Chapitre 2 : Recherche de plus court chemin

Cours Graphes & Applications

Unité Pédagogiques de Mathématiques,
Ecole Supérieure Privée d'Ingénierie et de Technologies (ESPRIT)

Année Universitaire : 2024-2025

Plan

A l'issu du chapitre 2, l'étudiant sera capable de :

- Identifier des problèmes dont les solutions sont données via une recherche du plus court chemin.
- Modéliser des situations réelles par des graphes.
- Application de l'algorithme de Dijkstra pour la recherche d'un plus court chemin.
- Connaître les limitations de l'algorithme de DIJKSTRA.
- Appliquer l'algorithme de FORD BELLMAN pour la recherche d'un plus court chemin dans un graphe à poids mixtes.
- Appliquer l'algorithme de FLOYD pour la recherche de plus courts chemins entre les différents couples de sommets dans un graphe à poids mixtes.
- Savoir comparer les algorithmes sur la base de leurs complexités algorithmiques.

Plan

Exemple 1:

- Beaucoup de problèmes de la vie quotidienne peuvent être représentés sous forme de **graphes** · · ·
- Le calcul de distance (et donc un **plus court chemin**) en est un des plus courants:

Google's map : le plus rapide / court chemin vers la destination

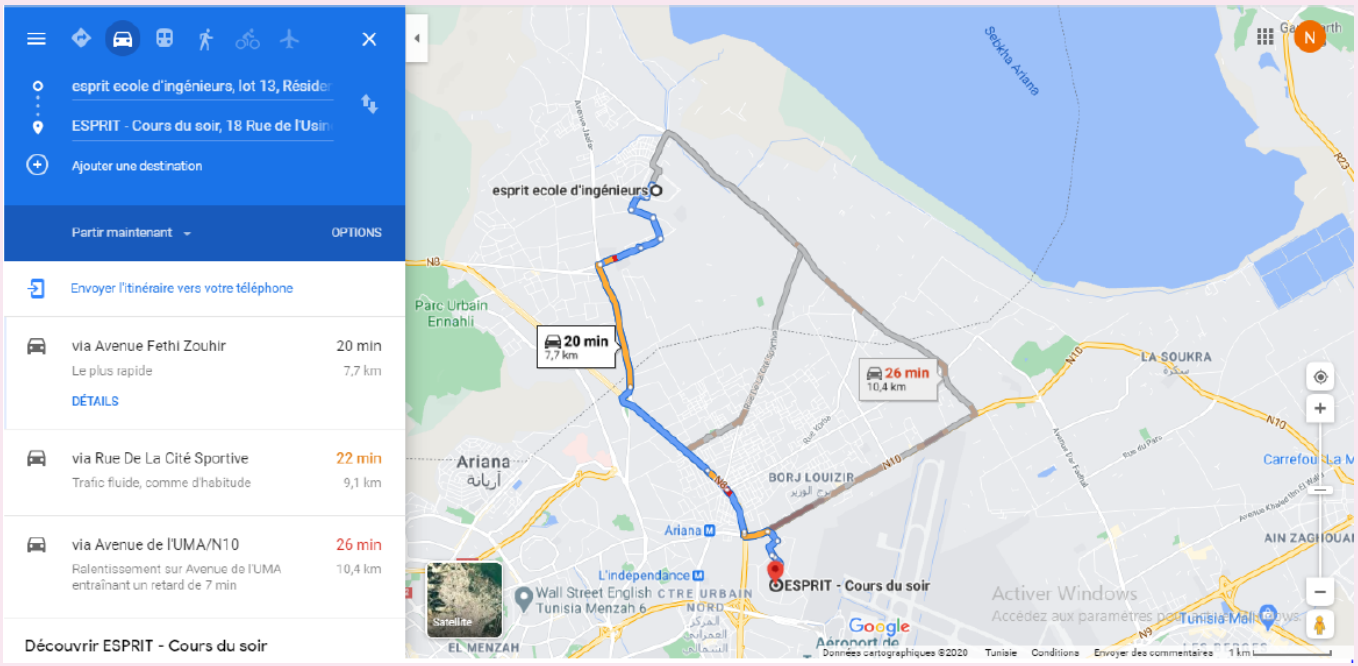


Figure: Itinéraire de ESPRIT Ghazela à ESPRIT Charguia

esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

Exemple 2:

Le routage : acheminement des paquets par le plus court chemin vers la destination

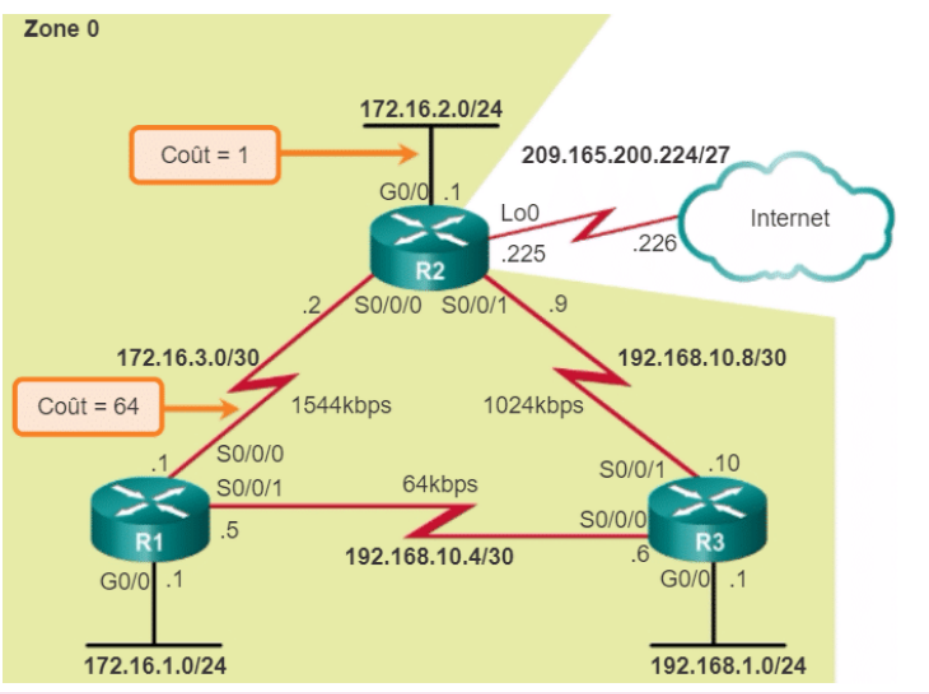


Figure: Illustration d'un réseaux de communication (source de l'image : internet)

Outline

Définition 1 (**Graphe pondéré (ou valué)**)

On appelle **graphe pondéré** un graphe $G = (V, E)$ muni d'une application $p : E \to X$ où X est l'ensemble des valeurs (la plupart du temps $\mathbb{R}$).

