1. Napisati klasu Vozac koja sadrži sljedeća polja: ime (String), tim (String) pobjede (cio broj – broj pobjeda), treninzi (cio broj – broj pobjeda na treninzima). Napisati klasu BinTree koja opisuje binarno drvo traženja u kojem se čuvaju objekti klase Vozac. Unutar te klase, napisati privatnu ugnježdenu klasu Cvor koja opisuje jedna čvor binarnog drveta traženja, sa poljima: info (objekat klase Vozac), left, right (reference na Cvor). Binarno drvo traženja zadaje se referencom na korijen. U klasi BinTree, pored konstruktora i prirtsupnih metoda za sva polja, napisati i sljedeće metode: dodaj(Vozac v) koja u drvo dodaje vozača v (kriterijum za dodavanje je broj pobjeda), brisi(String ime) koja iz drveta briše čvor koji sadrži vozača čije je ime dato argumentom ime i stampaj(int v) koja na standardni izlaz štampa imena vozača i broj pobjeda svih vozača koji imaju najmanje v pobjeda, u opadajućem redosljedu po broju pobjeda.
2. Napisati klasu BST koja implementira binarno drvo traženja u kojem se čuvaju cijeli brojevi. Unutar te klase, napisati privatnu ugnježdenu klasu Cvor koja opisuje jedna čvor binarnog drveta traženja, sa poljima: info (cio broj), cnt (cio broj), left, right (reference na Cvor). Polje cnt nam govori koliko se puta broj info pojavljuje u drvetu. Napisati sljedeće metode u klasi BST:

* insert(int v) koji u drvo dodaje broj v,
* delete(int v) koji iz drveta briše čvor koji sadrži argument v,
* print() koja na standardni izlaz štampa sadržaj drveta u opadajućem redosljedu, tako da svaki element štampa onoliko puta koliko se pojavljuje,
* symmetric() kojom se provjerava da li je binarno stablo simetrično,
* isFull() koja provjerava da li je dato binarno stablo kompletno,
* successor(int k) koja u binarnom stablu pretraživanja pronalazi i vraća vrijednost najmanjeg elementa stabla koji je veći od broja k, a ukoliko takav element ne postoji funkcija treba da vrati -1,
* cornerNodes()koji štampa krajnji lijevi i krajnji desni čvor na svakom nivou stabla,
* longestPath() koja štampa najdužu putanju od korijena binarnog stabla do najudaljenijeg lista,
* sumOnLevel(int h) koja vraća sumu elemenata stabla koji se nalaze na visini h,
* leavesLevel() koja vraća true ako su svi listovi stabla na istoj visini, a false inače.

Napomena: na slici 1 prikazano je drvo u koje su unijeti brojevi 5, 1, 4, 5, 1, 6, 1. Npr. u prvom čvoru polje cnt je 2 jer se broj 5 pojavljuje dva puta. U njegovom lijevom sinu polje cnt je 3 jer se broj 1 pojavljuje tri puta. Kod dodavanja broja x u drvo, ako već postoji čvor sa brojem x, samo se mijenja polje cnt. Ako je polje cnt=1, tada se čvor stvarno briše iz drveta, inače se samo mijenja polje cnt. Drvo sa slike kada se odštampa: 1 1 1 4 5 5 6.

5

2

1

3

6

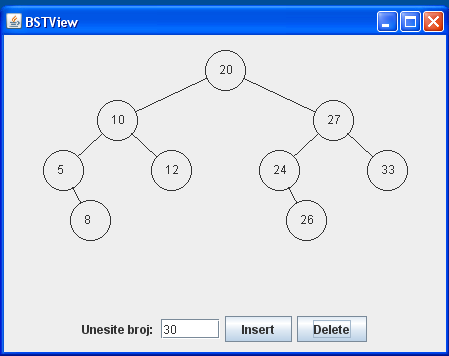
1

4

1

Slika 1

1. Napisati klasu BSTView izvedenu iz klase JFrame za dodavanje i brisanje elemenata u binarno drvo traženja i za crtanje drveta. Klikom na dugme Insert, u drvo se unosi broj iz tekstualnog polja i crta se novo drvo. Klikom na dugme Delete, iz drveta se briše broj iz tekstualnog polja i crta se novo drvo. U oba slučaja, ako broj ne postoji u drvetu, štampati poruku odgovarajuću poruku (koristite OptionPane.*showMessageDialog* ).



