

Plus courts chemins et arbres couvrants

Algorithmes et structures de données, 2025-2026

P. Albuquerque (B410) et O. Malaspinas (A401), ISC, HEPIA

2026-06-03

En partie inspiré des supports de cours de P. Albuquerque

Algorithme de Floyd–Warshall

Idée générale

- Soit l'ensemble de sommets $V = \{1, 2, 3, 4, \dots, n\}$.
- Pour toute paire de sommets, i, j , on considère tous les chemins passant par les sommets intermédiaires $\in \{1, 2, \dots, k\}$ avec $k \leq n$.
- On garde pour chaque k la plus petite valeur.

Principe

- A chaque étape, k , on vérifie s'il est plus court d'aller de i à j en passant par le sommet k .
- Si à l'étape $k - 1$, le coût du parcours est p , on vérifie si p est plus petit que $p_1 + p_2$, le chemin de i à k , et k à j respectivement.

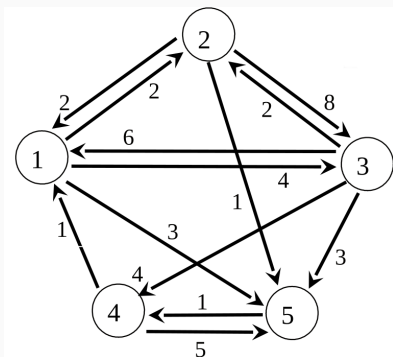
Algorithme de Floyd–Warshall

The algorithme

Soit $d_{ij}(k)$ le plus court chemin de i à j passant par les sommets $\in \{1, 2, \dots, k\}$

$$d_{ij}(k) = \begin{cases} w(i, j), & \text{si } k = 0, \\ \min(d_{ij}(k-1), d_{ik}(k-1) + d_{kj}(k-1)), & \text{sinon.} \end{cases}$$

Algorithme de Floyd–Warshall (exemple)



Que vaut $D^{(0)}$ (3min) ?

Figure 1 : Le graphe, $D = w$.

Algorithme de Floyd–Warshall (exemple)

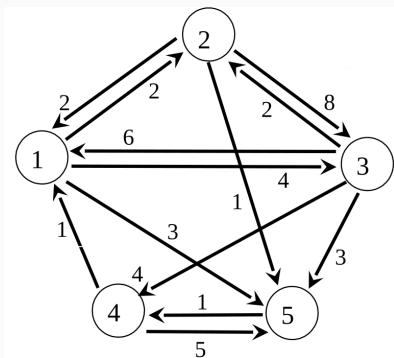


Figure 1 : Le graphe, $D = w$.

Que vaut $D^{(0)}$ (3min) ?

$$D^{(0)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(0)}$?

Que vaut $D^{(1)}$ (3min) ?

$$D^{(0)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(0)}$?

$$D^{(0)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Que vaut $D^{(1)}$ (3min) ?

$$D^{(1)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & \mathbf{6} & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \mathbf{3} & \mathbf{5} & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(0)}$

Que vaut $D^{(1)}$ (3min) ?

$$D^{(0)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(0)}$

$$D^{(0)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 8 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Que vaut $D^{(1)}$ (3min) ?

$$D^{(1)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & \mathbf{6} & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & \mathbf{3} & \mathbf{5} & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Exemple

$$D_{42}^{(1)} = D_{41}^{(0)} + D_{12}^{(0)} = 1 + 2 < \infty.$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(1)}$

Que vaut $D^{(2)}$ (3min) ?

$$D^{(1)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(1)}$

$$D^{(1)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 6 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Que vaut $D^{(2)}$ (3min) ?

$$D^{(2)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(2)}$

Que vaut $D^{(3)}$ (3min) ?

$$D^{(2)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(2)}$

$$D^{(2)} = \begin{bmatrix} 0 & 2 & 4 & \infty & 3 \\ 2 & 0 & 6 & \infty & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Que vaut $D^{(3)}$ (3min) ?

$$D^{(3)} = \begin{bmatrix} 0 & 2 & 4 & \mathbf{8} & 3 \\ 2 & 0 & 6 & \mathbf{10} & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(3)}$

Que vaut $D^{(4)}$ (3min) ?

$$D^{(3)} = \begin{bmatrix} 0 & 2 & 4 & 8 & 3 \\ 2 & 0 & 6 & 10 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(3)}$

$$D^{(3)} = \begin{bmatrix} 0 & 2 & 4 & 8 & 3 \\ 2 & 0 & 6 & 10 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Que vaut $D^{(4)}$ (3min) ?

$$D^{(4)} = \begin{bmatrix} 0 & 2 & 4 & 8 & 3 \\ 2 & 0 & 6 & 10 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ \mathbf{2} & \mathbf{4} & \mathbf{6} & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(4)}$

$$D^{(4)} = \begin{bmatrix} 0 & 2 & 4 & 8 & 3 \\ 2 & 0 & 6 & 10 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 2 & 4 & 6 & 1 & 0 \end{bmatrix}$$

Que vaut $D^{(5)}$ (3min) ?

Algorithme de Floyd–Warshall (exemple)

On part de $D^{(4)}$

$$D^{(4)} = \begin{bmatrix} 0 & 2 & 4 & 8 & 3 \\ 2 & 0 & 6 & 10 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 2 & 4 & 6 & 1 & 0 \end{bmatrix}$$

Que vaut $D^{(5)}$ (3min) ?

$$D^{(5)} = \begin{bmatrix} 0 & 2 & 4 & 4 & 3 \\ 2 & 0 & 6 & 2 & 1 \\ 4 & 2 & 0 & 4 & 3 \\ 1 & 3 & 5 & 0 & 4 \\ 2 & 4 & 6 & 1 & 0 \end{bmatrix}$$

Algorithme de Floyd–Warshall

The pseudo-code (10min)

- Quelle structure de données ?
- Quelle initialisation ?
- Quel est le code pour le calcul de la matrice D ?

Algorithme de Floyd–Warshall

The pseudo-code

- Quelle structure de données ?

```
int distance[n][n];
```

Algorithme de Floyd–Warshall

The pseudo-code

- Quelle structure de données ?

```
int distance[n][n];
```

- Quelle initialisation ?

```
matrice ini_floyd_warshall(distance, n, w)
    pour i de 1 à n
        pour j de 1 à n
            distance[i][j] = w(i,j)
    retourne distance
```

Algorithme de Floyd–Warshall

The pseudo-code

- Quel est le code pour le calcul de la matrice D ?

```
matrice floyd_warshall(distance, n, w)
  pour k de 1 à n
    pour i de 1 à n
      pour j de 1 à n
        distance[i][j] = min(distance[i][j],
                               distance[i][k] + distance[k][j])
  retourne distance
```

Algorithme de Floyd–Warshall

La matrice de précédence

- On a pas encore vu comment reconstruire le plus court chemin !
- On définit, $P_{ij}^{(k)}$, qui est le prédécesseur du sommet j depuis i avec les sommets intermédiaires $\in \{1, 2, \dots, k\}$.

$$P_{ij}^{(0)} = \begin{cases} \text{vide,} & \text{si } i = j, \text{ ou } w(i, j) = \infty \\ i, & \text{sinon.} \end{cases}$$

- Mise à jour

$$P_{ij}^{(k)} = \begin{cases} P_{ij}^{(k-1)}, & \text{si } d_{ij}^{(k)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ P_{kj}^{(k-1)}, & \text{sinon.} \end{cases}$$

Algorithme de Floyd–Warshall

La matrice de précedence

- On a pas encore vu comment reconstruire le plus court chemin !
- On définit, $P_{ij}^{(k)}$, qui est le prédécesseur du sommet j depuis i avec les sommets intermédiaires $\in \{1, 2, \dots, k\}$.

$$P_{ij}^{(0)} = \begin{cases} \text{vide,} & \text{si } i = j, \text{ ou } w(i, j) = \infty \\ i, & \text{sinon.} \end{cases}$$

- Mise à jour

$$P_{ij}^{(k)} = \begin{cases} P_{ij}^{(k-1)}, & \text{si } d_{ij}^{(k)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ P_{kj}^{(k-1)}, & \text{sinon.} \end{cases}$$

- Moralité : si le chemin est plus court en passant par k , alors il faut utiliser son prédécesseur !

