Algoritmi in podatkovne strukture 1 2022/2023

Seminarska naloga 1

Rok za oddajo programske kode prek učilnice je **sobota**, **3. 12. 2022.**

Zagovori seminarske naloge bodo potekali v terminu vaj v tednu **5. 12. – 9. 12. 2022**.

Navodila

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

- Uporabite programski jezik java.
- 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 (razen javanskih zbirk 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 2s za posamezno nalogo.

Naloga 1

Podana je mreža črk in seznam besed. Vsaka beseda iz seznama je vpisana v mrežo v eni izmed osmih smeri: vodoravno, navpično in diagonalno, naprej in nazaj. Naloga je poiskati vse besede iz seznama tako, da posamezna črka v mreži pripada samo eni besedi.

Implementirajte razred **Naloga1**, ki vsebuje metodo **main**. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]). Metoda naj prebere vhodne podatke, poišče besede v mreži in v izhodno datoteko zapiše njihove pozicije.

Tekstovna vhodna datoteka je podana v naslednjem formatu:

- V prvi vrstici sta zapisani dve celi števili, ločeni z vejico. Zapis V,S določa dimenzije mreže, pri čemer je V število vrstic in S število stolpcev mreže.
- V naslednjih V vrsticah so zapisani elementi mreže. Vsaka vrstica vsebuje S znakov, ločenih z vejicami.
- V naslednji vrstici je celo število B. Ta predstavlja število besed, ki jih iščemo v mreži.
- V vsaki izmed naslednjih B vrstic je zapisana ena beseda.

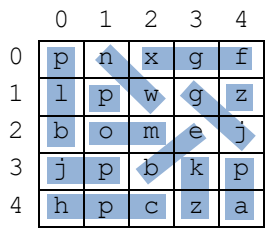
Tekstovna izhodna datoteka naj vsebuje B vrstic. V vsaki vrstici naj bo najprej zapisana beseda, nato še štiri cela števila, ločena z vejico. Prvi par števil V1,S1 določa pozicijo prve črke besede v mreži, drugi par V2,S2 pa pozicijo zadnje črke besede v mreži. **Pozor: indeksiranje začne z 0**.

Pri tej nalogi je možnih več enakovrednih rešitev.

Primer:

Vhodna datoteka:	Izhodna datoteka:
5,5 p,n,x,g,f l,p,w,g,z b,o,m,e,j j,p,b,k,p h,p,c,z,a 12 xgf mo zk jp plb nw hpc p pa z jg be	xgf,0,2,0,4 mo,2,2,2,1 zk,4,3,3,3 jp,3,0,3,1 plb,0,0,2,0 nw,0,1,1,2 hpc,4,0,4,2 p,1,1,1,1 pa,3,4,4,4 z,1,4,1,4 jg,2,4,1,3 be,3,2,2,3

Razlaga primera:



Naloga 2

Peter je poslal Bojanu zakodirano število. Število je zakodiral tako, da je uporabil dve vrsti in dva sklada. Na začetku je vrsto V1 napolnil s števkami po vrsti od leve proti desni. Vrsta V2 in sklada S1 in S2 so na začetku prazni. Potem je N krat ponovil sledeče korake:

Korak 1:

Števke je jemal iz vrste V1 ter jih izmenično postavljal v vrsto V2 in sklad S2. Prvo števko je dal v vrsto V2 drugo v sklad S2 in tako naprej, tako da so števke na lihih mestih šle v vrsto števke na sodih pa v sklad. To je ponavljal do izpraznitve vrste V1.

Korak 2:

Nato je polnil sklad S1 tako, da je iz vrste V2 in sklada S2 izmenično jemal števke, dokler ni bila vrsta V2 prazna. Števke je začel jemati iz vrste V2.

Korak 3:

Iz sklada S1 je nato jemal števke in jih izmenično postavljal najprej v sklad S2 in nato v vrsto V2, in sicer tako, da so šla prva **tri** števila v sklad S2, druga **tri** v vrsto V2, naslednja **tri** ponovno v sklad S2 in tako naprej. To je počel dokler ni bil sklad S1 prazen.

