Plus court chemin

Christophe Viroulaud

Terminale - NSI

Archi 15

Pour calculer rapidement les distances dans le réseau, les routeurs appliquent des algorithmes conçus au début de l'ère de l'informatique.

Comment fonctionnent les algorithmes de plus court chemin ?

Sommaire

Représentation des réseaux - Graphe non orienté

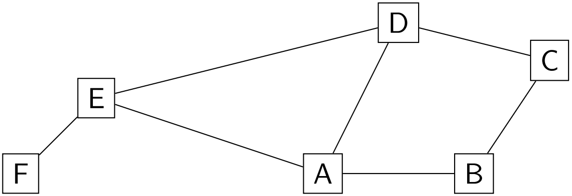


FIGURE 1 – graphe non orienté

À retenir

Un graphe est composé :

- de sommets ou nœuds,
- d'arêtes ou arcs qui relient les sommets.

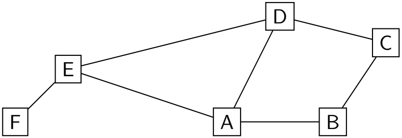


FIGURE 2 – graphe non orienté

À retenir

Pour chaque nœud on peut définir ses **prédécesseurs**.
E est le prédécesseur de **F**.

Sommaire

Graphe pondéré

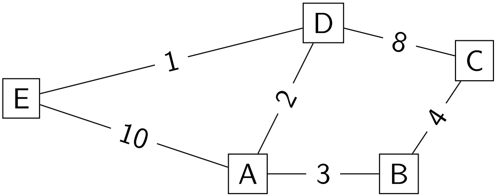


FIGURE 3 – graphe pondéré

À retenir

Selon le protocole mis en place la pondération pourra représenter :

- RIP : le nombre de réseaux traversés,
- OSPF : le coût d'un réseau.

Sommaire

Algorithme de Bellman-Ford : Principe

- 1956 - 1958

Algorithme de Bellman-Ford : Principe

- 1956 - 1958
- Richard Bellman (père programmation dynamique)

Algorithme de Bellman-Ford : Principe

- 1956 - 1958
- Richard Bellman (père programmation dynamique)
- Lester Ford (problème de flot maximum)

Algorithme de Bellman-Ford : Principe

- 1956 - 1958
- Richard Bellman (père programmation dynamique)
- Lester Ford (problème de flot maximum)
- redécouvert par Edward Moore en 1959

Algorithme de Bellman-Ford : Principe

- 1956 - 1958
- Richard Bellman (père programmation dynamique)
- Lester Ford (problème de flot maximum)
- redécouvert par Edward Moore en 1959
- Le protocole RIP applique cet algorithme.

Remarque

L'algorithme de Bellman-Ford est normalement appliqué dans un graphe orienté. Cependant, si les pondérations sont toutes positives, il est possible d'utiliser un graphe non orienté.

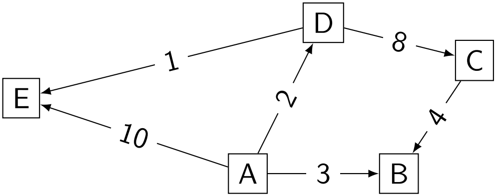


Figure 4 – graphe orienté

À retenir

La distance pour atteindre chaque nœud correspond à la distance pour atteindre son prédécesseur à laquelle on ajoute le poids de l'arête les séparant.

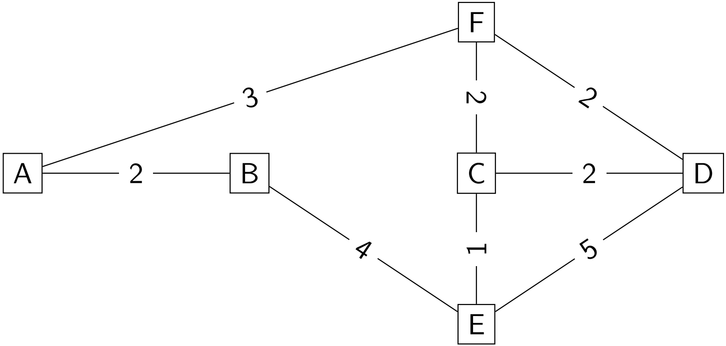


FIGURE 5 – graphe orienté pondéré

À retenir

L'algorithme de Bellman-Ford applique une méthode récursive.