1. Na slici je dato ternarno drvo traženja (ternary search tree). Čvorovi binarnog i ternarnog drveta opisani su klasama BNode i Tnode (vidi sliku ispod). Referenca left pokazuje na poddrvo koje sadrži vrijednosti manje od key1; referenca middle pokazuje na poddrvo koje sadrži vrijednosti veće od key1 i manje od key2; referenca right pokazuje na poddrvo koje sadrži vrijednosti veće od key2. Ako je key2 = -1, tada ne postoji poddrvo sa vrijednostima većim od key1. Reference na null prikazane su isprekidanom linijom. Sve vrijednosti u drvetu su nenegativne i različite. Napisati metode boolean FindKey(TNode t, int value) koja vraća true ako value postoji u ternarnom drvetu i false inače, metod void PrintInOrder(TNode t) koja štampa u rastućem redosljedu vrijednosti sačuvane u ternarnom drvetu i proceduru TNode BuildTernaryTree (TNode t, Bnode b) koja od binarnog drveta traženja b kreira odgovarajuće ternarno drvo t. Na posljednjoj slici dato je binarno drvo od kojeg je nastalo ternarno drvo sa prve slike. Pravila konverzije su sljedeća:

* Broj iz korijena drveta B postaje key1 u T (broj 14)
* Ključevi u lijevom poddrvetu drveta B biće u lijevom poddrvetu drveta T (brojevi 1,5,7,8)
* Neka je R desno poddrvo u B. Korijen poddrveta R postaje key2 u T (broj 20)
* Brojevi iz desnog poddrveta drveta R biće u poddrvetu na koje pokazuje pokazivač middle drveta T (brojevi 15, 16,18, 19)
* Brojevi iz desnog poddrveta drveta R biće u poddrvetu na koje pokazuje pokazivač right drveta T (brojevi 22 i 24)
* Ova pravila se primjenjuju rekurzivno kroz čitavo drvo

class BNode {

int key;

Bnode left,right;

}

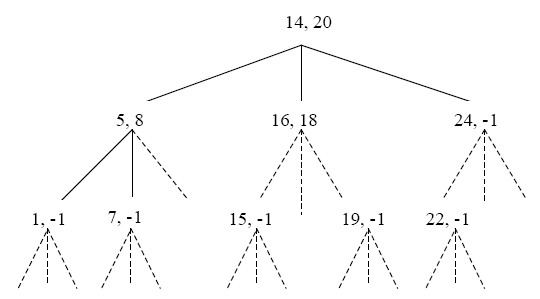
class TNode {

int key1, key2;

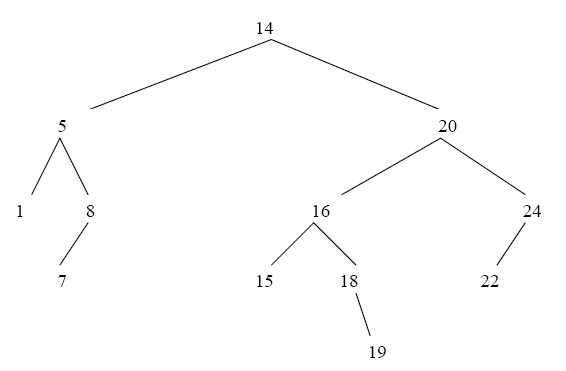
Tnode left, middle, right;

