Plus court chemin

Christophe Viroulaud

Terminale NSI

Plus court chemin

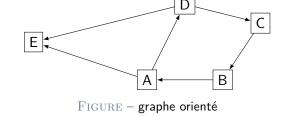
2/50

2021-04-28



Graphe orienté / non orienté

Graphe orienté / non orienté



Plus court chemin

Problématique

Retour des graphes

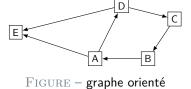
Graphe orienté

rotocole RIP : gorithme de

ise en application omplexité

SPF : algorithme e Dijkstra

1. Le nœud E ne possède pas de *successeur*.



À retenir

Pour chaque nœud on peut définir ses **prédécesseurs** et ses **successeurs**.

Mise en application Complexité

OSPF : algorithme de Dijkstra

Acti

Établir la matrice d'adjacence du graphe figure 2.

Établir le dictionnaire des successeurs du graphe

figure 2.

Déterminer un cycle dans le graphe.

on peut faire dictionnaire des prédécesseurs

Activité 1 :

- 1. Établir la matrice d'adjacence du graphe figure 2.
- 2. Établir le dictionnaire des successeurs du graphe figure 2.
- 3. Déterminer un cycle dans le graphe.

Plus court chemin

Problématique

Graphe orienté

Graphe pondéré

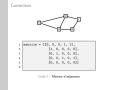
otocole RIP : gorithme de ellman-Ford

Principe Mise en application Complexité

OSPF : algorithme de Dijkstra

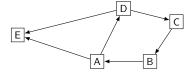
rincipe ise en application

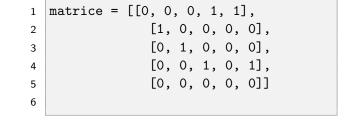
mplexité



La matrice n'est plus symétrique

Correction





Code 1 – Matrice d'adjacence

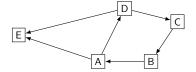
Plus court chemin

Graphe orienté

Correction dico = {"A": {"D", "E"}, "B": {"A"}, "C": {"B"}, "D": {"C", "E"}, "E": set()} Code 2 - Dictionnaire d'adjacence

On peut utiliser des tableaux en place des ensembles.

Correction



```
1 dico = {"A": {"D", "E"},
            "B": {"A"},
            "C": {"B"},
            "D": {"C", "E"},
            "E": set()}
```

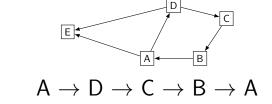
Code 2 – Dictionnaire d'adjacence

Plus court chemin

Graphe orienté

Correction

Correction



Plus court chemin

Problématique

Retour des graphe Graphe orienté

Graphe pondér

Graphe pondér

Sellman-Ford
Principe
Vise en application

SPF: algorithme

Dijkstra

rincipe lise en application omplexité



pondération peut être négative

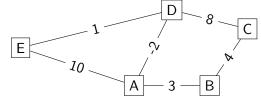


FIGURE – graphe non orienté pondéré

Plus court chemin

Graphe pondéré

Activité 2 :

- 1. Établir la matrice d'adjacence du graphe figure 3.
- Établir le dictionnaire d'adjacence du graphe figure
 3.

Plus court chemin

Problématique

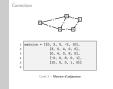
Retour des graphes Graphe orienté

Graphe pondéré

otocole RIP : gorithme de ellman-Ford

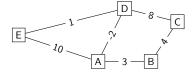
se en application mplexité

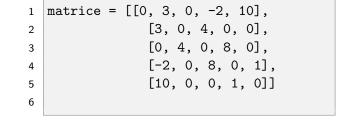
SPF: algorithme Dijkstra



La matrice est symétrique

Correction





Code 3 – Matrice d'adjacence

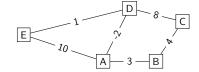
Plus court chemin

Graphe pondéré



Dictionnaire de dictionnaires

Correction



```
dico = {"A": {"B": 3, "D":
     -2, "E": 10},
          "B": {"A": 3, "C": 4},
          "C": {"B": 4, "D": 8},
          "D": {"A": -2, "C": 8, "E":
     1},
          "E": {"D": 1, "E": 10}}
6
```

Code 4 – Dictionnaire d'adjacence

Plus court chemin

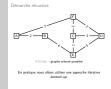
Graphe pondéré

-Algorithme de Bellman-Ford : Fin des années 50

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.

