Laborator 6:
Cele mai scurte drumuri. Algoritmul Floyd-Warshall

Întocmit de: Claudia Pârloagă

Îndrumător: Asist. Drd. Gabriel Danciu

I. NOŢIUNI TEORETICE

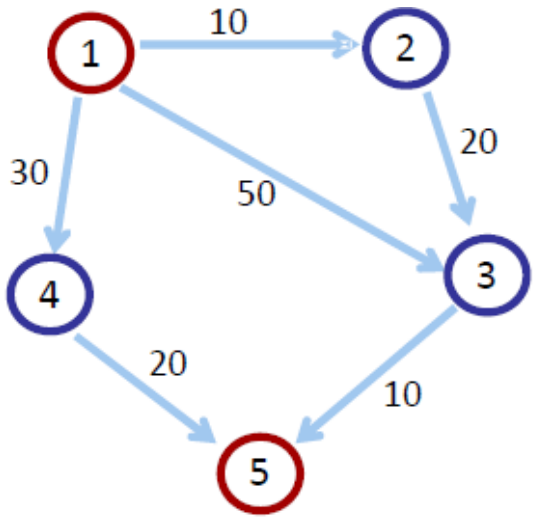
Fie un graf orientat $G = (V, E)$ cu N noduri. Se pune problema determinării, pentru orice pereche de noduri x si y, lungimea minimă a drumului de la nodul x la nodul y. Prin lungimea unui drum înţelegem suma costurilor arcelor care-l alcătuiesc.

CONCEPTE:

1. **Costul unei muchii şi al unui drum**

   Se considera funcţia $w : E \to R$ numită funcţie de cost(sau funcţia ponderilor muchiilor grafului G) care asociază fiecărei muchii o valoare numerică. Pentru perechile de noduri între care nu există muchie directă valoarea este ∞.

   Costul sau ponderea unui drum reprezintă suma tuturor costurilor muchiilor ce alcătuiesc drumul.



   În exemplul de mai sus, costul drumului de la 1 la 5 este:

   - drumul 1: (1,4),(4,5):30+20=50
   - drumul 2:(1,2),(2,3),(3,5):10+20+10=40
   - drumul 3: (1,3),(3,5):50+10=60

2. **Drumul de cost minim**

   Costul minim al al drumului dintre două noduri este minimul dintre costurile tuturor drumurilor dintre cele două noduri.

   În exemplul de mai sus, drumul de cost minim de la nodul 1 la nodul 5 este format din muchiile (1,2),(2,3),(3,5).

A. Algoritmul Floyd-Warshall

Algoritmul Floyd-Warshall determina cel mai scurt drum între oricare două noduri ale unui graf orientat. Algoritmul poate fi aplicat pe un graf ce conţine muchii cu cost negativ, dar nu se accepta cicluri de cost negativ.

Fie graful $G = (V, E)$ cu $V = \{1, 2, ..., n\}$.

NOTAŢII:

- functia cost w definită astfel:

$$w_{ij} = \begin{cases} 0 & \text{dacă i=j} \\ costulmuchiei(i,j) & \text{dacă i}\neq\text{j şi (i,j) muchie din E} \\ \infty & \text{dacă i}\neq\text{j şi (i,j) nu este muchie din E} \end{cases}$$

- π_{ij}^k: predecesorul nodului j pe drumul minim de la nodul i la j ce trece doar prin nodurile intermediare din mulţimea {1, 2, .., k}
- $d^k(i,j)$: costul celui mai scurt drum de la nodul i la nodul j

$$d_{ij}^k = \begin{cases} w_{ij} & \text{,dacă k=0, adică există cel mult o muchie de la nodul i la nodul j} \\ min\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\} & \text{,dacă } k \geq 1 \end{cases}$$

- matricea D mulţimea acestor drumuri. d_{ij}^n este distanţa dintre i şi j, atunci scopul este de a calcula D^n.
- matricea Π matricea de predecesori, unde π_{ij}^k poate fi formulat recursiv astfel:
  - dacă cel mai scurt drum dintre i şi j trece printr-un nod intermediar k atunci $\pi_{ij} = k$
  - daca nu trece prin nici un nod atunci $\pi_{ij} = NILL$

  Pentru a găsi cel mai scurt drum de la i la j consultăm π_{ij}. Dacă este NILL atunci cel mai scurt drum este doar muchia (i, j). Altfel, calculăm recursiv cel mai scurt drum de la i la π_{ij} şi cel mai scurt drum de la π_{ij} la j.

Principiul optimalităţii, pentru probleme de drum minim într-un graf orientat G:

- Dacă D_{ijp} este un drum minim de la nodul i la nodul j atunci oricare subdrum D_{klp} al său este un drum minim de la nodul k la nodul l.

