Algorithmique des graphes - Cours 5

Olivier Baudon

Université de Bordeaux

12 octobre 2020

Kruskal

Justification

Soit T_K un arbre obtenu par l'algorithme de Kruskal. Supposons que T_K ne soit pas de poids minimum. On note par $e_1, \dots e_{n-1}$ les arêtes de T_K telles que e_i est l'arête insérée à l'étape i.
Soit T un arbre de poids minimum tel que la première arête qui diffère entre T et T_K soit l'arête e_i avec i maximum.
Si on rajoute e_i à T, alors on crée un cycle fondamental C et donc il existe dans C une arête e_T n'appartenant pas à T_K. Cette arête a un poids $\omega(e_T) < \omega(e_i)$, sinon,

- si $\omega(e_T) > \omega(e_i)$, alors T ne serait pas optimal car on pourrait remplacer e_T par e_i ;
- si $\omega(e_T) = \omega(e_i)$, alors on pourrait remplacer e_T par e_i sans changer le poids de T et i ne serait maximum.

Donc quand on a choisit e_i, e_T était aussi candidate avec un poids inférieur et donc on aurait du choisir e_T.
Conclusion : T_K est de poids minimum.

Prim

Principe

L'algorithme de Prim construit un arbre T en rajoutant à chaque étape le sommet qui n'est pas encore dans l'arbre et qui possède l'arête de poids minimum parmi celles reliant les sommets de T aux sommets de $G - T$.

Énoncé

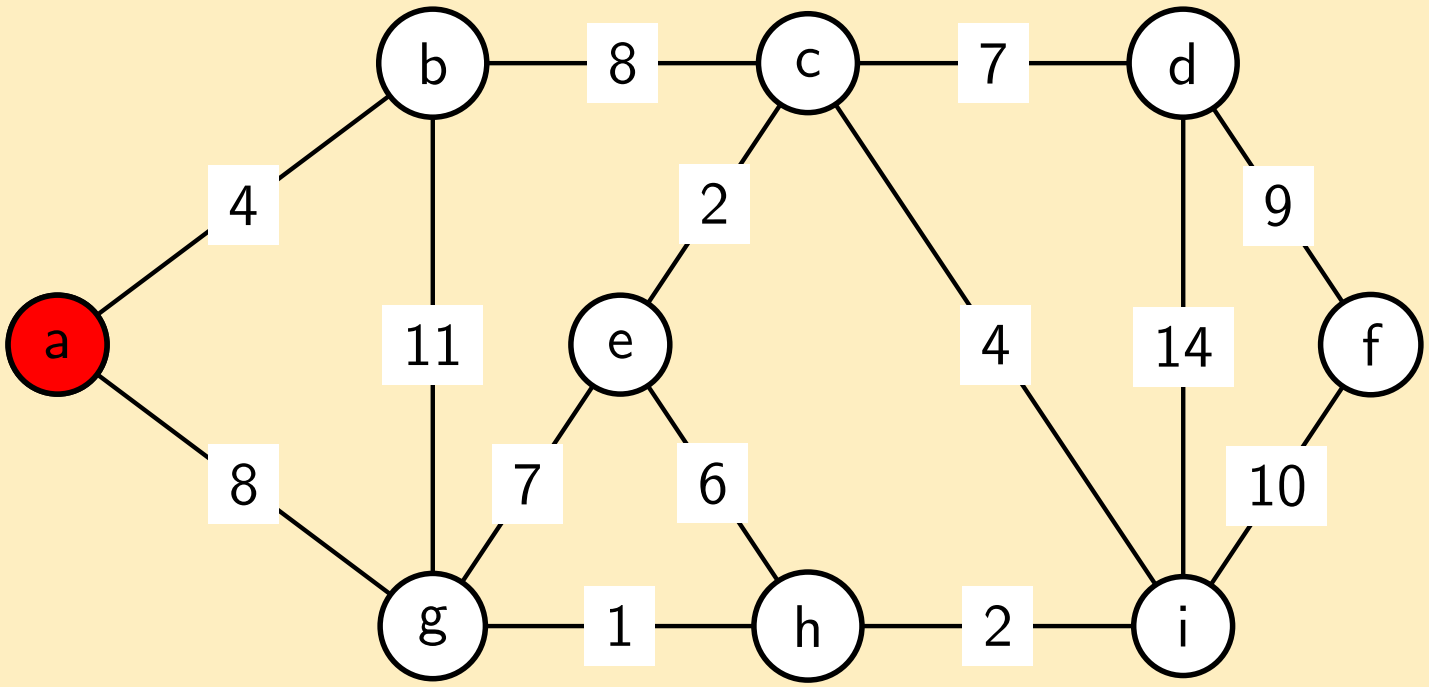
Voir le document "Algorithmes de graphes" page 5.

Complexité

L'algorithme de Prim est en $O(m + n \times log(n))$, à condition d'utiliser un tas de Fibonacci pour gérer les arêtes candidates à chaque étape.

