Plus court chemin

Christophe Viroulaud

Terminale NSI

$\boxed{\text{Comment fonctionnent les algorithmes de plus court chemin ?}}$

Graphe orienté / non orienté

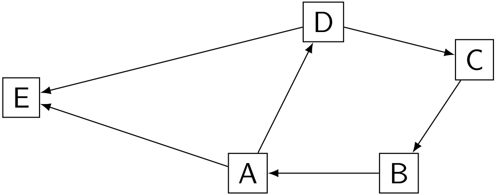


FIGURE – graphe orienté

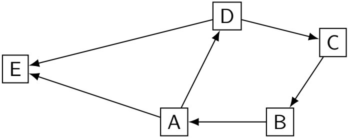


FIGURE – graphe orienté

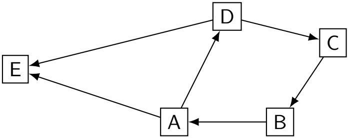
À retenir

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.

Correction

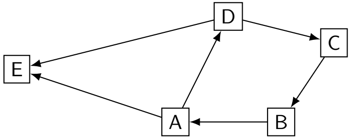



```
matrice = [[0, 0, 0, 1, 1],
           [1, 0, 0, 0, 0],
           [0, 1, 0, 0, 0],
           [0, 0, 1, 0, 1],
           [0, 0, 0, 0, 0]]

```

Code 1 – Matrice d'adjacence

Correction

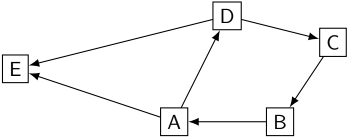



```python
dico = {"A": {"D", "E"},
        "B": {"A"},
        "C": {"B"},
        "D": {"C", "E"},
        "E": set()}

```

Code 2 – Dictionnaire d'adjacence

Correction



$$A \to D \to C \to B \to A$$

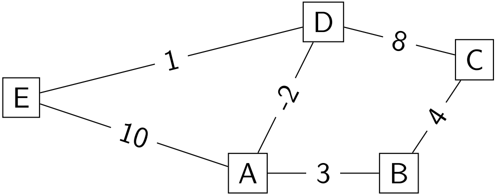
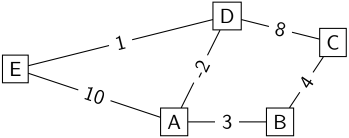


FIGURE – graphe non orienté pondéré

Activité 2 :

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

Correction

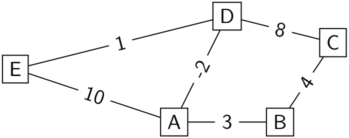



```
matrice = [[0, 3, 0, -2, 10],
           [3, 0, 4, 0, 0],
           [0, 4, 0, 8, 0],
           [-2, 0, 8, 0, 1],
           [10, 0, 0, 1, 0]]

```

Code 3 – Matrice d'adjacence

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}}
```

Code 4 – Dictionnaire d'adjacence

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.

Démarche récursive

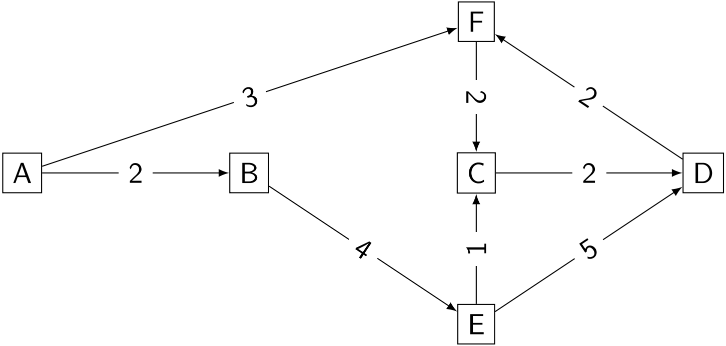


FIGURE – graphe non orienté pondéré

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

Pour chaque routeur

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

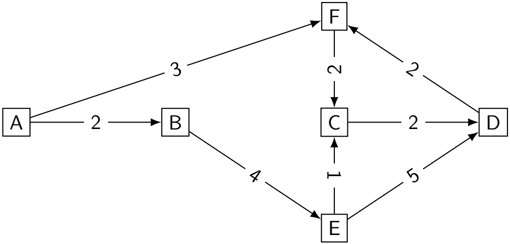
```
Créer un tableau des distances entre A et les routeurs (A inclus), initialisées à l'infini.
Dans le tableau modifier la distance vers A à 0.

