Plus court chemin

Christophe Viroulaud

Terminale - NSI

Archi 15

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 ?

RIP déjà utilisé avec ARPANET !

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 ?

2022-01-12

Plus court chemin
└Représentation des réseaux
└Sommaire

Sommaire

1. Représentation des réseaux
1.1 Graphe non orienté
1.2 Graphe pondéré

2. Protocole RIP : algorithme de Bellman-Ford

3. OSPF : algorithme de Dijkstra

Sommaire

Représentation des réseaux - Graphe non orienté

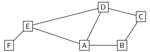


Figure 1 – graphe non orienté

À retenir
Un graphe est composé :
- de sommets ou nœuds,
- d'arêtes ou arcs qui relient les sommets.

Représentation des réseaux - Graphe non orienté

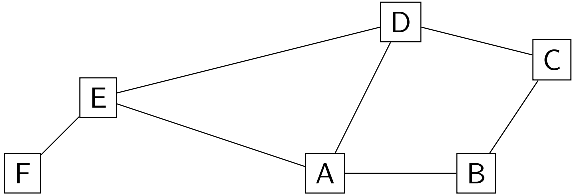


Figure 1 – graphe non orienté

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.

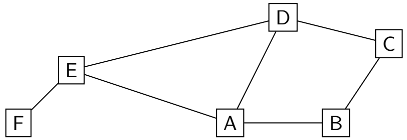


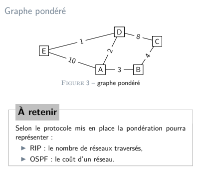
FIGURE 2 – graphe non orienté

À retenir

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

Sommaire
1. Représentation des réseaux
1.1 Graphe non orienté
1.2 Graphe pondéré
2. Protocole RIP : algorithme de Bellman-Ford
3. OSPF : algorithme de Dijkstra

Sommaire



pondération peut être négative

Graphe pondéré

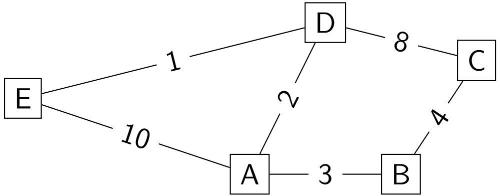


FIGURE 3 – graphe pondéré

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

1. Représentation des réseaux
2. Protocole RIP : algorithme de Bellman-Ford
2.1 Principe
2.2 Mise en application
2.3 Complexité
3. OSPF : algorithme de Dijkstra

Sommaire

redécouvert par Moore en 59 dans autre contexte

Algorithme de Bellman-Ford : Principe

- 1956 - 1958

Algorithme de Bellman-Ford : Principe

- 1956 - 1958

redécouvert par Moore en 59 dans autre contexte

Algorithme de Bellman-Ford : Principe

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

Algorithme de Bellman-Ford : Principe

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

redécouvert par Moore en 59 dans autre contexte

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 Moore en 59 dans autre contexte

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

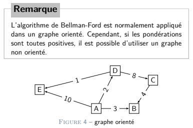
redécouvert par Moore en 59 dans autre contexte

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.

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.

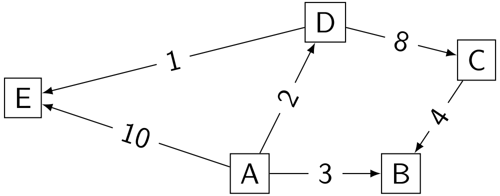


FIGURE 4 – graphe orienté

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.

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.

1. Bellman « père de l'approche dynamique »

Remarque

2. Dans le graphe figure 5 les pondérations représentent un nombre de routeurs traversés pour atteindre le nœud (routeur) suivant.

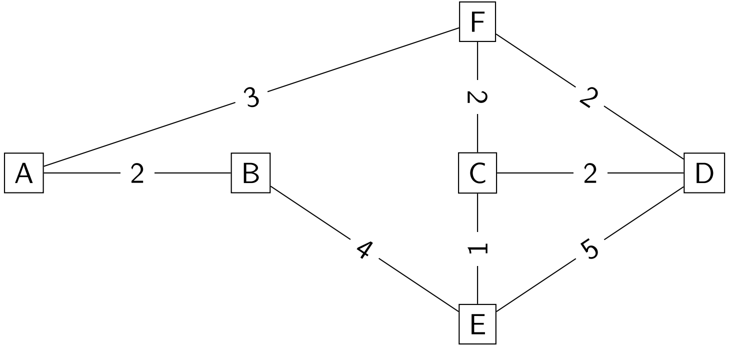


FIGURE 5 – graphe orienté pondéré

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.

peut servir pour retrouver chemin, distance mini...