Korak 4:

Na koncu je ponovno napolnil vrsto V1 tako, da je iz sklada S2 in vrste V2 izmenično jemal števke, dokler ni bil sklad S2 prazen. Števke je začel jemati iz sklada S2.

Pomagaj Bojanu poiskati prvotno (nekodirano) število.

Implementirajte razred **Naloga2**, ki vsebuje metodo **main**. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]), prebere vhodne podatke, ter izpiše iskano (nekodirano) število v izhodno datoteko.

Tekstovna vhodna datoteka je podana v naslednjem formatu:

- V prvi vrstici je zapisano število N, ki določa število ponovitev opisanih korakov.
- V naslednji vrstici je zapisano zakodirano število.

Tekstovna izhodna datoteka naj vsebuje vrstico z nekodiranim številom.

Primer:

Vhodna datoteka:	Izhodna datoteka:
4 185246739	123456789

Razlaga kodiranja:

Prvotno število: 123456789

	V1	S1	V2	S2
Inicializacija	1, 2, 3, 4, 5, 6, 7, 8, 9			
Korak 1	/	/	1, 3, 5, 7, 9	2, 4, 6, 8
Korak 2	/	1, 8, 3, 6, 5, 4, 7, 2, 9	/	/
Korak 3			4, 5, 6	9, 2, 7, 3, 8, 1
Korak 4	1, 4, 8, 5, 3, 6, 7, 2, 9			

Opomba: Z znakom »/« označimo prazen sklad ali vrsto. Vrstni red števil od leve proti desni pomeni vrstni red polnjenja. Vrstica prikazuje stanje po izvedenem koraku.

Naloga 3

Implementirajte seznam celih števil, ki podpira naslednje metode:

- public void preslikaj(char op, int val)
- public void ohrani(char op, int val)
- public void zdruzi(char op)

Metoda `void preslikaj(op,val)` se sprehodi čez elemente seznama ter vsak element `e` nadomesti z vrednostjo `e op val`. Parameter `op` lahko zavzame dve možni vrednosti: `'+'` in `'*'`. Drugače povedano, metoda preslikaj vsakemu elementu seznama bodisi prišteje vrednost `val` bodisi ga z `val` pomnoži.

Metoda `void ohrani(op,val)` v seznamu obdrži samo elemente za katere velja `e op val`. Parameter `op` lahko zavzame tri možne vrednosti: `'>'`, `'<'` in `'='`. Drugače povedano, metoda ohrani obdrži le tiste elemente seznama, ki so večji, manjši oziroma enaki parametru `val`.

Metoda `void združi(op)` bodisi sešteje bodisi zmnoži vse elemente seznama, v odvisnosti od vrednosti parametra `op` (`'+'` ali `'*'`). Ob zaključku seznam vsebuje en sam element, ki predstavlja rezultat operacije (seštevek ali zmnožek elementov).

Implementirajte razred **Naloga3**, ki vsebuje metodo **main**. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]). Vhodna datoteka vsebuje začetno stanje seznama in navodila za operacije, ki naj se nad njim izvedejo. Izhodna datoteka vsebuje elemente seznama po končani izvedbi vsake zahtevane operacije.

Tekstovna vhodna datoteka je podana v naslednjem formatu:

- V prvi vrstici je zapisano začetno stanje seznama. Vrednosti posameznih elementov so ločene z vejico.
- V drugi vrstici je število operacij.
- Sledijo navodila za operacije. Vsako navodilo je v eni izmed spodnjih oblik:

  p,A,B označuje klic `preslikaj(A,B)`
  o,A,B označuje klic `ohrani(A,B)`
  z,A označuje klic `zdruzi(A)`

Tekstovna izhodna datoteka vsebuje enako število vrstic, kot je bilo navodil za operacije. Pri tem velja, da i-ta vrstica vsebuje zaporedje elementov, ločenih z vejico, ki predstavljajo stanje seznama po izvedbi i-te operacije.

