### Plus court chemin

Christophe Viroulaud

Terminale NSI

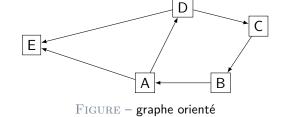
Plus court chemin

2/50

E A E

Graphe orienté / non orienté

Graphe orienté / non orienté



Plus court chemin

Problématique

Retour des graphes

Graphe orienté

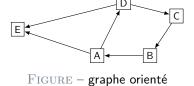
Protocole RIP

Principe
Mise en application

SPF : algorithme e Dijkstra

- 1. Le nœud A 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.

#### Activité 1 :

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Graphe pondéré

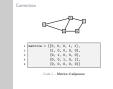
Protocole RIP : algorithme de Bellman-Ford

incipe ise en application omplexité

DSPF : algorithme le Dijkstra

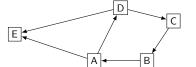
Principe Mise en application

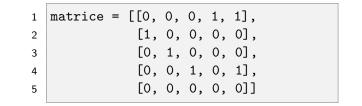
ise en application omplexité 2021-04-05



La matrice n'est plus symétrique

#### Correction





Code 1 – Matrice d'adjacence

Plus court chemin

Problématique

Retour des graphes

Graphe orienté

Graphe pondér

rotocole RIP : gorithme de ellman-Ford

Principe
Mise en application

OSPF : algorithme

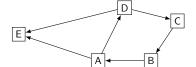
e Dijkstra Principe

2021-04-05

"D": {"C", "E"}, "E": set()} Code 2 - Dictionnaire d'adiacence

On peut utiliser des tableaux en place des ensembles.

#### Correction



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

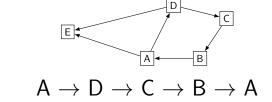
Code 2 – Dictionnaire d'adjacence

Plus court chemin

Graphe orienté

Correction

### Correction



Plus court chemin

Problématique

Retour des graphe

Graphe orienté

Protocole RII

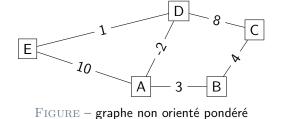
gorithme de ellman-Ford rincipe

omplexité
SPF: algorithme

SPF : algorithme e Dijkstra



pondération peut être négative

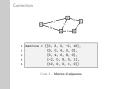


Plus court chemin

Graphe pondéré

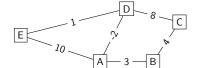
### Activité 2 :

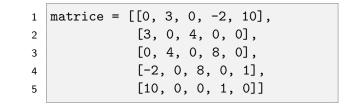
- 1. Établir la matrice d'adjacence du graphe figure 3.
- 2. Établir le dictionnaire d'adjacence du graphe figure



La matrice est symétrique

#### Correction



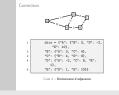


Code 3 – Matrice d'adjacence

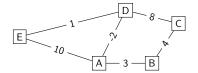
Plus court chemin

Graphe pondéré

Dictionnaire de dictionnaires



#### Correction



Code 4 – Dictionnaire d'adjacence

raphes

our des graphe

Plus court chemin

Graphe pondéré

Graphe pondéré

rotocole RIP : Igorithme de sellman-Ford

se en application mplexité

SPF : algorithme Dijkstra <sub>incipe</sub>

incipe se en application mplexité

12 / 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.

Fin des années 50

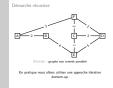
Fin des années 50

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

# 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

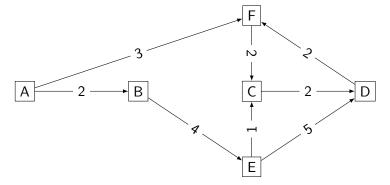


FIGURE – graphe non orienté pondéré

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

Plus court chemin

Problématique

Retour des graphes
Graphe orienté
Graphe pondéré

algorithme de Bellman-Ford

Vlise en application Complexité

SPF : algorithme e Dijkstra

peut servir pour retrouver chemin, distance mini...

Pour chaque routeur

Pour chaque routeur

On obtient un tableau contenant la distance minimale entre

le routeur de départ et chaque autre routeur.

