ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Техники за креирање алгоритми - 1

Алгоритми и податочни структури

Аудиториска вежба 3

Задача 1 (brute force)

- Дадена е nxn шаховска табла (n е цел број што се чита од влез). Пресметајте на колку различни начини можат да се постават две “кралици” без да се напаѓаат една со друга. Две “кралици” се напаѓаат ако се наоѓаат во ист ред, колона или дијагонала.

Задача 1 - решение

```java
import java.util.Scanner;

public class ChessBruteForce {

    public static boolean can_attack(int i1, int j1, int i2, int j2) {
        return (i1 == i2) || (j1 == j2) || (Math.abs(i1-i2) == Math.abs(j1-j2));
    }

    public static int number_of_combinations(int n) {
        int rez = 0;

        for(int i1=0;i1<n;i1++) {
            for(int j1=0;j1<n;j1++) {
                for(int i2=0;i2<n;i2++) {
                    for(int j2=0;j2<n;j2++) {
                        if(!can_attack(i1, j1, i2, j2)) {
                            rez++;
                        }
                    }
                }
            }
         }

        return rez;
    }
```

Задача 1 - решение - main дел

```
    public static void main(String [] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Vnesete ja goleminata na shahovskata tabla: ");
        int n = input.nextInt();

        int rez = number_of_combinations(n);
        System.out.println("Brojot na nachini za postavuvanje e:" + rez);
    }

}
```

Задача 2 - Water connection problem(greedy)

- На влез најпрво ни се дадени 2 цели броеви - кои означуваат број на куќи (n) и број на цевки (p) помеѓу тие куќи во една колонија. Потоа во следните p редови од влез ни се дадени информациите за цевките: од која куќа, до која куќа и кој е дијаметарот на цевката.
- Секоја куќа има најмногу една влезна и најмногу една излезна цевка. На куќите кои што имаат само влезна цевка се става чешма, а на куќите кои шти имаат само излезна цевка се става резервоар.

Задача 2 - Water connection problem (greedy)

- На излез треба да се испечати бројот на парови од чешми и тенкови кои ќе бидат ставени (t), и потоа во следните t линии излез треба да се испечати за секој пар: на која куќа е ставен резервоарот, на која куќа е ставена чешмата, и кој е минималниот дијаметар на цевка помеѓу тие две куќи.

Задача 2 - решение

```java
import java.util.ArrayList;
import java.util.Scanner;

public class WaterConnectionGreedy {

    public static ArrayList<ArrayList<Integer>> findTanksAndTaps(int houses[][]) {
        ArrayList<ArrayList<Integer>> rez = new ArrayList<>();

        for(int i=0;i<houses.length;i++) {
            if(houses[i][3] == 0 && houses[i][2] == -1) {
                int pom = i;
                int min_d = houses[i][1];
                int next = houses[i][0];
                while (next != -1) {
                    if(houses[pom][1] < min_d) {
                        min_d = houses[pom][1];
                    }
                    houses[pom][3] = 1;
                    pom = next;
                    next = houses[next][0];
                }
                if(pom != i) {
                    houses[pom][3] = 1;
                    ArrayList<Integer> tmp = new ArrayList<>();
                    tmp.add(i);
                    tmp.add(pom);
                    tmp.add(min_d);
                    rez.add(tmp);
                }
            }
        }

        return rez;
    }
```

Задача 2 - решение - main дел

```
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int n = input.nextInt();
        int p = input.nextInt();

        int houses[][] = new int[n][4];
        for(int i=0;i<n;i++) {
            houses[i][0] = -1; // kon koja kukja ima izlezna cevka
            houses[i][1] = 0; // dijametarot na izleznata cevka
            houses[i][2] = -1; //od koja kukja ima vlezna cevka
            houses[i][3] = 0; // dali e vekje povrzana kukjata
        }

        for(int i=0;i<p;i++) {
            int h1 = input.nextInt();
            int h2 = input.nextInt();
            houses[h1][0] = h2;
            houses[h1][1] = input.nextInt();
            houses[h2][2] = h1;
        }

        ArrayList<ArrayList<Integer>> tanksAndTaps = findTanksAndTaps(houses);

        System.out.println(tanksAndTaps.size());

        for(int i=0;i<tanksAndTaps.size();i++) {
            System.out.println(tanksAndTaps.get(i).get(0) + " " + tanksAndTaps.get(i).get(1) + " " + tanksAndTaps.get(i).get(2));
        }
    }
}
```

Задача 3 - Fractional knapsack (greedy)

- Да се даде решение за Fractional knapsack проблемот со користење на greedy алгоритам (алчност според односот профит/тежина). На влез најпрво добиваме цел број - број на пакети (n). Потоа во следните n линии влез добиваме по два цели броеви - профитот и тежината на секој пакет. На крај, добиваме уште еден цел број на влез - максималниот капацитет (C). На излез треба да се испечати максималниот профит што може да се добие.

Задача 3 - решение

```
import java.util.Scanner;

public class FractionalKnapsackGreedy {

    public static void sortProfitsAndWeights(int p[], int w[]) {
        for(int i=0;i<p.length;i++) {
            for(int j=i+1;j<p.length;j++) {
                if((p[i]/(float) w[i]) < (p[j]/(float) w[j])) {
                    int tmpP = p[i];
                    int tmpW = w[i];
                    p[i] = p[j];
                    w[i] = w[j];
                    p[j] = tmpP;
                    w[j] = tmpW;
                }
            }
        }
    }
```

Задача 3 - решение

```
public static float getFractKnpMaxProfit(int p[], int w[], int C) {
    sortProfitsAndWeights(p,w);
    float profit = 0;
    for(int i=0;i<p.length;i++) {
        if(C > w[i]) {
            C -= w[i];
            profit += p[i];
        } else {
            float x = C / (float) w[i];
            profit += x*p[i];
            C = 0;
            break;
        }
    }
    return profit;
}
```

Задача 3 - решение - main дел

```
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int p[] = new int[n];
        int w[] = new int[n];
        for(int i=0;i<n;i++) {
            p[i] = input.nextInt();
            w[i] = input.nextInt();
        }
        int C = input.nextInt();

        System.out.println(getFractKnpMaxProfit(p, w, C));
    }
}
```