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

Sommaire

1. Représentation des réseaux

2. Protocole RIP : algorithme de Bellman-Ford
2.1 Principe
2.2 Mise en application
2.3 Complexité

3. OSPF : algorithme de Dijkstra

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.

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

1. ligne 2 de l'algo : en effet distance de A à A $=$ 0
2. ligne 4 : on fait autant d'itérations qu'il y a de routeurs $=$ pire des cas $\rightarrow$ on pourra améliorer

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

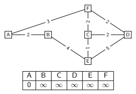
Observations
- On effectue autant d'itérations qu'il y a de routeurs.
- On regarde chaque arc à chaque tour.

Observations

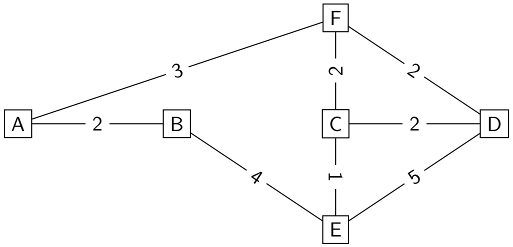
- On effectue autant d'itérations qu'il y a de routeurs.
- On regarde chaque arc à chaque tour.

2022-01-12

Plus court chemin
└─Protocole RIP : algorithme de Bellman-Ford
└─Mise en application
└─Initialisation

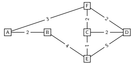


Initialisation



A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Première itération

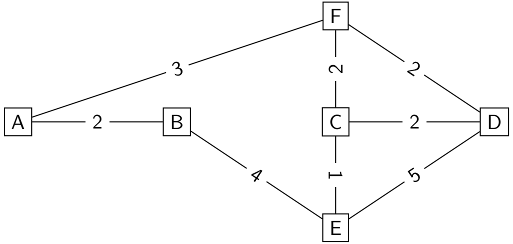


A	B	C	D	E	F
0	**2 (A)**	∞	∞	∞	∞

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

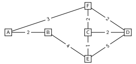


A	B	C	D	E	F
0	**2 (A)**	∞	∞	∞	∞

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

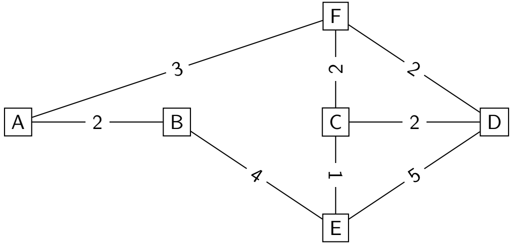


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Second prédécesseur de B : E

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

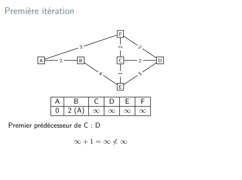
Première itération



A	B	C	D	E	F
0	**2 (A)**	∞	∞	∞	∞

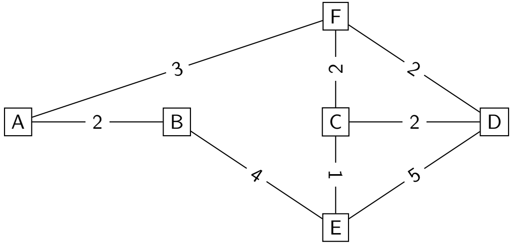
Second prédécesseur de B : E

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



On est toujours dans la première itération ; on vérifie chaque arc.

Première itération

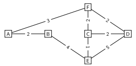


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Premier prédécesseur de C : D

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

Première itération

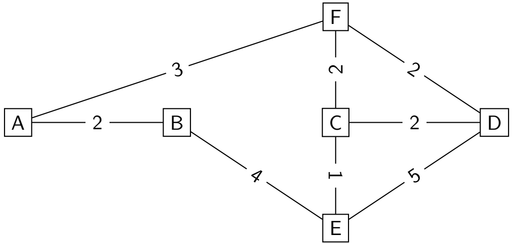


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Deuxième prédécesseur de C : E

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

Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Deuxième prédécesseur de C : E

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

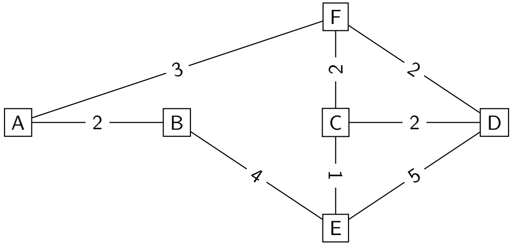
Première itération

A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Troisième prédécesseur de C : F

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

Première itération

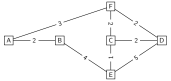


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

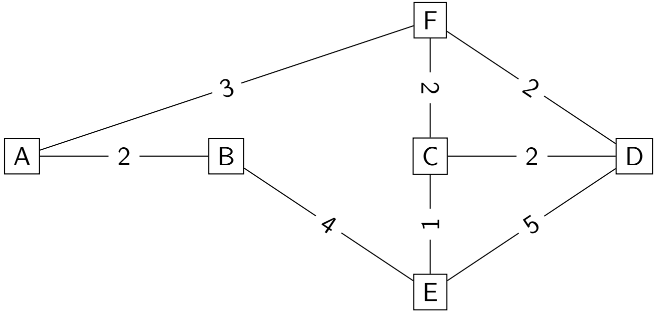
Troisième prédécesseur de C : F

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

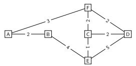
Activité 1 : Continuer de dérouler la première itération de l'algorithme sur le graphe.



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



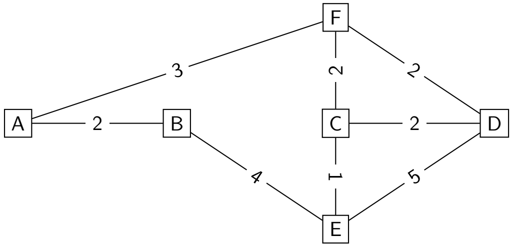
Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Même constat pour D

Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Même constat pour D

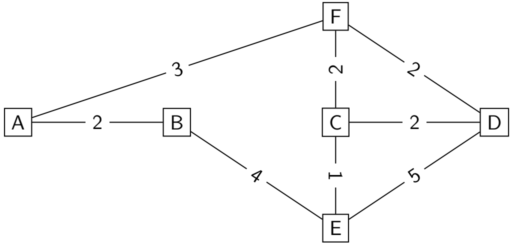
Première itération

A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Premier prédécesseur de E : B

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

Première itération



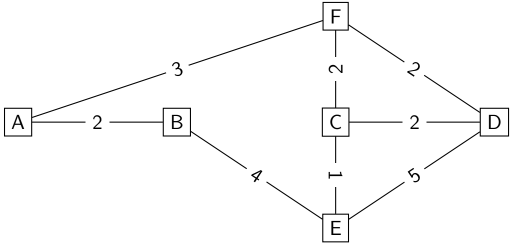
A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Premier prédécesseur de E : B

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

Première itération
Deuxième prédécesseur de E : C
$\infty + 1 = \infty > 6$

Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Deuxième prédécesseur de E : C

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

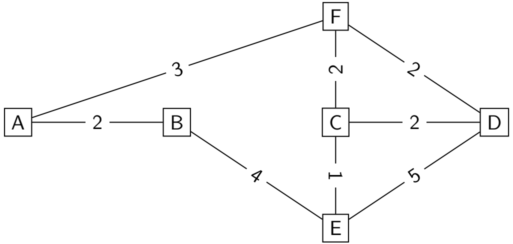
Première itération

A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Troisième prédécesseur de E : D

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

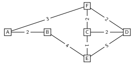
Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Troisième prédécesseur de E : D

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



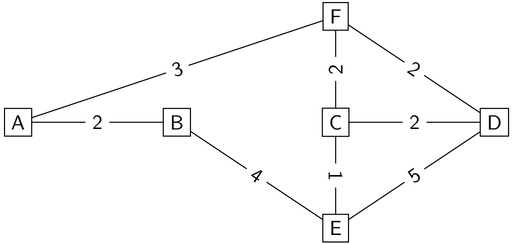
Première itération

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

Premier prédécesseur de F : A

$0 + 3 = 3 < \infty$

Première itération

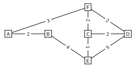


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	**3 (A)**

Premier prédécesseur de F : A

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

Première itération

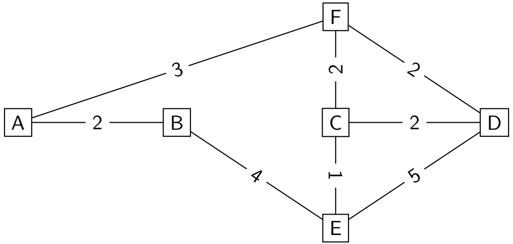


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Deuxième prédécesseur de F : C