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

Plus court chemin

Principe

15 / 50

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

ier un tableau den distazons estre A et les routeurs (A incha), intinà in en a l'infai.

nole i tableau modifier la distazon vers A à 0.

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

Four chaque arc dei graphe

Si (la distazon du routeurs) > (la distazon de son prédicenser + peids de l'arc estre les deux centre de la distazon de routeurs).

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), initialis ées à l'infini. Dans le tableau modifier la distance vers A à 0. 3 Tant que (le nombre d'itérations) < (nombre de routeurs) Pour chaque arc du graphe Si (la distance du routeur) > (la distance de son prédécesseur + 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

> algorithme de Bellman-Ford Principe

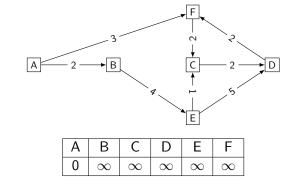
Mise en application Complexité

e Dijkstra rincipe

Principe Mise en application Complexité 2021-04-05



### Initialisation



Plus court chemin

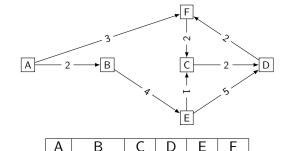
Mise en application



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



 $\infty$ 

 $\infty$ 

 $\infty$ 

 $\infty$ 

2 (A)

Plus court chemin

Mise en application

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

Plus court chemin

blématique

Graphe orienté Graphe pondéré

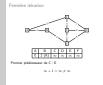
Mise en application

orithme de Iman-Ford

mplexité SPF: algorithme

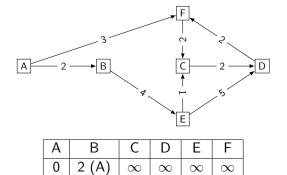
> Dijkstra ipe en application

> > 19 / 50



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

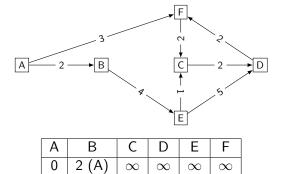
Protocole RIP : algorithme de Bellman-Ford

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

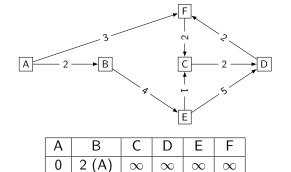
Mise en application

Complexité

OSPF : algorithme de Dijkstra



### Première itération



Même constat pour D

Plus court chemin

Problématique

Retour des graphes Graphe orienté

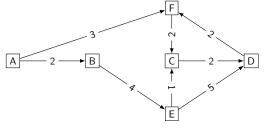
Protocole RIP : algorithme de

Principe Mise en application

OSPF : algorithme



### Première itération



Α	В	С	D	E	F
0	2 (A)	$\infty$	$\infty$	6 (B)	$\infty$

Prédécesseur de E : B

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

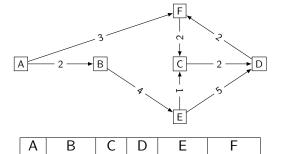
rotocole RIP : Igorithme de ellman-Ford

Mise en application Complexité

OSPF : algorithme de Dijkstra



### Première itération



 $\infty$ 

6 (B)

3 (A)

Premier prédécesseur de F : A

2 (A)

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

 $\infty$ 

Plus court chemin

Problématique

Retour des graphes Graphe orienté

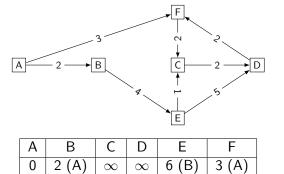
> rotocole RIP : gorithme de ellman-Ford

Mise en application

OSPF : algorithme de Dijkstra



### Première itération



Second	prédécesseur	de	F	:	D	

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

Plus court chemin

Problématique

Retour des graphe Graphe orienté

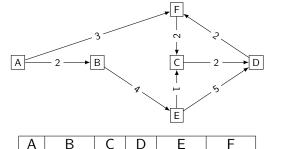
> rotocole RIP : Igorithme de Sellman-Ford