Pour chaque routeur, on obtient un tableau contenant la distance minimale entre le routeur de départ et chaque autre routeur.

Sommaire

Mise en application

Initialisation :

- Créer un tableau des distances entre A et les routeurs, initialisées à l'infini.
- Modifier la distance vers A à 0.

Initialisation :

- Créer un tableau des distances entre A et les routeurs, initialisées à l'infini.
- Modifier la distance vers A à 0.

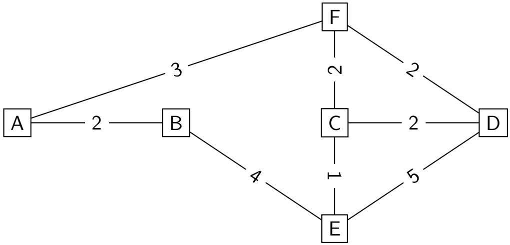
Déroulement :

- Tant que (nombre d'itérations) $<$ (nombre de routeurs)
  - Pour chaque arc du graphe
    - Si (distance du routeur) $>$ (distance de son prédécesseur $+$ poids de l'arc entre les deux routeurs)
      $\Rightarrow$ Mettre à jour distance du routeur

Observations

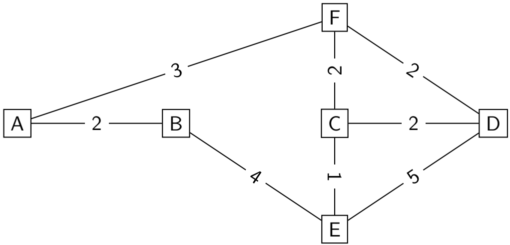
- On effectue autant d'itérations qu'il y a de routeurs.
- On regarde chaque arc à chaque tour.

Initialisation



A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Première itération

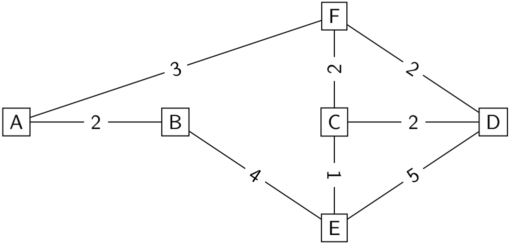


A	B	C	D	E	F
0	**2 (A)**	∞	∞	∞	∞

La distance calculée pour atteindre B correspond à la distance de son prédécesseur (A) augmentée du poids de l'arc A-B :

$$0 + 2 = 2 < \infty$$

Première itération

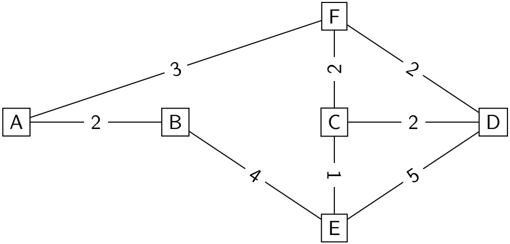


A	B	C	D	E	F
0	**2 (A)**	∞	∞	∞	∞

Second prédécesseur de B : E

$$\infty + 4 = \infty < \infty$$

Première itération

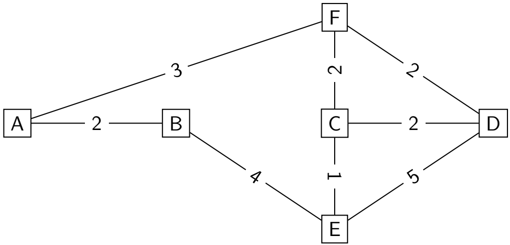


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Premier prédécesseur de C : D

$$\infty + 1 = \infty \not< \infty$$

Première itération

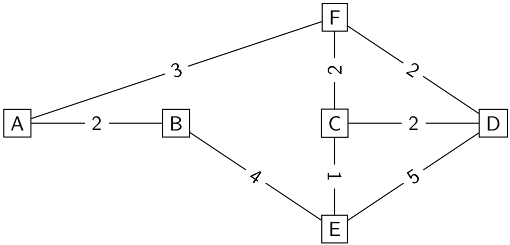


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Deuxième prédécesseur de C : E

$$\infty + 1 = \infty \not< \infty$$

Première itération

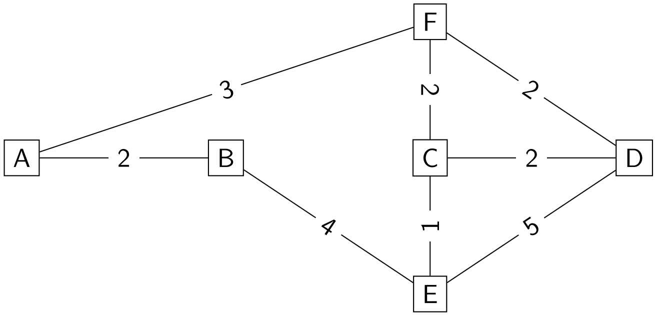


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

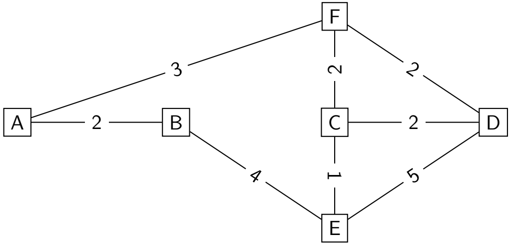
Troisième prédécesseur de C : F

$$\infty + 2 = \infty \nless \infty$$

Activité 1 : Continuer de dérouler la première itération de l'algorithme sur le graphe.



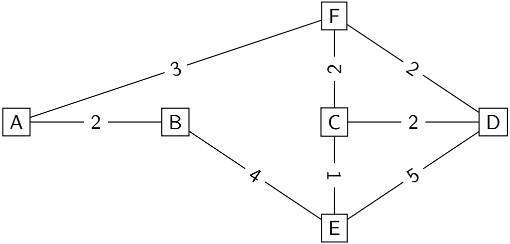
Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Même constat pour D

Première itération

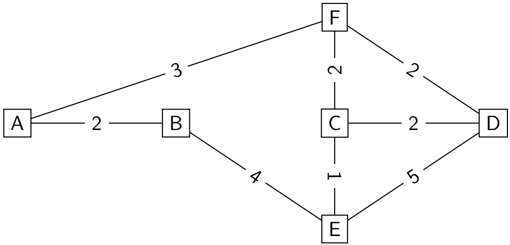


A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Premier prédécesseur de E : B

$$2 + 4 = 6 < \infty$$

Première itération

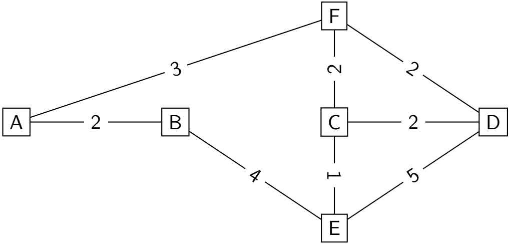


A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Deuxième prédécesseur de E : C

$$\infty + 1 = \infty > 6$$

Première itération

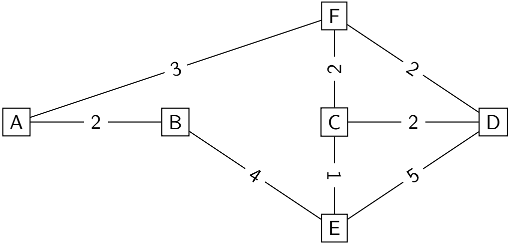


A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Troisième prédécesseur de E : D