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

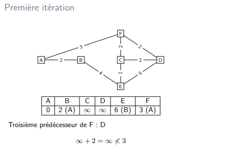
Première itération



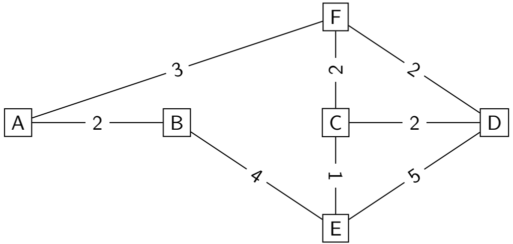
A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Deuxième prédécesseur de F : C

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



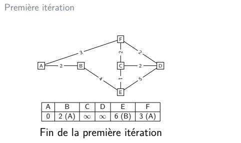
Première itération



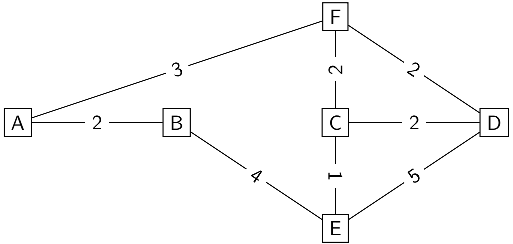
A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Troisième prédécesseur de F : D

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



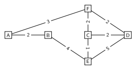
Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Fin de la première itération

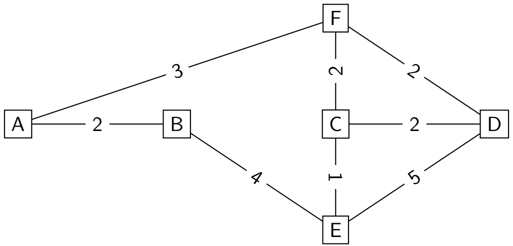
Deuxième itération



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)	∞	∞	6 (B)	3 (A)

Pas de modification pour A et B

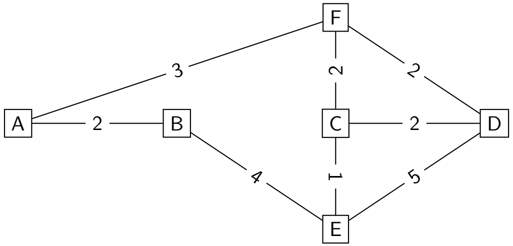
Deuxième itération

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)

Premier prédécesseur de C : D

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

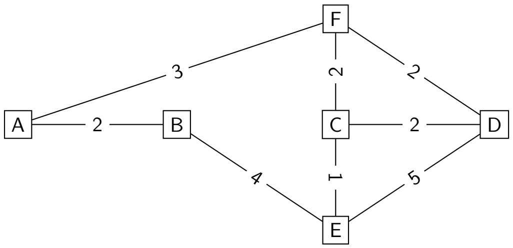
Deuxième itération

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)	**7 (E)**	∞	6 (B)	3 (A)

Deuxième prédécesseur de C : E

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

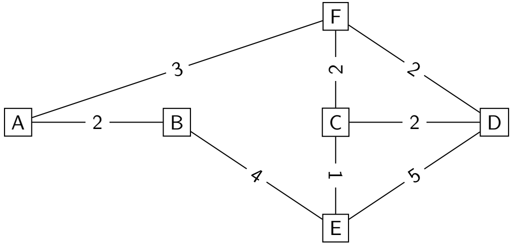
Deuxième itération

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

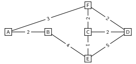


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	**5 (F)**	∞	6 (B)	3 (A)