Primer:

Vhodna datoteka:	Izhodna datoteka:
5,8,3,2,9,3,4,8 4 o,>,5 p,+,2 o,=,10 z,*	8,9,8 10,11,10 10,10 100

Naloga 4

Napisali bomo simulator dogajanja v trgovini z B blagajnami (označene z indeksi od 1 do B). Blagajni z indeksom *i* je prirejeno celo število V_i, ki predstavlja čas skeniranja enega izdelka. Vsaka blagajna ima svojo čakalno vrsto, v katere se postavljajo kupci, ki so nabrali svoje izdelke in želijo zaključiti nakup. Kupec za blagajno (prvi v čakalni vrsti) skenira svoje izdelke, nato zapusti trgovino.

V naši simulaciji se bodo periodično generirali kupci, ki nameravajo vstopiti v trgovino. Kupec je opisan s štirimi parametri:

- ID – enolično določa kupca
- S – dolžina nakupovalnega seznama (število izdelkov, ki jih namerava kupiti),
- H – čas nabiranja enega izdelka z nakupovalnega seznama,
- G – toleranca do gneče pred blagajno.

Pred vstopom v trgovino, kupec najprej preveri gnečo - prešteje ljudi, ki trenutno čakajo v vrstah pri blagajnah. Če je dolžina najkrajše vrste strogo večja od njegove tolerance do gneče, kupec takoj odide. V nasprotnem primeru vstopi v trgovino in prične z nabiranjem izdelkov. Čez S * H časovnih enot kupec zaključi z nabiranjem izdelkov in se postavi v (tem trenutku) najkrajšo čakalno vrsto. Če le-ta ni enolično določena, kupec izbere tisto z najnižjim indeksom.

Ko enkrat kupec pride do blagajne *i* (postane prvi v čakalni vrsti pri tej blagajni), potrebuje S * V_i časovnih enot za skeniranje svojih izdelkov, s čemer dokonča nakup in zapusti trgovino, s skeniranjem izdelkov pa začne naslednji v vrsti.

Vhodni parametri simulacije so:

- T – število korakov (časovnih enot) simulacije
- L_V – seznam vrednosti V_1, V_2, ..., V_B, ki predstavljajo čase skeniranja enega izdelka na posamezni blagajni (v korakih simulacije).
- L_T – seznam z zamiki prihodov kupcev (v korakih simulacije). Seznam L_T lahko vsebuje enega ali več elementov. Na primer, če velja L_T = [X, Y, Z], pomeni, da bodo novi kupci generirali v korakih X, X+Y, X+Y+Z, X+Y+Z+X, X+Y+Z+X+Y, X+Y+Z+X+Y+Z,…
- L_S – seznam s podatki o dolžinah nakupovalnih seznamov. Seznam L_S lahko vsebuje enega ali več elementov. Na primer, če velja L_S = [A,B,C], pomeni, da bo prvi kupec imel na nakupovalnem seznamu A izdelkov, drugi kupec bo imel B izdelkov, tretji kupec C izdelkov, četrti kupec A izdelkov, peti kupec B izdelkov, in tako naprej.
- L_H – seznam s podatki o tem, koliko časa kupec nabira en izdelek (v korakih simulacije). Seznam L_H lahko vsebuje enega ali več elementov. Na primer, če velja L_H = [A,B], pomeni, da bo prvi kupec potreboval A korakov simulacije za izbiro izdelka, druga kupec B korakov, tretji kupec A korakov, četrti kupec B korakov, in tako naprej.
- L_G – seznam s podatki o toleranci do gneče za generirane kupce. Seznam L_G lahko vsebuje enega ali več elementov. Na primer, če velja L_G = [A], pomeni, da bodo imeli vsi kupci toleranco nastavljeno na A.