}

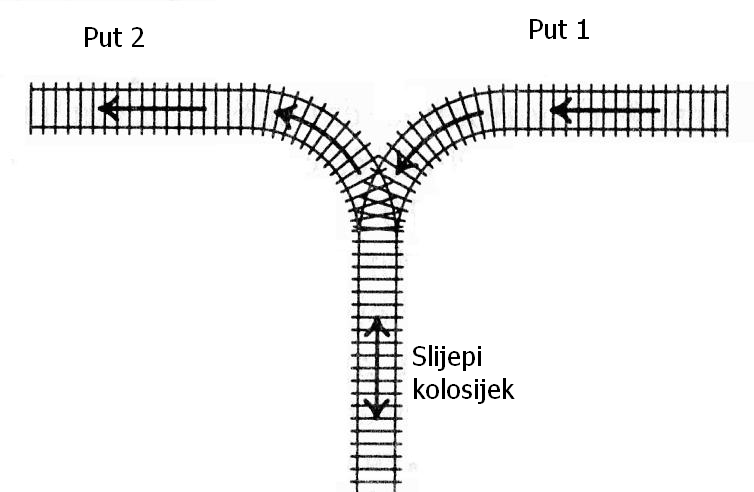
Ternarno drvo



Binarno drvo



1. Ka slijepom kolosijeku dolazi voz sa strane označene kao Put 1 (vidi sliku).



Dozvoljeno je od voza otkačiti jedan ili više vagona (moguće i cio voz) sa prednje strane voza i uvesti ih na slijepi kolosijek. Poslije toga je moguće dio vagona sa slijepog kolosijeka izvesti na put 2. Zatim dio preostalih vagona sa Puta 1 ponovo možemo uvesti na slijepi kolosijek, pa dio vagona sa kolosijeka izvesti na Put 2, itd. Nije dozvoljeno pomjerati vagone sa Puta 2 na slijepi kolosijek niti izvoziti vagone sa kolosijeka na Put 1 (vidi strelice na slici). Poznato je kojim redom dolaze vagoni. Potrebno je, pomoću opisanih operacija, reorganizovati voz tako da vagoni idu u poretku 1,2,3.... Redosljed se posmatra na Putu 2, od slijepog kolosijeka. **Ulaz**: U prvom redu unosi se N – broj vagona u vozu (1≤N≤2000). U drugom redu su dati vagoni koji dolaze po putu 1 ka slijepom kolosijeku. Vagoni su neumerisni brojevima od 1 do N. **Izlaz**: Štampati plan koji raspoređuje vagone u željeni raspored. Dopuštene su dvije operacije:

* 1 K – pomjeranje K vagona sa puta 1 na slijepi kolosijek, K≥1
* 2 K – pomjeranje K vagona sa slijepog kolosijeka na put 2, K≥1

Ako postoji više rješenja, štampati bilo koje od njih. Ako ne postoji rješenje, štampati samo broj 0.

**Primjeri:**

|  |  |  |
| --- | --- | --- |
| **ulaz**  3  3 2 1  **izlaz**  1 3  2 3 | **ulaz**  4  4 1 3 2  **izlaz**  1 2  2 1  1 2  2 3 | **ulaz**  3  2 3 1  **izlaz**  0 |

1. Napisati program koji učitava ispravan aritmetički izraz koji sadrži 4 osnovne računske operacije, prirodne brojeve i male zagrade i računa njegovu vrijednost.

Napomena: Prethodno je potrebno napisati program koji pretvara dati aritmetički izraz u ekvivalentan postfiksni izraz. Za više detalja pogledati fajl InfixToPostfixExamples.pdf

1. Napisati klasu AVL koja implementira AVL stablo. Vrijednosti koji se čuvaju u drvetu moraju implementirati interfejs Comparable. Napisati konstruktore i metode za dodavanje u AVL stablo, brisanje iz AVL stabla i obilazak stabla.
2. Napisati klasu koja implementira strukturu Heap.
3. Data je matrica dimenzija NxN popunjena brojevima 0, 1, 2 i 3 koja predstavlja mapu po kojoj se kreće robot. Polje na kojem piše broj 1 je startna pozicija robota. Polje na kojem piše broj 2 je odredište robota. Polja sa brojem 3 su polja na koja robot može da stane, dok su polja sa brojem 0 zid i robot ne može stati na njih. Robot se sa određenog polja može pomjeriti na bilo koje od osam susjednih polja ukoliko na tom polju nije upisan broj 0.

Napisati program koji provjerava da li postoji putanja između početne i odredišne pozicije robota.

U prvom redu ulaza unosi se broj N, a zatim se u svakom od sljedećih N redova unosi N elemenata matrice.

U jedinom redu izlaza treba štampati string “Postoji” ukoliko put između datih polja postoji ili string “Ne postoji” ukoliko takav put ne postoji.

Primjeri:

ULAZ: ULAZ: ULAZ:

3 4 4

0 3 2 0 3 1 0 0 3 1 0

3 3 0 3 0 3 3 0 0 3 3

1 3 0 2 3 0 3 2 0 0 3

0 3 3 3 0 3 0 3

IZLAZ: IZLAZ: IZLAZ:

“Postoji” “Postoji” “Ne postoji”

1. Data je matrica dimenzija MxN popunjena brojevima 0 i 1. Polje na kojem piše broj 0 predstavlja kopno, a polje na kojem piše broj 1 predstavlja vodu. Napisati funkciju largestLake(int [][] map) koja računa veličinu najveće vodene površine u matrici map. Veličina vodene površine je broj polja vrijednosti 1 koje ta vodena površina sadrži. Dvije vodene ćelije se smatraju povezanim, ako su susjedne horizontalno, vertikalno ili dijagonalno.

**Napomena:** Vremenska složenost rješenja će se uzimati u obzir prilikom bodovanja. Najbolje rješenje ima složenost O(M\*N).