Algorithme de Floyd–Warshall

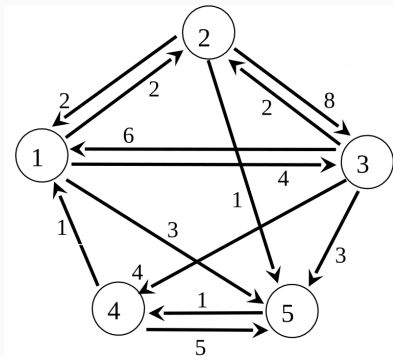
La matrice de précédence (pseudo-code, 3min)

Algorithme de Floyd–Warshall

La matrice de précédence (pseudo-code, 3min)

```
matrice, matrice floyd_warshall(distance, n, w)
  pour k de 1 à n
    pour i de 1 à n
      pour j de 1 à n
        n_distance = distance[i][k] + distance[k][j]
        if n_distance < distance[i][j]
          distance[i][j] = n_distance
          précédence[i][j] = précédence[k][j]
  retourne distance, précédence
```

Algorithme de Floyd–Warshall (exercice)



Que vaut $P^{(0)}$ (3min) ?

Figure 2 : Le graphe, $D = w$.

Algorithme de Floyd–Warshall (exercice)

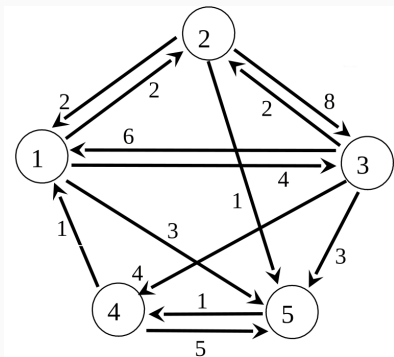
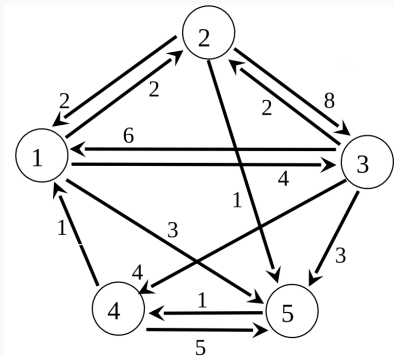


Figure 2 : Le graphe, $D = w$.

Que vaut $P^{(0)}$ (3min) ?

$$P^{(0)} = \begin{bmatrix} - & 1 & 1 & - & 1 \\ 2 & - & 2 & - & 2 \\ 3 & 3 & - & 3 & 3 \\ 4 & - & - & - & 4 \\ - & - & - & 5 & - \end{bmatrix}$$

Algorithme de Floyd–Warshall (exercice)



Que vaut $P^{(5)}$ (10min) ?

Figure 3 : Le graphe, $D = w$.

Algorithme de Floyd–Warshall (exercice)

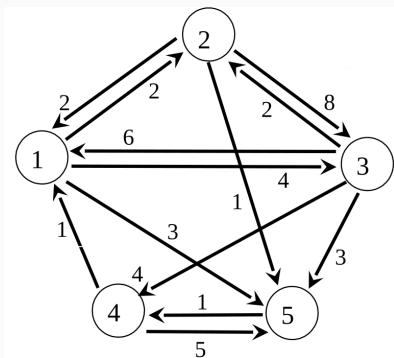


Figure 3 : Le graphe, $D = w$.

Que vaut $P^{(5)}$ (10min) ?

$$P^{(5)} = \begin{bmatrix} - & 1 & 1 & 5 & 1 \\ 2 & - & 1 & 5 & 2 \\ 2 & 3 & - & 3 & 3 \\ 4 & 1 & 1 & - & 1 \\ 4 & 1 & 1 & 5 & - \end{bmatrix}$$

Exercice : retrouver le chemin entre 1 et 4 (5min)

$$P = \begin{bmatrix} - & 1 & 1 & 5 & 1 \\ 2 & - & 1 & 5 & 2 \\ 2 & 3 & - & 3 & 3 \\ 4 & 1 & 1 & - & 4 \\ 4 & 1 & 1 & 5 & - \end{bmatrix}$$

Exercice : retrouver le chemin entre 1 et 4 (5min)

$$P = \begin{bmatrix} - & 1 & 1 & 5 & 1 \\ 2 & - & 1 & 5 & 2 \\ 2 & 3 & - & 3 & 3 \\ 4 & 1 & 1 & - & 4 \\ 4 & 1 & 1 & 5 & - \end{bmatrix}$$

Solution

- Le sommet $5 = P_{14}$, on a donc, $5 \rightarrow 4$, on veut connaître le prédécesseur de 5.
- Le sommet $1 = P_{15}$, on a donc, $1 \rightarrow 5 \rightarrow 4$. The end.

Exercice complet

Appliquer l'algorithme de Floyd–Warshall au graphe suivant

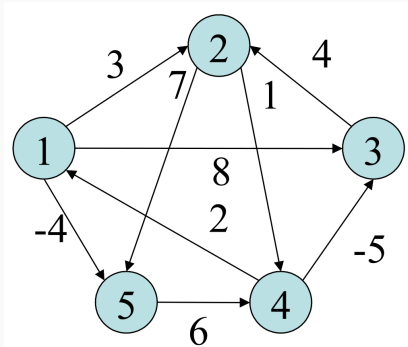


Figure 4 : The exorcist.

- Bien indiquer l'état de D et P à chaque étape !
- Ne pas oublier de faire la matrice d'adjacence évidemment...

Les arbres couvrants

Trouver un réseau électrique pour

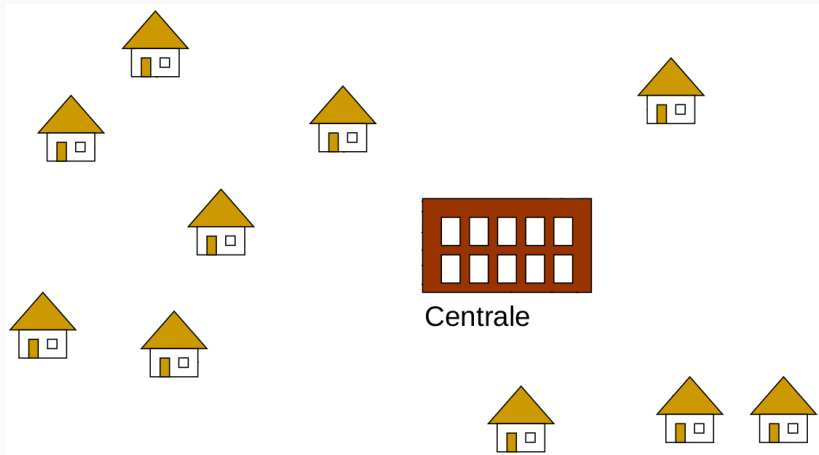


Figure 5 : Ces maisons n'ont pas d'électricité.

Solution : pas optimale

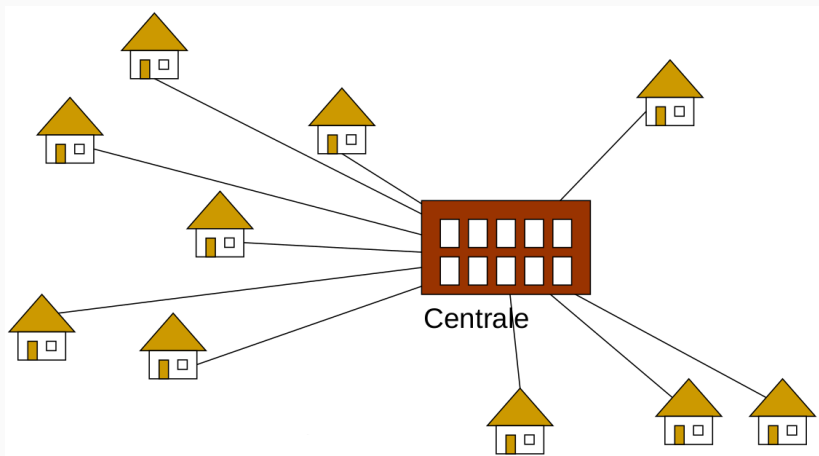


Figure 6 : Le réseau simple, mais nul.

- La longueur totale des câbles est super longue !

Solution : optimale

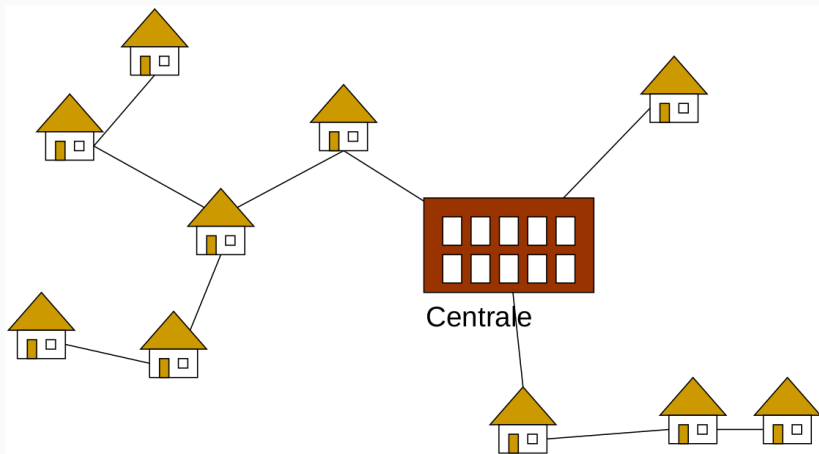


Figure 7 : Le meilleur réseau.

Application : minimisation des coûts

- Équipement d'un lotissement avec des lignes électriques/téléphoniques, des canalisations, ...

Application : minimisation des coûts

- Équipement d'un lotissement avec des lignes électriques/téléphoniques, des canalisations, ...
- Pour réduire les coûts, on cherche à minimiser la longueur totale des câbles/tuyaux.

Application : minimisation des coûts

- Équipement d'un lotissement avec des lignes électriques/téléphoniques, des canalisations, ...
- Pour réduire les coûts, on cherche à minimiser la longueur totale des câbles/tuyaux.
- Les lignes/tuyaux forment un *arbre couvrant*.

Application : minimisation des coûts

- Équipement d'un lotissement avec des lignes électriques/téléphoniques, des canalisations, ...
- Pour réduire les coûts, on cherche à minimiser la longueur totale des câbles/tuyaux.
- Les lignes/tuyaux forment un *arbre couvrant*.
- La meilleure option est un *arbre couvrant minimal*.

Formalisation : Les arbres couvrants

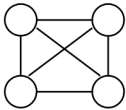
- Qu'est-ce qu'un arbre couvrant ? Des idées ? De quel objet part-on ?
Où va-t-on ?

Formalisation : Les arbres couvrants

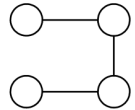
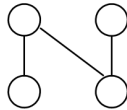
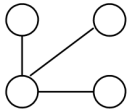
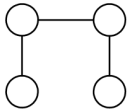
- Qu'est-ce qu'un arbre couvrant ? Des idées ? De quel objet part-on ? Où va-t-on ?
- Un arbre couvrant d'un graphe non-orienté et connexe est :
 - un arbre inclus dans le graphe qui connecte tous les sommets du graphe.

Formalisation : Les arbres couvrants

- Qu'est-ce qu'un arbre couvrant ? Des idées ? De quel objet part-on ? Où va-t-on ?
- Un arbre couvrant d'un graphe non-orienté et connexe est :
 - un arbre inclus dans le graphe qui connecte tous les sommets du graphe.



Graphe non-orienté
connexe



Quatre arbres couvrants de ce graphe

Figure 8 : Exemple d'arbres couvrants d'un graphe connexe.

Arbres couvrants

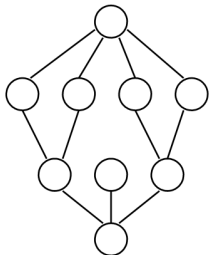
- Quels algorithmes que nous avons déjà vus, permettent de construire des arbres couvrants ?

Arbres couvrants

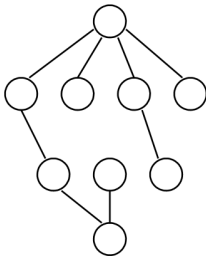
- Quels algorithmes que nous avons déjà vus, permettent de construire des arbres couvrants ?
- Les parcours en largeur et en profondeur !

Arbres couvrants

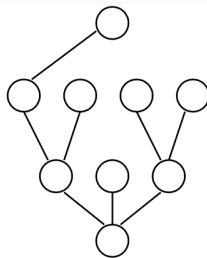
- Quels algorithmes que nous avons déjà vus, permettent de construire des arbres couvrants ?
- Les parcours en largeur et en profondeur !



Graphe non-orienté



Arbre couvrant
obtenu via un BFS



Arbre couvrant
obtenu via un DFS

Figure 9 : Graphe et parcours comme arbres couvrants.

Arbres couvrants minimaux

- Un *arbre couvrant minimal* est un sous-graphe d'un graphe non-orienté pondéré $G(V, E)$ tel quel :
 - C'est un arbre (graphe acyclique) ;
 - Il couvre tous les sommets de G et contient $|V| - 1$ arêtes ;
 - Le coût total associé aux arêtes de l'arbre est minimum parmi tous les arbres couvrants possibles.

Arbres couvrants minimaux

- Un *arbre couvrant minimal* est un sous-graphe d'un graphe non-orienté pondéré $G(V, E)$ tel quel :
 - C'est un arbre (graphe acyclique) ;
 - Il couvre tous les sommets de G et contient $|V| - 1$ arêtes ;
 - Le coût total associé aux arêtes de l'arbre est minimum parmi tous les arbres couvrants possibles.
- Est-il unique ?

Arbres couvrants minimaux

- Un *arbre couvrant minimal* est un sous-graphe d'un graphe non-orienté pondéré $G(V, E)$ tel quel :
 - C'est un arbre (graphe acyclique) ;
 - Il couvre tous les sommets de G et contient $|V| - 1$ arêtes ;
 - Le coût total associé aux arêtes de l'arbre est minimum parmi tous les arbres couvrants possibles.
- Est-il unique ?
- Pas forcément.

Arbres couvrants minimaux

- Comment générer un arbre couvrant minimal ?

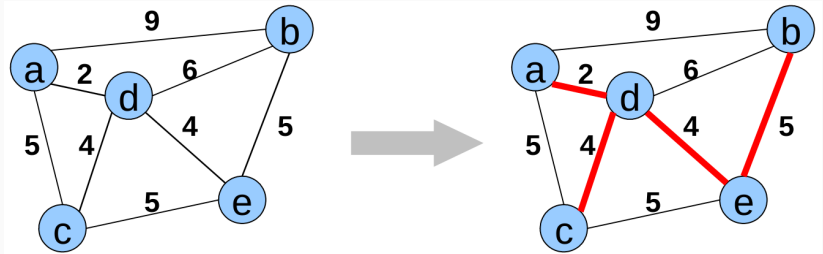


Figure 10 : Un graphe, connexe, non-orienté, pondéré, et un arbre couvrant minimal.

Algorithme de Prim

Un exemple

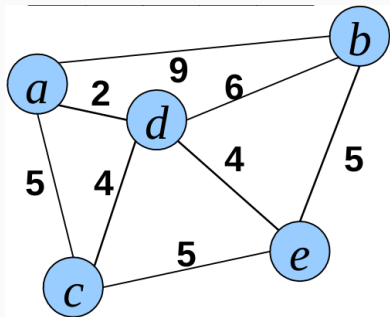


Figure 11 : Le graphe de départ.

On part de *e* (au hasard)

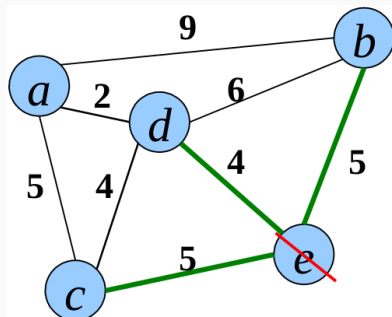


Figure 12 : Le sommet *e* est couvert.

Algorithme de Prim

On choisit comment ?

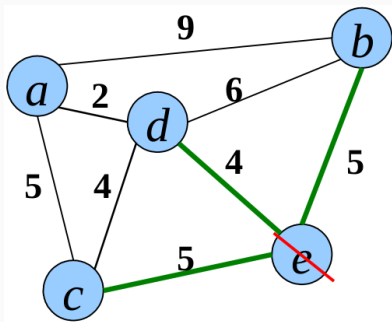


Figure 13 : Quelle arête choisir ?

Algorithme de Prim

On choisit comment ?

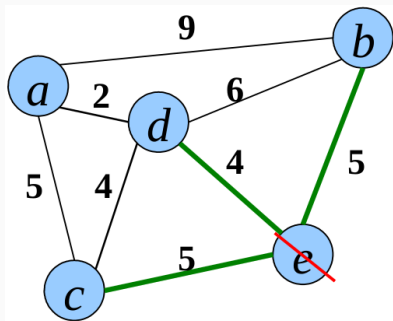


Figure 13 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet

non visité

Algorithme de Prim

On choisit comment ?

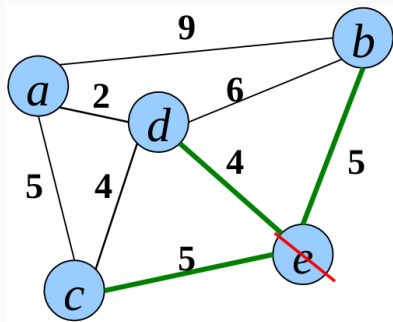


Figure 13 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet

non visité

L'arête $e \rightarrow d$

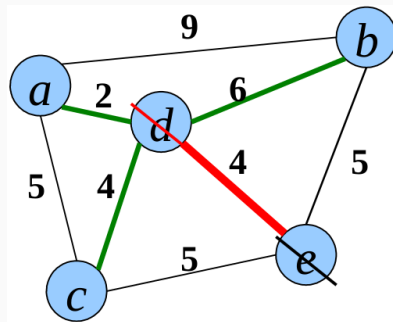


Figure 14 : Le sommet d est couvert.

Algorithme de Prim

On choisit comment ?

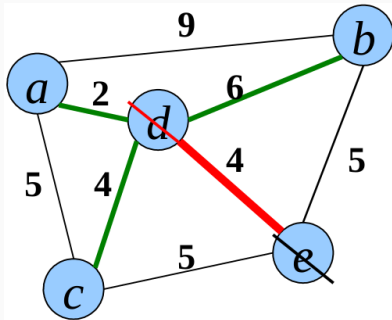


Figure 15 : Quelle arête choisir ?

Algorithme de Prim

On choisit comment ?

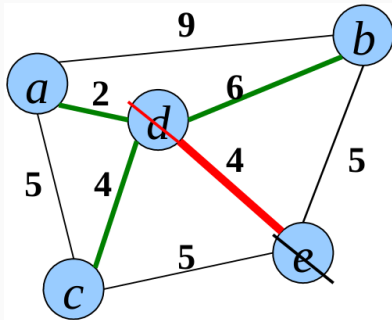


Figure 15 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet non visité

Algorithme de Prim

On choisit comment ?

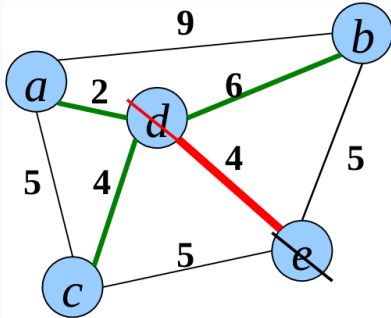


Figure 15 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet non visité

L'arête d→a

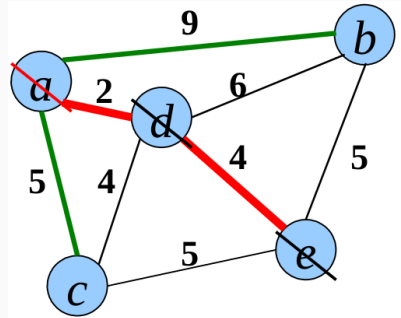


Figure 16 : Le sommet a est couvert.

Algorithme de Prim

On choisit comment ?

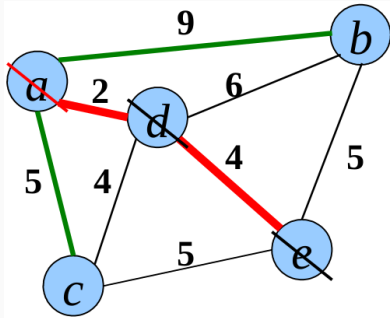


Figure 17 : Quelle arête choisir ?

Algorithme de Prim

On choisit comment ?

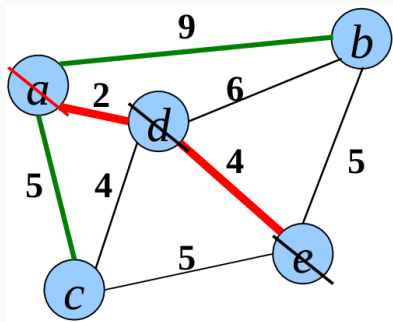


Figure 17 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet

non visité

Algorithme de Prim

On choisit comment ?

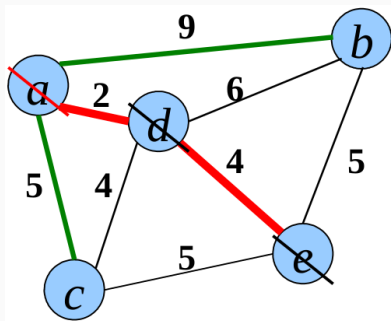


Figure 17 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet

non visité

L'arête $d \rightarrow c$

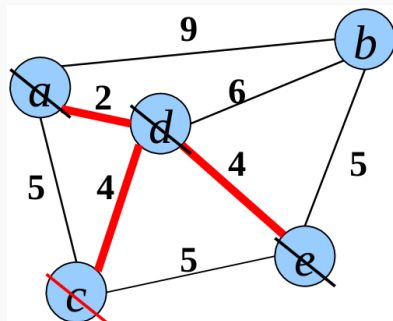


Figure 18 : Le sommet c est couvert.

Algorithme de Prim

On choisit comment ?

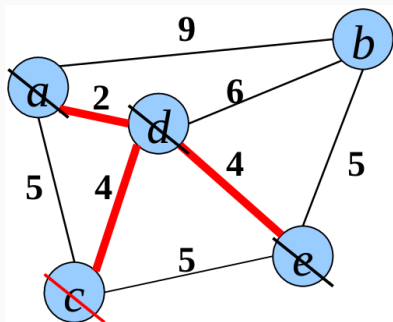


Figure 19 : Quelle arête choisir ?

Algorithme de Prim

On choisit comment ?

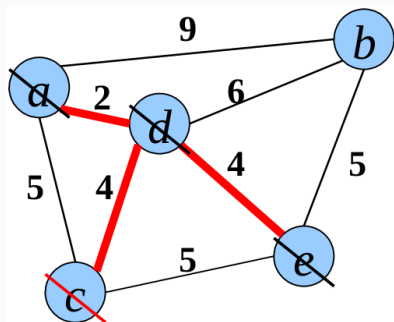


Figure 19 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet

Algorithme de Prim

On choisit comment ?

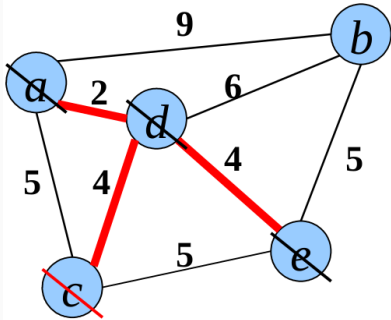


Figure 19 : Quelle arête choisir ?

- L'arête la plus courte sortant d'un sommet déjà visité, et entrant dans un sommet

L'arête $e \rightarrow b$

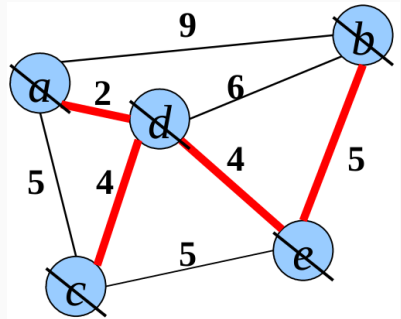


Figure 20 : Le sommet b est couvert.

- Game over !

Exemple d'algorithme de Prim

Un exemple

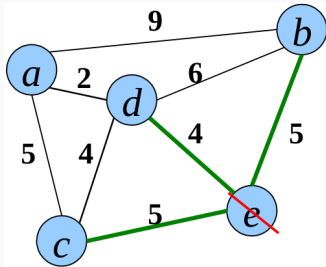


Figure 21 : Étape 1.

FP		e		d		b		c		a	

D		0		inf		inf		inf		inf	
---	--	---	--	-----	--	-----	--	-----	--	-----	--

		e		d		b		c		a	

P		-		-		-		-		-	
---	--	---	--	---	--	---	--	---	--	---	--

Devient ?

Exemple d'algorithme de Prim

Un exemple

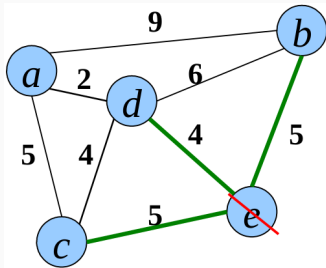


Figure 21 : Étape 1.

FP		e		d		b		c		a	
----	--	---	--	---	--	---	--	---	--	---	--

D		0		inf		inf		inf		inf	
---	--	---	--	-----	--	-----	--	-----	--	-----	--

		e		d		b		c		a	
--	--	---	--	---	--	---	--	---	--	---	--

P		-		-		-		-		-	
---	--	---	--	---	--	---	--	---	--	---	--

Devient ?

FP		d		b		c		a	
----	--	---	--	---	--	---	--	---	--

D		4		5		5		inf	
---	--	---	--	---	--	---	--	-----	--

		e		d		b		c		a	
--	--	---	--	---	--	---	--	---	--	---	--

P		-		e		e		e		-	
---	--	---	--	---	--	---	--	---	--	---	--

Exemple d'algorithme de Prim

Un exemple

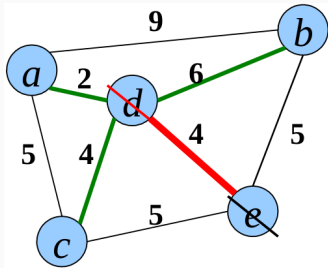


Figure 22 : Étape 2.

FP		d		b		c		a	
----	--	---	--	---	--	---	--	---	--

D		4		5		5		inf	
---	--	---	--	---	--	---	--	-----	--

		e		d		b		c		a	
--	--	---	--	---	--	---	--	---	--	---	--

P		-		e		e		e		-	
---	--	---	--	---	--	---	--	---	--	---	--

Devient ?

Exemple d'algorithme de Prim

Un exemple

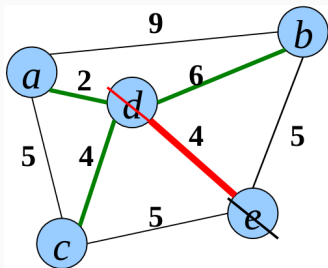


Figure 22 : Étape 2.

FP	d	b	c	a
----	---	---	---	---

D	4	5	5	inf
---	---	---	---	-----

	e	d	b	c	a
--	---	---	---	---	---

P	-	e	e	e	-
---	---	---	---	---	---

Devient ?

FP	a	c	b
----	---	---	---

D	2	4	5
---	---	---	---

	e	d	b	c	a
--	---	---	---	---	---

P	-	e	e	d	d
---	---	---	---	---	---

Exemple d'algorithme de Prim

Un exemple

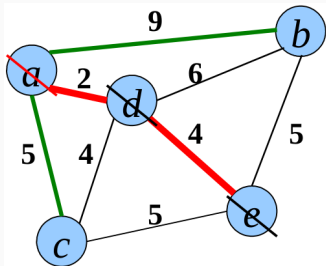


Figure 23 : Étape 3.

FP		a		c		b	

D		2		4		5	
---	--	---	--	---	--	---	--

		e		d		b		c		a	

P		-		e		e		d		d	
---	--	---	--	---	--	---	--	---	--	---	--

Devient ?

Exemple d'algorithme de Prim

Un exemple

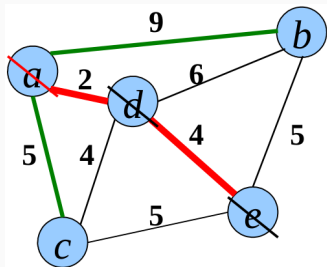


Figure 23 : Étape 3.

FP		a		c		b	
----	--	---	--	---	--	---	--

D		2		4		5	
---	--	---	--	---	--	---	--

		e		d		b		c		a	
--	--	---	--	---	--	---	--	---	--	---	--

P		-		e		e		d		d	
---	--	---	--	---	--	---	--	---	--	---	--

Devient ?

FP		c		b	
----	--	---	--	---	--

D		4		5	
---	--	---	--	---	--

		e		d		b		c		a	
--	--	---	--	---	--	---	--	---	--	---	--

P		-		e		e		d		d	
---	--	---	--	---	--	---	--	---	--	---	--

Exemple d'algorithme de Prim

Un exemple

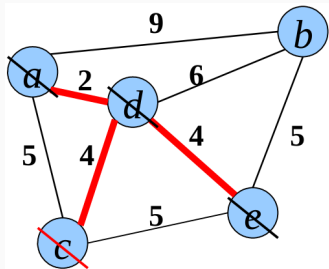


Figure 24 : Étape 4.

FP		c		b	

D		4		5	
---	--	---	--	---	--

		e		d		b		c		a	

P		-		e		e		d		d	
---	--	---	--	---	--	---	--	---	--	---	--

Devient ?

Exemple d'algorithme de Prim

Un exemple

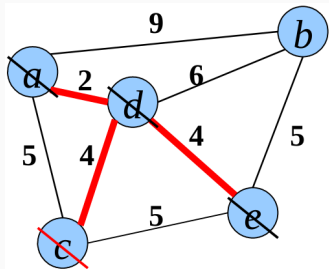


Figure 24 : Étape 4.

FP		c		b	
----	--	---	--	---	--

D		4		5	
---	--	---	--	---	--

		e		d		b		c		a	
--	--	---	--	---	--	---	--	---	--	---	--

P		-		e		e		d		d	
---	--	---	--	---	--	---	--	---	--	---	--

Devient ?

FP		b	
----	--	---	--

D		5	
---	--	---	--

		e		d		b		c		a	
--	--	---	--	---	--	---	--	---	--	---	--

P		-		e		e		d		d	
---	--	---	--	---	--	---	--	---	--	---	--

Exemple d'algorithme de Prim

Un exemple

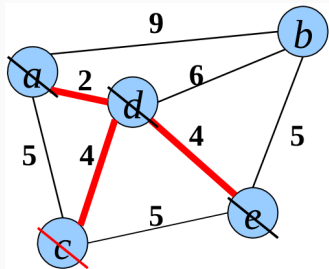


Figure 25 : Étape 5.

FP | b |

D | 5 |

| e | d | b | c | a |

P | - | e | e | d | d |

Devient ?

Exemple d'algorithme de Prim

Un exemple

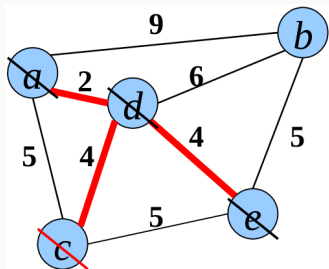


Figure 25 : Étape 5.

FP | b |

D | 5 |

| e | d | b | c | a |

P | - | e | e | d | d |

Devient ?

FP |

D |

| e | d | b | c | a |

P | - | e | e | d | d |

Structures de données

- Dans quoi allons-nous stocker les sommets ?

Structures de données

- Dans quoi allons-nous stocker les sommets ?
- File de priorité min.
- Autre chose ?

Structures de données

- Dans quoi allons-nous stocker les sommets ?
- File de priorité min.
- Autre chose ?
- Tableau des coûts (comme pour Dijkstra).
- Autre chose ?

Structures de données

- Dans quoi allons-nous stocker les sommets ?
- File de priorité min.
- Autre chose ?
- Tableau des coûts (comme pour Dijkstra).
- Autre chose ?
- Tableau des parents (presque comme pour Dijkstra).
- Autre chose ?

Structures de données

- Dans quoi allons-nous stocker les sommets ?
- File de priorité min.
- Autre chose ?
- Tableau des coûts (comme pour Dijkstra).
- Autre chose ?
- Tableau des parents (presque comme pour Dijkstra).
- Autre chose ?
- Non.

Algorithme de Prim

Initialisation : Pseudo-code (2min)

Algorithme de Prim

Initialisation : Pseudo-code (2min)

```
file_priorité, coût, parent initialisation(graphe)
  s_initial = aléatoire(graphe)
  coût[s_initial] = 0
  fp = file_p_vide()
  pour s_courant dans sommets(graphe)
    si s_courant != s_initial
      coût[s_courant] = infini
      parent[s_courant] = indéfini
      fp = enfiler(fp, s_courant, coût[s_courant])
  retourne fp, coût, parent
```


Algorithme de Prim

Algorithme : Pseudo-code (5min)

Algorithme de Prim

Algorithme : Pseudo-code (5min)

```
coût, parent prim(graphe)
  fp, coût, parent initialisation(graphe)
  sommets = vide
  tant que !est_vide(fp)
    s_courant, fp = défiler(fp)
    sommets = insérer(sommets, s_courant)
    pour s_voisin dans voisinage(s_courant) et pas dans sommets
      // ou dans fp
      si poids(s_courant, s_voisin) < coût[s_voisin]
        parent[s_voisin] = s_courant
        coût[s_voisin] = poids(s_courant, s_voisin)
        fp = changer_priorité(fp, s_voisin,
                               poids(s_courant, s_voisin))
  retourne coût, parent
```

Exercice : algorithme de Prim

Appliquer l'algorithme de Prim à (15min) :

