ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

HONORIS UNITED UNIVERSITIES

RECHERCHE OPERATIONNELLE

4ème année

Ingénierie Informatique et Réseaux

Algorithme de Floyd-Warshall

REALISER PAR : CHAKOUR Imad

TUTEUR DE L'ECOLE : M. BENOUAHMANE Brahim

CAMPUS : EMSI AGDAL 2

GROUPE : G4

PERIODE DU PROJET : DU 21/11/2024 AU 12/12/2024

Résumé

Ce rapport présente l'application de l'algorithme de Floyd-Warshall, qui calcule les plus courts chemins entre toutes les paires de sommets dans un graphe orienté pondéré. Après une introduction au principe de fonctionnement de l'algorithme, nous illustrons chaque étape à l'aide d'une matrice de poids initiale et des itérations successives, permettant d'obtenir les matrices des distances minimales et des prédécesseurs. Le processus est détaillé afin de démontrer comment l'algorithme met à jour les distances et les prédécesseurs pour chaque sommet intermédiaire. Enfin, les résultats sont analysés pour mettre en évidence l'efficacité de cet algorithme et son importance pour résoudre des problèmes pratiques dans divers domaines.

Introduction Générale

Le problème des plus courts chemins dans un graphe est une question centrale en théorie des graphes et en informatique, avec des applications dans de nombreux domaines tels que les réseaux de transport, la logistique, les télécommunications et la planification. Dans le cadre de ce rapport, nous nous intéressons à l'algorithme de Floyd-Warshall, un algorithme fondamental qui permet de déterminer les distances minimales entre toutes les paires de sommets d'un graphe orienté pondéré. Cet algorithme se distingue par sa simplicité et son efficacité pour résoudre ce problème de manière exhaustive, en manipulant des matrices représentant les poids et les chemins dans le graphe.

Ce rapport vise à appliquer l'algorithme de Floyd-Warshall à un exemple concret et à en analyser les résultats pour mieux comprendre son fonctionnement et son utilité dans la résolution des problèmes liés aux graphes pondérés.

Table des matières

Chapitre I : Présentation de l'Algorithme de Floyd-Warshall

1. INTRODUCTION AU PROBLEME DES PLUS COURTS CHEMINS

Dans les graphes orientés pondérés, le problème des plus courts chemins consiste à déterminer les chemins les moins coûteux entre toutes les paires de sommets, en fonction des poids associés aux arcs.
Ce problème est fondamental dans de nombreux domaines, tels que :

- Les réseaux de transport (trouver le chemin le plus rapide entre deux villes)
- Les réseaux informatiques (optimisation des routes pour les données)
- La logistique et la planification (réduction des coûts ou des délais).

L'algorithme de Floyd-Warshall est une solution efficace pour résoudre ce problème dans les graphes orientés pondérés, pourvu qu'ils ne contiennent pas de circuits absorbants (cycles dont le poids total est négatif).

2. PRINCIPE ET FONCTIONNEMENT DE L'ALGORITHME

Cet algorithme permet de calculer l'ensemble des plus courts chemins entre toute paire de sommets dans un graphe donné orienté, pondéré et sans circuit absorbant.

Voici une synthèse du fonctionnement de l'algorithme et une explication détaillée des matrices produites :

Données d'entrée :

- **Graphe G = (S,A,w)**
  - S : Ensemble des sommets (S= {1, 2,…,n}).
  - A : Ensemble des arêtes.
  - w : Fonction de poids (w: A→R).
- **Matrice des poids initiale W** :
  - Chaque w (i, j) représente le poids de l'arête entre les sommets i et j. Si aucune arête n'existe, w (i, j) = ∞.

Résultats :

- **Matrice D des plus courts chemins :**
  - d{i, j} est la distance minimale entre i et j.
- **Matrice π des prédécesseurs :**
  - π i,j indique le prédécesseur immédiat de j sur le plus court chemin partant de i.

