Plus court chemin Christophe Viroulaud Terminale NSI

Plus court chemin

Christophe Viroulaud

Terminale NSI

Plus court chemin

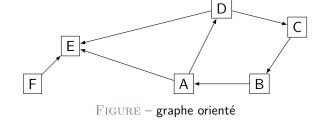
Plus court chemin

Problématique

Graphe orienté / non orienté



Graphe orienté / non orienté



Plus court chemin

Problematique

Retour des graphes Graphe orienté

Graphe pondér

Graphe pondéré

gorithme de ellman-Ford rincipe

SPF: algorithme

SPF : algorithme e Dijkstra

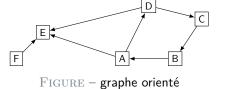
rincipe lise en application omplexité

Plus court chemin -Retour des graphes Graphe orienté



- 1. Le nœud F ne possède pas de prédécesseur.
- 2. Le nœud E ne possède pas de successeur.





À retenir

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

Plus court chemin

Graphe orienté

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

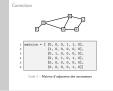
Retour des graphes Graphe orienté

Graphe pondéré

gorithme de ellman-Ford incipe

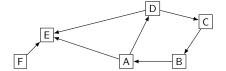
Mise en application Complexité

OSPF : algorithme de Dijkstra



La matrice n'est plus symétrique

Correction

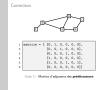


```
matrice = [ [0, 0, 0, 1, 1, 0],
            [1, 0, 0, 0, 0, 0],
            [0, 1, 0, 0, 0, 0],
            [0, 0, 1, 0, 1, 0],
            [0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0]
```

Code 1 – Matrice d'adjacence des successeurs

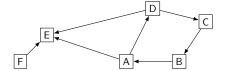
Plus court chemin

Graphe orienté



La matrice n'est plus symétrique

Correction



Code 2 – Matrice d'adjacence des **prédécesseurs**

Plus court chemin

Problématique

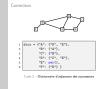
Retour des graphes

Graphe orienté
Graphe pondéré

Protocole RIP : algorithme de

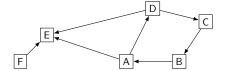
Principe Mise en application Complexité

DSPF : algorithme le Dijkstra



On peut utiliser des tableaux en place des ensembles.

Correction



Code 3 – Dictionnaire d'adjacence des successeurs

Plus court chemin

Problématique

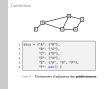
Retour des graphes Graphe orienté

Graphe pondén

Protocole RIP : algorithme de

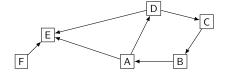
Principe Mise en application

OSPF : algorithme de Dijkstra



On peut utiliser des tableaux en place des ensembles.

Correction



Code 4 – Dictionnaire d'adjacence des **prédécesseurs**

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Graphe pondén

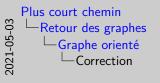
Protocole RIP : algorithme de Bellman-Ford

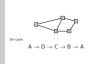
Principe Mise en application Complexité

DSPF : algorithme de Dijkstra

Principe Mise en application

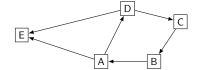
9 / 52





Correction

Correction



Un cycle

$$\mathsf{A}\to\mathsf{D}\to\mathsf{C}\to\mathsf{B}\to\mathsf{A}$$

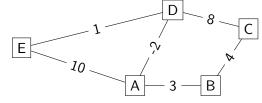
Plus court chemin

Graphe orienté

10 / 52



pondération peut être négative



 $\label{eq:figure} Figure - \text{graphe non orient\'e pond\'er\'e}$

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Graphe pondéré

Graphe pondere

lgorithme de Jellman-Ford

Mise en application Complexité

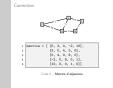
SPF : algorithme Dijkstra

incipe

incipe se en application mplexité

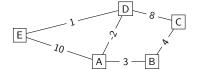
Activité 2 :

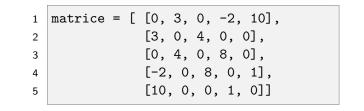
- 1. Établir la matrice d'adjacence du graphe figure 3.
- 2. Établir le dictionnaire d'adjacence du graphe figure



La matrice est symétrique

Correction





Code 5 – Matrice d'adjacence

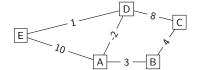
Plus court chemin

Graphe pondéré



Dictionnaire de dictionnaires

Correction



```
dico = {\text{"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}}
```

Code 6 – Dictionnaire d'adjacence

Plus court chemin

Graphe pondéré

Protocole RIP : algorithme de Bellman-Ford
Principe

incipe

- 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.