$$\infty + 5 = \infty > 6$$

Première itération

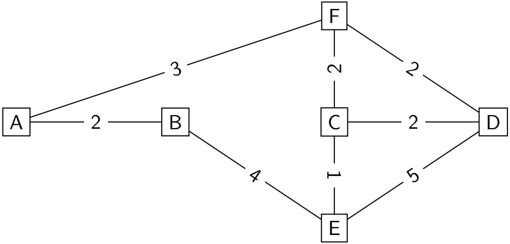


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	**3 (A)**

Premier prédécesseur de F : A

$$0 + 3 = 3 < \infty$$

Première itération

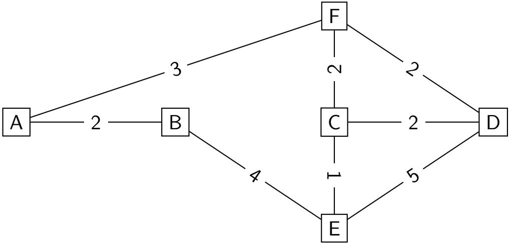


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Deuxième prédécesseur de F : C

$$\infty + 2 = \infty \not< 3$$

Première itération

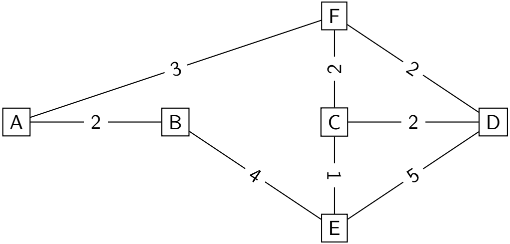


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Troisième prédécesseur de F : D

$$\infty + 2 = \infty \not< 3$$

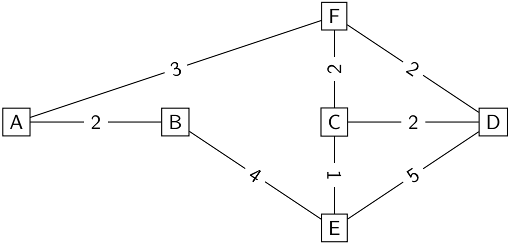
Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Fin de la première itération

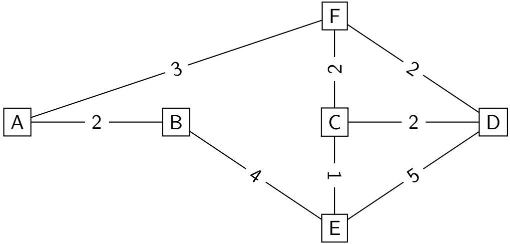
Deuxième itération



A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	∞	∞	6 (B)	3 (A)

Pas de modification pour A et B

Deuxième itération

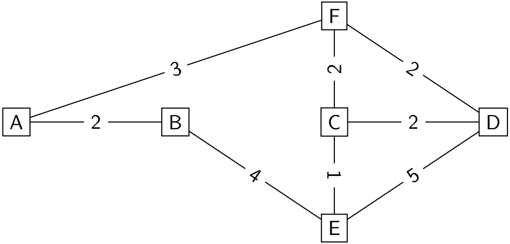


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	**7 (E)**	∞	6 (B)	3 (A)

Premier prédécesseur de C : D

$$\infty + 2 = \infty \not< \infty$$

Deuxième itération

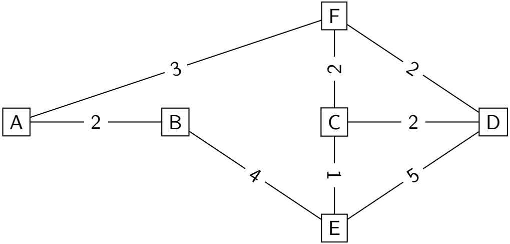


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	**7 (E)**	∞	6 (B)	3 (A)

