Algoritmi in podatkovne strukture 1 2021/2022

Seminarska naloga 2

Rok za oddajo programske kode prek učilnice je **sobota**, **8. 1. 2022.**

Zagovori seminarske naloge bodo potekali v terminu vaj v tednu **10. 1. – 14. 1. 2022**.

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.
- V drugi seminarski nalogi boste vse vhodne podatke prebrali iz tekstovne datoteke in vse izhode zapisali v tekstovno datoteko.

Ocena nalog je odvisna od pravilnosti izhoda in učinkovitosti implementacije (čas izvajanja). Čas izvajanja je omejen na 2s za posamezno nalogo.

Naloga 6

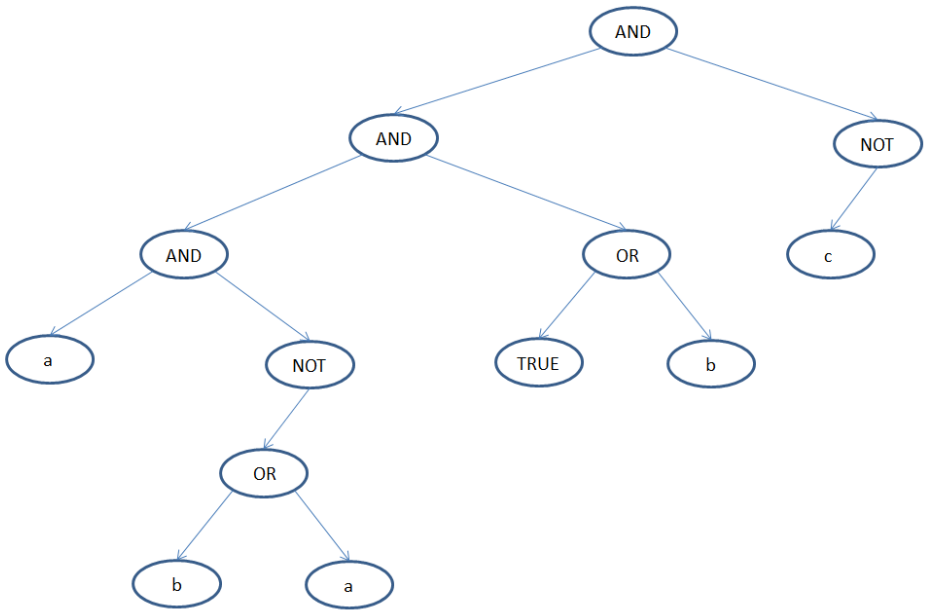
Vhodna tekstovna datoteka vsebuje logični izraz, sestavljen iz konstant ("TRUE" in "FALSE"), spremenljivk (simbolična imena, ki začnejo z malo črko, nadaljujejo se pa lahko s poljubnimi črkami ali ciframi), operatorjev ("AND", "OR" in "NOT") in oklepajev.

Implementirajte razred ***Naloga6***, ki vsebuje metodo ***main***. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]) in na podlagi prebranega vhoda sestavi binarno izrazno drevo (pri tem se držite dogovora, da sta operatorja "AND" in "OR" **levo asociativna**). Ko je drevo zgrajeno, naj se v izhodno datoteko najprej izpišejo oznake vseh vozlišč drevesa v premem (preorder) vrstnem redu (ločeno z vejicami, brez presledkov), nato se v novi vrstici izpiše še višina drevesa.

Primer

Vhodna datoteka:	Izhodna datoteka:
`a AND NOT (b or a) AND (TRUE or b) AND NOT c`	`AND,AND,AND,a,NOT,OR,b,a,OR,TRUE,b,NOT,c` `6`

Razlaga primera
Upoštevamo levo asociativnost operatorjev, zato je rekonstruirano drevo oblike:



Naloga 7

V nekem mestu ima javni prevoz **N** linij. Po mestu je razporejenih **M** postajališč, ki so označena s celimi števili. Potek proge posamezne linije je podan s seznamom oznak postajališč. Vsaka linija obratuje v obeh smereh. Za dani postaji **A** (vstop) in **B** (izstop) poiščite dve poti: pot z najmanjšim številom prestopanj in pot z najmanjšim številom postankov.

