljena. Uporaba internih knjižnic java.* je dovoljena (vključno z javanskimi zbirkami iz paketa java.util).
- Razred naj bo v privzetem (default) paketu. Ne definirajte svojega.
- Uporabljajte kodni nabor utf-8.

Ocena nalog je odvisna od pravilnosti izhoda in učinkovitosti implementacije (čas izvajanja). Čas izvajanja je omejen na 1s za posamezno nalogo.

Naloga 6

Transportno podjetje prevaža blago med mesti, povezanimi z dvosmernimi cestami. Nekatere ceste vključujejo predore, ki omejujejo prehod previsokim tovornjakom.

Zaradi pogostih vzdrževalnih del v predorih je treba spreminjati pot, po kateri bo blago prepeljano iz enega mesta v drugo. V ta namen želimo prešteti, na koliko različnih načinov lahko pridemo iz enega mesta do drugega. Pri tem preštevamo samo poti, ki ne vsebujejo ciklov (mesta se v njem ne ponavljajo) in so vsi morebitni predori na poti višji od višine vozila (podan parameter).

Želimo torej prešteti število različnih sekvenc id_1, id_2, …., id_n brez ponavljajočih elementov, kjer je id_1 začetna lokacija, id_n pa končna lokacija. Pri tem mora veljati, da sta sosednja id-ja povezana s cesto, ki ne vsebuje predora, strogo nižjega od višine vozila.

V vhodni datoteki je v prvi vrstici zapisano število cest na območju M. V naslednjih M vrsticah so zapisani podatki o cestah. Vsaka vrstica je oblike *ID_M1,ID_M2,H* (vse vrednosti so cela števila). *ID* predstavlja oznako ceste; *ID_M1* in *ID_M2* sta oznaki mest ki ju ta cesta povezuje; *H* je višina predora (za ceste brez predorov je vrednost *H* enaka -1). Podatkom o cestah sledi vrstica oblike *ID1,ID2*, ki določata izhodiščno in ciljno mesto. Tej vrstici sledi vrstica z višino vozila, ki bo poslano na pot.

Lahko predpostavite, da bo za povezani mesti M1 in M2 samo en zapis v vhodni datoteki (če je v datoteki podana povezava iz M1 v M2, ne bo eksplicitno podane povezave iz M2 v M1, saj so povezave dvosmerne).

V izhodno datoteko zapišite število različnih poti med izhodiščnim in ciljnim mestom (to je *ID1* in *ID2*).

Implementirajte razred ***Naloga6***, ki vsebuje metodo ***main***. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]), prebere vhodne podatke in sestavi izhodno datoteko.

Primer:

Vhodna datoteka	Izhodna datoteka
5 1,3,2 1,4,1 3,4,3 3,2,-1 4,2,3 1,2 2	2

Razlaga primera: Iz mesta 1 do mesta 2 lahko pridemo po poti 1-3-2 ali 1-3-4-2. Iz mesta 1 v mesto 4 ne moremo neposredno zaradi prenizkega predora.

Če bi v zadnji vrstici vhodne datoteke bila vrednost 1 (namesto 2), bi lahko ubrali tudi poti 1-4-2 in 1-4-3-2, tako da bi v izhodno datoteko zapisali vrednost 4.

Naloga 7

Želimo poskrbeti za lepši izris splošnih binarnih dreves (med elementi ni nujna urejenost). V ta namen želimo vsakemu vozlišču v drevesu določiti koordinati (x,y) v izrisu. Veljajo naslednja pravila:

- Koordinata y je enaka globini vozlišča v drevesu. Koren je na globini 0.
- Za vsako poddrevo s korenom *k* velja:
  - Koordinate x vozlišč v levem poddrevesu so manjše od koordinate x korena *k.*
  - Koordinate x vozlišč v desnem poddrevesu so večje od koordinate x korena *k.*
- Noben par vozlišč v drevesu nima enakih koordinat x.
- Zaloga vrednosti koordinat x je od 0 do N - 1, pri čemer je N število vozlišč v drevesu.

Drevo je podano v tekstovni vhodni datoteki. V prvi vrstici je zapisano celo število N, ki označuje število vozlišč v drevesu. V naslednjih N vrsticah so zapisani podatki o vozliščih v poljubnem vrstnem redu. Posamezna vrstica je oblike *ID,V,ID_L,ID_R* (vse vrednosti so cela števila). *ID* predstavlja identifikator vozlišča; *V* je vrednost, zapisana v tem vozlišču; *ID_L* je identifikator levega sina; *ID_R* je identifikator desnega sina. Za identifikatorje velja, da so enolično določeni. Identifikator -1 označuje prazno poddrevo (vozlišče nima ustreznega sina).