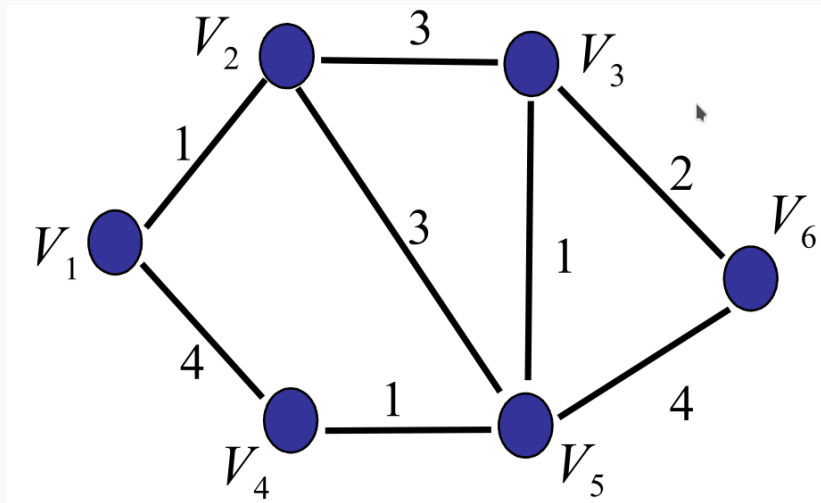
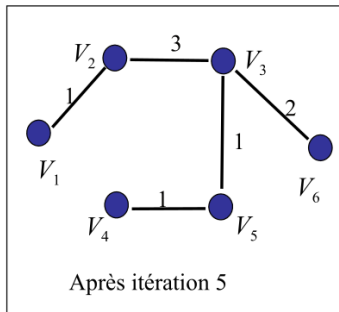
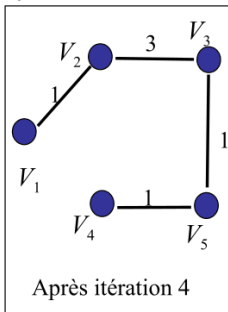
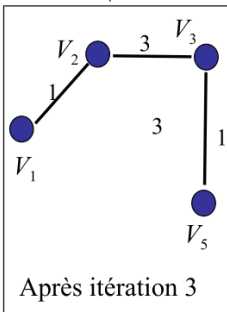
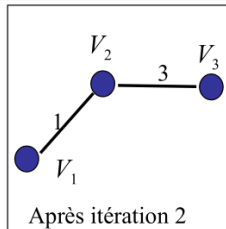
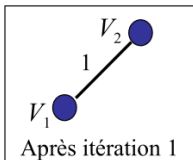
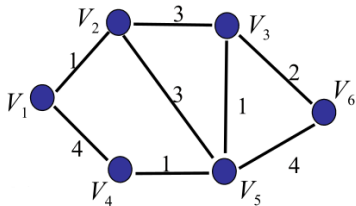


Figure 26 : En démarrant du sommet V_1 .

Exercice : algorithme de Prim

Solution



Complexité de l'algorithme de Prim

```
file_priorité, coût, parent initialisation(graphe)
    // choix r et initialisation
    pour v dans sommets(graphe)
        // initialisation coût et parent en  $O(|V|)$ 
        fp = enfiler(fp, v, coût[v])
    retourne fp, coût, parent
coût, parent prim(graphe)
    fp, coût, parent initialisation(graphe) //  $O(|V|)$ 
    sommets = vide
    tant que !est_vide(fp)
        u, fp = défiler(fp) //  $O(|V|)$ 
        sommets = insérer(sommets, u)
        pour v dans voisinage de u et pas dans sommets
            si poids(u, v) < coût[v] //  $O(|E|)$ 
                // m à j coût + parent
                fp = changer_priorité(fp, v, poids(u, v)) //  $O(|V|)$ 
    retourne coût, parent
```

- $O(|V|) + O(|E|) + O(|V|^2) = O(|E| + |V|^2)$
- Remarque : $O(|E|)$ n'est pas multiplié par $O(|V|)$, car les arêtes ne sont traitées qu'une fois en **tout**.

Algorithme de Kruskal

- On ajoute les arêtes de poids minimal :
 - si cela ne crée pas de cycle ;
 - on s'arrête quand on a couvert tout le graphe.

Algorithme de Kruskal

- On ajoute les arêtes de poids minimal :
 - si cela ne crée pas de cycle ;
 - on s'arrête quand on a couvert tout le graphe.
- Comment on fait ça ?

Algorithme de Kruskal

- On ajoute les arêtes de poids minimal :
 - si cela ne crée pas de cycle ;
 - on s'arrête quand on a couvert tout le graphe.
- Comment on fait ça ?
- Faisons un exemple pour voir.

Algorithme de Kruskal : exemple

Un exemple

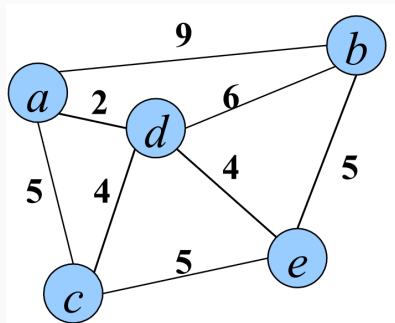


Figure 27 : Le graphe de départ.

On part de (a, d) (poids le plus faible)

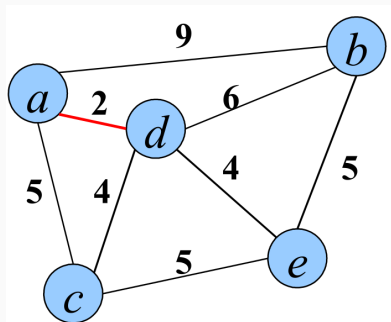


Figure 28 : Les sommets a , d sont couverts.

Algorithme de Kruskal : exemple

On continue avec (c, d)

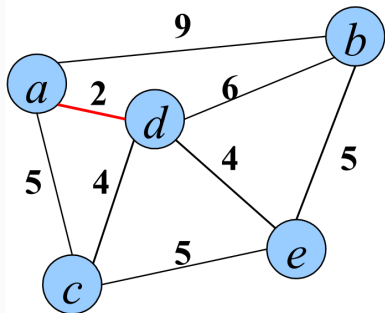


Figure 29 : On aurait pu choisir (d, e) aussi.

Résultat

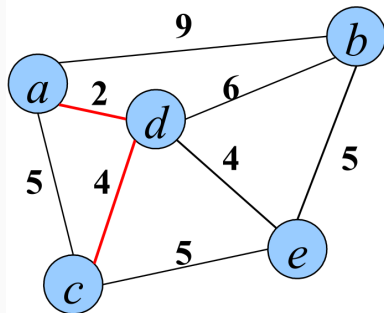


Figure 30 : Les sommets *a*, *d*, *c* sont couverts.

Algorithme de Kruskal : exemple

On continue avec (d, e)

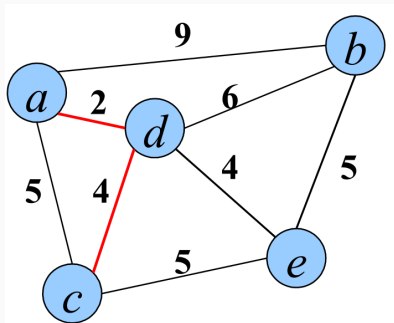


Figure 31 : Le poids de (d, e) est le plus bas.

Résultat

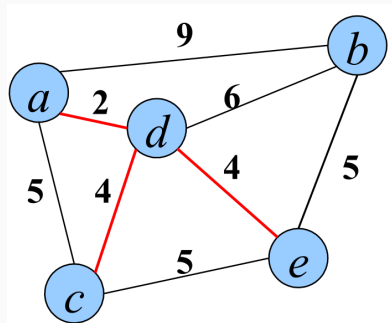


Figure 32 : Les sommets *a*, *d*, *c*, *e* sont couverts.

Algorithme de Kruskal : exemple

On continue avec (b, e)

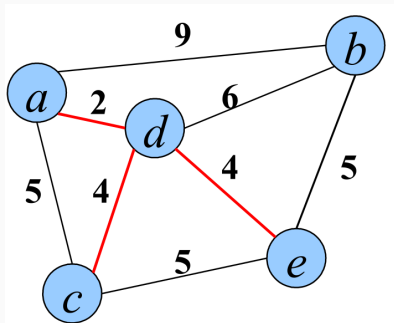


Figure 33 : Le poids de (b, e) est le plus bas.