Cela revient à affecter un **"poids"** à chaque arête (ou arcs).

Définition 2

Soit $G = (V, E)$ un graphe pondéré. Soit $v_0, v_1, \cdots, v_n$ un chemin du graphe, **la longueur du chemin** est la somme des poids des arêtes (ou arcs) qui constituent le chemin.

Soit $G = (V, E)$ un graphe un graphe orienté.

Définition 3 (**Longueur du chemin**)

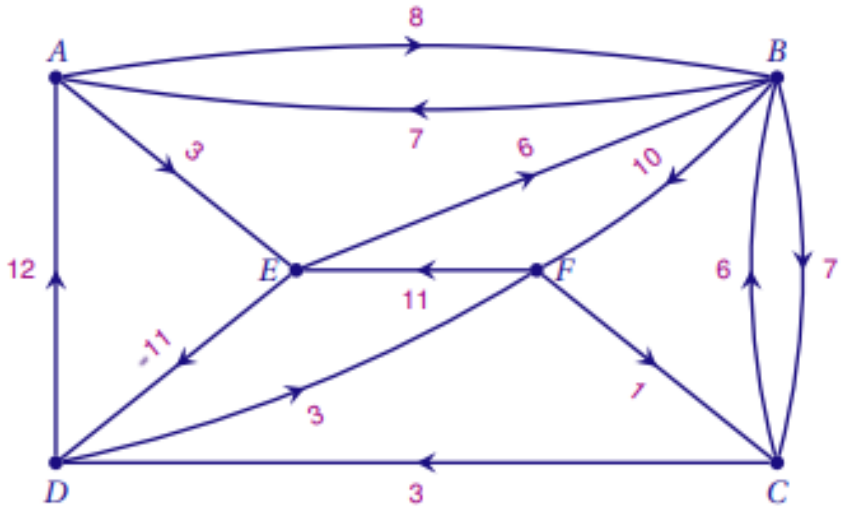
Soit $m =< v_1, v_2, \cdots, v_k >$ un chemin reliant k sommets de $G = (V, E)$ avec $|V| = n$:

$$v_i \in V, 1 \leq i \leq k, k \leq n.$$

La **longueur du chemin** m, notée $l(m)$, correspond à la **somme des poids** (ou encore **les coûts**) associés aux arcs qui le composent:

$$l(m) = \sum_{e \in m} l(e) = \sum_{i=1}^{k-1} p(v_i, v_{i+1}).$$

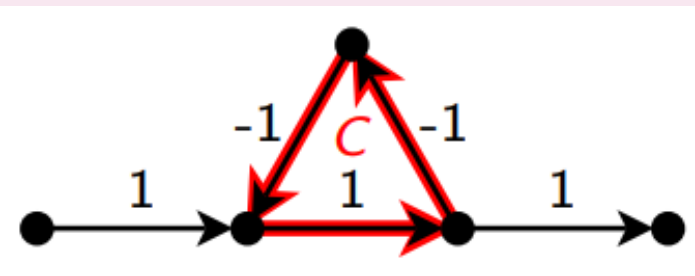
Exemple:



- la longueur du chemin $m_1 =< A, E, B, F >$ est $l(m_1) = p(A,E) + p(E,B) + p(B,F) = 3 + 6 + 10 = 19$.
- la longueur du chemin $m_2 =< A, E, B, C, F, D >$ est $l(m_2) = p(A,E)+p(E,B)+p(B,C)+p(C,F)+p(F,D) = 3+6+7-1-3 = 12$.

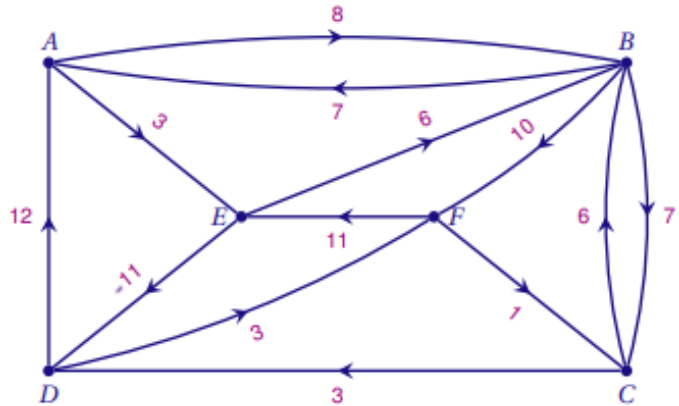
Définition 4 (**Distance/ Circuit absorbant**)

- La **distance** entre deux sommets v_i et v_j avec $1 \leq i,j \leq n$ de V notée $\delta(v_i, v_j)$ est la plus petite longueur de tous les chemins qui les relie.
- $\delta(v_i, v_j) = +\infty$ S'il n'existe pas de chemin entre v_i et v_j ,
- $\delta(v_i, v_j) = -\infty$ S'il existe un chemin entre deux sommets v_i et v_j contenant un circuit de coût total négatif.
- Un **circuit absorbant** est un circuit dont le coût total est négatif.



Remarque: Une condition nécessaire d'existence de plus court chemin est l'absence de circuit absorbant.

Exemple:



- la distance entre A et B est $\delta(A,B) = 8$
- la distance entre D et C est $\delta(D,C) = \delta(D,F) + \delta(F,C) = 3 + 1 = 4$.
- la distance entre A et D est $\delta(A,D) = -\infty$: : il n'existe pas de plus court chemin entre A et D (existence de circuit absorbant).

Propriété des sous-chemins optimaux

Tout sous-chemin d'un plus court chemin est un plus court chemin.

Définition 5 (**Problème du plus court chemin (PCC)**)

Le **problème de plus court chemin** entre deux sommets i et j consiste à déterminer un chemin de i à j de longueur minimale.

Remarque: Avec un graphe de taille importante, ceci risque de devenir rapidement impossible. Pour résoudre ce problème, on fait appel à des algorithmes.

Outline

Quelques algorithmes de recherche de PCC:

Les algorithmes de résolution du problème de recherche du plus court chemin seront différents suivant:

- les propriétés du graphe:
  - graphe valué avec des poids positives,
  - graphe valué avec des poids de signes quelconques.
- le problème considéré:
  - recherche du plus court chemin d'un sommet à un autre,
  - recherche du plus court chemin d'un sommet à tous les autres,
  - recherche du plus court chemin entre tous les couples de sommets.

Outline

Algorithme de DIJKSTRA : Principe

On considère le graphe orienté et valué $G = (V, E)$ avec $|V| = n$ et $V = (x_1, x_2, \cdots, x_n)$.

- Recherche du plus court chemin partant de la source x_1 à toutes les autres destinations.
- On désigne par $dist(j)$:la longueur optimale du chemin entre le sommet x_1 et le sommet x_j.
- P: ensemble de sommets dont les distances sont fixées de manière Permanente: pour lesquels le plus court chemin a été déterminé.
- $T = V \backslash P$: ensemble de sommets dont les distances sont fixées de manière Temporaire: pour lesquels le plus court chemin n'a été pas encore déterminé.
- A chaque itération un sommet est transféré de T vers P.
- on note par $P_{cc}(j)$: le plus court chemin de $x_i \longrightarrow x_j$.
- Le calcul des plus courtes distances se fait de proche en proche par ajustements successifs.

Pseudo-code Algorithme de Dijkstra-Moore (1959)

Initialisation :

- $dist(1) = 0$
- Pour j de 2 à n faire

$$dist(j) = \begin{cases} C_{1,j} & \text{si } (x_1, x_j) \in A, \\ +\infty & \text{sinon.} \end{cases}$$

- fin pour
- $Pcc(1) = (x_1)$, $P = \{x_1\}$, et $T = \{x_2, \cdots, x_n\}$

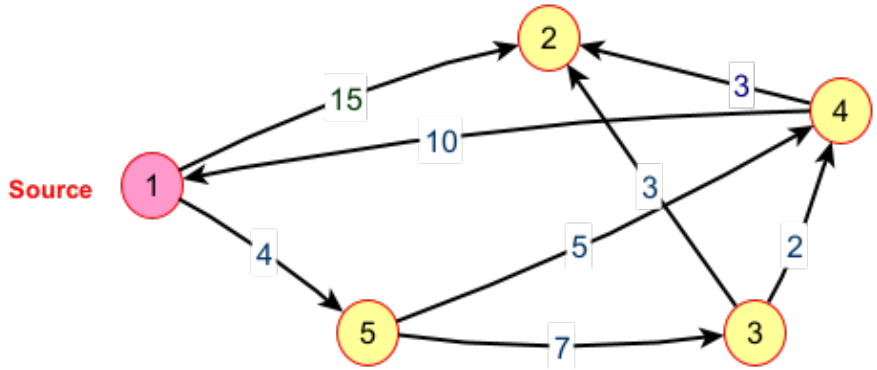
Procédure itérative :

- Tant que $T \neq \emptyset$ faire

  étape 1 : Choix de $dist(k)$ optimale
  - Détermination de x_k / $dist(k) = \min_{x_j \in T} dist(j)$
  - Mémorisation de $Pcc(k)$, $P = P \cup \{x_k\}$ et $T = T/\{x_k\}$

  étape 2 : Mise à jour des $dist$
  - Pour $x_j \in T$ et x_j successeur de x_k faire
    - Si $dist(k) + C_{k,j} < dist(j)$ alors
      $dist(j) = dist(k) + C_{k,j}$
    - fin Si
  - fin Pour
- fin Tant que
- Afficher $dist$ et Pcc.

Illustration de l'algorithme de DIJKSTRA

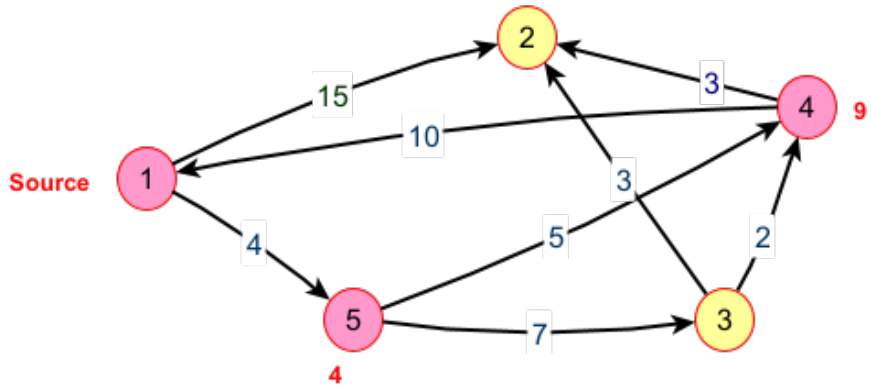


		itération 0		itération 1		itération 2		itération 3		itération 4	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
2	2	15	(1, 2)								
3	3	∞	(1, 3)								
4	4	∞	(1, 4)								
5	5	4	(1, 5)								
	P	{1}									
	T	{2, 3, 4, 5}									

Illustration de l'algorithme de DIJKSTRA

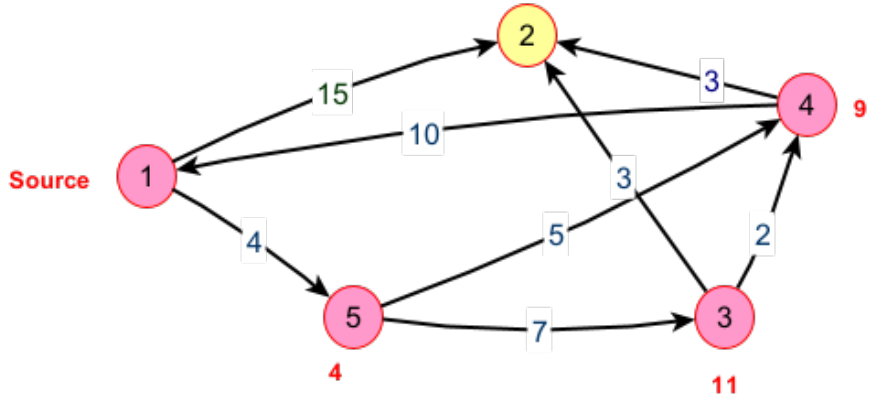
		itération 0		itération 1		itération 2		itération 3		itération 4	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
2	2	15	(1, 2)	15	(1, 2)	12	(1, 5,4,2)				
3	3	∞	(1, 3)	11	(1, 5,3)	11	(1, 5,3)				
4	4	∞	(1, 4)	9	(1, 5,4)	-	-	-	-	-	-
5	5	4	(1, 5)	-	-	-	-	-	-	-	-
	P	{1}		{1,5}		{1,5,4}					
	T	{2, 3, 4, 5}		{2,3, 4}		{2,3}					

Illustration de l'algorithme de DIJKSTRA



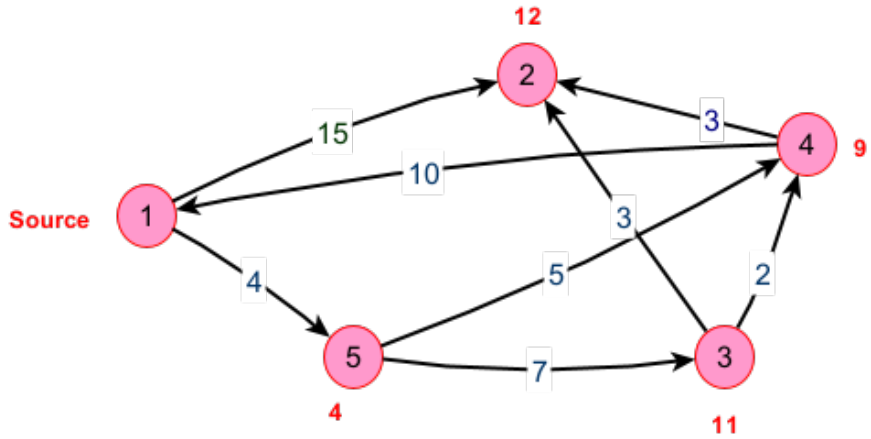
		itération 0		itération 1		itération 2		itération 3		itération 4	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
2	2	15	(1, 2)	15	(1, 2)	12	(1, 5,4,2)	12	(1, 5,4,2)		
3	3	∞	(1, 3)	11	(1, 5,3)	11	(1, 5,3)	-	-	-	-
4	4	∞	(1, 4)	9	(1, 5,4)	-	-	-	-	-	-
5	5	4	(1, 5)	-	-	-	-	-	-	-	-
	P	{1}		{1,5}		{1,5,4}		{1,5,4,3}			
	T	{2, 3, 4, 5}		{2,3, 4}		{2,3}		{2}			

Illustration de l'algorithme de DIJKSTRA



		itération 0		itération 1		itération 2		itération 3		itération 4	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
2	2	15	(1, 2)	15	(1, 2)	12	(1, 5,4,2)	12	(1, 5,4,2)	-	-
3	3	∞	(1, 3)	11	(1, 5,3)	11	(1, 5,3)	-	-	-	-
4	4	∞	(1, 4)	9	(1, 5,4)	-	-	-	-	-	-
5	5	4	(1, 5)	-	-	-	-	-	-	-	-
	P	{1}		{1,5}		{1,5,4}		{1,5,4,3}		{1,5,4,3,2}	
	T	{2, 3, 4, 5}		{2,3, 4}		{2,3}		{2}		{}	

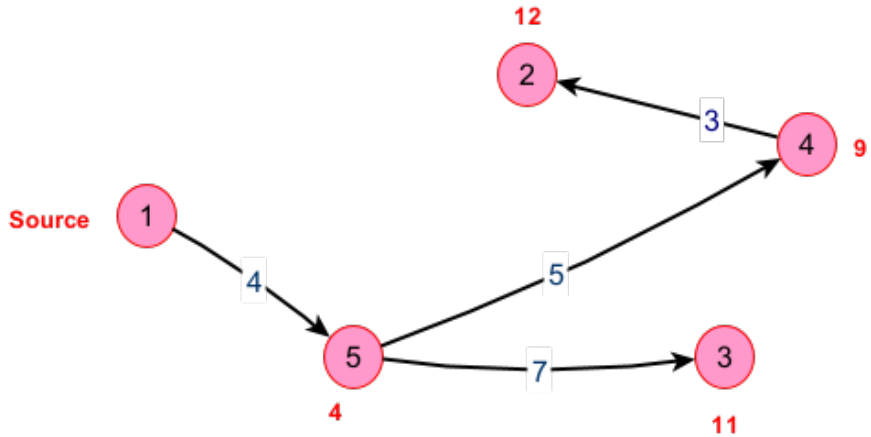
Illustration de l'algorithme de DIJKSTRA



		itération 0		itération 1		itération 2		itération 3		itération 4	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
2	2	15	(1, 2)	15	(1, 2)	12	(1, 5,4,2)	12	(1, 5,4,2)	-	-
3	3	∞	(1, 3)	11	(1, 5,3)	11	(1, 5,3)	-	-	-	-
4	4	∞	(1, 4)	9	(1, 5,4)	-	-	-	-	-	-
5	5	4	(1, 5)	-	-	-	-	-	-	-	-
	P	{1}		{1,5}		{1,5,4}		{1,5,4,3}		{1,5,4,3,2}	
	T	{2, 3, 4, 5}		{2,3, 4}		{2,3}		{2}		{}	

Illustration de l'algorithme de DIJKSTRA

Arborescence optimale :

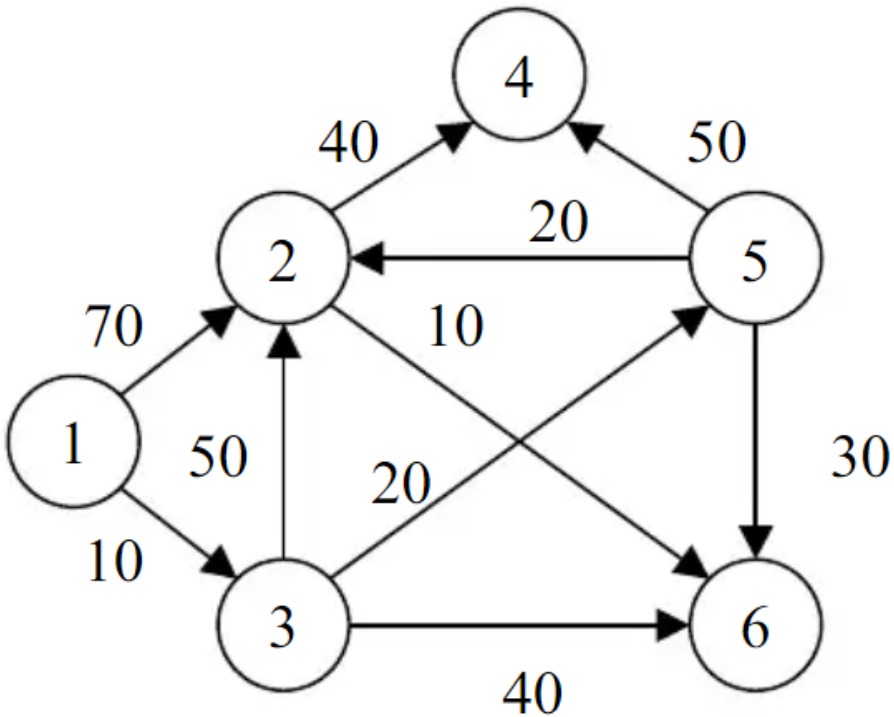


		itération 0		itération 1		itération 2		itération 3		itération 4	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
2	2	15	(1, 2)	15	(1, 2)	12	(1, 5,4,2)	12	(1, 5,4,2)	-	-
3	3	∞	(1, 3)	11	(1, 5,3)	11	(1, 5,3)	-	-	-	-
4	4	∞	(1, 4)	9	(1, 5,4)	-	-	-	-	-	-
5	5	4	(1, 5)	-	-	-	-	-	-	-	-
	P	{1}		{1,5}		{1,5,4}		{1,5,4,3}		{1,5,4,3,2}	
	T	{2, 3, 4, 5}		{2,3, 4}		{2,3}		{2}		{}	

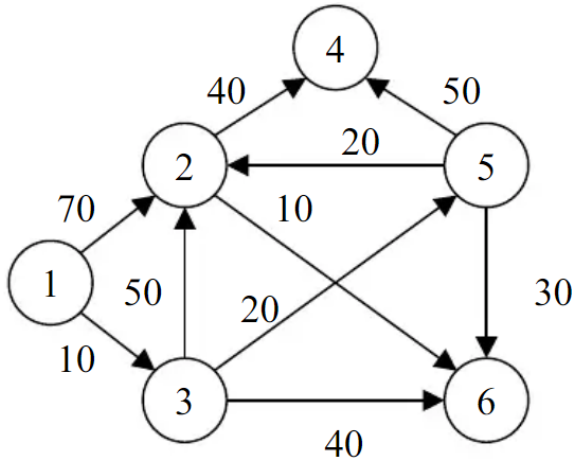
Exemple d'application

Une entreprise située au sommet 1 doit distribuer la marchandise à ses clients situés aux sommets 2 à 6. La distance à parcourir d'un point à un autre et le sens de circulation sont indiqués dans la figure ci-contre.

- Déterminer les plus courts chemins entre 1 et 2 et entre 1 et 6?



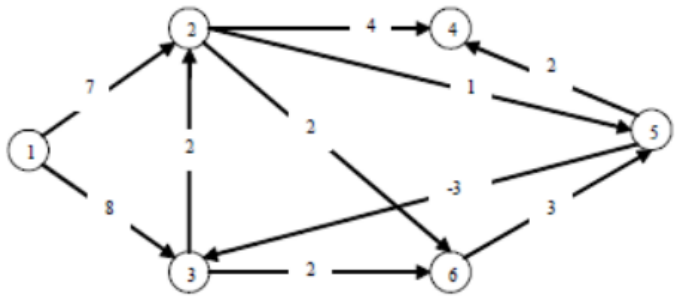
Exemple d'application



		itération 0		itération 1		itération 2		itération 3		itération 4		itération 5	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
2	2	70	(1, 2)	60	(1,3,2)	50	(1,3,5,2)	50	(1,3,5,2)	-	-	-	-
3	3	10	(1, 3)	-	-	-	-	-	-	-	-	-	-
4	4	∞	(1, 4)	∞	(1, 4)	80	(1, 5, 4)	80	(1, 5, 4)	80	(1, 5, 4)	-	-
5	5	∞	(1, 5)	30	(1,3,5)	-	-	-	-	-	-	-	-
	P	{1}		{1, 3}		{1, 3, 5}		{1, 3, 5, 6}		{1, 3, 5, 6, 2}		{1, 3, 5, 6, 2, 4}	
	T	{2, 3, 4, 5, 6}		{2, 4, 5, 6}		{2, 4, 6}		{2, 4}		{4}		{}	

- Le plus court chemin de 1 à 2 est $< 1, 3, 5, 2 >$ de longueur 50.
- Le plus court chemin de 1 à 6 est $< 1, 3, 6 >$ de longueur 50.

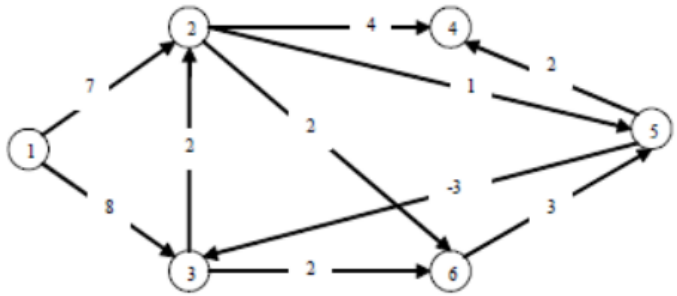
Limite de l'algorithme de DIJKSTRA



Application de l'algorithme de DIJKSTRA : Itération 2

		itération 0		itération 1		itération 2		itération 3		itération 4		itération 5	
j	x_j	dist(x_j)	Pcc(x_j)	dist(x_j)	Pcc(x_j)	dist(x_j)	Pcc(x_j)	dist(x_j)	Pcc(x_j)	dist(x_j)	Pcc(x_j)	dist(x_j)	Pcc(x_j)
2	2	7	(1, 2)	-	-	-	-	-	-	-	-	-	-
3	3	8	(1, 3)	8	(1, 3)	-	-	-	-	-	-	-	-
4	4	∞	(1, 4)	11	(1, 2, 4)	11	(1, 2, 4)						
5	5	∞	(1, 5)	8	(1, 2, 5)	8	(1, 2, 5)						
6	6	∞	(1, 6)	9	(1, 2, 6)	9	(1, 2, 6)						
	P	{1}		{1, 2}		{1, 2, 3}							
	T	{2, 3, 4, 5, 6}		{3, 4, 5, 6}		{4, 5, 6}							

Limite de l'algorithme de DIJKSTRA



- L'algorithme de DIJKSTRA n'a pas détecté le plus court chemin $p =< 1, 2, 5, 3 >$ pour aller du sommet 1 vers le sommet 3.
- $\ell(p) = 5$ alors que L'algorithme de DIJKSTRA nous a retourné $\pi(1, 3) = 8$.
- L'algorithme DIJKSTRA **n'est plus utilisable** dans le cas d'un graphe **valué à poids mixtes**.
- D'où la nécessité d'une autre démarche qui prend en considération la présence de poids négatifs, aussi bien des circuits absorbants : Algorithme de **FORD BELLMAN**.

Outline

Algorithme de FORD BELLMAN: Principe

On considère le graphe orienté et valué $G = (X, A)$ avec $|X| = n$ et $C : A \longrightarrow \mathbb{R}$,la fonction coût associée.

Notation:

- Pour tout $j \in \{1, \cdots, n\}$, on note par $dist^k(j)$: la longueur optimale du chemin reliant la source x_1 à la destination x_j, composé d'au plus k sommets autre que l'origine. Cette longueur sera mise à jour tout au long du processus afin de déterminer $\delta(x_1, x_j)$, le plus court chemin reliant les deux sommets x_1 et x_j.
- Pour tout $j \in \{1, \cdots, n\}$, on note par $Pcc(j)$: le plus court chemin associé au sommet x_j.

- Graphe ayant des coûts quelconques,
- Recherche du plus court chemin entre un sommet et tous les autres,
- Il y a convergence **en absence de circuit absorbant**,

Pseudo-code Algorithme de FORD BELLMAN

Étape 0 : Initialisation :

- $k = 1$
- $dist^k(1) = 0$
- Pour j de 2 à n faire

$$dist^1(j) = \begin{cases} C_{1,j} & \text{si } (x_1, x_j) \in A, \\ +\infty & \text{sinon.} \end{cases}$$

- fin pour

Étape 1 : Mise à jour des $dist^k(j)$:

- Pour j de 1 à n faire
  $dist^{k+1}(j) = \min(dist^k(j), \min(dist^k(l) + C_{l,j}))$, x_l sont les prédécesseurs de x_j.
- fin pour

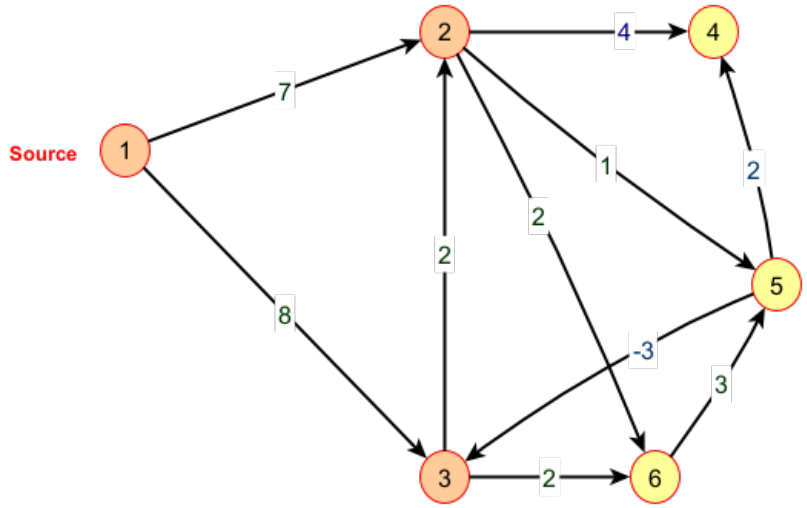
Étape 2 : Test de convergence :

- Si $\forall j \in \{1, \cdots, n\}$, $dist^{k-1}(j) = dist^k(j)$ alors fin : optimalité atteinte
- Sinon et Si $k = n$ alors : il existe un circuit absorbant
- Sinon, faire $k = k + 1$ et aller à l'étape 1.

Remarque: L'algorithme converge en n itérations.

Illustration de l'algorithme de FORD BELLMAN

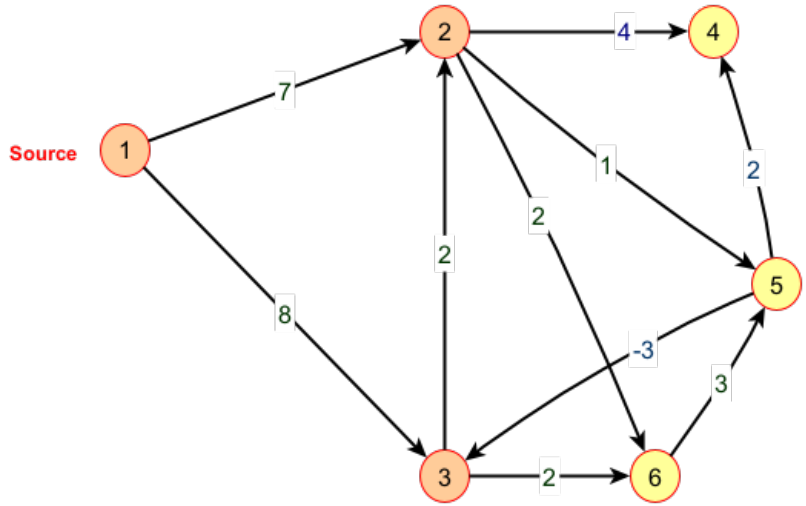
k=1: Détermination des chemins composés d'au plus 2 sommets



		itération 1									
j	x_j	dist(j)	Pcc(j)								
1	1	0	(1, 1)								
2	2	7	(1, 2)								
3	3	8	(1, 3)								
4	4	∞	(1, 4)								
5	5	∞	(1, 5)								
6	6	∞	(1, 6)								

Illustration de l'algorithme de FORD BELLMAN

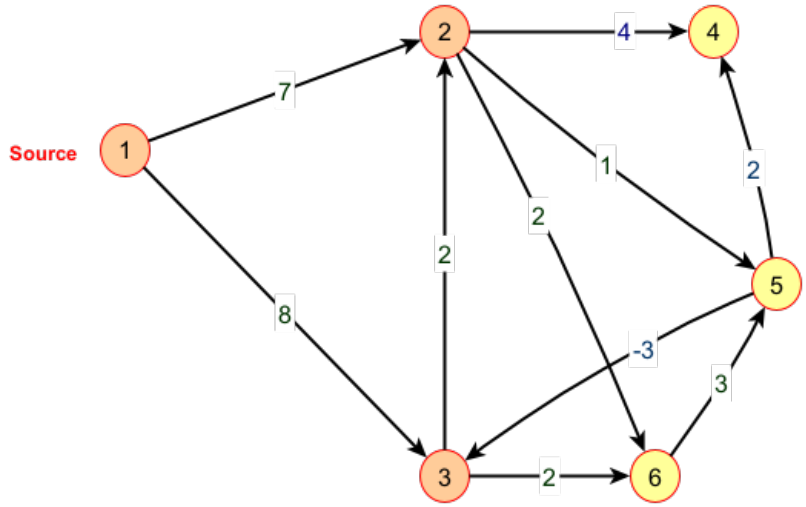
k=2 : Détermination des chemins composés d'au plus 3 sommets



		itération 1		itération 2							
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)						
1	1	0	(1, 1)	0	(1, 1)						
2	2	7	(1, 2)	7	(1, 2)						
3	3	8	(1, 3)	8	(1, 3)						
4	4	∞	(1, 4)	11	(1,2, 4)						
5	5	∞	(1, 5)	8	(1,2, 5)						
6	6	∞	(1, 6)	9	(1,2, 6)						

Illustration de l'algorithme de FORD BELLMAN

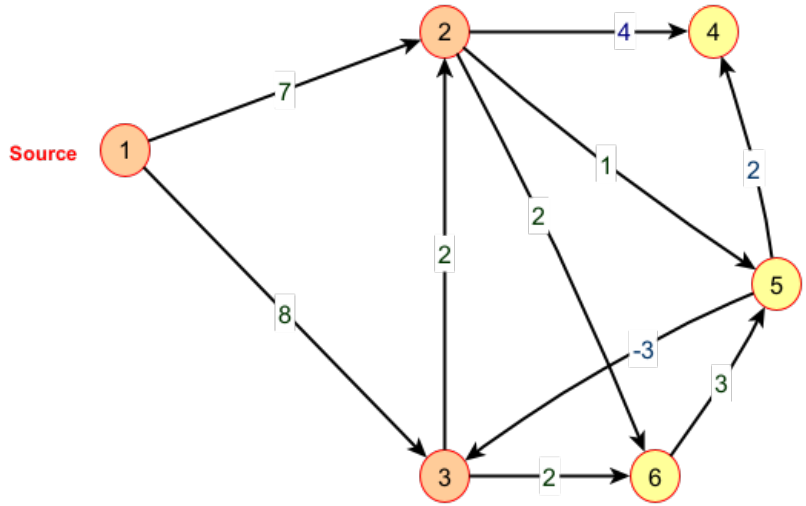
k=3: Détermination des chemins composés d'au plus 4 sommets



		itération 1		itération 2		itération 3					
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)				
1	1	0	(1, 1)	0	(1, 1)	0	(1, 1)				
2	2	7	(1, 2)	7	(1, 2)	7	(1, 2)				
3	3	8	(1, 3)	8	(1, 3)	5	(1,2, 5,3)				
4	4	∞	(1, 4)	11	(1,2, 4)	10	(1,2,5, 4)				
5	5	∞	(1, 5)	8	(1,2, 5)	8	(1,2, 5)				
6	6	∞	(1, 6)	9	(1,2, 6)	9	(1,2, 6)				

Illustration de l'algorithme de FORD BELLMAN

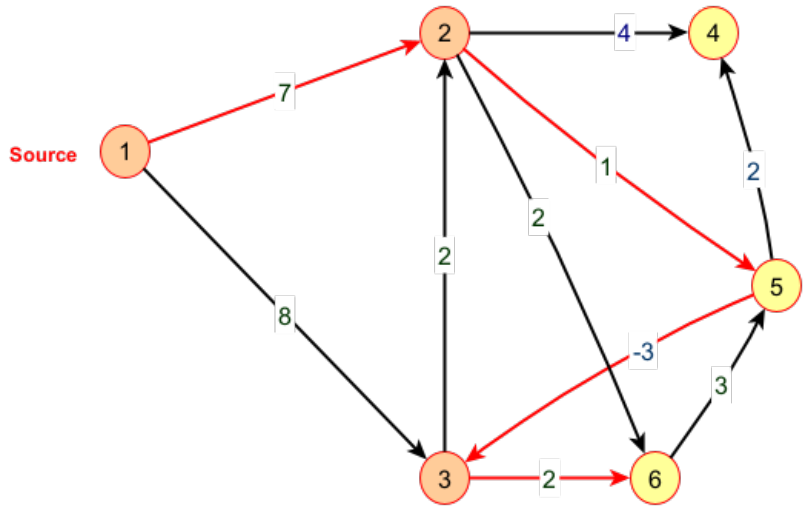
k=4 : Détermination des chemins composés d'au plus 5 sommets



		itération 1		itération 2		itération 3		itération 4			
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)		
1	1	0	(1, 1)	0	(1, 1)	0	(1, 1)	0	(1, 1)		
2	2	7	(1, 2)	7	(1, 2)	7	(1, 2)	7	(1, 2)		
3	3	8	(1, 3)	8	(1, 3)	5	(1,2, 5,3)	5	(1,2, 5,3)		
4	4	∞	(1, 4)	11	(1,2, 4)	10	(1,2,5, 4)	10	(1,2,5, 4)		
5	5	∞	(1, 5)	8	(1,2, 5)	8	(1,2, 5)	8	(1,2, 5)		
6	6	∞	(1, 6)	9	(1,2, 6)	9	(1,2, 6)	7	(1,2,5,3, 6)		

Illustration de l'algorithme de FORD BELLMAN

k=5 : Détermination des chemins composés d'au plus 6 sommets



		itération 1		itération 2		itération 3		itération 4		itération 5	
j	x_j	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)	dist(j)	Pcc(j)
1	1	0	(1, 1)	0	(1, 1)	0	(1, 1)	0	(1, 1)	0	(1, 1)
2	2	7	(1, 2)	7	(1, 2)	7	(1, 2)	7	(1, 2)	7	(1, 2)
3	3	8	(1, 3)	8	(1, 3)	5	(1,2, 5,3)	5	(1,2, 5,3)	5	(1,2, 5,3)
4	4	∞	(1, 4)	11	(1,2, 4)	10	(1,2,5, 4)	10	(1,2,5, 4)	10	(1,2,5, 4)
5	5	∞	(1, 5)	8	(1,2, 5)	8	(1,2, 5)	8	(1,2, 5)	8	(1,2, 5)
6	6	∞	(1, 6)	9	(1,2, 6)	9	(1,2, 6)	7	(1,2,5,3, 6)	7	(1,2,5,3, 6)

Outline

Algorithme de FLOYD-WARSHALL: Principe

- Une généralisation de l'algorithme de FORD-BELLMAN.
- Détermination d'un plus court chemin entre n'importe quel couple de sommets du graphe.
- Cet algorithme prend en entrée un graphe orienté et valué sous forme d'une matrice d'adjacence donnant le poids d'un arc lorsque l'arc existe et la valeur infinie sinon.

Variables de l'algorithme de FLOYD-WARSHALL :

On considère le graphe orienté et valué $G = (X, A)$ avec $|X| = n$ et $C : A \longrightarrow \mathbb{R}$,la fonction coût associée.

Notation:

- Pour tout $j \in \{1, \cdots, n\}$, on note par $dist(i,j)$: la longueur optimale du chemin reliant la source x_i à la destination x_j. Cette longueur sera mise à jour tout au long du processus afin de déterminer $\delta(x_i, x_j)$, le plus court chemin reliant les deux sommets x_i et x_j.
- Pour tout $i,j \in \{1, \cdots, n\}$, on note par $Pcc(i,j)$: le plus court chemin reliant les sommets x_i et x_j.