Izhodna datoteka naj vsebuje N vrstic. Posamezna oblika naj bo sestavljena iz podatkov: vrednost v vozlišču, x koordinata vozlišča, y koordinata vozlišča (ločeno z vejicama). Vozlišča izpisujte v vrstnem redu, ki ustreza obhodu drevesa po nivojih - vozlišča izpisujemo nivo po nivo (vozlišča znotraj istega nivoja naj bodo izpisana od skrajno levega proti skrajno desnem).

Implementirajte razred ***Naloga7***, ki vsebuje metodo ***main***. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]), prebere vhodne podatke in sestavi izhodno datoteko.

Primer:

Vhodna datoteka:	Izhodna datoteka:

`5`
`14,5,-1,210`
`302,2,14,-1`
`42,1,302,8`
`210,3,-1,-1`
`8,4,-1,-1`
	`1,3,0`
	`2,2,1`
	`4,4,1`
	`5,0,2`
	`3,1,3`

Izhodna datoteka ustreza naslednji razporeditvi vozlišč ob izrisu:

x y	0	1	2	3	4
0				1	
1			2		4
2	5				
3		3			

Naloga 8

Logistično podjetje razvaža blago med N mesti, ki so povezana z M dvosmernimi cestnimi odseki z znanimi dolžinami v kilometrih. Velja, da za poljubni mesti obstaja zaporedje cestnih odsekov, ki ju povezuje. Podjetje želi avtomatizirati svojo dejavnost z uporabo (N-1) avtonomnih električnih vozil. Vsako avtonomno vozilo je mogoče naučiti, da izvaja prevoz na enem cestnem odseku v obe smeri (tj. zagotavlja povezavo med mestoma). Vozila imajo omejen doseg, ki predstavlja najdaljšo dolžino cestnega odseka, ki ga vozilo še lahko prevozi z vgrajeno baterijo. Vozila imajo tudi določeno verjetnost, da bo med vožnjo prišlo do okvare. Ob vsakem prihodu vozila v mesto se vgrajena baterija napolni.

Vsa vozila imajo enak doseg D in enako verjetnost P okvare na prevoženih 1000 kilometrov.

Cilj naloge je vozilom dodeliti cestne odseke na tak način, da je omogočen transport med vsemi mesti (bodisi neposredno bodisi preko drugih mest), pri čemer pa naj bo skupna verjetnost okvare čim manjša. Če zaradi podanih omejitev takšna dodelitev ne obstaja in posledično transport med nekaterimi pari mest ni možen, mora program to situacijo prepoznati.

V vhodni datoteki je v prvi vrstici podano število cestnih odsekov M. V naslednjih M vrsticah sledijo opisi odsekov v obliki *ID,ID_M1,ID_M2,C*. *ID* predstavlja identifikator cestnega odseka; *ID_M1 in ID_M2* sta identifikatorja mest, ki ju ta odsek povezuje; *C* pa predstavlja dolžino tega odseka. Vsi podatki v opisu cestnega odseka so cela števila. Lahko privzamete, da dve mesti povezuje največ en cestni odsek. Opisom cestnih odsekov sledi vrstica, ki vsebuje celo število D in predstavlja doseg vozil. V naslednji vrstici je zapisana verjetnost napake na 1000km, ki je podano kot decimalno število (kot decimalno ločilo je uporabljena pika). Opozorilo: število mest N (in s tem število avtonomnih vozil N-1) lahko razberete iz podanih podatkov, saj do vseh mest pelje vsaj ena cesta.

V izhodno datoteko zapišite identifikatorje cestnih odsekov z dodeljenimi vozili, ločene z vejico. Identifikatorji naj bodo zapisani v naraščajočem vrstnem redu. Če naloga ni rešljiva, v izhodno datoteko zapišite samo vrednost -1.

Implementirajte razred ***Naloga8***, ki vsebuje metodo ***main***. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]), prebere vhodne podatke in sestavi izhodno datoteko.

Primer:

Vhodna datoteka:	Izhodna datoteka:
5 1,1,2,2 2,1,4,1 3,2,4,3 4,2,3,5 5,3,4,3 3 0.00001	1,2,5

Naloga 9

Podana je množica oseb M = {o_1, o_2, …, o_m} in zaporedje dejstev o prijateljstvih N = {p_1, p_2, …, p_n}. Vsako prijateljstvo p_i je zapisano v obliki para <o_{i1},o_{i2}> in določa, da sta osebi o_{i1} in o_{i2} prijatelja. Na podlagi zaporedja dejstev N lahko osebe iz M razdelimo v skupine prijateljev. Privzemite , da velja, da je prijatelj mojega prijatelja tudi moj prijatelj.

V zaporedju N je lahko navedba nekaterih dejstev odveč, saj te sledijo iz predhodnih dejstev. Naloga je odvečna dejstva identificirati in jih zapisati v izhodno datoteko. Odvečne trditve identificiramo tako, da dejstva iz N upoštevamo **v podanem** vrstnem redu.

