Algoritmi in podatkovne strukture 1 2017/2018

Seminarska naloga 2

Rok za oddajo programske kode prek učilnice je **sobota**, **13. 1. 2018.**

Zagovori seminarske naloge bodo potekali v terminu vaj v tednu **15. 1. – 19. 1. 2018**.

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 5s za posamezno nalogo.

Naloga 6

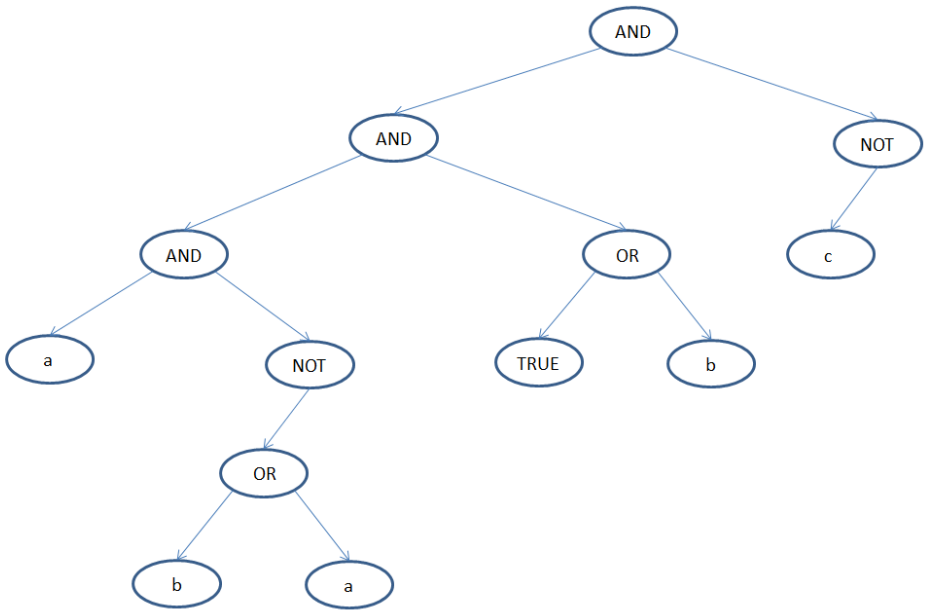
Vhodna tekstovna datoteka vsebuje logični izraz, sestavljen iz konstant ("TRUE" in "FALSE"), spremenljivk (simbolična imena, ki začnejo z malo črko, nadaljujejo se pa lahko s poljubnimi črkami ali ciframi), operatorjev ("AND", "OR" in "NOT") in oklepajev.

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.

Na primer, če je vsebina vhodne datoteke

```
a AND NOT (b or a) AND (TRUE or b) AND NOT c
```

je rekonstruirano drevo oblike



zato bo v izhodni datoteki zapisano

```
AND,AND,AND,a,NOT,OR,b,a,OR,TRUE,b,NOT,c
6
```

Naloga 7

Vhodna datoteka je enake oblike kot v nalogi 6. Implementirajte razred *Naloga7* z metodo *main*, ki v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]). Program naj najprej prebere logični izraz, nato naj v izhodno datoteko zapišite število različnih opredelitev spremenljivk, pri katerih je vrednost logičnega izraza resnična.

Na primer, če je vsebina vhodne datoteke

```
a AND NOT b or NOT c AND b
```

bo v izhodni datoteki zapisano

```
4
```

saj imamo 4 različne nastavitve spremenljivk, pri katerih je izraz resničen

a	b	c
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

Naloga 8

Vhodna datoteka je enake oblike kot v nalogi 6. Implementirajte razred *Naloga8* z metodo *main*, ki v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]). Program naj najprej prebere logični izraz in sestavi binarno izrazno drevo (spet velja dogovor o levi asociativnosti operatorjev "AND" in "OR"), nato naj transformira drevo z upoštevanjem **izključno** naslednjih pravil:

- NOT TRUE → FALSE
- NOT FALSE → TRUE
- NOT NOT X → X
- NOT (X OR Y) → NOT X AND NOT Y
- NOT (X AND Y) → NOT X OR NOT Y

S temi pravili se želimo znebiti odvečnih negacij, oziroma jih potisniti čim bližje listom drevesa. Pravila začnite uporabljati pri korenu, nato pa rekurzivno v poddrevesih. Končno drevo naj bo izpisano v izhodni datoteki s premim (preorder) obhodom (oznake vozlišč naj bodo ločene z vejicami brez presledkov).