3. DEROULEMENT DE L'ALGORITHME

Cet algorithme calcule les matrices $D^{(k)}$:

$$D^{(0)} = W^{(0)} = (w(\{i,j\}))_{1\leq i,j\leq n} : \text{matrice des poids}$$
$$D^{(k)} = (d_{i,j}^{(k)})_{1\leq i,j\leq n} \quad \text{où : } d_{i,j}^{(k)} = \min(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)})$$

En même temps, il est possible de calculer les matrices $\Pi^{(k)}$:

— Pour $k = 0$: $\pi_{i,j}^{(0)} = \begin{cases} N \text{ si } i = j \text{ ou } w(\{i,j\}) = \infty \\ i \text{ si } i \neq j \text{ et } w(\{i,j\}) < \infty \end{cases}$

— Pour $k \geq 1$: $\pi_{i,j}^{(k)} = \begin{cases} \pi_{i,j}^{(k-1)} \text{ si } d_{i,j}^{(k-1)} \leq d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \\ \pi_{k,j}^{(k-1)} \text{ si } d_{i,j}^{(k-1)} > d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \end{cases}$

Après n itérations (k = n), les matrices finales D et π contiennent respectivement les distances minimales et les prédécesseurs pour tous les couples de sommets.

4. CONCLUSION

L'algorithme de Floyd-Warshall offre une solution élégante et systématique au problème des plus courts chemins pour toutes les paires de sommets dans un graphe orienté pondéré. Grâce à l'utilisation de matrices et à une relation de récurrence, il garantit des résultats précis.

Cet algorithme est particulièrement utile dans des contextes où une analyse globale des chemins est nécessaire.

Chapitre II : Application Pratique et Résultats

1. DESCRIPTION DU GRAPHE

Données d'entrée :

Le graphe G = (S, A, w)

- S : Ensemble des sommets avec S= {A, B, C, D, E}.

- A : Ensemble des arêtes avec :

A = {(A,C),(B,C),(A,B),(B,A),(E,B),(E,A),(A,D),(D,E),(C,D)(C,E)}

- w : Fonction de poids (w : A→R).

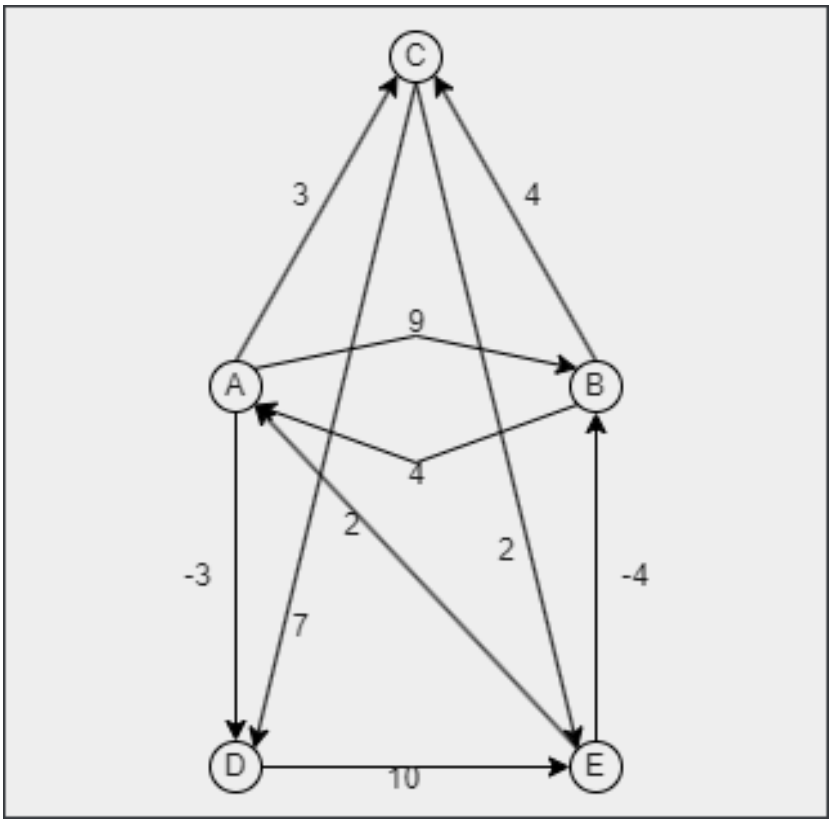

Figure 1: Graphe dessiner avec le logiciel draw.io

2. LA MATRICE DES POIDS DU GRAPHE

$$W=\begin{pmatrix} 0 & 9 & 3 & -3 & \infty \\ 4 & 0 & 4 & \infty & \infty \\ \infty & \infty & 0 & 7 & 2 \\ \infty & \infty & \infty & 0 & 10 \\ 2 & -4 & \infty & \infty & 0 \end{pmatrix}$$