Troisième prédécesseur de C : F

$$3 + 2 = 5 < 7$$

Deuxième itération

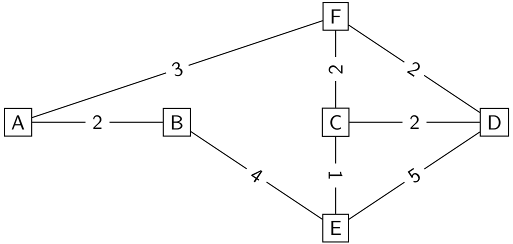


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	**7 (C)**	6 (B)	3 (A)

Premier prédécesseur de D : C

$5+2=7<\infty$

Deuxième itération

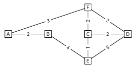


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	**7 (C)**	6 (B)	3 (A)

Premier prédécesseur de D : C

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

Deuxième itération

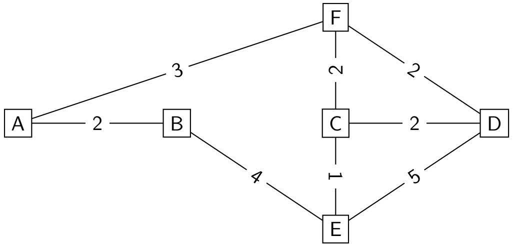


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)

Second prédécesseur de D : E

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

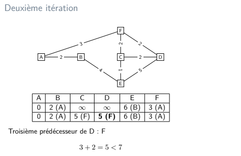
Deuxième itération



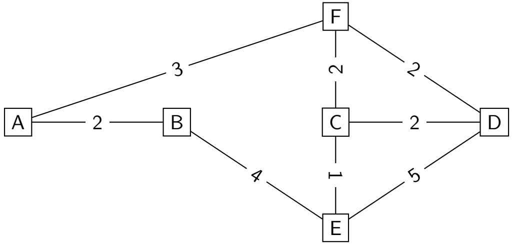
A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)

Second prédécesseur de D : E

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



Deuxième itération

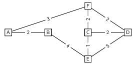


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	**5 (F)**	6 (B)	3 (A)

Troisième prédécesseur de D : F

$$3 + 2 = 5 < 7$$

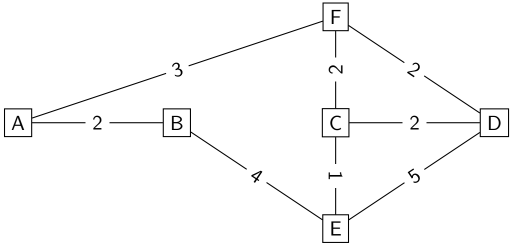
Deuxième itération



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

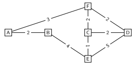
Deuxième itération



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

Pas de changement pour E et F

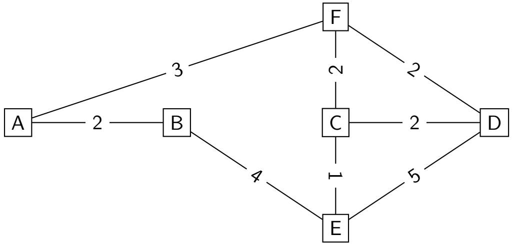
Deuxième itération



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

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

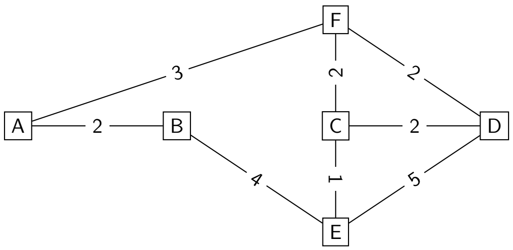
Troisième itération

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

1. amélioration de l'algorithme : on peut s'arrêter là
2. On peut retracer le chemin en partant de la fin : D $\rightarrow$ C $\rightarrow$ F $\rightarrow$ A

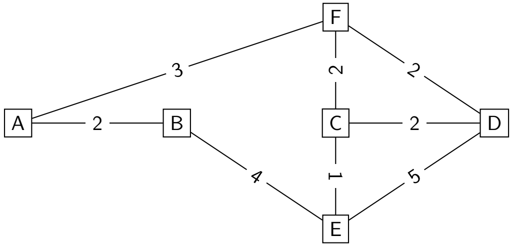
Troisième itération



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

On remonte dans le tableau pour trouver le chemin vers une destination.
$D \to C \to F \to A$



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.