Na primer, če je vsebina vhodne datoteke

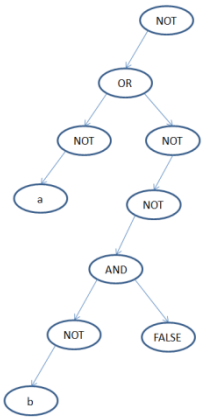
```
NOT (NOT a OR NOT NOT (NOT b AND FALSE))
```

bo v izhodni datoteki zapisano

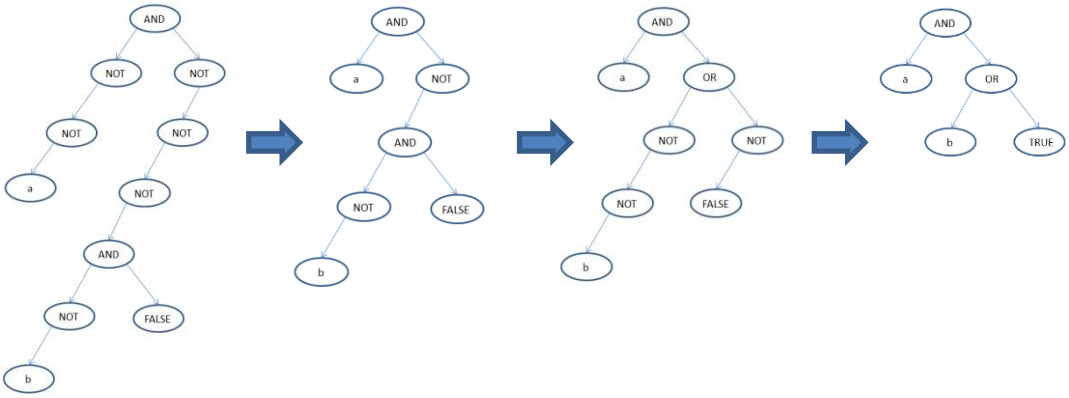
```
AND,a,OR,b,TRUE
```

Analiza primera

Na podlagi vhodnega izraza sestavimo izrazno drevo



Sledi zaporedje transformacij



Naloga 9

Na izlet želimo povabiti N prijateljev (vsak je enolično označen s številom od 1 do N). Eni bi svojo udeležbo brezpogojno potrdili, nekateri pa svoj pristanek pogojujejo s potrjeno udeležbo drugih ljudi. Na voljo imamo N telefonskih klicev, s katerimi želimo pridobiti potrditve prijateljev o udeležbi izleta. Cilj naloge je določiti vrstni red klicev tako, da nihče ne zavrne udeležbe. Ob tem velja dogovor, da če je v danem trenutku možno poklicati več prijateljev, pokličemo tistega z najnižjo identifikacijsko številko!

Implementirajte razred *Naloga9* z metodo *main*, ki v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]). V vhodni datoteki je v prvi vrstici zapisano število prijateljev N. V drugi vrstici je zapisano število omejitev M. V naslednjih M-tih vrsticah so zapisane omejitve oblike A,B. Posamezna omejitev pomeni, da oseba A svojo udeležbo pogojuje z udeležbo osebe B. Ena oseba lahko postavi več omejitev in jih je potrebno vse upoštevati. Program naj v izhodno datoteko zapiše zaporedje identifikacijskih števil (ločeno z vejico, brez presledkov), ki določa vrstni red klicev, pri katerem nihče ne bo zavrnil povabila. Če takega zaporedja ni (do te situacije pride, ko se pogojevanje zacikla), naj se v izhodno datoteko zapiše -1.

Na primer, če je vsebina vhodne datoteke

```
5
8
1,4
2,1
2,3
2,4
3,4
5,1
5,3
5,4
```

bo v izhodni datoteki zapisano

```
4,1,3,2,5
```

Naloga 10

Podan je graf, v katerem vozlišča predstavljajo osebe, povezave med vozlišči pa dolgove med osebami. Pri tem velja, da oznaka na povezavi iz vozlišča A proti vozlišču B označuje znesek, ki ga oseba A dolguje osebi B. Cilj naloge je minimizirati število povezav v grafu tako, da nihče ni oškodovan.

Implementirajte razred *Naloga10* z metodo *main*, ki v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]). V vhodni datoteki je v prvi vrstici zapisano število povezav N. Nato sledijo opisi posameznih povezav, ki so oblike A,B,C. Pri tem velja, da sta A in B oznaki vozlišč, C pa je oznaka povezave med vozliščema.

Na primer, če je vsebina vhodne datoteke:

```
12
V1,V3,6
V1,V5,9
V2,V3,1
V2,V4,3
V3,V1,4
V3,V2,10
V3,V4,10
V4,V1,6
V4,V2,10
V4,V3,1
V5,V1,5
V5,V3,5
```

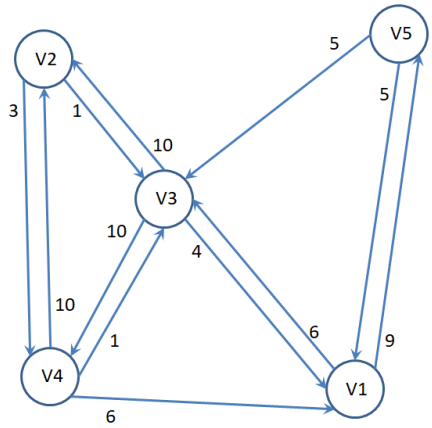
bo v izhodni datoteki zapisano:

```
3
```

saj z minimalno tremi povezavami poravnamo vse dolgove med osebami.

Analiza primera

Izhodiščni graf:



Možna poravnava s tremi povezavami: