Problema podurilor din Königsberg

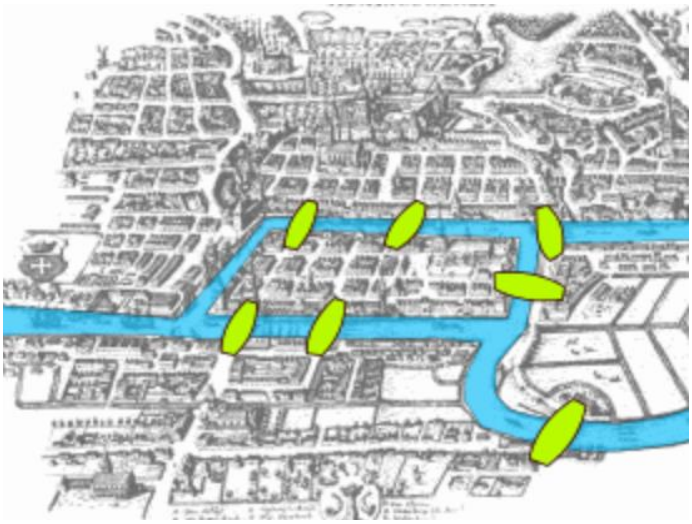
Problema podurilor din Königsberg este o problemă clasică de matematică și teorie a grafurilor, care a condus la dezvoltarea ulterioară a acestei ramuri a matematicii. Problema a fost formulată în secolul al XVIII-lea și a fost rezolvată de matematicianul elvețian Leonhard Euler în 1736.

Descriere a problemei:

Königsberg (în prezent Kaliningrad, Rusia) este situat pe râul Pregel și este traversat de două insule. Orașul era cunoscut pentru cele șapte poduri care îl conectau și pentru configurarea lor specifică.

Problema era să se determine dacă era posibil să se traverseze fiecare pod o singură dată și să se revină la punctul de plecare. Pentru a rezolva această problemă, Euler a dezvoltat un model abstract reprezentat prin grafuri.

Modelul matematic:



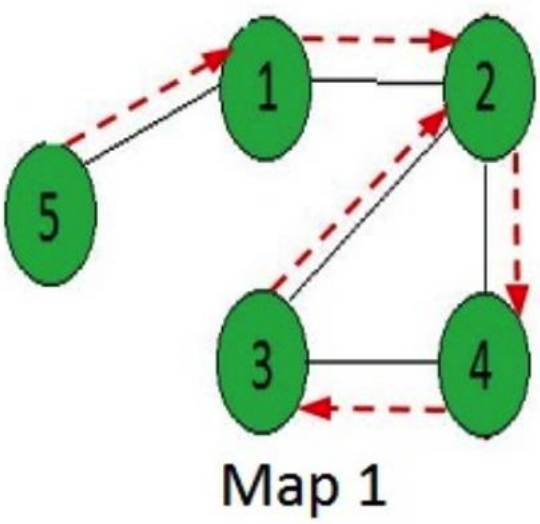
Noduri: Fiecare porțiune de pământ (inclusiv insulele și malurile) a fost reprezentată ca un nod.
Muchii: Podurile au fost reprezentate ca muchii care conectează nodurile. Fiecare pod era modelat ca o muchie între două noduri.

Soluția lui Euler:

Euler a arătat că pentru a exista un traseu care traversează fiecare pod o singură dată și revine la punctul de plecare, un graf asociat problemei trebuie să aibă anumite proprietăți. Acesta a enunțat două condiții necesare pentru ca o astfel de călătorie să fie posibilă:
Gradul nodurilor: Toate nodurile trebuie să aibă un grad par (un număr par de muchii conectate la ele).
Conectivitatea: Graficul trebuie să fie conex (adica, să existe o cale între oricare două noduri).

Concluzie:

Datorită configurării specifice a podurilor din Königsberg, era evident că nu toate nodurile aveau un grad par, ceea ce a indicat că nu există un traseu care să îndeplinească condițiile impuse. Această rezolvare a problemii a fost un pas important în dezvoltarea teoriei grafurilor și a avut un impact semnificativ în matematică și informatică.