Simulacija se izvaja po korakih. Posamezen korak se izvede v tem zaporedju:

1. Če obstaja kupec za blagajno (prvi v vrsti), ki je ravnokar končal s skeniranjem vseh izdelkov s svojega seznama, ta zapusti trgovino in naslednji v vrsti (če obstaja) takoj prične s skeniranjem svojih izdelkov.
2. Če obstaja kupec v trgovini (in ni še v čakalni vrsti), ki je ravnokar nabral vse izdelke s svojega seznama, se ta postavi v najkrajšo čakalno vrsto. Če je uvrščen kot prvi v vrsti, takoj prične s skeniranjem svojih izdelkov. **Opomba**: če v enem koraku več kupcev zaključi z nabiranjem svojih izdelkov, jih obravnavamo (razvrščamo po čakalnih vrstah) naraščajoče po njihovem ID-ju.
3. Če je nastopil čas za prihod novega kupca, ga kreiramo v skladu s podatki v L_S, L_H in L_G. Prvi kupec dobi ID=1, vsak naslednji kupec ima vrednost ID povečano za 1. Če ni prevelike gneče pri blagajnah, kupec vstopi v trgovino in začne z izbiranjem izdelkov s svojega seznama. V nasprotnem primeru odide in se ne vrača več.

Implementirajte razred **Naloga4**, ki vsebuje metodo **main**. Argumenti metode main vsebujejo poti do vhodne in izhodne datoteke (args[0] in args[1]).

Tekstovna vhodna datoteka v prvi vrstici vsebuje celoštevilčno vrednost parametra T. V drugi vrstici so zapisani celoštevilčni elementi seznama L_V, ločeni z vejico. Tretja, četrta, peta in šesta vrstica vsebujejo celoštevilčne elemente seznamov L_T, L_S , L_H ter L_G, ki so prav tako ločeni z vejicami.

V tekstovno izhodno datoteko zapišite (v kronološkem vrstnem redu) ID-je kupcev, ki so zaključili nakup na posamezni blagajni (torej, upoštevajte samo kupce, ki so izbrali in skenirali **vse izdelke** s svojega seznama). V prvi vrstici naj bodo podatki za prvo blagajno, v drugi vrstici podatki za drugo blagajno in tako naprej. V B+1 vrstici naj bodo v kronološkem vrstnem redu zapisani ID-ju kupcev, ki zaradi prevelike gneče niso vstopili v trgovino.

Primer 1:

Vhodna datoteka:	Izhodna datoteka:
5 1 1 1 1 1	1,2,3 0

Razlaga primera:

Korak	Opis dogajanja
1	• Generiranje kupca 1 (dolžina 1, hitrost 1, toleranca 1). • Kupec 1 vstopi in začne z nabiranjem izdelkov.
2	• Kupec 1 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 1). • Kupec 1 začne s skeniranjem izdelkov (blagajna 1). • Generiranje kupca 2 (dolžina 1, hitrost 1, toleranca 1). • Kupec 2 vstopi in začne z nabiranjem izdelkov.
3	• Kupec 1 dokonča skeniranje izdelkov in zapusti trgovino (blagajna 1). • Kupec 2 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 1). • Kupec 2 začne s skeniranjem izdelkov (blagajna 1). • Generiranje kupca 3 (dolžina 1, hitrost 1, toleranca 1). • Kupec 3 vstopi in začne z nabiranjem izdelkov.
4	• Kupec 2 dokonča skeniranje izdelkov in zapusti trgovino (blagajna 1). • Kupec 3 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 1). • Kupec 3 začne s skeniranjem izdelkov (blagajna 1). • Generiranje kupca 4 (dolžina 1, hitrost 1, toleranca 1). • Kupec 4 vstopi in začne z nabiranjem izdelkov.
5	• Kupec 3 dokonča skeniranje izdelkov in zapusti trgovino (blagajna 1). • Kupec 4 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 1). • Kupec 4 začne s skeniranjem izdelkov (blagajna 1). • Generiranje kupca 5 (dolžina 1, hitrost 1, toleranca 1). • Kupec 5 vstopi in začne z nabiranjem izdelkov.