Deuxième prédécesseur de C : E

$$6 + 1 = 7 < \infty$$

Deuxième itération

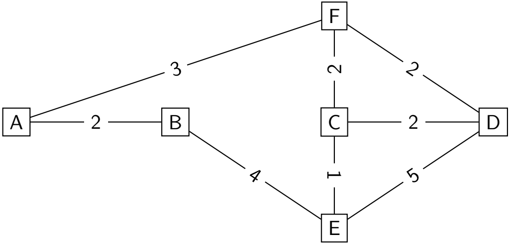


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	**5 (F)**	∞	6 (B)	3 (A)

Troisième prédécesseur de C : F

$$3 + 2 = 5 < 7$$

Deuxième itération

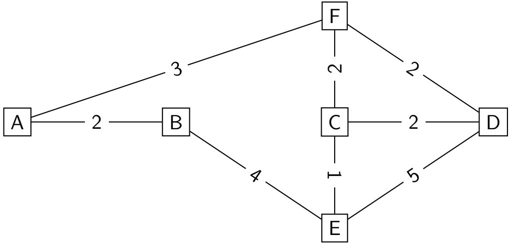


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	**7 (C)**	6 (B)	3 (A)

Premier prédécesseur de D : C

$$5 + 2 = 7 < \infty$$

Deuxième itération

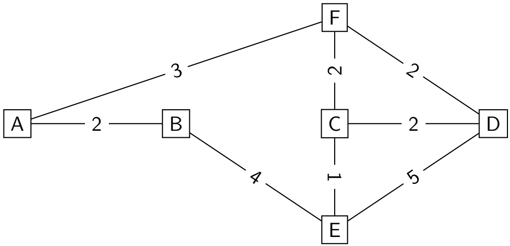


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)

Second prédécesseur de D : E

$$6 + 5 = 11 \not< 7$$

Deuxième itération

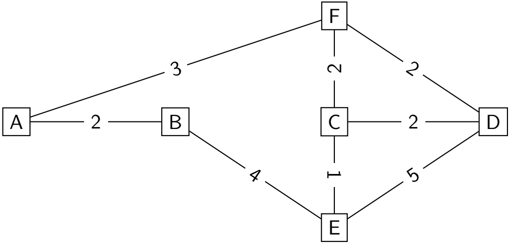


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	**5 (F)**	6 (B)	3 (A)

Troisième prédécesseur de D : F

$$3 + 2 = 5 < 7$$

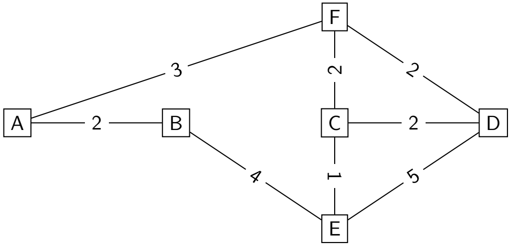
Deuxième itération



A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	5 (F)	6 (B)	3 (A)

Pas de changement pour E et F

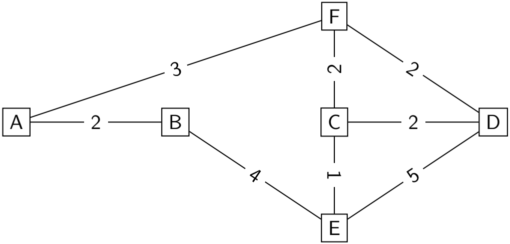
Deuxième itération



A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	5 (F)	6 (B)	3 (A)

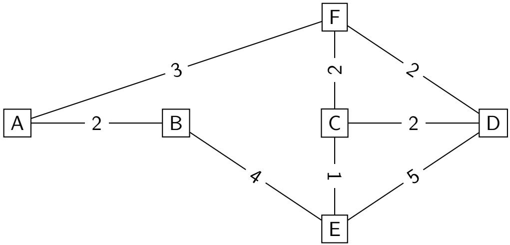
Fin de la deuxième itération

Troisième itération



A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	5 (F)	6 (B)	3 (A)
0	2 (A)	5 (F)	5 (F)	6 (B)	3 (A)