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

2022-01-12

Plus court chemin
└─Protocole RIP : algorithme de Bellman-Ford
  └─Complexité
    └─Sommaire

Sommaire

1. Représentation des réseaux
2. Protocole RIP : algorithme de Bellman-Ford
2.1 Principe
2.2 Mise en application
2.3 Complexité
3. OSPF : algorithme de Dijkstra

Sommaire

ligne 4 = on peut faire une amélioration : si pas de modification pendant l'itération, on peut s'arrêter

Complexité

La complexité dépend de :

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

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

Complexité

La complexité dépend de :

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

```
1 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
```


ligne 4 = on peut faire une amélioration : si pas de modification pendant l'itération, on peut s'arrêter

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
```

$O(S.A)$

précisément chaque arc est parcouru deux fois car on est dans un graphe non orienté

Complexité de l'algorithme de Bellman Ford

$$O(S.A)$$

1. Représentation des réseaux
2. Protocole RIP : algorithme de Bellman-Ford
3. OSPF : algorithme de Dijkstra
3.1 Principe
3.2 Mise en application
3.3 Complexité

Sommaire

1. peut se contenter de renvoyer la distance départ/arrivée (et le chemin)
2. graphe non orienté possible
3. parfois appelé Moore-Dijkstra

OSPF : algorithme de Dijkstra - principe

- Edsger Dijkstra : mathématicien néerlandais
- Algorithme utilisé dans GPS

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.

À retenir

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

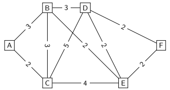


Figure 6 – graphe non orienté et pondéré

Remarque

Les poids de chaque arc représentent le coût de chaque connexion.

coût 5 $\rightarrow$ 20Mbit/s
coût 2 $\rightarrow$ 50Mbit/s

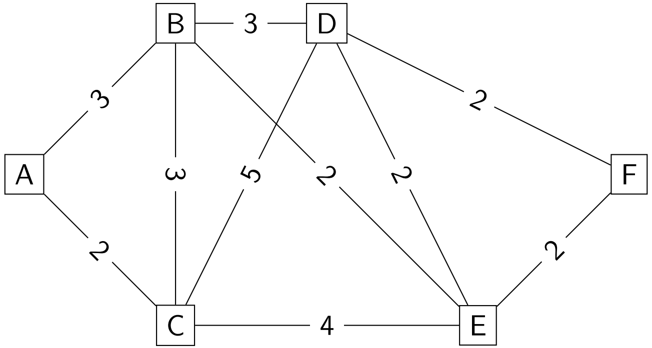


Figure 6 – graphe non orienté et pondéré

Sommaire
1. Représentation des réseaux
2. Protocole RIP : algorithme de Bellman-Ford
3. OSPF : algorithme de Dijkstra
3.1 Principe
3.2 Mise en application
3.3 Complexité

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.

1. S = suivant ; V = voisin
2. ligne 7 : on compare encore la distance déjà enregistrée à distance du voisin + arc

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) ⇒ Mettre à jour la distance de V.

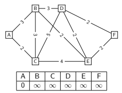
1. S = suivant ; V = voisin
2. ligne 7 : on compare encore la distance déjà enregistrée à distance du voisin + arc

Initialisation :

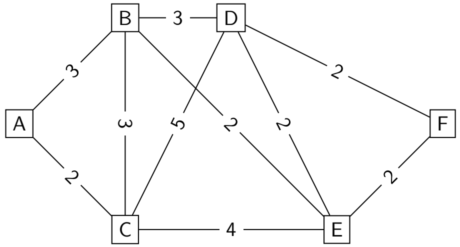
- Créer un tableau des distances entre A et les routeurs, initialisées à l'infini.
- Modifier la distance vers A à 0.

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)
      ⇒ Mettre à jour la distance de V.



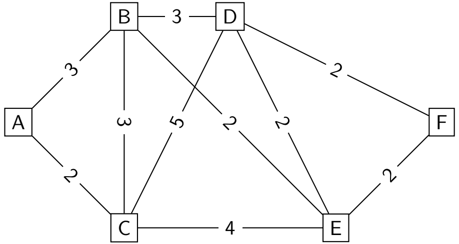
Initialisation



A	B	C	D	E	F
0	∞	∞	∞	∞	∞

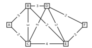
Sélection de A

Sélection de A



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

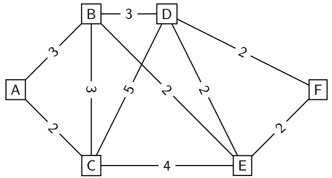