3. DEROULEMENT DE L'ALGORITHME

$$D^0 = \begin{pmatrix} 0 & 9 & 3 & -3 & \infty \\ 4 & 0 & 4 & \infty & \infty \\ \infty & \infty & 0 & 7 & 2 \\ \infty & \infty & \infty & 0 & 10 \\ 2 & -4 & \infty & \infty & 0 \end{pmatrix}, \quad \pi^0 = \begin{pmatrix} N & 1 & 1 & 1 & N \\ 2 & N & 2 & N & N \\ N & N & N & 3 & 3 \\ N & N & N & N & 4 \\ 5 & 5 & N & N & N \end{pmatrix}$$

$d^{(1)}_{B,C} = \min(d^{(1-1)}_{B,C}, d^{(1-1)}_{B,A} + d^{(1-1)}_{A,C}) = \min(4, 4+3) = 4$
$d^{(1)}_{B,D} = \min(d^{(1-1)}_{B,D}, d^{(1-1)}_{B,A} + d^{(1-1)}_{A,D}) = \min(\infty, 4-3) = 1$
$d^{(1)}_{B,E} = \min(d^{(1-1)}_{B,E}, d^{(1-1)}_{B,A} + d^{(1-1)}_{A,E}) = \min(\infty, 4+\infty) = \infty$
$d^{(1)}_{C,B} = \min(d^{(1-1)}_{C,B}, d^{(1-1)}_{C,A} + d^{(1-1)}_{A,B}) = \min(\infty, \infty+9) = \infty$
$d^{(1)}_{C,D} = \min(d^{(1-1)}_{C,D}, d^{(1-1)}_{C,A} + d^{(1-1)}_{A,D}) = \min(7, \infty-3) = 7$
$d^{(1)}_{C,E} = \min(d^{(1-1)}_{C,E}, d^{(1-1)}_{C,A} + d^{(1-1)}_{A,E}) = \min(2, \infty+\infty) = 2$
$d^{(1)}_{D,B} = \min(d^{(1-1)}_{D,B}, d^{(1-1)}_{D,A} + d^{(1-1)}_{A,B}) = \min(\infty, \infty + 9) = \infty$
$d^{(1)}_{D,C} = \min(d^{(1-1)}_{D,C}, d^{(1-1)}_{D,A} + d^{(1-1)}_{A,C}) = \min(\infty, \infty + 3) = \infty$
$d^{(1)}_{D,E} = \min(d^{(1-1)}_{D,E}, d^{(1-1)}_{D,A} + d^{(1-1)}_{A,E}) = \min(10, \infty + \infty) = 10$
$d^{(1)}_{E,B} = \min(d^{(1-1)}_{E,B}, d^{(1-1)}_{E,A} + d^{(1-1)}_{A,B}) = \min(-4, 2+9) = -4$
$d^{(1)}_{E,C} = \min(d^{(1-1)}_{E,C}, d^{(1-1)}_{E,A} + d^{(1-1)}_{A,C}) = \min(\infty, 2 + 3) = 5$
$d^{(1)}_{E,D} = \min(d^{(1-1)}_{E,D}, d^{(1-1)}_{E,A} + d^{(1-1)}_{A,D}) = \min(\infty, 2 - 3) = -1$

$$D^1 = \begin{pmatrix} 0 & 9 & 3 & -3 & \infty \\ 4 & 0 & 4 & 1 & \infty \\ \infty & \infty & 0 & 7 & 2 \\ \infty & \infty & \infty & 0 & 10 \\ 2 & -4 & 5 & -1 & 0 \end{pmatrix}, \quad \pi^1 = \begin{pmatrix} N & 1 & 1 & 1 & N \\ 2 & N & 2 & 1 & N \\ N & N & N & 3 & 3 \\ N & N & N & N & 4 \\ 5 & 5 & 1 & 1 & N \end{pmatrix}$$