Mise en application
Complexité

OSPF : algorithme de Dijkstra



#### Première itération



Fin de la première itération

 $\infty$ 

 $\infty$ 

6 (B)

3 (A)

В

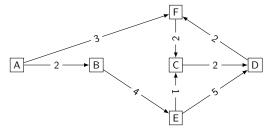
2 (A)

Plus court chemin

Mise en application



### Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	$\infty$	$\infty$	6 (B)	3 (A)
0	2 (A)	$\infty$	$\infty$	6 (B)	3 (A)

Pas de modification pour A et B

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Protocole RIP

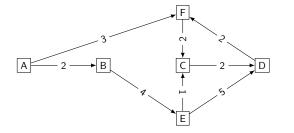
Bellman-Ford Principe

Mise en application Complexité

OSPF : algorithme de Dijkstra



### Deuxième itération



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

Premier prédécesseur de C : E

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

Plus court chemin

Problématique

Retour des graphes Graphe orienté

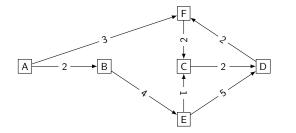
gorithme de ellman-Ford

Mise en application Complexité

OSPF : algorithme de Dijkstra



### Deuxième itération



Α	В	С	D	E	F
0	2 (A)	$\infty$	$\infty$	6 (B)	3 (A)
0	2 (A)	5 (F)	$\infty$	6 (B)	3 (A)

Second prédécesseur de C : F

$$3+2=5<7$$

Plus court chemin

Problématique

Retour des graphes Graphe orienté

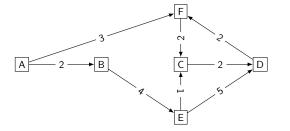
> algorithme de Bellman-Ford Principe

Mise en application Complexité

OSPF : algorithme de Dijkstra



### Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	$\infty$	$\infty$	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

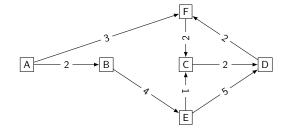
algorithme de Bellman-Ford

Mise en application

OSPF : algorithmede Dijkstra



### Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	$\infty$	$\infty$	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

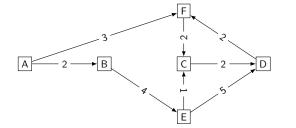
algorithme de Bellman-Ford Principe

Mise en application Complexité

OSPF : algorithme de Dijkstra



#### Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	$\infty$	$\infty$	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

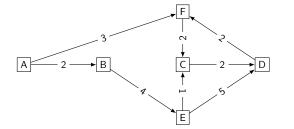
Protocole RIP : algorithme de Bellman-Ford

Mise en application

OSPF : algorithmede Dijkstra



### Deuxième itération



Α	В	С	D	Е	F
0	2 (A)	$\infty$	$\infty$	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 graphe Graphe orienté

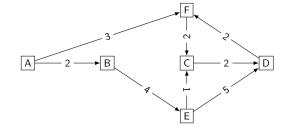
> algorithme de Bellman-Ford

Mise en application

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)	$\infty$	$\infty$	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

► du nombre de sommets (motée S): on visite chaque sommet (ligne 4); 1 Tant que (le nombre d'itérations) < (mombre 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);

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

Plus court chemin

Problématique

Graphe orienté

gorithme de ellman-Ford incipe

Mise en application Complexité

OSPF : algorithme de Dijkstra



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

Problématique

Graphe orienté
Graphe pondéré

Bellman-Ford
Principe
Mise en application
Complexité

SPF : algorithme e Dijkstra

# Complexité de l'algorithme de Bellman Ford

Plus court chemin

Complexité

2021-04-05

Plus court chemin

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

principe : construire un sous-graphe en ajoutant à chaque



### Remarque

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

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

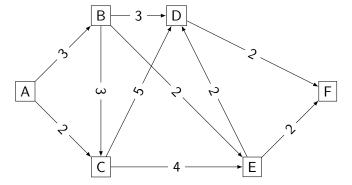


FIGURE – graphe orienté et pondéré

Plus court chemin

Problématique

Retour des graphes Graphe orienté

Protocole RIP : algorithme de Bellman-Ford

Principe Mise en application Complexité

OSPF : algorithme de Diikstra

Principe

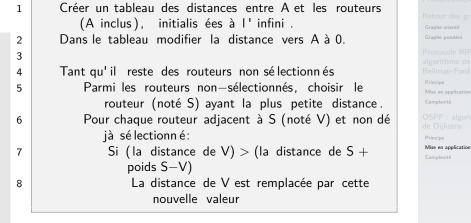
incipe ise en application

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

Crier un tabless des distances entre A et les routeurs (A inclus), initials fas 1º feder. Durn le tabless modifier la distance ver A h G. Test qu'il reste des resteurs non-altestionnée par le control de la commentation de