Plus court chemin

Principe



1. Bellman « père de l'approche dynamique »

Remarque

2. Dans le graphe figure 4 les pondérations représentent un nombre de routeurs traversés pour atteindre le nœud (routeur) suivant.

Démarche récursive

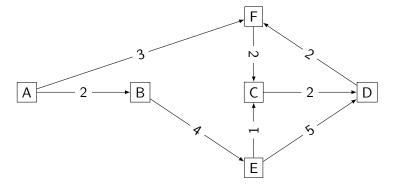


FIGURE – graphe orienté pondéré

En pratique nous allons utiliser une approche itérative bottom-up.

Plus court chemin

Problématique

Graphe orienté
Graphe pondéré

Protocole RIP : Ilgorithme de Bellman-Ford

Principe
Mise en application
Complexité

OSPF : algorithme de Dijkstra

Pour chaque routeur

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...

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

Plus court chemin

Principe

Plus court chemin —Protocole RIP : algorithme de Bellman-Ford —Mise en application

iéer un tableau den distances entre A et les routeurs (A inclus .), initialisées à l'était.

In italisées à l'était.

In tableau modifier le distance vers A à 0.

Int que (le nombre d'étaision) < (nombre de routeurs)

Pour chaque ar ce de graphe

Si (la distance du routeur) > (la distance de son prédicesser » pois de l'àre untre le noter routeurs)

Le distance du routeur » (la distance de son prédicesser » pois de l'àre untre le noter routeurs)

Le distance du routeur est rereplacée par cette nouvelle valeur.

Code 5 - Algorithme de Bellman Ford

- 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

Créer un tableau des distances entre A et les routeurs (A inclus), initialisées à l'infini.

Dans le tableau modifier la distance vers A à 0.

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

Pour chaque arc du graphe
Si (la distance du routeur) > (la distance de son prédé cesseur + poids de l'arc entre les deux routeurs)

La distance du routeur est remplacée par cette nouvelle valeur

Code 5 – Algorithme de Bellman Ford

Plus court chemin

Problématique

Graphe orienté
Graphe pondéré

gorithme de ellman-Ford

Mise en application

OSPF : algorithme de Dijkstra

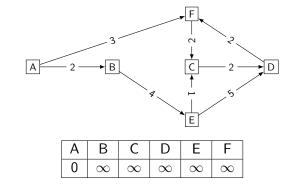
Principe Mise en application

16 / 50

2021-04-28



Initialisation



Plus court chemin

Problématique

Retour des graphes Graphe orienté

Graphe pondér

Protocole RIP : algorithme de Bellman-Ford

Mise en application

Complexité

OSPF : algorithme de Dijkstra

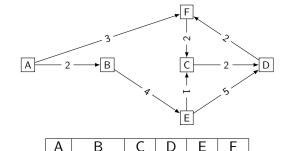
Plus court chemin Protocole RIP : algorithme de Bellman-Ford Mise en application Première itération



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

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

Première itération



 ∞

 ∞

 ∞

 ∞

2 (A)

Plus court chemin

Problématique

Retour des graphes
Graphe orienté

Protocole RIP : Ilgorithme de Bellman-Ford

Mise en application

OSPF : algorithme de Diikstra

Activité 3 : Continuer de dérouler l'algorithme sur le graphe figure 4.

Plus court chemin

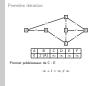
blématique

Graphe orienté Graphe pondéré

Bellman-Ford
Principe
Mise en application

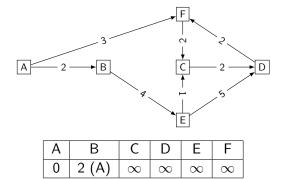
omplexité SPF : algorithme

Dijkstra cipe en application



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

Première itération



Premier prédécesseur de C : E

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

Plus court chemin

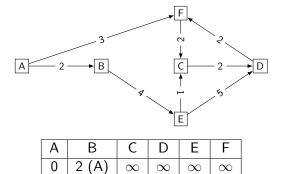
Mise en application

20 / 50

Plus court chemin -Protocole RIP: algorithme de Bellman-Ford ☐Mise en application -Première itération