$d^{(2)}_{A,C} = \min(d^{(1)}_{A,C}, d^{(1)}_{A,B} + d^{(1)}_{B,C}) = \min(3, 9+4) = 3$
$d^{(2)}_{A,D} = \min(d^{(1)}_{A,D}, d^{(1)}_{A,B} + d^{(1)}_{B,D}) = \min(-3, 9+1) = -3$
$d^{(2)}_{A,E} = \min(d^{(1)}_{A,E}, d^{(1)}_{A,B} + d^{(1)}_{B,E}) = \min(\infty, 9 + \infty) = \infty$
$d^{(2)}_{C,A} = \min(d^{(1)}_{C,A}, d^{(1)}_{C,B} + d^{(1)}_{B,A}) = \min(\infty, \infty+4) = \infty$
$d^{(2)}_{C,D} = \min(d^{(1)}_{C,D}, d^{(1)}_{C,B} + d^{(1)}_{B,D}) = \min(7, \infty+1) = 7$
$d^{(2)}_{C,E} = \min(d^{(1)}_{C,E}, d^{(1)}_{C,B} + d^{(1)}_{B,E}) = \min(2, \infty+\infty) = 2$
$d^{(2)}_{D,A} = \min(d^{(1)}_{D,A}, d^{(1)}_{D,B} + d^{(1)}_{B,A}) = \min(\infty, \infty + 4) = \infty$
$d^{(2)}_{D,C} = \min(d^{(1)}_{D,C}, d^{(1)}_{D,B} + d^{(1)}_{B,C}) = \min(\infty, \infty + 4) = \infty$
$d^{(2)}_{D,E} = \min(d^{(1)}_{D,E}, d^{(1)}_{D,B} + d^{(1)}_{B,E}) = \min(10, \infty + \infty) = 10$
$d^{(2)}_{E,A} = \min(d^{(1)}_{E,A}, d^{(1)}_{E,B} + d^{(1)}_{B,A}) = \min(2, -4+4) = 0$
$d^{(2)}_{E,C} = \min(d^{(1)}_{E,C}, d^{(1)}_{E,B} + d^{(1)}_{B,C}) = \min(5, -4 + 4) = 0$
$d^{(2)}_{E,D} = \min(d^{(1)}_{E,D}, d^{(1)}_{E,B} + d^{(1)}_{B,D}) = \min(-1, -4 + 1) = -3$

$$D^2 = \begin{pmatrix} 0 & 9 & 3 & -3 & \infty \\ 4 & 0 & 4 & 1 & \infty \\ \infty & \infty & 0 & 7 & 2 \\ \infty & \infty & \infty & 0 & 10 \\ 0 & -4 & 0 & -3 & 0 \end{pmatrix}, \quad \pi^2 = \begin{pmatrix} N & 1 & 1 & 1 & N \\ 2 & N & 2 & 1 & N \\ N & N & N & 3 & 3 \\ N & N & N & N & 4 \\ N & 5 & N & 1 & N \end{pmatrix}$$

$d^{(3)}_{A,B} = \min(d^{(2)}_{A,B}, d^{(2)}_{A,C} + d^{(2)}_{C,B}) = \min(9,3+\infty) = 9$
$d^{(3)}_{A,D} = \min(d^{(2)}_{A,D}, d^{(2)}_{A,C} + d^{(2)}_{C,D}) = \min(-3,3+7) = -3$
$d^{(3)}_{A,E} = \min(d^{(2)}_{A,E}, d^{(2)}_{A,C} + d^{(2)}_{C,E}) = \min(\infty,3+2) = 5$
$d^{(3)}_{B,A} = \min(d^{(2)}_{B,A}, d^{(2)}_{B,C} + d^{(2)}_{C,A}) = \min(4,4+\infty) = 4$
$d^{(3)}_{B,D} = \min(d^{(2)}_{B,D}, d^{(2)}_{B,C} + d^{(2)}_{C,D}) = \min(1,4+7) = 1$
$d^{(3)}_{B,E} = \min(d^{(2)}_{B,E}, d^{(2)}_{B,C} + d^{(2)}_{C,E}) = \min(\infty,4+2) = 6$
$d^{(3)}_{D,A} = \min(d^{(2)}_{D,A}, d^{(2)}_{D,C} + d^{(2)}_{C,A}) = \min(\infty,\infty+\infty) = \infty$
$d^{(3)}_{D,B} = \min(d^{(2)}_{D,B}, d^{(2)}_{D,C} + d^{(2)}_{C,B}) = \min(\infty, \infty + \infty) = \infty$
$d^{(3)}_{D,E} = \min(d^{(2)}_{D,E}, d^{(2)}_{D,C} + d^{(2)}_{C,E}) = \min(10, \infty + 2) = 10$
$d^{(3)}_{E,A} = \min(d^{(2)}_{E,A}, d^{(2)}_{E,C} + d^{(2)}_{C,A}) = \min(0,0+\infty) = 0$
$d^{(3)}_{E,B} = \min(d^{(2)}_{E,B}, d^{(2)}_{E,C} + d^{(2)}_{C,B}) = \min(-4,0+\infty) = -4$
$d^{(3)}_{E,D} = \min(d^{(2)}_{E,D}, d^{(2)}_{E,C} + d^{(2)}_{C,D}) = \min(-3, 0+7) = -3$