Pseudo-code Algorithme de FLOYD-WARSHALL

Étape 0 : Initialisation :

- Pour j de 1 à n faire

$$dist(i,j) = \begin{cases} 0 & \text{si } i = j, \\ C_{i,j} & \text{si } (x_i, x_j) \in A, \\ +\infty & \text{sinon.} \end{cases}$$

- fin si

$\forall i \neq j,\ Pcc(i,j) = (x_i, x_j)$

Étape 1 : Mise à jour des $dist(i,j)$:

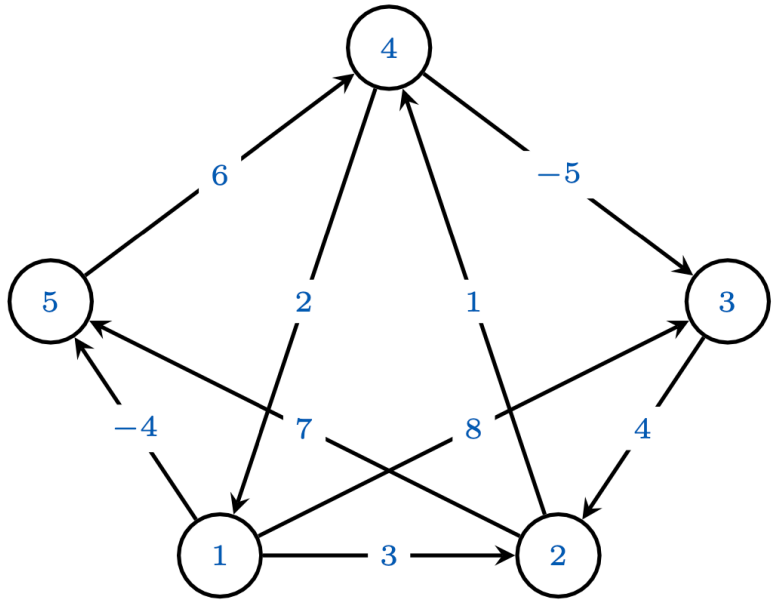
- $k = 1$
- $\forall i \neq k$ et $j \neq k$, si $dist(i,k) + dist(k,j) < dist(i,j)$ faire
  $dist(i,j) = dist(i,k) + dist(k,j)$, et $Pcc(i,j) = Pcc(i,k) \cup Pcc(k,j)$.
- fin pour

Étape 2 : Test de convergence :

- Si $\exists i$ tel que $dist(i,i) < 0$ alors fin : il existe un circuit absorbant
- Sinon et Si $k = n$ alors : optimalité atteinte
- Sinon, faire $k = k + 1$ et aller à l'étape 1.

Exemple d'application

Initialisation selon l'ordre des sommets 1-2-3-4-5



$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(0)} = \begin{pmatrix} X & 1 & 1 & X & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & X & X \\ 4 & X & 4 & X & X \\ X & X & X & 5 & X \end{pmatrix}$$

k=1 : insertion du sommet d'indice k (le premier sommet : ici c'est 1)

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(0)} = \begin{pmatrix} X & 1 & 1 & X & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & X & X \\ 4 & X & 4 & X & X \\ X & X & X & 5 & X \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(1)} = \begin{pmatrix} X & 1 & 1 & X & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & X & X \\ 4 & 1 & 4 & X & 1 \\ X & X & X & 5 & X \end{pmatrix}$$

k=2 : insertion du sommet d'indice k (le deuxième sommet : ici c'est 2)

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(1)} = \begin{pmatrix} X & 1 & 1 & X & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & X & X \\ 4 & 1 & 4 & X & 1 \\ X & X & X & 5 & X \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(2)} = \begin{pmatrix} X & 1 & 1 & 2 & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & 2 & 2 \\ 4 & 1 & 4 & X & 1 \\ X & X & X & 5 & X \end{pmatrix}$$

k=3 : insertion du sommet d'indice k (le troisième sommet : ici c'est 3)

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(2)} = \begin{pmatrix} X & 1 & 1 & 2 & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & 2 & 2 \\ 4 & 1 & 4 & X & 1 \\ X & X & X & 5 & X \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(3)} = \begin{pmatrix} X & 1 & 1 & 2 & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & 2 & 2 \\ 4 & 3 & 4 & X & 1 \\ X & X & X & 5 & X \end{pmatrix}$$

k=4 : insertion du sommet d'indice k (le quatrième sommet : ici c'est 4)

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \pi^{(3)} = \begin{pmatrix} X & 1 & 1 & 2 & 1 \\ X & X & X & 2 & 2 \\ X & 3 & X & 2 & 2 \\ 4 & 3 & 4 & X & 1 \\ X & X & X & 5 & X \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \pi^{(4)} = \begin{pmatrix} X & 1 & 4 & 2 & 1 \\ 4 & X & 4 & 2 & 4 \\ 4 & 3 & X & 2 & 4 \\ 4 & 3 & 4 & X & 1 \\ 4 & 4 & 4 & 5 & X \end{pmatrix}$$

k=5 : insertion du sommet d'indice k (le cinquième sommet : ici c'est 5)

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \pi^{(4)} = \begin{pmatrix} X & 1 & 4 & 2 & 1 \\ 4 & X & 4 & 2 & 4 \\ 4 & 3 & X & 2 & 4 \\ 4 & 3 & 4 & X & 1 \\ 4 & 4 & 4 & 5 & X \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \pi^{(5)} = \begin{pmatrix} X & 5 & 5 & 5 & 1 \\ 4 & X & 4 & 2 & 4 \\ 4 & 3 & X & 2 & 4 \\ 4 & 3 & 4 & X & 1 \\ 4 & 4 & 4 & 5 & X \end{pmatrix}$$

Outline

Conclusion:

On considère un graphe orienté valué $G = (X, A)$ avec $|X| = n$, $|A| = m$ et $\mathcal{C} : A \longrightarrow \mathbb{R}$, la fonction coût associée.

	DIJKSTRA	FORD BELLMAN	FLOYD WARSHALL
Type de chemins	d'un sommet vers les autres	d'un sommet vers les autres	paires de sommets
Complexité représentation du graphe par listes d'adjacence	$\theta(n + m\log(n))$	$\theta(n \times m)$	
Complexité représentation du graphe par matrice d'adjacence	$\theta(n^2)$	$\theta(n^3)$	$\theta(n^3)$
Cas d'utilisation	Graphes à poids positifs	Graphes à poids quelconques	Graphes à poids quelconques