Tant que (le nombre d'itérations) < (nombre de routeurs)
    Pour chaque arc du graphe
        Si (la distance du routeur) > (la distance de son prédécesseur + poids de l'arc entre les deux routeurs)
            La distance du routeur est remplacée par cette nouvelle valeur

```

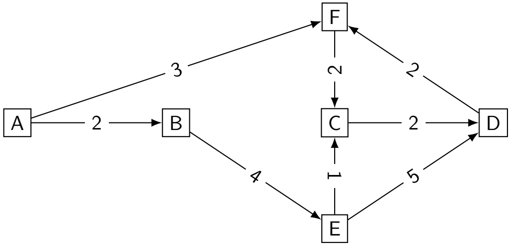
Code 5 – Algorithme de Bellman Ford

Initialisation



A	B	C	D	E	F
0	∞	∞	∞	∞	∞

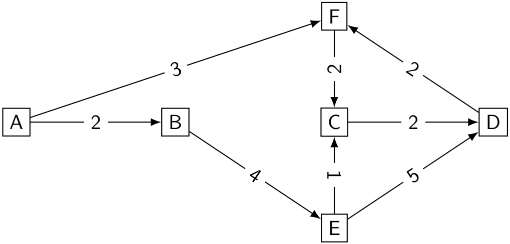
Première itération



A	B	C	D	E	F
0	**2 (A)**	∞	∞	∞	∞

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

Première itération

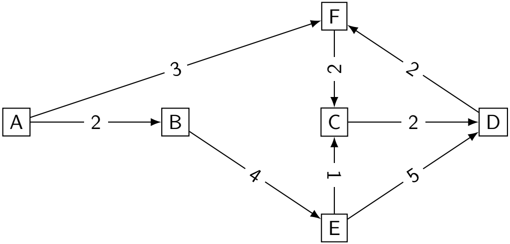


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Premier prédécesseur de C : E

$$\infty + 1 = \infty \nless \infty$$

Première itération

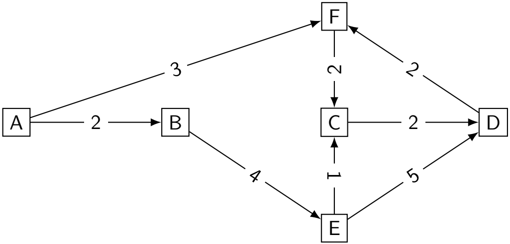


A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Second prédécesseur de C : F

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

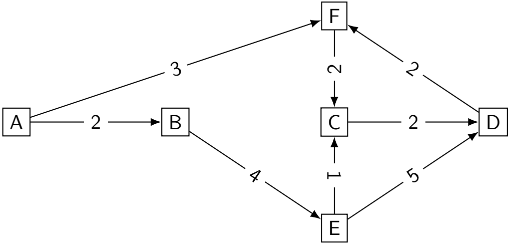
Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	∞	∞

Même constat pour D

Première itération

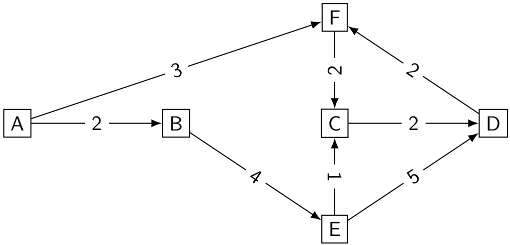


A	B	C	D	E	F
0	2 (A)	∞	∞	**6 (B)**	∞

Prédécesseur de E : B

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

Première itération

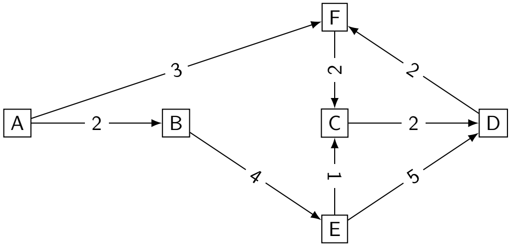


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	**3 (A)**

Premier prédécesseur de F : A

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

Première itération

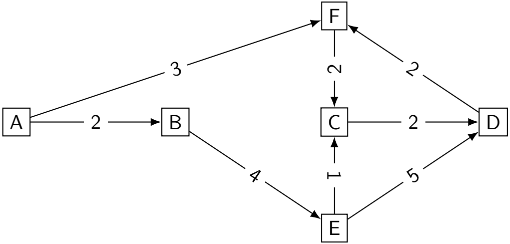


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Second prédécesseur de F : D