Algorithme de Bellman-Ford : Fin des années 50

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

Problèmatique

Retour des graphe Graphe orienté

Graphe pondéré

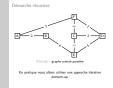
rotocole RIP : gorithme de ellman-Ford

Principe Mise en application

SPF : algorithme

Dijkstra

Plus court chemin —Protocole RIP : algorithme de Bellman-Ford —Principe —Démarche récursive



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

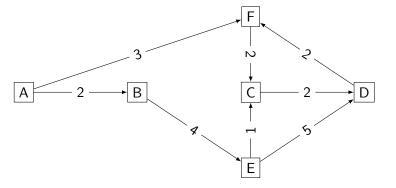


FIGURE – graphe orienté pondéré

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

Plus court chemin

Problématique

Graphe orienté
Graphe pondéré

lgorithme 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

sier un tableau des distances entre A et les rootsers (A nobis), includiers à l'infeil.

In tella modifier de distance vers A à 0.

et que l'es nombre d'étations) (combre de rootsers)

et que l'es nombre d'étations) (combre de rootsers)

Si (la distance du souteur) ou de distance de souteurs

Si (la distance du souteur) (de distance de sou prédictance de souteur) pois de l'étatione de souteur) et de distance de souteur pois de l'est centrée la deux contents)

La distance de rootsers et completés par cette nouteurs)

Code 7 - 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 7 – Algorithme de Bellman Ford

Plus court chemin

Problématique

Graphe orienté Graphe pondéré

Bellman-Ford
Principe

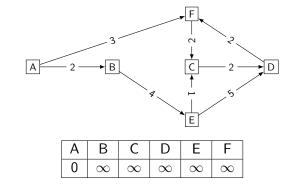
Mise en application Complexité

OSPF : algorithme de Dijkstra

Principe Mise en application

18 / 52

Initialisation



Plus court chemin

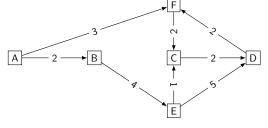
Mise en application



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



Α	В	С	D	Е	F
0	2 (A)	∞	∞	∞	∞

Plus court chemin

Problématique

Retour des graphes
Graphe orienté

'rotocole RIP : Igorithme de Bellman-Ford

Mise en application

DSPF : algorithme de Dijkstra

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

Plus court chemin

roblématique

Graphe orienté

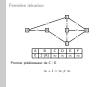
rotocole RIP :

Principe Mise en application

SPF: algorithme

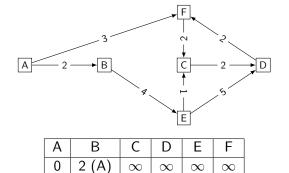
Dijkstra

21 / 52



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

Retour des graphes Graphe orienté

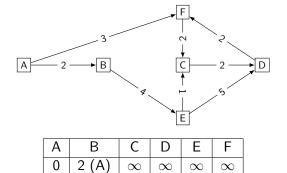
Protocole RIP : algorithme de Bellman-Ford

Mise en application Complexité

OSPF : algorithme de Dijkstra



Première itération



Second prédécesseur de C : F

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

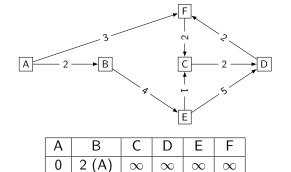
> rotocole RIP : gorithme de ellman-Ford

Mise en application
Complexité

DSPF : algorithme de Dijkstra



Première itération



Même constat pour D

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Protocole RIP : algorithme de

Principe Mise en application

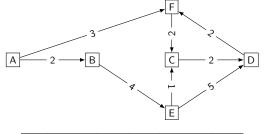
OSPF : algorithme de Dijkstra

Principe Mise en application

24 / 52



Première itération



Α	В	С	D	E	F
0	2 (A)	∞	∞	6 (B)	∞

Prédécesseur de E : B

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

rotocole RIP : gorithme de ellman-Ford

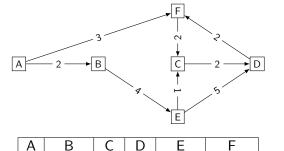
Mise en application Complexité

OSPF : algorithme de Dijkstra

Principe
Mise en application
Complexité



Première itération



 ∞

6 (B)