$$D^3 = \begin{pmatrix} 0 & 9 & 3 & -3 & 5 \\ 4 & 0 & 4 & 1 & 6 \\ \infty & \infty & 0 & 7 & 2 \\ \infty & \infty & \infty & 0 & 10 \\ 0 & -4 & 0 & -3 & 0 \end{pmatrix}, \qquad \pi^3 = \begin{pmatrix} N & 1 & 1 & 1 & 3 \\ 2 & N & 2 & 1 & 3 \\ N & N & N & 3 & 3 \\ N & N & N & N & 4 \\ N & 5 & N & 1 & N \end{pmatrix}$$

$d^{(4)}_{A,B} = \min(d^{(3)}_{A,B}, d^{(3)}_{A,D} + d^{(3)}_{D,B}) = \min(9,-3+\infty) = 9$
$d^{(4)}_{A,C} = \min(d^{(3)}_{A,C}, d^{(3)}_{A,D} + d^{(3)}_{D,C}) = \min(3,-3+\infty) = 3$
$d^{(4)}_{A,E} = \min(d^{(3)}_{A,E}, d^{(3)}_{A,D} + d^{(3)}_{D,E}) = \min(5,-3+10) = 5$
$d^{(4)}_{B,A} = \min(d^{(3)}_{B,A}, d^{(3)}_{B,D} + d^{(3)}_{D,A}) = \min(4,1+\infty) = 4$
$d^{(4)}_{B,C} = \min(d^{(3)}_{B,C}, d^{(3)}_{B,D} + d^{(3)}_{D,C}) = \min(4,1+\infty) = 4$
$d^{(4)}_{B,E} = \min(d^{(3)}_{B,E}, d^{(3)}_{B,D} + d^{(3)}_{D,E}) = \min(6,1+10) = 6$
$d^{(4)}_{C,A} = \min(d^{(3)}_{C,A}, d^{(3)}_{C,D} + d^{(3)}_{D,A}) = \min(\infty,7+\infty) = \infty$
$d^{(4)}_{C,B} = \min(d^{(3)}_{C,B}, d^{(3)}_{C,D} + d^{(3)}_{D,B}) = \min(\infty, 7 + \infty) = \infty$
$d^{(4)}_{C,E} = \min(d^{(3)}_{C,E}, d^{(3)}_{C,D} + d^{(3)}_{D,E}) = \min(2,7+10) = 2$
$d^{(4)}_{E,A} = \min(d^{(3)}_{E,A}, d^{(3)}_{E,D} + d^{(3)}_{D,A}) = \min(0,-3+\infty) = 0$
$d^{(4)}_{E,B} = \min(d^{(3)}_{E,B}, d^{(3)}_{E,D} + d^{(3)}_{D,B}) = \min(-4,-3+\infty) = -4$
$d^{(4)}_{E,C} = \min(d^{(3)}_{E,C}, d^{(3)}_{E,D} + d^{(3)}_{D,C}) = \min(0, -3+\infty) = 0$

$$D^4 = \begin{pmatrix} 0 & 9 & 3 & -3 & 5 \\ 4 & 0 & 4 & 1 & 6 \\ \infty & \infty & 0 & 7 & 2 \\ \infty & \infty & \infty & 0 & 10 \\ 0 & -4 & 0 & -3 & 0 \end{pmatrix}, \qquad \pi^4 = \begin{pmatrix} N & 1 & 1 & 1 & 3 \\ 2 & N & 2 & 1 & 3 \\ N & N & N & 3 & 3 \\ N & N & N & N & 4 \\ N & 5 & N & 1 & N \end{pmatrix}$$