nœuds visités
A



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



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

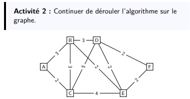
A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	**7 (C)**	**6 (C)**	∞

nœuds visités
A - C

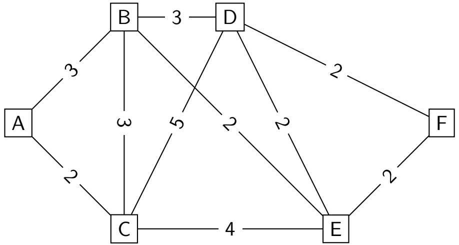
Observation
La route la plus courte a déjà été déterminée pour les nœuds déjà visités. Ils ne seront plus modifiés (cellule grise).

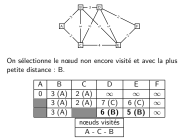
Observation

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



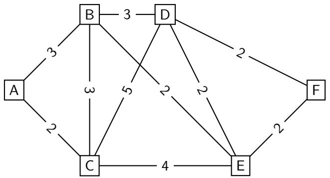
Activité 2 : Continuer de dérouler l'algorithme sur le graphe.





C n'est pas regardé : on a déjà trouvé la + petite distance pour lui = il a déjà été visité

Correction



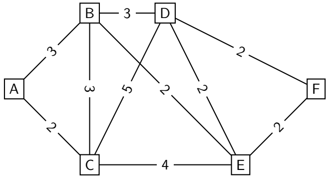
On sélectionne le nœud non encore visité et avec la plus petite distance : B.

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.

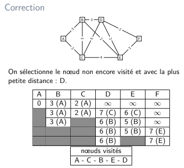
Correction



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

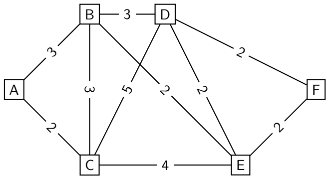
A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		6 (B)	5 (B)	∞
			6 (B)	5 (B)	**7 (E)**

nœuds visités
A - C - B - E



pas de modification

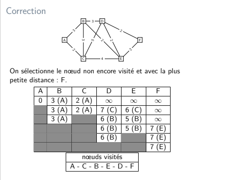
Correction



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

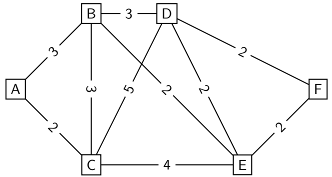
A	B	C	D	E	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		6 (B)	5 (B)	∞
			6 (B)	5 (B)	7 (E)
			6 (B)		7 (E)

nœuds visités
A - C - B - E - D



pas de modification

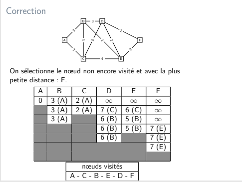
Correction



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)

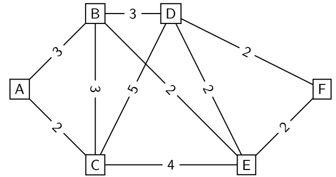
nœuds visités
A - C - B - E - D - F



tous les nœuds ont été visités = fin de l'algorithme. On peut reconstruire le chemin.

pour F : F → E → B → A

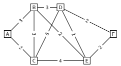
Correction



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)

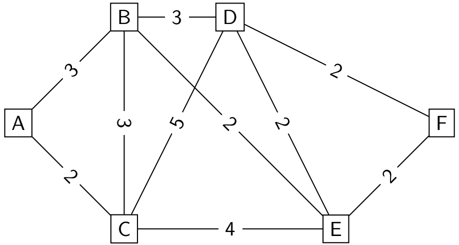
nœuds visités
A - C - B - E - D - F



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.

$$F \rightarrow E \rightarrow B \rightarrow A$$



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.

$$F \rightarrow E \rightarrow B \rightarrow A$$

Sommaire

1. Représentation des réseaux
2. Protocole RIP : algorithme de Bellman-Ford
3. OSPF : algorithme de Dijkstra
3.1 Principe
3.2 Mise en application
3.3 Complexité

Sommaire

1. hors programme
2. utilisation de tas par exemple

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.

1. hors programme
2. utilisation de tas par exemple

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.

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

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.

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

Complexité de l'algorithme de Dijkstra

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

Complexité de l'algorithme de Dijkstra

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