Conform acestui principiu, drumul minim $D_{ijp} = (i, ..., k, ..., j)$ poate fi exprimat $D_{ijp} = D_{ikp} \cup D_{kjp}$ cu D_{ikp}, D_{kjp} drumuri minime de la nodul i la nodul k, respectiv de la nodul k la nodul j. Această exprimare duce la recurenţa exprimată mai sus: $d_{ij}^k = min\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\}, pentruk \geq 1, k, i, j \epsilon \{1, ..., n\}$

Prezentăm mai jos pseudocodul algoritmului Floyd-Warshall:

```
1
2  PROGRAM FLOYD–WARSHALL
3  BEGIN
4     FOR i:=1 TO n DO
5          FOR j:=1 TO n DO
6                    d(i,j):= w(i,j)
7                    IF i ≠ j AND d(i,j)< \infty
8                         THEN π(i,j):=i
9                         ELSE π(i,j):=NILL
10                   ENDIF;
11         ENDFOR;
12    ENDFOR;
13
14    FOR k:=1 TO n DO
15        FOR i:=1 TO n DO
16             FOR j:=1 TO n DO
17                  IF d(i,k)+ d(k,j)< d(i,j)
18                     THEN
19                        d(i,j)=d(i,k)+ d(k,j)
20                        π(i,j)= π(k,j)
21                     END;
22                ENDIF;
23             ENDFOR;
24        ENDFOR;
25    ENDFOR;
26 END.
```

II. PREZENTAREA LUCRĂRII DE LABORATOR

A. Algoritmul Floyd-Warshall

Codul de mai jos exemplifică un mod de implementare al algoritmului Floy-Warshall:

- Implementarea algoritmului:

```
1  package floyd;
2  import java.io.*;
3
4  import java.util.*;
5
6  public class FloydWarshall {
7     public static void main(String[] args) throws IOException{
8        Scanner sc=new Scanner(new FileReader("warshall2.txt"));
9        int n=sc.nextInt();
10       int[][] d=new int[n][n];
11
12       for(int i=0;i<n;i++)
13          for(int j=0;j<n;j++)
14          {
15             d[i][j]=sc.nextInt();
16             if(d[i][j]==-10)// Conventie: valoarea -10 citita din //fisier inseamna ca nu exista muchie de
                   la i la j
17                d[i][j]=Integer.MAX_VALUE;
18          }
19       sc.close();
20
21
22       //aplicarea algoritmului
23       for(int k=0;k<n;k++)
24          for(int i=0;i<n;i++)
25             for(int j=0;j<n;j++)
26                if(d[i][k]<Integer.MAX_VALUE && d[k][j]<Integer.MAX_VALUE && d[i][k]+d[k][j]<d[i][j])
27                   d[i][j]=d[i][k]+d[k][j];
28
29       //afisarea matricei d finala
30       System.out.println("Matricea D finala:");
31       System.out.println();
32       for(int i=0;i<n;i++)
33       {
34          for(int j=0;j<n;j++)
35             if(d[i][j]!=Integer.MAX_VALUE)
36                System.out.printf("%4d", d[i][j]);
37             else
38                System.out.print(" inf");
39          System.out.println();
40       }
41    }
42 }
```

- fişierul warshall2.txt ce conţine graful asupra căruia se aplică algoritmul. Din fişier se citesc numărul de noduri al grafului(prima linie) după care costurile fiecărei muchii. S-a folosit convenţia următoare: dacă nu există muchie de la nodul i la nodul j atunci costul asociat celei muchii este -10 însemnând ∞.

```
1 5
2 0 3 8 -10 -4
3 -10 0 -10 1 7
4 -10 4 0 -10 -10
5 2 -10 -5 0 -10
6 -10 -10 -10 6 0
```

III. TEMĂ

Pentru fiecare din următoarele probleme se va scrie pseudocodul şi apoi codul în Java corespunzător.

Problema 1:

Implementaţi algoritmul Dijkstra. Rulaţi acest algoritm pe graful de mai jos utilizând mai întâi nodul s ca sursă iar apoi nodul y ca sursă. Arătaţi valorile matricelor d şi π.

Problema 2:

Fie un graf ponderat, orientat $G = (V, E)$ care nu conţine cicluri negative. Modificaţi algoritmul Bellman-Ford astfel încât execuţia acestuia să se termine în $m + 1$ paşi. m este maximul peste toate perechile de noduri u, v din V dintre numarul minim de muchii într-un cel mai scurt drum de la u la v.(Cel mai scurt drum este în funcţie de pondere nu de numărul de muchii).

Problema 3:

Modificaţi algoritmul Bellman-Ford aşa încât $d(v)$ va fi setat la ∞ pentru toate nodurile v pentru care există un ciclu negativ pe vreun drum de la sursă la v.