**Primjer:**

ULAZ:

4 5

0 0 1 1 0

1 0 1 1 0

0 1 0 0 0

0 0 0 1 1

IZLAZ:

6

1. Napisati program koji učitava niz cijelih brojeva A dužine N, a zatim napisati funkciju koja za svaki element niza štampa prvi veći element koji se nalazi poslije njega u nizu. Za elemente za koje ne postoji takav element, štampati -1.

Napomena: Pokušati implementaciju optimalnijeg rješenje od onog koje koristi dvije ugnježdene for petlje. Potrebno je iskoristiti određenu strukturu podataka. Najbolje rješenje ima složenost O(N).

**Primjeri:**

ULAZ: ULAZ:

4 4

4 5 2 25 13 7 6 12

IZLAZ: IZLAZ:

5 25 25 -1 -1 12 12 -1

1. Napisati program koji u datom izrazu, koji se sastoji od otvorenih i zatvorenih malih zagrada, brojem 0 mijenja otvorene zagrade koje su pravilno zatvorene, brojem 1 mijenja zatvorene zagrade za koje postoji odgovarajuća otvorena zagrada, a brojem -1 nepravilno postavljene otvorene i zatvorene zagrade.

**Napomena:** Najbolje rješenje ima složenost O(n), gdje je n broj zagrada.

**Primjeri:**

ULAZ: ULAZ: ULAZ:

(() ()) ((())(()))))

IZLAZ: IZLAZ: IZLAZ:

-101 01-1 0001100111-1-1

1. Tri štapa, zabodena u zemlju, sastoje se iz N1, N2 i N3 djelova respektivno. Djelovi štapova se mogu odlamati kako bi se štapovi napravili jednakim. Napisati program koji računa maksimalnu moguću visinu na koju se mogu svesti sva tri štapa.

U prvom redu ulaza dati u brojevi N1, N2 i N3. U sledeća tri reda su dijelovi prvog, drugog i trećeg štapa poređani od dna prema vrhu.

**Napomena:** Vremenska složenost rješenja će se uzimati u obzir prilikom bodovanja. Najbolje rješenje ima složenost O(N1+N2+N3).

**Primjer:**

ULAZ:

5 3 4

1 1 1 2 3

2 3 4

1 4 1 1

IZLAZ:

5

1. Data su dva cijela broja M i N, napisati program koji štampa sve brojeve iz segmenta [M, N] čije sve susjedne cifre imaju apsolutnu razliku jednaku 1.

**Napomena:** Vremenska složenost rješenja će se uzimati u obzir prilikom bodovanja. Najbolje rješenje ima složenost .

**Primjeri:**

ULAZ: ULAZ:

0 21 10 12

IZLAZ: IZLAZ:

0 1 2 3 4 5 6 7 8 9 10 12 21 10 12

1. Dato je N kanapa različitih dužina. Potrebno je povezati sve kanape u jedan. Cijena povezivanja dva kanapa jednaka je sumi njihovih dužina. Napisati program koji štampa minimalnu cijenu povezivanja N kanapa.

**Primjeri:**

ULAZ:

4

4 3 2 6

IZLAZ:

29

**Napomena:**

Prvo se spoje kanapi dužina 2 i 3 da bi se dobio kanap dužine 5, zatim se spoje kanapi dužine 4 i 5 da bi se dobio kanap dužine 9. Na kraju se spoje kanapi dužina 9 i 6. Ukupna cijena je 5+9+15=29.

**ZADACI SA TAKMIČENJA:**

1. Zli Kaktus car dočepao se Čarobne bačvice i poplavio Čudesnu šumu! Slikar Paleta i tri mala ježa sada se moraju što brže vratiti do Dabrove kolibe, gdje će se skloniti od bujice.

Karta Čudesne šume se sastoji od R redova i S kolona. Na karti su prazna polja označena znakom '.', polja pokrivena vodom znakom '\*', a polja na kojima se nalazi stijena znakom 'X'. Takođe, dva su istaknuta polja, polje na kojem se nalazi Dabrova koliba (označeno znakom 'D') i polje na kojem se nalaze Slikar i tri mala ježa (označeno znakom 'S'). Slikar Paleta i tri mala ježa se u svakoj minuti mogu s polja na kojem se nalaze pomjeriti na neko od na 4 susjedna polja (gore, dolje, lijevo, desno). Svake minute se takođe i poplava proširi, i to tako da svako prazno polje koje je susjedno barem jednom već poplavljenom polju postaje poplavljeno. Ni voda, ni Slikar Paleta i tri mala ježa ne mogu proći kroz polje na kojem se nalazi stijena. Naravno, Slikar Paleta i tri mala ježa ne mogu proći kroz poplavljeno polje. Voda ne može poplaviti Dabrovu kolibu.