3 (A)

Premier	prédécesseur	de	F	:	Α

2 (A)

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

 ∞

Plus court chemin

Problématique

Retour des graphes Graphe orienté

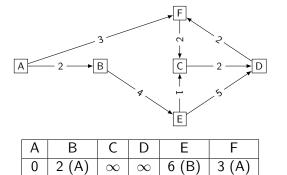
> rotocole RIP : gorithme de ellman-Ford

Mise en application Complexité

OSPF : algorithme de Dijkstra



Première itération



Second prédécesseur de F : D

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

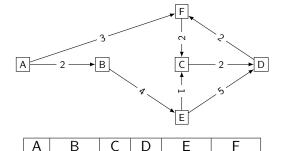
> rotocole RIP : gorithme de ellman-Ford

Mise en application Complexité

OSPF : algorithme de Dijkstra



Première itération



Fin de la première itération

 ∞

 ∞

6 (B)

3 (A)

В

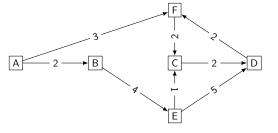
2 (A)

Plus court chemin

Mise en application



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

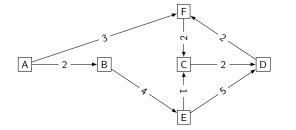
Plus court chemin

Mise en application

29 / 52



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

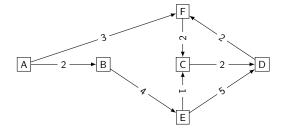
> gorithme de ellman-Ford

Mise en application

OSPF : algorithme de Dijkstra



Deuxième itération



Α	В	С	D	E	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

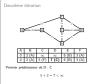
Problématique

Retour des graphes Graphe orienté

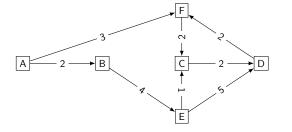
> rotocole RIP : gorithme de ellman-Ford

Mise en application

OSPF : algorithme de Dijkstra



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

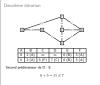
Problématique

Retour des graphes Graphe orienté

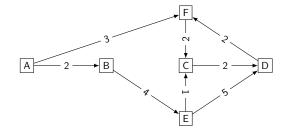
> Protocole RIP : algorithme de Bellman-Ford

Mise en application
Complexité

OSPF : algorithme de Dijkstra



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

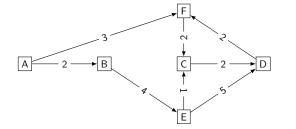
> gorithme de ellman-Ford

Mise en application Complexité

OSPF : algorithme de Dijkstra



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

Retour des graphes
Graphe orienté
Graphe pondésé

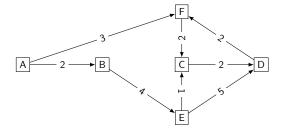
Igorithme de Bellman-Ford

Mise en application

OSPF : algorithmede Dijkstra



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

Graphe pondéré

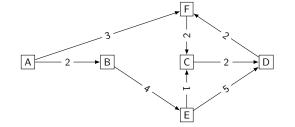
algorithme de Bellman-Ford Principe

Mise en application Complexité

OSPF : algorithme de Dijkstra

- 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	С	D	Е	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

Mise en application Complexité

OSPF : algorithme de Dijkstra

 du nombre de sommets (notée S) : on visite chaque sommet (ligne 4) :
 Tant que (le nombre d'inérations) < (nombre de routeurs)

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 pondéré

orithme de Ilman-Ford

Principe
Mise en application
Complexité

OSPF: algorithme

incipe

Principe Mise en application Complexité ► du nombre de sommets (notée S) : on visite chaque Tant que (le nombre d'itérations) < (nombre du nombre d'arcs (notée A) : pour chaque sommet on regarde tous les arcs du graphe (ligne 5). Pour chaque arc du graphe

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);
 - 1 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

 $\mathcal{I}(S.A)$

iematique

Plus court chemin

Retour des graphes Graphe orienté

Graphe pondéré
Protocole Ri

Principe
Mise en application
Complexité

PF: algorithme Dijkstra

Dijkstra cipe en application 2021-05-03

principe : construire un sous-graphe en ajoutant à chaque

Plus court chemin

Principe

1. peut se contenter de renvoyer la distance départ/arrivée (et le chemin)