Map 1

Examples :

```
Input : [[0, 1, 0, 0, 1],
         [1, 0, 1, 1, 0],
         [0, 1, 0, 1, 0],
         [0, 1, 1, 0, 0],
         [1, 0, 0, 0, 0]]

Output : 5 -> 1 -> 2 -> 4 -> 3 -> 2
```

```
import java.util.ArrayList;
import java.util.Arrays;

public class EulerianCircuit {

    static class Graf {
```

```
int numarNoduri;
ArrayList<ArrayList<Integer>> listaAdiacenta;

Graf(int numarNoduri) {
    this.numarNoduri = numarNoduri;
    listaAdiacenta = new ArrayList<>();

    for (int i = 0; i < numarNoduri; i++) {
        listaAdiacenta.add(new ArrayList<>());
    }
}

void adaugaMuchie(int u, int v) {
    listaAdiacenta.get(u).add(v);
    listaAdiacenta.get(v).add(u);
}

void stergeMuchie(int u, int v) {
    int indiceV = listaAdiacenta.get(u).indexOf(v);
    listaAdiacenta.get(u).set(indiceV, -1);

    int indiceU = listaAdiacenta.get(v).indexOf(u);
    listaAdiacenta.get(v).set(indiceU, -1);
}

int gaseste(ArrayList<Integer> lista, int v) {
    for (int i = 0; i < lista.size(); i++) {
        if (lista.get(i) == v) {
            return i;
        }
    }
    return -1;
}

void afiseazaCircuitEulerian() {
    int nodStart = 0;

    for (int i = 0; i < numarNoduri; i++) {
        if (listaAdiacenta.get(i).size() % 2 == 1) {
            nodStart = i;
            break;
        }
    }

    afiseazaCircuitEulerianUtil(nodStart);
    System.out.println();
}

void afiseazaCircuitEulerianUtil(int nod) {
    for (int i = 0; i < listaAdiacenta.get(nod).size(); ++i) {
        int vecin = listaAdiacenta.get(nod).get(i);
```

```
            if (vecin != -1 && esteMuchieValida(nod, vecin)) {
                System.out.print(nod + "-" + vecin + " ");
                stergeMuchie(nod, vecin);
                afiseazaCircuitEulerianUtil(vecin);
            }
        }
    }

    int numaraNoduriReachable(int nod, boolean vizitat[]) {
        vizitat[nod] = true;
        int numarNoduri = 1;

        for (int i = 0; i < listaAdiacenta.get(nod).size(); ++i) {
            int vecin = listaAdiacenta.get(nod).get(i);

            if (vecin != -1 && !vizitat[vecin]) {
                numarNoduri += numaraNoduriReachable(vecin, vizitat);
            }
        }

        return numarNoduri;
    }

    boolean esteMuchieValida(int nod, int vecin) {
        int numarVecini = 0;
        for (int i = 0; i < listaAdiacenta.get(nod).size(); ++i) {
            if (listaAdiacenta.get(nod).get(i) != -1) {
                numarVecini++;
            }
        }
        if (numarVecini == 1) {
            return true;
        }

        boolean vizitat[] = new boolean[numarNoduri];
        Arrays.fill(vizitat, false);
        int numarNoduri1 = numaraNoduriReachable(nod, vizitat);

        stergeMuchie(nod, vecin);

        Arrays.fill(vizitat, false);
        int numarNoduri2 = numaraNoduriReachable(nod, vizitat);

        adaugaMuchie(nod, vecin);

        return (numarNoduri1 > numarNoduri2) ? false : true;
    }
}

public static void main(String args[]) {
```

```
        Graf graf1 = new Graf(4);
        graf1.adaugaMuchie(0, 1);
        graf1.adaugaMuchie(0, 2);
        graf1.adaugaMuchie(1, 2);
        graf1.adaugaMuchie(2, 3);
        graf1.afiseazaCircuitEulerian();

        Graf graf3 = new Graf(4);
        graf3.adaugaMuchie(0, 1);
        graf3.adaugaMuchie(1, 0);
        graf3.adaugaMuchie(0, 2);
        graf3.adaugaMuchie(2, 0);
        graf3.adaugaMuchie(2, 3);
        graf3.adaugaMuchie(3, 1);

        graf3.afiseazaCircuitEulerian();
    }
}
```

Iesire:
2-0 0-1 1-2 2-3
1-0 0-2 2-3 3-1 1-0 0-2