Première itération



Second prédécesseur de C : F

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

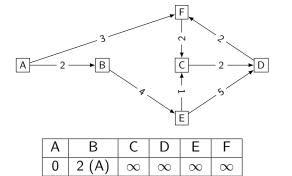
Plus court chemin

Mise en application

21 / 50



Première itération



Même constat pour D

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Protocole RIP : algorithme de

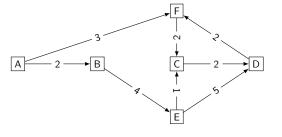
Mise en application

OSPF : algorithme de Dijkstra

Plus court chemin Protocole RIP: algorithme de Bellman-Ford Mise en application Première itération



Première itération



Α	В	С	D	E	F
0	2 (A)	∞	∞	6 (B)	∞

Prédécesseur de E : B

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

Plus court chemin

Droblómatique

Retour des graphes Graphe orienté

rotocole RIP : gorithme de ellman-Ford

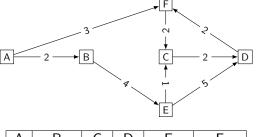
Mise en application Complexité

OSPF : algorithme de Dijkstra

Plus court chemin Protocole RIP : algorithme de Bellman-Ford Mise en application Première itération



Première itération



A	В	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Premier prédécesseur de F : A

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

rotocole RIP : Igorithme de Jellman-Ford

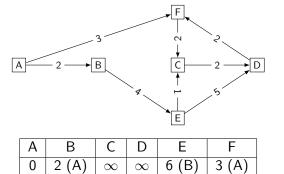
Mise en application Complexité

OSPF : algorithme de Dijkstra

Plus court chemin Protocole RIP : algorithme de Bellman-Ford Mise en application Première itération



Première itération



						_	
Second pré					г. г	`	
secona pre	NACE	CCAI	ir (78	- · ı	,	
)—(() () +	~	. ~ ~ - 1	11 (,	

$$\infty + 2 = \infty \nless 3$$

Plus court chemin

Problématique

Retour des graphes Graphe orienté

> rotocole RIP : Igorithme de sellman-Ford

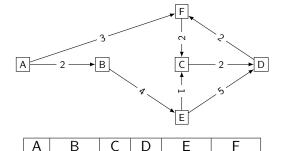
Mise en application

OSPF : algorithme de Dijkstra

Plus court chemin —Protocole RIP : algorithme de Bellman-Ford —Mise en application —Première itération



Première itération



Fin de la première itération

 ∞

 ∞

6 (B)

2 (A)

3 (A)

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Protocole RIP :

Principe

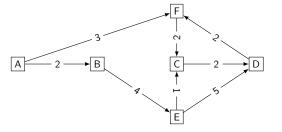
Mise en application

OSPF: algorithme

OSPF : algorithme de Dijkstra



Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	∞	∞	6 (B)	3 (A)

Pas de modification pour A et B

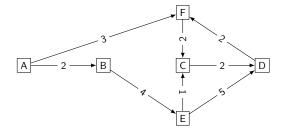
Plus court chemin

Mise en application

Plus court chemin —Protocole RIP : algorithme de Bellman-Ford —Mise en application —Deuxième itération



Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	7 (E)	∞	6 (B)	3 (A)

Premier prédécesseur de C : E

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

rotocole RIP : Igorithme de sellman-Ford

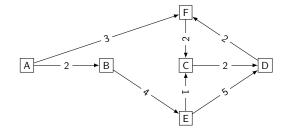
Mise en application Complexité

OSPF : algorithme de Dijkstra

Principe
Mise en application
Complexité



Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	∞	6 (B)	3 (A)

Second prédécesseur de C : F

$$3+2=5<7$$

Plus court chemin

Problématique

Retour des graphes Graphe orienté

gorithme de ellman-Ford

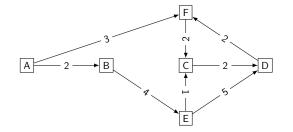
Mise en application

OSPF : algorithme de Dijkstra

Plus court chemin —Protocole RIP : algorithme de Bellman-Ford —Mise en application —Deuxième itération



Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)