Code 6 - Algorithme de Dijkstra

- 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

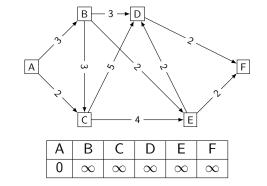
39 / 50

Plus court chemin

2021-04-05

Initialisation

# Initialisation



Plus court chemin

Problématique

Retour des graphes Graphe orienté

Graphe pondéré

Protocole RIP : algorithme de Bellman-Ford

> ise en application implexité

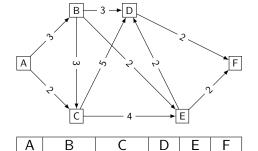
OSPF : algorithme de Dijkstra

Principe

2021-04-05



#### Sélection de A



2 (A)

nœuds visités

 $\infty$ 

 $\infty$ 

 $\infty$ 

3 (A)

Plus court chemin

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

Plus court chemin

tour des graphe

algorithme de Bellman-Ford

F: algorithme

Dijkstra ncipe

Principe

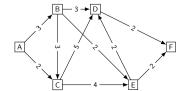
Mise en application

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.

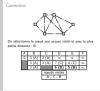
Α	В	С	D	Е	F
0	3 (A)	2 (A)	$\infty$	$\infty$	$\infty$
	3 (A)	2 (A)	7 (C)	6 (C)	$\infty$

nœuds visités A - C

Plus court chemin

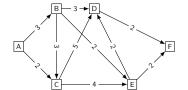
Mise en application

43 / 50



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

#### Correction



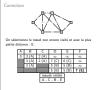
On sélectionne le nœud non encore visité et avec la plus petite distance : B.

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

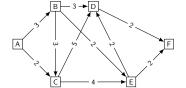
nœuds visités A - C - B

Plus court chemin

# 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)	$\infty$	$\infty$	$\infty$
	3 (A)	2 (A)	7 (C)	6 (C)	$\infty$
	3 (A)		6 (B)	5 (B)	$\infty$
			6 (B)	5 (B)	7 (E)

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

Problématique

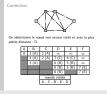
letour des graphe Graphe orienté Graphe pondéré

algorithme de Bellman-Ford

Mise en application Complexité

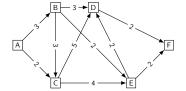
OSPF : algorithme de Dijkstra

Principe Mise en application



pas de modification

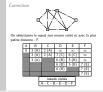
#### Correction



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

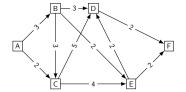
Α	В	С	D	Е	F
0	3 (A)	2 (A)	$\infty$	$\infty$	$\infty$
	3 (A)	2 (A)	7 (C)	6 (C)	$\infty$
	3 (A)		6 (B)	5 (B)	$\infty$
			6 (B)	5 (B)	7 (E)
			6 (B)		7 (E)
noude vicitée					

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



pas de modification

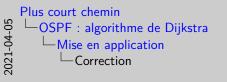
#### Correction



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

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

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

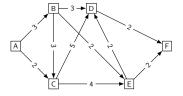




tous les nœuds ont été visités = fin de l'algorithme. On peut reconstruire le chemin.

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)	$\infty$	$\infty$	$\infty$
	3 (A)	2 (A)	7 (C)	6 (C)	$\infty$
	3 (A)		6 (B)	5 (B)	$\infty$
			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

- 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
- 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 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 : sorithme de Illman-Ford

plexité

PF: algorithme

Dijkstra

Principe
Mise en application
Complexité