Primer 2:

Vhodna datoteka:	Izhodna datoteka:
20 2,2 2,3,1 4,3 1,3 1	1,2 3 8,9

Razlaga primera:

Korak	Opis dogajanja
1	
2	• Generiranje kupca 1 (dolžina 4, hitrost 1, toleranca 1). • Kupec 1 vstopi in začne z nabiranjem izdelkov.
3	
4	
5	• Generiranje kupca 2 (dolžina 3, hitrost 3, toleranca 1). • Kupec 2 vstopi in začne z nabiranjem izdelkov.
6	• Kupec 1 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 1). • Kupec 1 začne s skeniranjem izdelkov (blagajna 1). • Generiranje kupca 3 (dolžina 4, hitrost 1, toleranca 1). • Kupec 3 vstopi in začne z nabiranjem izdelkov.
7	
8	• Generiranje kupca 4 (dolžina 3, hitrost 3, toleranca 1). • Kupec 4 vstopi in začne z nabiranjem izdelkov.
9	
10	• Kupec 3 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 2). • Kupec 3 začne s skeniranjem izdelkov (blagajna 2).
11	• Generiranje kupca 5 (dolžina 4, hitrost 1, toleranca 1). • Kupec 5 vstopi in začne z nabiranjem izdelkov.
12	• Generiranje kupca 6 (dolžina 3, hitrost 3, toleranca 1). • Kupec 6 vstopi in začne z nabiranjem izdelkov.
13	
14	• Kupec 1 dokonča skeniranje izdelkov in zapusti trgovino (blagajna 1). • Kupec 2 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 1). • Kupec 2 začne s skeniranjem izdelkov (blagajna 1). • Generiranje kupca 7 (dolžina 4, hitrost 1, toleranca 1). • Kupec 7 vstopi in začne z nabiranjem izdelkov.
15	• Kupec 5 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 1).

16	
17	• Kupec 4 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 2). • Generiranje kupca 8 (dolžina 3, hitrost 3, toleranca 1). • Kupec 8 se premisli (prevelika gneča).
18	• Kupec 3 dokonča skeniranje izdelkov in zapusti trgovino (blagajna 2). • Kupec 4 začne s skeniranjem izdelkov (blagajna 2). • Kupec 7 dokonča nabiranje izdelkov in se postavi v čakalno vrsto (blagajna 2). • Generiranje kupca 9 (dolžina 4, hitrost 1, toleranca 1). • Kupec 9 se premisli (prevelika gneča).
19	
20	• Kupec 2 dokonča skeniranje izdelkov in zapusti trgovino (blagajna 1). • Kupec 5 začne s skeniranjem izdelkov (blagajna 1). • Generiranje kupca 10 (dolžina 3, hitrost 3, toleranca 1). • Kupec 10 vstopi in začne z nabiranjem izdelkov.

Naloga 5

Na spodnji sliki je prikazan načrt skladišča.

Imamo *N* odstavnih trakov (označeni z indeksi od 1 do *N*), na vsak trak lahko shranimo *P* enako velikih predmetov. Skladišče upravljamo z robotskim vozičkom, s katerim lahko dostopamo le do prvega predmeta na posameznem traku. Trak se torej obnaša po LIFO principu: v primeru traku 1 lahko v tem trenutku dostopamo le do elementa A, do B ne moremo priti. Med predmeti imamo lahko tudi prazen prostor, kot je to v primeru tretjega traku.

Če robotski voziček stoji pred odstavnim trakom z indeksom *i*, rečemo, da je voziček na lokaciji *i*.

Skladišče upravljamo z naslednjimi ukazi:

1) LEVO; premakne voziček z lokacije *i* na lokacijo *i*-1. Če je pred to akcijo voziček na lokaciji 1, se ne premakne.

2) DESNO; premakne voziček z lokacije *i* na lokacijo *i*+1. Če je pred to akcijo voziček na lokaciji *N*, se ne premakne.