Premier prédécesseur de D : C

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

Plus court chemin

Problématique

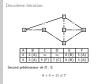
Retour des graphes Graphe orienté

algorithme de Bellman-Ford

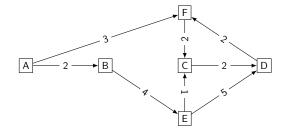
Mise en application
Complexité

OSPF : algorithme de Dijkstra

Plus court chemin Protocole RIP : algorithme de Bellman-Ford Mise en application Deuxième itération



Deuxième itération



Α	В	С	D	Е	F	
0	2 (A)	∞	∞	6 (B)	3 (A)	
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)	

Second prédécesseur de D : E

$$6 + 5 = 11 \nless 7$$

Plus court chemin

Problématique

Retour des graphes Graphe orienté

algorithme de Bellman-Ford

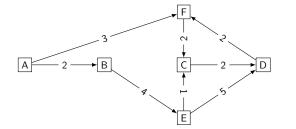
Mise en application Complexité

OSPF : algorithme de Dijkstra

Plus court chemin —Protocole RIP : algorithme de Bellman-Ford —Mise en application —Deuxième itération



Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)

Pas de changement pour E et F

Plus court chemin

Problématique

Graphe orienté

algorithme de Bellman-Ford

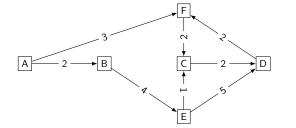
Mise en application

OSPF : algorithm de Dijkstra

Plus court chemin Protocole RIP : algorithme de Bellman-Ford Mise en application Deuxième itération



Deuxième itération



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

Fin de la deuxième itération

Plus court chemin

Problématique

Retour des graphes Graphe orienté

> Protocole RIP : algorithme de Bellman-Ford

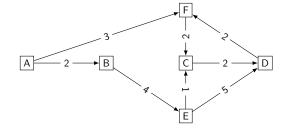
Mise en application

OSPF : algorithmede Dijkstra

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



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

Pas de changement lors de la troisième itération

Plus court chemin

Problématique

Retour des graphes
Graphe orienté

algorithme de Bellman-Ford Principe

Mise en application Complexité

OSPF : algorithme de Dijkstra

Principe Mise en application Complexité



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

du nombre de sommets (notée S) : on visite chaque sommet (ligne 4);

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

> gorithme de Illman-Ford Incipe

Complexité

OSPF : algorithme



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

▶ du nombre de sommets (notée S) : on visite chaque sommet (ligne 4);

```
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 (ligne 5).