$d^{(5)}_{A,B} = \min(d^{(4)}_{A,B}, d^{(4)}_{A,E} + d^{(4)}_{E,B}) = \min(9,5-4) = 1$
$d^{(5)}_{A,C} = \min(d^{(4)}_{A,C}, d^{(4)}_{A,E} + d^{(4)}_{E,C}) = \min(3,5+0) = 3$
$d^{(5)}_{A,D} = \min(d^{(4)}_{A,D}, d^{(4)}_{A,E} + d^{(4)}_{E,D}) = \min(-3,5-3) = -3$
$d^{(5)}_{B,A} = \min(d^{(4)}_{B,A}, d^{(4)}_{B,E} + d^{(4)}_{E,A}) = \min(4,6+0) = 4$
$d^{(5)}_{B,C} = \min(d^{(4)}_{B,C}, d^{(4)}_{B,E} + d^{(4)}_{E,C}) = \min(4,6+0) = 4$
$d^{(5)}_{B,D} = \min(d^{(4)}_{B,D}, d^{(4)}_{B,E} + d^{(4)}_{E,D}) = \min(1,6-3) = 1$
$d^{(5)}_{C,A} = \min(d^{(4)}_{C,A}, d^{(4)}_{C,E} + d^{(4)}_{E,A}) = \min(\infty,2+0) = 2$
$d^{(5)}_{C,B} = \min(d^{(4)}_{C,B}, d^{(4)}_{C,E} + d^{(4)}_{E,B}) = \min(\infty, 2-4) = -2$
$d^{(5)}_{C,D} = \min(d^{(4)}_{C,D}, d^{(4)}_{C,E} + d^{(4)}_{E,D}) = \min(7,2-3) = -1$
$d^{(5)}_{D,A} = \min(d^{(4)}_{D,A}, d^{(4)}_{D,E} + d^{(4)}_{E,A}) = \min(\infty,10+0) = 10$
$d^{(5)}_{D,B} = \min(d^{(4)}_{D,B}, d^{(4)}_{D,E} + d^{(4)}_{E,B}) = \min(\infty,10-4) = 6$
$d^{(5)}_{D,C} = \min(d^{(4)}_{D,C}, d^{(4)}_{D,E} + d^{(4)}_{E,C}) = \min(\infty, 10+0) = 10$

$$D^5 = \begin{pmatrix} 0 & 1 & 3 & -3 & 5 \\ 4 & 0 & 4 & 1 & 6 \\ 2 & -2 & 0 & -1 & 2 \\ 10 & 6 & 10 & 0 & 10 \\ 0 & -4 & 0 & -3 & 0 \end{pmatrix}, \qquad \pi^5 = \begin{pmatrix} N & 5 & 1 & 1 & 3 \\ 2 & N & 2 & 1 & 3 \\ N & 5 & N & 1 & 3 \\ N & 5 & N & N & 4 \\ N & 5 & N & 1 & N \end{pmatrix}$$

$$D = D^{(5)}, \Pi = \Pi^{(5)}$$

FIN DE L'ALGORITHME

Interprétation des résultats (java)

En exécutant le programme avec la matrice d'entrée, La matrice des plus courts chemins obtenue est :