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

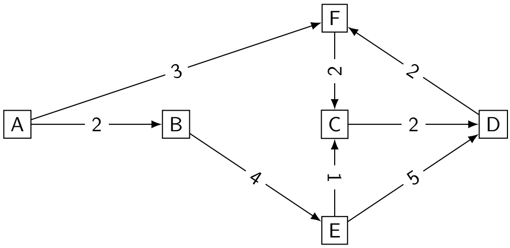
Première itération



A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)

Fin de la première itération

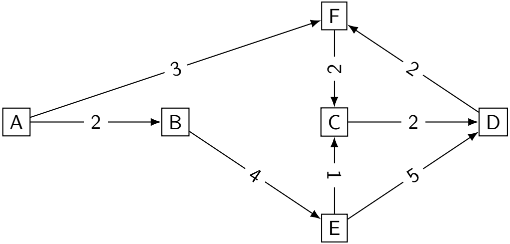
Deuxième itération



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

Pas de modification pour A et B

Deuxième itération

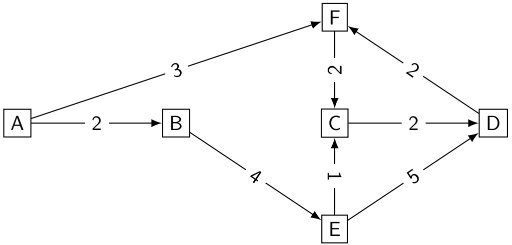


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	**7 (E)**	∞	6 (B)	3 (A)

Premier prédécesseur de C : E

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

Deuxième itération

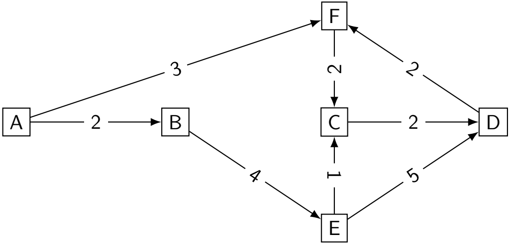


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	**5 (F)**	∞	6 (B)	3 (A)

Second prédécesseur de C : F

$$3 + 2 = 5 < 7$$

Deuxième itération

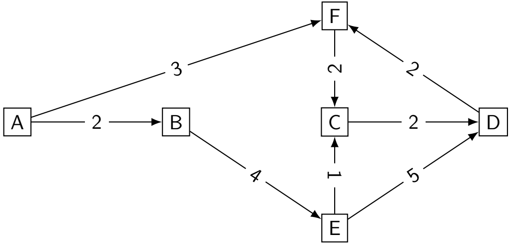


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	**7 (C)**	6 (B)	3 (A)

Premier prédécesseur de D : C

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

Deuxième itération

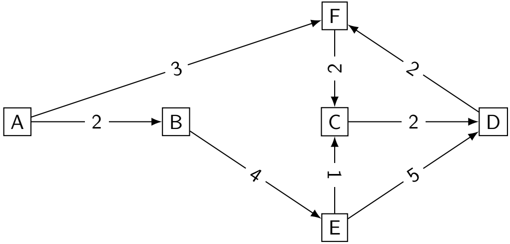


A	B	C	D	E	F
0	2 (A)	∞	∞	6 (B)	3 (A)
0	2 (A)	5 (F)	7 (C)	6 (B)	3 (A)

Second prédécesseur de D : E

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

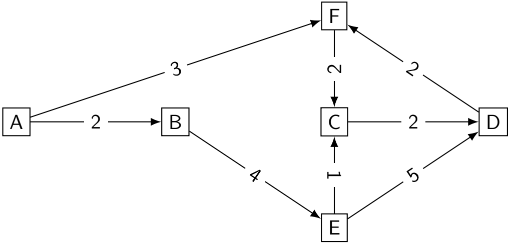
Deuxième itération



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

Pas de changement pour E et F

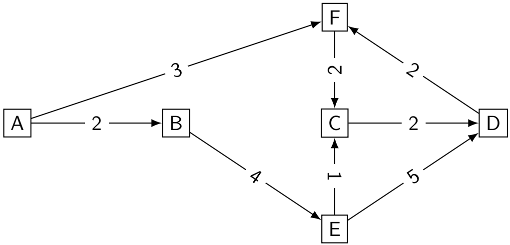
Deuxième itération



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

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