Edsger Diikstra

principe : construire un sous-graphe en ajoutant à chaque

- 2. graphe non orienté possible
- 3. parfois appelé Moore-Dijkstra



Remarque

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

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

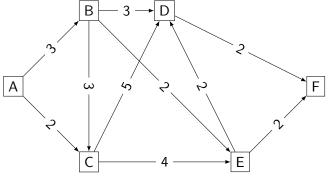


FIGURE – graphe orienté et pondéré

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Protocole RIP : algorithme de Bellman-Ford

lise en application omplexité

OSPF : algorithme de Dijkstra

Principe

rincipe ise en application

Plus court chemin OSPF: algorithme de Dijkstra Mise en application

order un tableau den distancese entre A et les routeaurs (A inclus.

). Intableaires à Priefel.

aut to blance au l'autre de l'autre en et à 0.0

est qu'il reau des routeaus non sélectionnées

set qu'il reau des routeaus non sélectionnées

5) syeat la plus perite déstance.

Pour chappe routear adjonnée 15 (noué 1) y et one digit à utilité.

\$1 (il distance de V) (c) (il distance de 5 points 5 v) (a distance de 5 points 5 v)

La distance du V est remplacée par cette nouveille valeur.

Code 8 – Algorithme de Dijkstra

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

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 qu'il reste des routeurs non sélectionnés

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

Pour chaque routeur adjacent à S (noté V) et non déjà sé lectionné:

Si (la distance de V) > (la distance de S + poids S-V)

Code 8 – Algorithme de Dijkstra

valeur

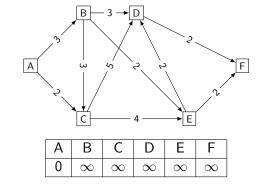
La distance de V est remplacée par cette nouvelle

Plus court chemin

2021-05-03



Initialisation



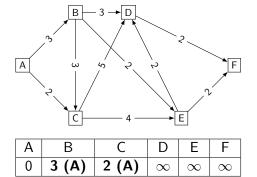
Plus court chemin

Principe Mise en application

2021-05-03



Sélection de A



nœuds visités

Plus court chemin

Mise en application

43 / 52

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

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

tour des graphe

Plus court chemin

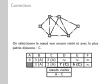
Protocole R

Mise en application Complexité

PF : algorithme Dijkstra

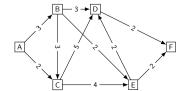
Dijkstra ncipe

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

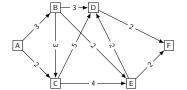
Mise en application

45 / 52



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	Е	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

Bellman-Ford
Principe

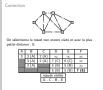
Complexité

OSPE : algorithme

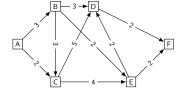
OSPF : algorithme de Dijkstra

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

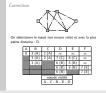
Problématique

Retour des graphe Graphe orienté

algorithme de Bellman-Ford

Principe Mise en application

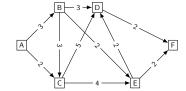
OSPF : algorithme de Dijkstra



pas de modification

2021-05-03

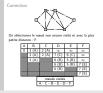
Correction



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)

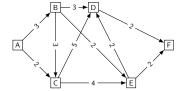
nœuds visités A - C - B - E - D Plus court chemin



pas de modification

2021-05-03

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 Plus court chemin

Problématique

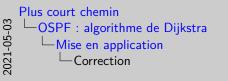
Retour des graphe Graphe orienté

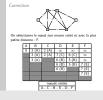
rotocole RIP : gorithme de

Principe Mise en application

DSPF : algorithme de Dijkstra

Principe

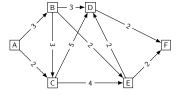




tous les nœuds ont été visités = fin de l'algorithme. On peut reconstruire le chemin.

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	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

- 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

Plus court chemin OSPF: algorithme de Dijkstra -Complexité

- nombre d'arcs A. ► Le point clé de l'algorithme tient dans la recherche de la
- Parmi les routeurs non-sélectionnés, choisir le routeur (noté S) avant la plus petite distance

La complexité dépend du nombre de sommets S et du

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

- 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.

Plus court chemin

Complexité

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 dexité F: algorithme

Dijkstra

Principe
Mise en application
Complexité