3) NALOZI; naložimo zaboj iz trenutnega traka na voziček. Če voziček ni prazen, se ne zgodi nič. Če je prvi prostor na traku prazen, se ne zgodi nič.

4) ODLOZI; odložimo zaboj iz vozička na trenutni trak. Če na traku ni prostora, zaboj ostane na vozičku.

5) GOR; trenutni trak (kjer je voziček) premakne vse zaboje za en korak gor (glej sliko). Zaboj, ki je pred to akcijo na zadnjem mestu, pade iz traku in izgine. Npr., če bi 2x izvedli ukaz GOR na traku 3, bi s tem izgubili zaboj D.

6) DOL; trak se premakne dol. Zaboj, ki je pred to akcijo na prvem mestu, pade iz traku in izgine.

Napišite program, ki sprejme začetno in končno konfiguracijo skladišča ter poišče **najkrajše zaporedje** ukazov, ki vodijo od začetne do končne konfiguracije. Robotski voziček naj **vedno začne na lokaciji 1**. Lahko predpostavite, da bo vedno možno doseči končno ureditev.

Implementirajte razred **Naloga5**, ki vsebuje metodo **main**. Metoda v argumentih prejme poti do vhodne in izhodne datoteke (args[0] in args[1]). Metoda naj prebere vhodne podatke, poišče najkrajše zaporedje ukazov, ki rešijo nalogo, in jih zapiše v izhodno datoteko.

Tekstovna vhodna datoteka je podana v naslednjem formatu:

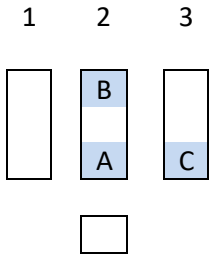
- V prvi vrstici bosta podana število trakov ($N \leq 5$) in dolžina trakov ($P \leq 10$).
- V naslednjih N vrsticah je opisana vsebina posameznih trakov začetne konfiguracije. Vsaka vrstica se začne s številko traku, za njo pride dvopičje, sledijo oznake zabojev ločene z vejico. Zaboji so označeni z eno črko ('A' do 'Z').
- V naslednjih N vrsticah je opisana vsebina posameznih trakov končne konfiguracije.

Tekstovna izhodna datoteka naj vsebuje najkrajše zaporedje ukazov, ki dosežejo zahtevano ureditev. Vsak ukaz naj bo v svoji vrstici. Pri tej nalogi je možnih več enakovrednih rešitev.

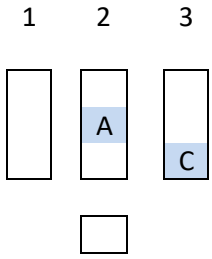
Primer:

Vhodna datoteka:	Izhodna datoteka:
3,3 1: 2:A,,B 3:C 1:C 2:A 3:	DESNO GOR DOL DESNO NALOZI LEVO LEVO ODLOZI

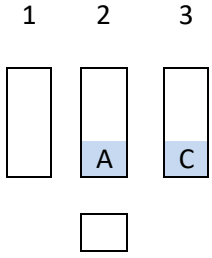
Razlaga primera:



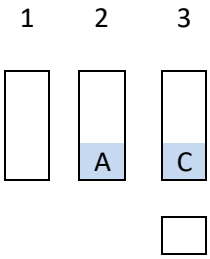
DESNO



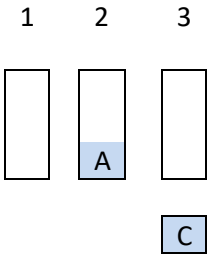
GOR



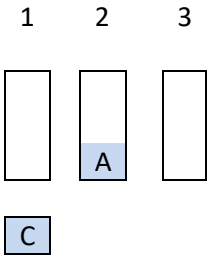
DOL



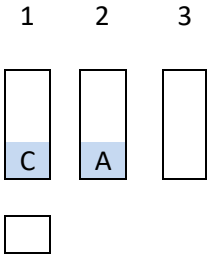
DESNO



NALOZI



LEVO
LEVO



ODLOZI