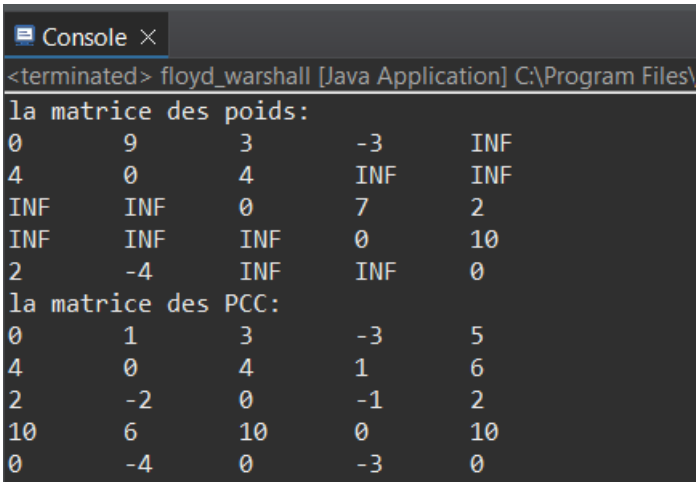

Figure 2: Resultat obtenue avec le programme en java

4. LES PLUS COURTS CHEMINS (PCC)

Sommets	Valeurs du PCC	Le PCC
A (initial)	0	-
B	1	A-C-E-B
C	3	A-C
D	-3	A-D
E	5	A-C-E

Sommets	Valeurs du PCC	Le PCC
A	4	B-A
B (initial)	0	-
C	4	B-C
D	1	B-A-D
E	6	B-C-E

Sommets	Valeurs du PCC	Le PCC
A	2	C-E-B-A
B	-2	C-E-B
C (initial)	0	-
D	-1	C-E-A-D
E	2	C-E

Sommets	Valeurs du PCC	Le PCC
A	10	D-E-B-A
B	6	D-E-B
C	10	D-E-B-C
D (initial)	0	-
E	10	D-E

Sommets	Valeurs du PCC	Le PCC
A	0	E-B-A
B	-4	E-B
C	0	E-B-C
D	-3	E-B-A-D
E (initial)	0	-

5. IMPLEMENTATION DE L'ALGORITHME EN JAVA

```
package tp1;

import java.util.Arrays;

public class floyd_warshall {

    public static void floyds(int[][] graph) {
        int n = graph.length;
        int INF = Integer.MAX_VALUE;
        for (int k = 0; k < n; k++) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    if (graph[i][k] != INF && graph[k][j] != INF &&
graph[i][k] + graph[k][j] < graph[i][j]) {
                        graph[i][j] = graph[i][k] + graph[k][j];
                    }
                }
            }
        }
        System.out.println("la matrice des PCC:");
        printMatrix(graph);
    }

    public static void printMatrix(int[][] matrix) {
        int n = matrix.length;
        int INF = Integer.MAX_VALUE;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (matrix[i][j] == INF) {
                    System.out.print("INF\t");
                } else {
                    System.out.print(matrix[i][j] + "\t");
                }
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] graph = new int[5][5];
        int INF = Integer.MAX_VALUE;

        for (int i = 0; i < 5; i++) {
            Arrays.fill(graph[i], INF);
            graph[i][i] = 0;
        }
        graph[0][1] = 9;
        graph[0][2] = 3;
        graph[0][3] = -3;
        graph[1][0] = 4;
        graph[1][2] = 4;
        graph[2][3] = 7;
        graph[2][4] = 2;
        graph[3][4] = 10;
        graph[4][0] = 2;
        graph[4][1] = -4;

        System.out.println("la matrice des poids:");
        printMatrix(graph);

        floyds(graph);
    }
}
```

Figure 3: Code source java

CONCLUSION GENERALE

En conclusion, l'algorithme de Floyd-Warshall est une méthode puissante et polyvalente pour résoudre le problème des plus courts chemins entre toutes les paires de sommets dans un graphe orienté et pondéré. Il se distingue par sa capacité à gérer des poids négatifs, ce qui le rend adapté à de nombreux scénarios complexes. Cependant, sa sensibilité aux cycles absorbants impose une vigilance particulière dans l'analyse des graphes avant son application. Ce travail nous a permis d'approfondir notre compréhension des concepts fondamentaux des graphes, tout en mettant en pratique des notions clés de la programmation et de l'optimisation algorithmique.

BIBLIOGRAPHIE

- **Tutorialspoint :**

  https://www.tutorialspoint.com/data_structures_algorithms/floyd_warshall_algorithm.htm

- **Programiz :**

  https://www.programiz.com/dsa/floyd-warshall-algorithm

Le code source complet de l'algorithme de Floyd-Warshall, ainsi que les autres parties du projet, est disponible sur le dépôt GitHub :
https://github.com/imad-chakour/floyd-warshall.git