```
Pour chaque arc du graphe
```

Plus court chemin

Complexité

Complexité de l'algorithme de Bellman Ford

O(S.A)

èmatique

Plus court chemin

tour des graphe

Graphe pondéré

nan-Ford pe n application

Complexité

OSPF : algorithme de Dijkstra

Dijkstra cipe en application

36 / 50

Edsger Dijkstra

principe : construire un sous-graphe en ajoutant à chaque

itération un sommet de distance minimale.

Plus court chemin

Principe

principe : construire un sous-graphe en ajoutant à chaque

Edsger Diikstra

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



Remarque

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

 $\begin{array}{l} \text{coût 5} \rightarrow 20 \text{Mbit/s} \\ \text{coût 2} \rightarrow 50 \text{Mbit/s} \end{array}$

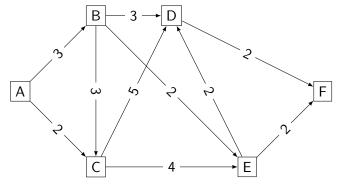


FIGURE – graphe orienté et pondéré

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Protocole RIP : algorithme de Bellman-Ford

Mise en application Complexité

OSPF : algorithme de Dijkstra

Principe

Criter on tablese den dificaces auten A et les rentreus (A inclus), incluinless l'erfeni.

Ten tables mediatris d'étances even A à il.

Ten qu'il mai technes mediatris d'étances even A à il.

Ten qu'il ma des rentemes non-silenticosisés.

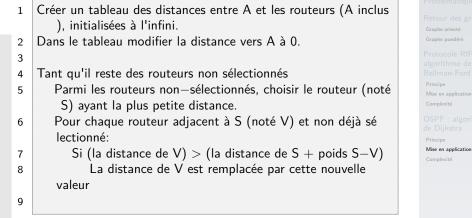
Ten qu'il ma des rentemes non-silenticosisés.

Ten qu'il ma silentiesse, décisé le renteur (roist Pour chapes moisters algient 1.5 (roist 3) y et con diju et lectrones.

S (to destance de V) - chi delamen de S + poids S - V) se defances de S + poids S - V) se defances de V en respitable per rette mountle subteur.

Contri 1 - Algorithms de Dijuless.

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



Code 6 – Algorithme de Dijkstra

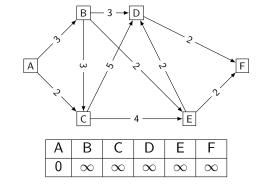
/ 50

Plus court chemin

2021-04-28

Initialisation

Initialisation



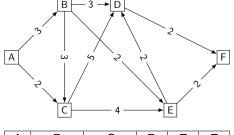
Plus court chemin

Principe

2021-04-28



Sélection de A



Α	В		С	D	E	F	
0	3 (A)		2 (A) ∞		∞	∞	
nœuds visités							
			Α				

Plus court chemin

Mise en application

41 / 50

Activité 4 : Continuer de dérouler l'algorithme sur le graphe figure 5.

Plus court chemin

our des graphes

Graphe pondéré
Protocole RIP

Principe Mise en application

PF: algorithme

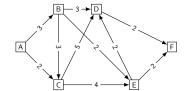
Dijkstra incipe

Mise en application
Complexité



cellule grisée = nœud déjà visité, on a déjà sa route la + courte

Correction



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

Α	В	С	D	Е	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞

nœuds visités A - C

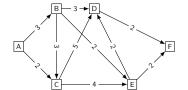
Plus court chemin

Plus court chemin OSPF: algorithme de Dijkstra Mise en application Correction



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.

Α	В	С	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 Plus court chemin

Problématiqu

Retour des graphe
Graphe orienté
Graphe pondéré

Igorithme de
Bellman-Ford
Principe
Mise en application

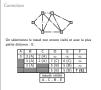
Complexité

OSPF: algorithme

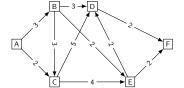
de Dijkstra

Principe
Mise en application

Plus court chemin OSPF: algorithme de Dijkstra Mise en application Correction



Correction



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

Α	В	С	D	Е	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 Plus court chemin

Problématique

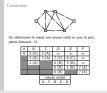
etour des graphe

Graphe pondéré

algorithme de Bellman-Ford Principe

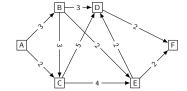
Mise en application Complexité

OSPF : algorithme de Dijkstra



pas de modification

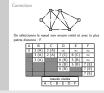
Correction



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

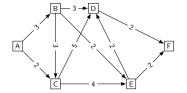
Α	В	С	D	Е	F
0	3 (A)	2 (A)	∞	∞	∞
	3 (A)	2 (A)	7 (C)	6 (C)	∞
	3 (A)		6 (B)	5 (B)	∞
			6 (B)	5 (B)	7 (E)
			6 (B)		7 (E)

nœuds visités A - C - B - E - D Plus court chemin



pas de modification

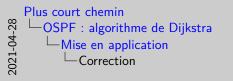
Correction

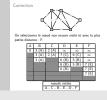


On sélectionne le nœud non encore visité et avec la plus petite distance : F.

Α	В	С	D	Е	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 Plus court chemin

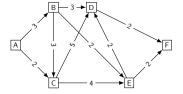




tous les nœuds ont été visités = fin de l'algorithme. On peut reconstruire le chemin.

pour $F: F \rightarrow E \rightarrow B \rightarrow A$

Correction



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

Α	В	С	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

48 / 50

Plus court chemin

nombre d'arcs A.

- 1. hors programme
- 2. utilisation de tas par exemple

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

Plus court chemin

Complexité

- 1. hors programme
- 2. utilisation de tas par exemple

- ► La complexité dépend du nombre de sommets S et du
- 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.

- 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.

blématique

robiematique

Plus court chemin

Graphe orienté
Graphe pondéré

lgorithme de ellman-Ford

Mise en application Complexité

SPF : algorithme e Dijkstra

incipe ise en application

Complexité

(

Complexité de l'algorithme de Diikstra

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

Complexité de l'algorithme de Dijkstra

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

Plus court chemin

oblématique

Graphe orienté

otocole RIP : gorithme de ellman-Ford

en application
plexité
PF: algorithme

Dijkstra incipe

Principe
Mise en application
Complexité