Vhodna datoteka v prvi vrstici vsebuje celo število P, ki določa število dejstev o prijateljstvih. V naslednjih P vrsticah so zapisani pari oseb *O1,O2*, ki sta v prijateljskem odnosu. Pri tem velja, da sta *O1* in *O2* pozitivni celi števili, ločeni z vejico.

V izhodno datoteko izpišite identificirana odvečna dejstva, po eno dejstvo na vrstico.

Implementirajte razred ***Naloga9***, ki vsebuje metodo ***main***. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]), prebere vhodne podatke in sestavi izhodno datoteko.

Primer:

Vhodna datoteka:	Izhodna datoteka:
6 **1,2** **4,2** **5,6** **1,4** **3,4** **1,3**	**1,4** **1,3**

Razlaga primera: trditev <1,4> sledi iz dejstev <1,2> in <4,2>, trditev <1,3> pa iz dejstev <1,2>, <4,2> in <3,4>.

Naloga 10

Podan je usmerjen graf z *n* vozlišči in *m* povezavami, kjer je vsaki povezavi *(a,b)* pripisana cena *c(a,b)*. Vsako vozlišče ima dodeljeno oznako *id* (pozitivno celo število). Želimo poiskati pot z najmanjšo vsoto cen povezav od začetnega vozlišča *s* do končnega vozlišča *t*.

Naloge ne bomo reševali z Dijkstrovim algoritmom, ampak s postopkom, ki ponazarja premikanje agenta po grafu. Pri premikanju agent upošteva oceno razdalje vozlišč do cilja *t* (v nadaljevanju označena kot *h*). Na začetku postopka je ocena *h* za vsa vozlišča enaka 0.

Iskanje optimalne poti se izvaja v iteracijah. Vsako iteracijo agent prične v vozlišču *s* in se premika po povezavah, dokler ne prispe v vozlišče *t*, kar predstavlja konec iteracije. Med premiki agent posodablja ocene *h* obiskanih vozlišč. V eni iteraciji agent obišče posamezno vozlišče kvečjemu enkrat. Po zaključku ene iteracije se agent prestavi v izhodiščno vozlišče *s* in se prične nova iteracija. Celoten postopek se zaključi, ko agent dvakrat zapovrstjo ubere isto pot od *s* do *t*, ne da bi se ob tem spremenile ocene *h* obiskanih vozlišč.

Agent na svoji poti od *s* do *t* v vsakem vozlišču *a* izbira, v katero vozlišče so bo premaknil v naslednjem koraku. Najprej za vsa sosednja, v trenutni iteraciji še ne obiskana, vozlišča *b* (tista, v katera lahko pride z enim premikom) izračuna vsoto *v(b) = c(a,b) + h(b)* (vrednost *v(b)* predstavlja seštevek cene premika iz *a* v *b* in ocenjene razdalje med *b* in *t*). Izmed vseh sosednjih vozlišč agent izbere tisto z **najmanjšo** vrednostjo *v(b)*, poimenujmo ga (tisto vozlišče) *b_min*. V primeru, da ima več vozlišč enako oceno *v(b)*, se izbere tisto z **najmanjšo** oznako *id*. Agent se nato premakne v vozlišče *b_min.* Obenem se v primeru, da je *h(a)* **manjše od** *v(b_min),* nastavi *h(a) = v(b_min).*

Kadar agent med premiki pride v vozlišče, ki ni cilj *t* in nima izhodnih povezav, se ocena *h* tega vozlišča nastavi na neskončno, trenutna iteracija pa se zaključi. Iteracija se prav tako zaključi v primeru, ko agent pride v vozlišče, ki ni ciljno in ima vsaj eno izhodno povezavo, ampak s premiki ne more nadaljevati, ker so vsa sosednja vozlišča že obiskana. V tem primeru se ob zaključku iteracije ocena *h* zadnjega vozlišča ne posodobi.

V vhodni datoteki je v prvi vrstici podano število povezav M. V naslednjih M vrsticah sledijo opisi povezav v obliki treh celih števil, ločenih z vejico. Zapis *ID_V1,ID_V2,C* predstavlja usmerjeno povezavo iz vozlišča z oznako *ID_V1* v vozlišče z oznako *ID_V2* s ceno *C*. Opisom povezav sledi vrstica oblike *ID_S,ID_T*, ki določa izhodiščno (*ID_S*) in ciljno vozlišče *(ID_T)*.

V izhodno datoteko za vsako iteracijo izpišite vozlišča, ki jih je obiskal agent (vključno z zadnjima dvema, identičnima, iteracijama).

Implementirajte razred ***Naloga10***, ki vsebuje metodo ***main***. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]), prebere vhodne podatke in sestavi izhodno datoteko.