Pas de changement lors de la troisième itération



A	B	C	D	E	F
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)

On remonte dans le tableau pour trouver le chemin vers une destination.

$$D \to C \to F \to A$$

Sommaire

Complexité

La complexité dépend de :

- du nombre de sommets (notée S) : on visite chaque sommet

```
Tant que (le nombre d'itérations) < (nombre de routeurs)
```

Complexité

La complexité dépend de :

- du nombre de sommets (notée S) : on visite chaque sommet

```
Tant que (le nombre d'itérations) < (nombre de routeurs)
```

- du nombre d'arcs (notée A) : pour chaque sommet on regarde tous les arcs du graphe

```
Pour chaque arc du graphe
```

Complexité de l'algorithme de Bellman Ford

$$O(S.A)$$

Sommaire

OSPF : algorithme de Dijkstra - principe

- Edsger Dijkstra : mathématicien néerlandais
- Algorithme utilisé dans GPS

À retenir

principe : construire un sous-graphe en ajoutant à chaque itération un sommet de distance minimale.

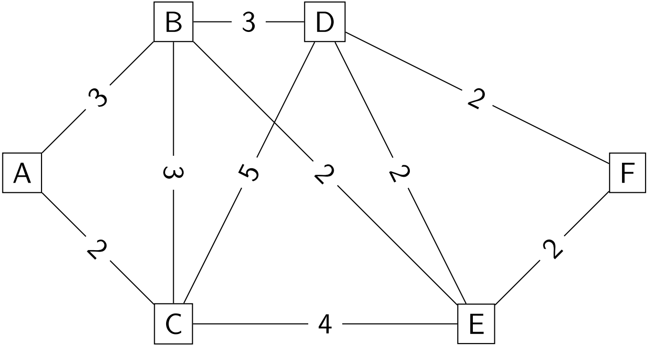


FIGURE 6 – graphe non orienté et pondéré

Sommaire

Mise en application

Initialisation :

- Créer un tableau des distances entre A et les routeurs, initialisées à l'infini.
- Modifier la distance vers A à 0.

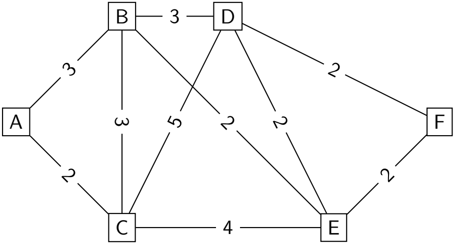
Initialisation :

- Créer un tableau des distances entre A et les routeurs, initialisées à l'infini.
- Modifier la distance vers A à 0.

Déroulement :

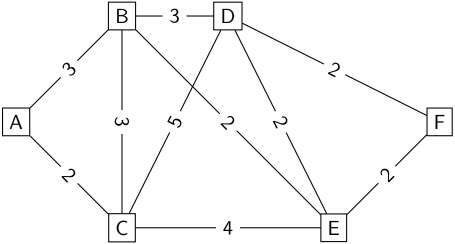
- Tant qu'il reste des routeurs non sélectionnés
  - Parmi les routeurs non-sélectionnés, choisir celui (noté S) ayant la plus petite distance.
  - Pour chaque routeur adjacent à S (noté V) et non déjà sélectionné :
    - Si (la distance de V) $>$ (la distance de S $+$ poids S-V)
      $\Rightarrow$ Mettre à jour la distance de V.

Initialisation



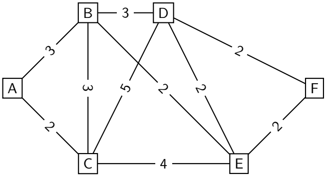
A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Sélection de A



A	B	C	D	E	F
0	**3 (A)**	**2 (A)**	∞	∞	∞

nœuds visités
A



On sélectionne le nœud non encore visité et avec la plus petite distance : C.