```
1 Tant que (le nombre d'itérations)
      < (nombre de routeurs)
2
```

- 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

```

Complexité de l'algorithme de Bellman Ford

$$O(S.A)$$

Edsger Dijkstra

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

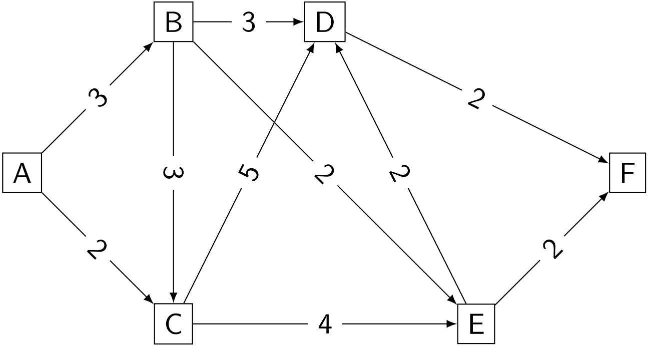


FIGURE – graphe orienté et pondéré

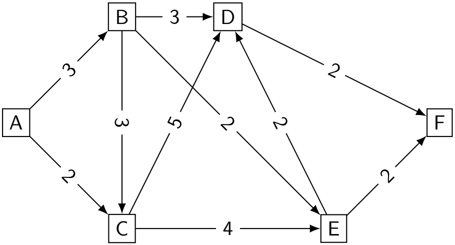
```
Créer un tableau des distances entre A et les routeurs (A inclus), initialisées à l'infini.
Dans le tableau modifier la distance vers A à 0.

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 sélectionné:
        Si (la distance de V) > (la distance de S + poids S−V)
            La distance de V est remplacée par cette nouvelle valeur

```

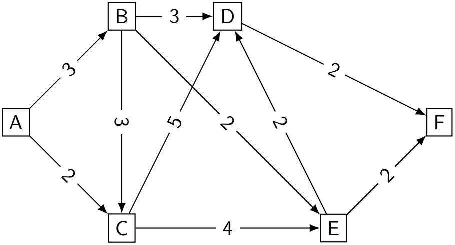
Code 6 – Algorithme de Dijkstra

Initialisation



A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Sélection de A

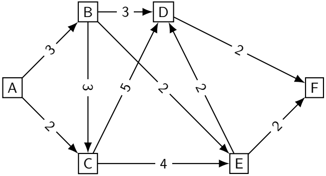


A	B	C	D	E	F
0	**3 (A)**	**2 (A)**	∞	∞	∞

nœuds visités
A

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

Correction



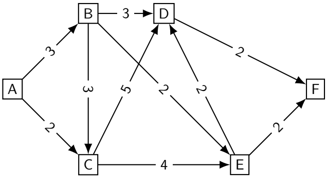
On sélectionne le nœud non encore visité et avec la plus petite distance : C.

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

nœuds visités
A - C

Plus court chemin

Correction

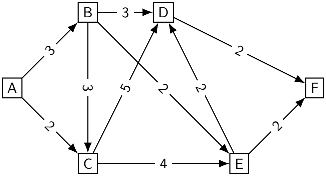


On sélectionne le nœud non encore visité et avec la plus petite distance : B.

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

nœuds visités
A - C - B

Correction

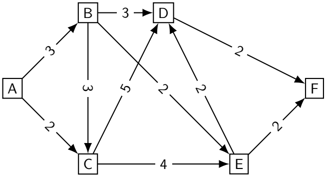


On sélectionne le nœud non encore visité et avec la plus petite distance : E.

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

nœuds visités
A - C - B - E

Correction

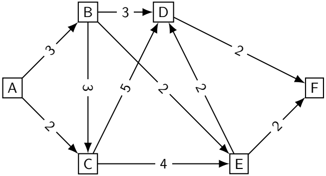


On sélectionne le nœud non encore visité et avec la plus petite distance : D.

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

nœuds visités
A - C - B - E - D

Correction

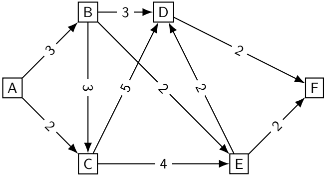


On sélectionne le nœud non encore visité et avec la plus petite distance : F.

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

nœuds visités
A - C - B - E - D - F

Correction



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

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

nœuds visités
A - C - B - E - D - F

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

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

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

```

Complexité de l'algorithme de Dijkstra

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