Napišite program koji će za unesenu kartu Čudesne šume ispisati **najkraće** vrijeme potrebno da Slikar Paleta i tri mala ježa sigurno dođu do Dabrove kolibe. **Napomena:** Slikar Paleta i tri mala ježa se ne smiju pomjeriti u polje koje će te minute biti poplavljeno. **Ulaz:** U prvom redu nalaziće se R i S, prirodni brojevi manji od ili jednaki 50. U sljedećih R redova nalaziće se po S znakova ('.', '\*', 'X', 'D' i 'S'). U cijeloj karti nalaziće se tačno jedan znak 'D' i tačno jedan znak 'S'. **Izlaz:** U jedan red treba štampati **najkraće** vrijeme potrebno da Slikar Paleta i tri mala ježa sigurno stignu do Dabrove kolibe. Ukoliko je to nemoguće potrebno je ispisati riječ “KAKTUS”.

**Primjeri:**

|  |  |  |
| --- | --- | --- |
| **ulaz**  3 3  D.\*  ...  .S.  **izlaz**  3 | **ulaz**  3 3  D.\*  ...  ..S  **izlaz**  KAKTUS | **ulaz**  3 6  D...\*.  .X.X..  ....S.  **izlaz**  6 |

1. Vuk Vjekoslav se nalazi u sumi s mnogo lovaca. Svaki od lovaca krije se iza nekog stabla i vreba na vuka. Vjekoslav to zna, ali ne zna koliko lovaca ima u sumi i iza kojih se stabala oni nalaze. Stoga on u svakom trenutku yeli biti sto dalje od najblizeg stabla. Ipak, Vjekoslav zeli nekako doci do svoje jazbine. Sumu mozemo prikazati kao polje sa N redaka i M stupaca. Oznacimo na polju znakom '.' prazno sumsko polje, znakom '+' polje na kojem se nalazi stablo, znakom 'V' polje na kojem se nalazi Vjekoslav, a znakom 'J' polje na kojem se nalazi jazbina. Vjekoslav se moze iz polja u kojem se nalazi pomaknuti na polje koje se nalazi iznad, ispod, ljevo ili desno. Vjekoslava zanima koliko se najvise mora pribliziti nekom stablu da bi dosao do svoje jazbine. Ako se Vjekoslav nalazi na polju (R1, C1), a stablo na polju (R2, C2), udaljenost Vjekoslava od tog stabla iznosi: |R1-R2| + |C1-C2| Stablo ne zauzima cijelo polje pa Vjekoslav moze proci poljem na kojem se nalazi stablo, ali je tada njegova udaljenost od stabla 0. Pomozite Vjekoslavu i napisite program koji ce izracunati koliko se u najgorem slucaju Vjekoslav mora pribliziti stablu.

ULAZNI PODACI

U prvom retku nalazi se brojevi N i M (1 <= N, M <= 500), dimenzije sume. U sljedecih N redaka nalazi se po M znakova: '.' (tocka), '+', 'V', 'J'. U ulaznim podacima ce se nalaziti tocno jedan znak 'V' i 'J' te barem jedan znak '+'.

IZLAZNI PODACI

U prvom i jedinom retku ispisa potrebno je ispisati najveci moguci cijeli broj X, takav da Vjekoslav moze doci do jazbine tako da u svakom trenutku udaljenost izmedju njega i svakog stabla iznosi najmanje X.

PRIMJERI

ulaz

4 4

+...

....

....

V..J

izlaz

3

ulaz

4 5

.....

.+++.

.+.+.

V+.J+

izlaz

0

RJESENJE

Ovaj zadatak rjesavamo primjenom algoritma pretrazivanja. Za svako polje odredimo koliko je udaljeno od najblizeg stabla. Svakom polju u kome je stablo pridruzimo 0, pa zatim sirimo dalje (red Q).

Krenimo od polja na kojem se trenutno nalazi vuk. Sva njemu susjedna polja unosimo u sortiranu strukturu PQ (Priority Queue)u kojoj polja sortiramo prema udaljenosti od stabla. U svakom trenutku posjecujemo polje koje se nalazi trenutno na najvecoj udaljenosti od stabla. Prilikom posjete nekog polja, sva polja koja se nalaze oko njega a nisu vec u strukturi PQ se dodaju u PQ. Rjesenje za neko polje je minimum sljedecih vrijednosti: udaljenost tog polja do stabla ili rjessnja za polje sa kojeg se doslo na to polje. Na ovaj nacin u O(n log n) slozenosti mozemo za svako polje izracunati trazenu informaciju.