Vhodna datoteka v prvi vrstici vsebuje število linij javnega prevoza **N**. Nato so po vrsticah podana zaporedja postajališč, ki določajo potek posameznih linij (ena linija v eni vrstici). Oznake postajališč so ločene z vejicami. V zadnji vrstici vhodne datoteke sta zapisani in z vejico ločeni celi števili **A** in **B**, ki predstavljata oznaki vstopnega in izstopnega postajališča.

Implementirajte razred ***Naloga7***, ki vsebuje metodo ***main***. Metoda prejme poti do vhodne in izhodne datoteke (args[0] in args[1]), prebere vhodne podatke o linijah javnega prevoza in nato v izhodno datoteko zapiše tri vrednosti, vsako v svoji vrstici.

V prvi vrstici naj bo zapisano minimalno število prestopanj, ki ga potrebujemo za pot od vstopnega postajališča **A** do izstopnega postajališča **B** (pri tem nas ne zanima dejanska dolžina poti oziroma število prevoženih postaj). Če sta obe postajališči na isti progi, je število prestopanj enako 0. Če z uporabo linij javnega prevoza ni možno priti iz enega postajališča do drugega, je število prestopanj enako -1.

V drugi vrstici naj bo zapisano število prevoženih postaj na najkrajši vožnji med vstopnim postajališčem **A** in izstopnim postajališčem **B** (pri tem nas ne zanima število prestopanj na tej poti). Če sta postajališči sosednji na isti liniji, je dolžina najkrajše poti enaka 1. Če z uporabo linij javnega prevoza ni možno priti iz enega postajališča do drugega, je dolžina najkrajše poti enaka -1.

V tretji vrstici naj bo zapisano število 1, če ima pot z najmanj postajami tudi najmanj prestopanj. V nasprotnem primeru naj bo zapisano število 0. Če z uporabo linij javnega prevoza ni možno priti iz enega postajališča do drugega, naj bo v tretji vrstici zapisano število -1.

Primer:

Vhodna datoteka:	Izhodna datoteka:
4 1,2,3,4,5 6,7 8,7,4 2,6 1,7	1 3 0

Razlaga primera:
Za pot med postajališči A=1 in B=7 je dovolj samo eno prestopanje (v postajališču 4 prestopimo s prve na tretjo linijo), najkrajša pot pa zahteva 3 postaje (1 → 2 s prvo linijo, 2 → 6 s četrto linijo, 6 → 7 s tretjo linijo). Pot z najmanj postajami ima tri prestopanja, medtem, ko je najmanjše število prestopanj 1. Zato v tretjo vrstico zapišemo 0.

Naloga 8

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 vrstica 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 ***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 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

Razlaga primera
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 9

Podan je neusmerjen povezan graf, kjer vozlišča predstavljajo mesta, povezave pa cestne odseke med njimi. Podana je tudi množica dejstev oblike, **A,B,C**, kjer sta **A** in **B** (naravni števili) oznaki mest, **C** pa predstavlja število potnikov (vozil), ki bodo potovali iz mesta **A** v mesto B. Vsak potnik vedno izbere najkrajšo pot iz mesta **A** v mesto **B**. Če najkrajša pot ni enolično določena, potniki izberejo pot, ki se pojavi prej v urejenem seznamu najkrajših poti. Urejen seznam dobimo tako, da razvrstimo poti naraščajoče najprej po velikosti ključa prvega mesta na poti, potem naraščajoče po velikosti drugega ključa in tako naprej do ključa zadnjega mesta. Na vsakem cestnem odseku se nahaja cestninska postaja, ki vsem vozilom zaračuna enako cestnino. Poiščite cestninske postaje, ki bodo pobrale največ cestnin.