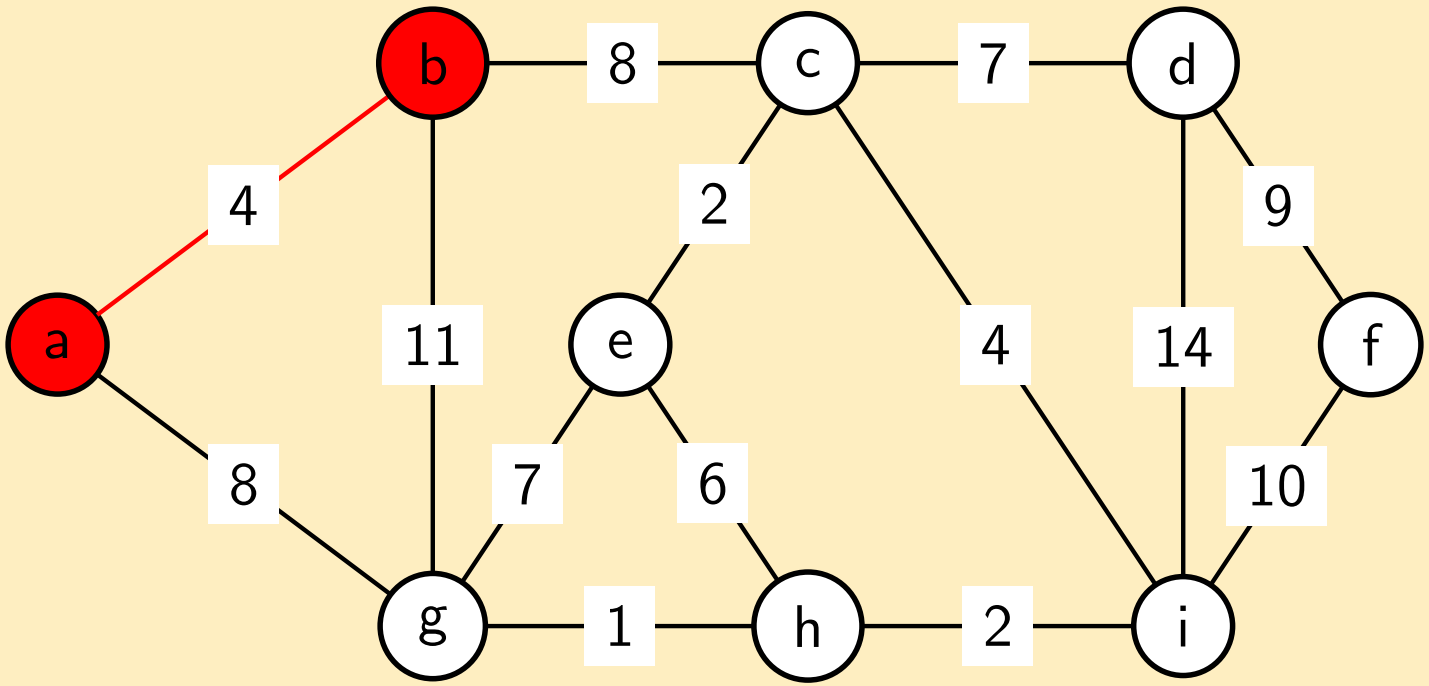
Prim

Exemple



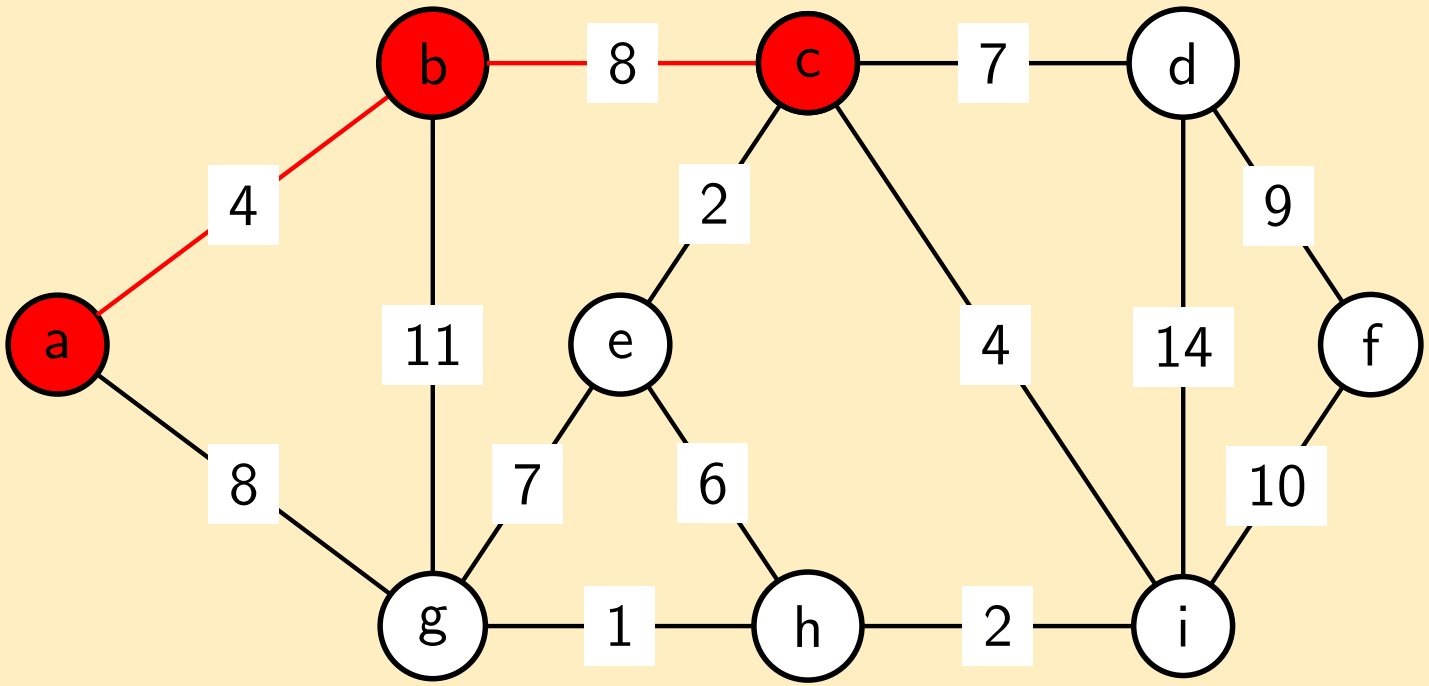