Résultat

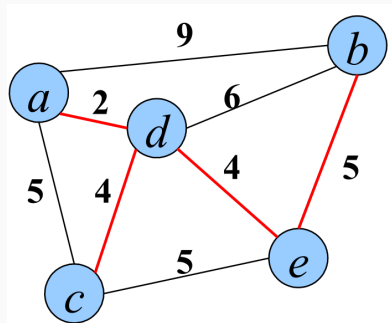


Figure 34 : Les sommets a, d, c, e, b sont couverts.

Algorithme de Kruskal : exemple

Mais pourquoi pas (c, e) ?

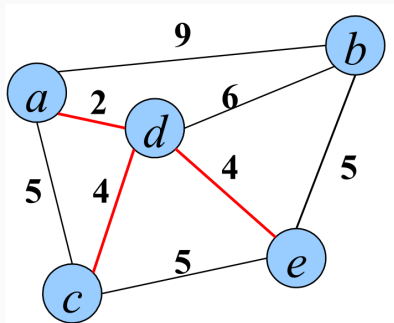


Figure 35 : Le poids de (b, e) ou (a, c) est le même.

Résultat : un cycle

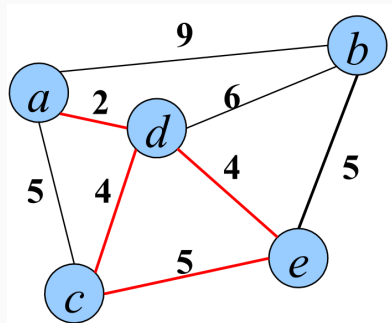


Figure 36 : Les sommets a, d, c, e sont couverts.

- Comment faire pour empêcher l'ajout de (c, e) ou (a, c) ?

Algorithme de Kruskal : exemple

Mais pourquoi pas (c, e) ?

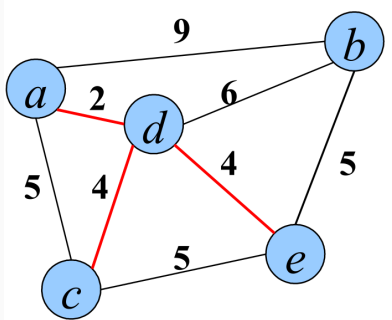


Figure 35 : Le poids de (b, e) ou (a, c) est le même.

Résultat : un cycle

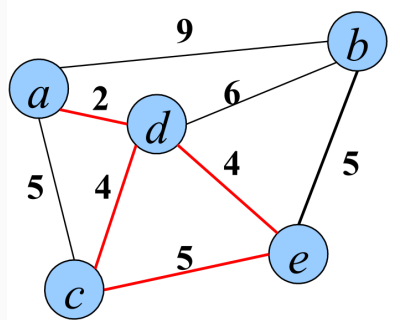


Figure 36 : Les sommets a, d, c, e sont couverts.

- Comment faire pour empêcher l'ajout de (c, e) ou (a, c) ?
- Si les deux sommets sont déjà couverts nous sommes sauvés (presque) !

Algorithme de Kruskal

L'initialisation

- Créer un ensemble de sommets pour chaque de sommet du graphe (V_1, V_2, \dots) :
 - $V_1 = \{v_1\}, V_2 = \{v_2\}, \dots$
 - S'il y a n sommets, il y a n V_i .
- Initialiser l'ensemble A des arêtes "sûres" constituant l'arbre couvrant minimal, $A = \emptyset$.
- Initialiser l'ensemble des sommets couverts $F = \emptyset$.
- Trier les arêtes par poids croissant dans l'ensemble E .

Mise à jour

- Tant qu'il reste plus d'un V_i :
 - Pour $(u, v) \in E$ à poids minimal :
 - Retirer (u, v) de E .
 - Si $u \in V_i$ et $v \in V_j$ avec $V_i \cap V_j = \emptyset$:
 - Ajouter (u, v) à A ;
 - Fusionner V_i et V_j dans F .

Algorithme de Kruskal : exemple

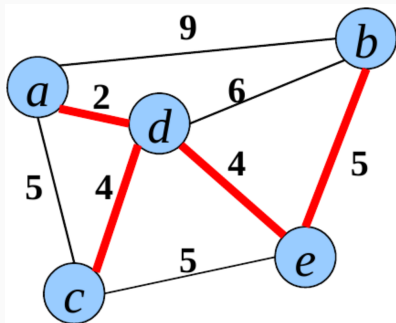


Figure 37 : Couvrir cet arbre bon sang !

Algorithme de Kruskal : solution

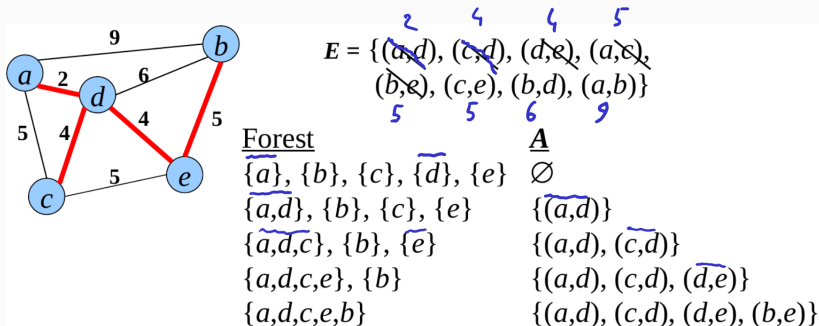


Figure 38 : La solution !

Algorithme de Kruskal : exercice

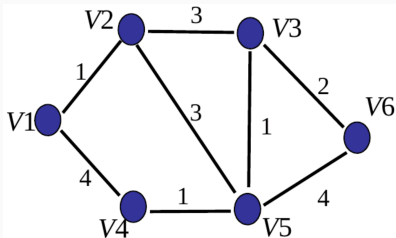


Figure 39 : Couvrir cet arbre bon sang !

Algorithme de Kruskal : solution

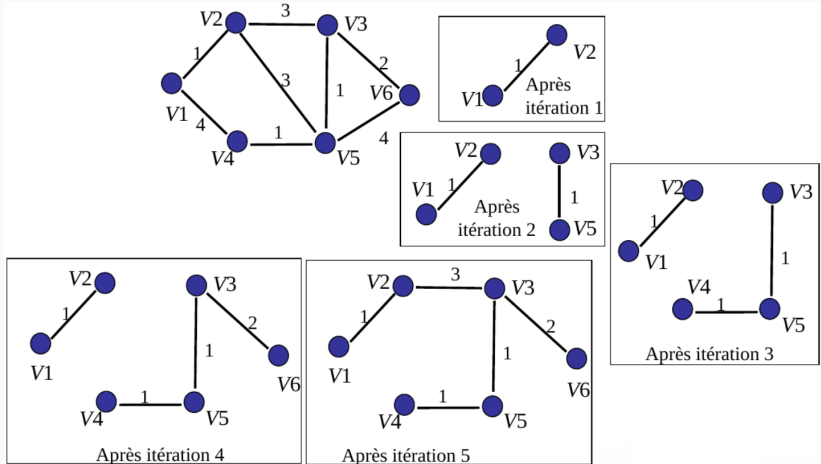


Figure 40 : La solution !