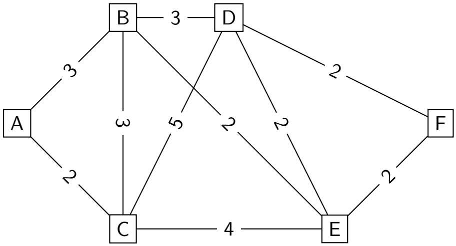
A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	**7 (C)**	**6 (C)**	∞

nœuds visités
A - C

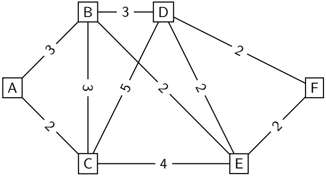
Observation

La route la plus courte a déjà été déterminée pour les nœuds déjà visités. Ils ne seront plus modifiés (cellule grise).

Activité 2 : Continuer de dérouler l'algorithme sur le graphe.



Correction

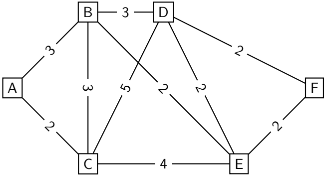


On sélectionne le nœud non encore visité et avec la plus petite distance : B.

A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		**6 (B)**	**5 (B)**	∞

nœuds visités
A - C - B

Correction

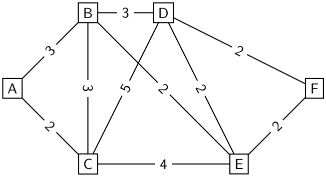


On sélectionne le nœud non encore visité et avec la plus petite distance : E.

A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		6 (B)	5 (B)	∞
			6 (B)	5 (B)	**7 (E)**

nœuds visités
A - C - B - E

Correction

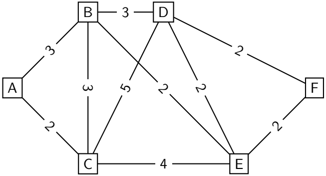


On sélectionne le nœud non encore visité et avec la plus petite distance : D.

A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		6 (B)	5 (B)	∞
			6 (B)	5 (B)	7 (E)
			6 (B)		7 (E)

nœuds visités
A - C - B - E - D

Correction

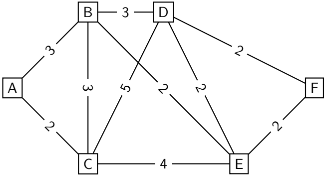


On sélectionne le nœud non encore visité et avec la plus petite distance : F.

A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		6 (B)	5 (B)	∞
			6 (B)	5 (B)	7 (E)
			6 (B)		7 (E)
					7 (E)

nœuds visités
A - C - B - E - D - F

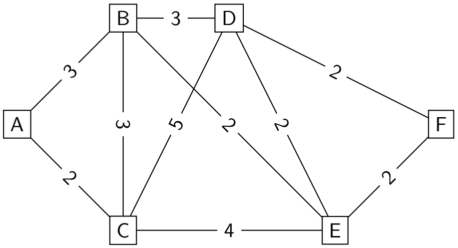
Correction



On sélectionne le nœud non encore visité et avec la plus petite distance : F.

A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		6 (B)	5 (B)	∞
			6 (B)	5 (B)	7 (E)
			6 (B)		7 (E)
					7 (E)

nœuds visités
A - C - B - E - D - F



A	B	C	D	E	F
0	3 (A)	2 (A)	6 (B)	5 (B)	7 (E)

On remonte dans le tableau pour trouver le chemin vers une destination.

$$F \to E \to B \to A$$

Sommaire

Complexité

- La complexité dépend du nombre de sommets S et du nombre d'arcs A.

Complexité

- La complexité dépend du nombre de sommets S et du nombre d'arcs A.
- Le point clé de l'algorithme tient dans la recherche de la distance minimale.

```
Parmi les routeurs non−sélectionnés, choisir le
    routeur (noté S) ayant la plus petite distance.
```

Complexité de l'algorithme de Dijkstra

$$O((A + S) \times \log S)$$