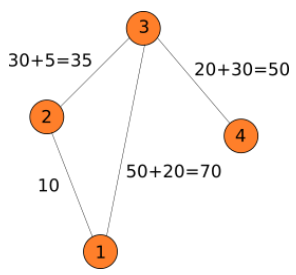
Implementirajte razred ***Naloga9***, ki vsebuje metodo ***main***. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]).

Vhodna datoteka ima v prvi vrstici zapisani in z vejico ločeni naravni števili **N** in **M**, ki določata število povezav grafa (**N**) in število dejstev (**M**). V naslednjih **N** vrsticah so zapisane povezave neusmerjenega grafa, ločene z vejico. Nato pa v naslednjih **M** vrsticah dejstva oblike **A**,**B**,**C**.

Izhodna datoteka naj vsebuje eno vrstico v formatu **A,B**, pri čemer velja, da je **A** < **B**. Zapis **A,B** označuje cestni odsek med mesti **A** in **B**, na katerem se bo pobralo največ cestnine. V primeru, da je takšnih odsekov več, naj bo vsak zapisan v svojo vrstico naraščajoče urejeno najprej po oznaki **A** nato pa **B**.
Primer:

Vhodna datoteka:	Izhodna datoteka:
4,5 1,2 2,3 3,4 1,3 1,2,10 1,3,50 1,4,20 2,4,30 3,2,5	1,3

Razlaga primera
Na spodnjem grafu smo na povezavah sešteli število vseh potnikov, ki potujejo na ustreznih odsekih. Odsek (povezava) 1,3 ima vrednost (število potnikov) 70, kar je največ med vsemi.



Naloga 10

Podana je množica N-točk v 2d prostoru. Vsaka točka je podana s koordinatama (x, y). Točke želimo razdeliti v K skupin. Razdalja med skupinama S1 in S2 je definirana kot evklidska razdalja med najbližjima točkama t1 iz S1 in t2 iz S2. Naloga je poiskati dodelitev točk skupinam tako, da so razdalje med skupinami maksimalne. Ideja: na začetku vsaka točka predstavlja svojo skupino, nato združujemo najbližje skupine dokler ne dobimo zahtevanega števila skupin.

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

Tekstovna vhodna datoteka v prvi vrstici vsebuje celo število N, ki predstavlja število točk. V naslednjih N vrsticah sledijo koordinate točk. Posamezna koordinata je podana v obliki X,Y. Pri tem velja, da sta X in Y števili tipa double. Vsem točkam se implicitno dodelijo enolične oznake (id): prva točka ima id=1, druga točka ima id=2 in tako naprej to zadnje točke, ki ima id=N. V zadnji vrstici vhodne datoteke je zapisano celo število K, ki določa število zahtevanih skupin.

Tekstovna izhodna datoteka naj vsebuje K vrstic. V vsaki vrstici naj bodo izpisane oznake točk, ki pripadajo isti skupini. Oznake točk naj bodo izpisane v naraščajočem vrstnem redu in ločene z vejico. Vrstni red izpisanih skupin naj bo prav tako urejen naraščajoče glede na najmanjšo oznako točke, ki pripada skupini. To pomeni, da bo v prvi vrstici izpisana skupina, ki vsebuje točko z id=1. V vsaki naslednji vrstici bo izpisana tista skupina, ki vsebuje točko z najmanjšo oznako, ki še ni izpisana v prejšnjih vrsticah. Drugače povedano, prve vrednosti v vsaki izpisani vrstici bodo naraščale od prve do zadnje vrstice.

Prvi primer

Vhodna datoteka:	Izhodna datoteka:
5 -1.4,2.5 -2.5,-2.0 1.9,1.7 3.2,-1.2 0.3,1.3 3	1,3,5 2 4

Drugi primer (enake točke ampak drugo število skupin):

Vhodna datoteka:	Izhodna datoteka:
5 -1.4,2.5 -2.5,-2.0 1.9,1.7 3.2,-1.2 0.3,1.3 4	1 2 3,5 4

Razlaga primera

Izris prvega primera (točke ene skupine so izrisane z isto barvo):	Izris drugega primera (točke ene skupine so izrisane z isto barvo):