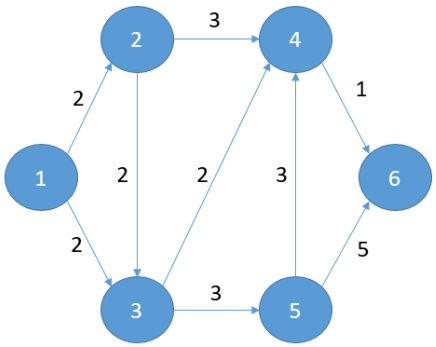
Primer:

Vhodna datoteka:	Izhodna datoteka:

`9`
`1,2,2`
`1,3,2`
`2,3,2`
`2,4,3`
`3,4,2`
`3,5,3`
`5,4,3`
`5,6,5`
	`1,2,3,4,6`
	`1,2,3,4,6`
	`1,3,4,6`
	`1,3,4,6`

4,6,1 1,6	

Razlaga primera:



Na začetku velja h(1) = h(2) = h(3) = h(4) = h(5) = h(6) = 0. Agent se nahaja v vozlišču 1 in oceni neposredno dosegljivi vozlišči 2 in 3. Dobljeni oceni sta v(2) = 2 + 0 = 2 in v(3) = 2 + 0 = 2. Vozlišči sta ocenjeni enako, zato se agent premakne v vozlišče 2, ki ima manjšo oznako. Obenem agent nastavi novo hevristično oceno za vozlišče 1 na h(1) = 2. Iz vozlišča 2 sta neposredno dosegljivi vozlišči 3 in 4. Agent ju oceni in dobi v(3) = 2 + 0 = 2 ter v(4) = 3 + 0 = 3. Ker je vozlišče 3 bolje ocenjeno, se agent premakne vanj in nastavi h(2) = 2. Iz vozlišča 3 sta neposredno dosegljivi vozlišči 4 in 5. Agent ju oceni in dobi v(4) = 2 + 0 = 2 ter v(5) = 3 + 0 = 3. Sedaj je vozlišče 4 tisto bolje ocenjeno, zato se agent premakne vanj in nastavi h(3) = 2. Iz vozlišča 4 je dosegljivo le vozlišče 6. Agent ga oceni in dobi v(6) = 1 + 0 = 1, se premakne vanj ter nastavi h(4) = 1. S tem se zaključi prva iteracija in se v izhodno datoteko zapiše sekvenca 1,2,3,4,6.

Na začetku druge iteracije je agent spet v vozlišču 1. Agent oceni vozlišči 2 in 3 in dobi v(2) = 2 + 2 = 4 ter v(3) = 2 + 2 = 4. Ponovno izbere premik v vozlišče 2 in nastavi h(1) = 4. V vozlišču 2 agent oceni vozlišči 3 in 4 ter dobi v(3) = 2 + 2 = 4 in v(4) = 3 + 1 = 4. Ker sta oceni enaki, agent izbere vozlišče 3 (manjša oznaka) in se premakne vanj, obenem nastavi h(2) = 4. V vozlišču 3 agent oceni vozlišči 4 in 5 in dobi v(4) = 2 + 1 = 3 ter v(5) = 3 + 0 = 3. Oceni sta enaki, zato agent izbere vozlišče 4 (manjša oznaka) in se premakne vanj, obenem nastavi h(3) = 3. V vozlišču 4 agent izračuna v(6) = 1 + 0 = 1 ter se premakne v vozlišče 6, ocena h(4) ostane nespremenjena. S tem se je zaključila druga iteracija in se v izhodno datoteko zapiše sekvenca 1,2,3,4,6. Agent je dvakrat zapored izbral enako pot, vendar se je med sprehodom vsaj ena hevristična ocena h(v) spremenila, zato postopek še vedno ni zaključen.

Na začetku tretje iteracije agent oceni vozlišči 2 in 3 ter dobi v(2) = 2 + 4 = 6 in v(3) = 2 + 3 = 5, se premakne v vozlišče 3 in nastavi h(1) = 5. V vozlišču 3 oceni vozlišči 4 in 5 ter dobi v(4) = 2 + 1 = 3 in v(5) = 3 + 0 = 3. Izbere vozlišče 4 (manjša oznaka) ter se premakne vanj, ocena h(3) pa ostane nespremenjena. Iz vozlišča 4 je neposredno dosegljivo samo vozlišče 6, agent ga oceni in dobi v(6) = 1 + 0 = 1. Agent se premakne v vozlišče 6, ocena h(4) ostane nespremenjena, iteracija se zaključi in se v izhodno datoteko zapiše sekvenca 1,3,4,6.

Četrta iteracija se izvede na enak način kot tretja, s to razliko, da se nobena hevristična ocena h(v) ne spremeni. To je pogoj za konec postopka. V izhodno datoteko zapišemo izbrano pot 1,3,4,6 in zaključimo.