Prim

Exemple



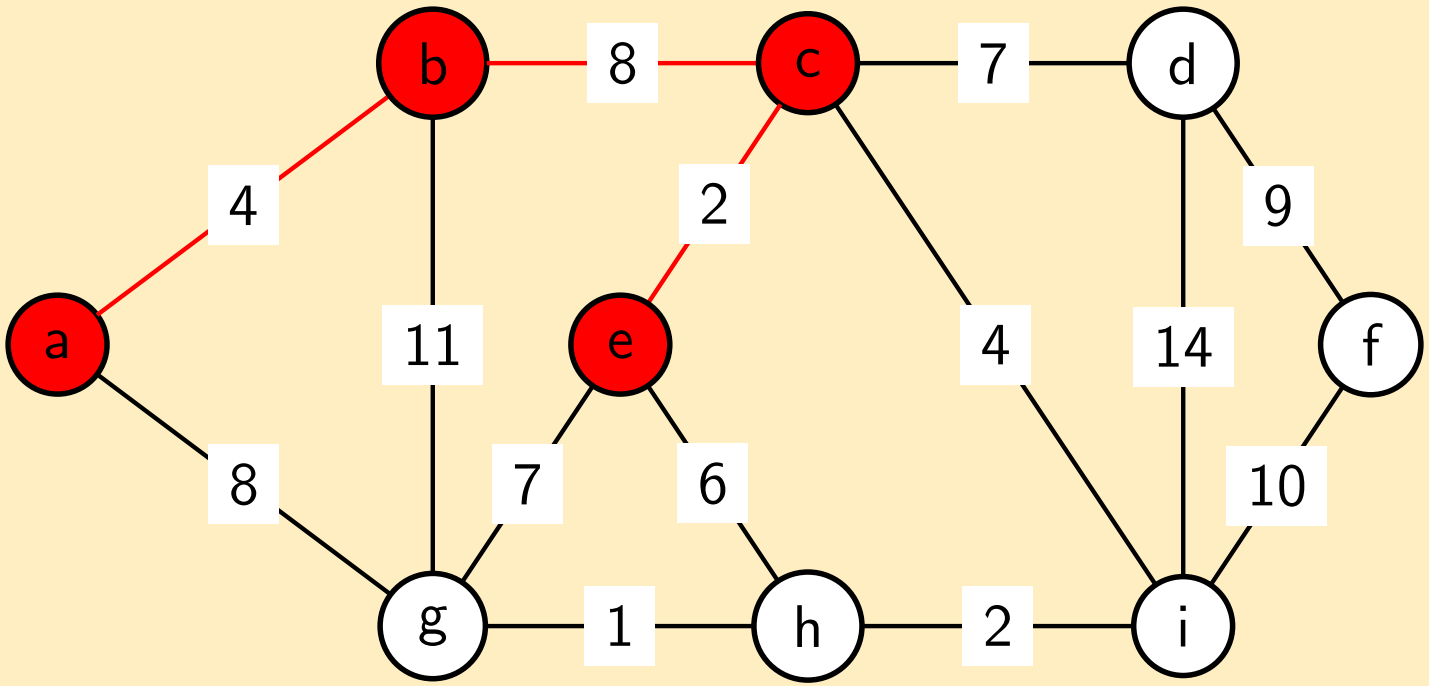
Prim

Exemple



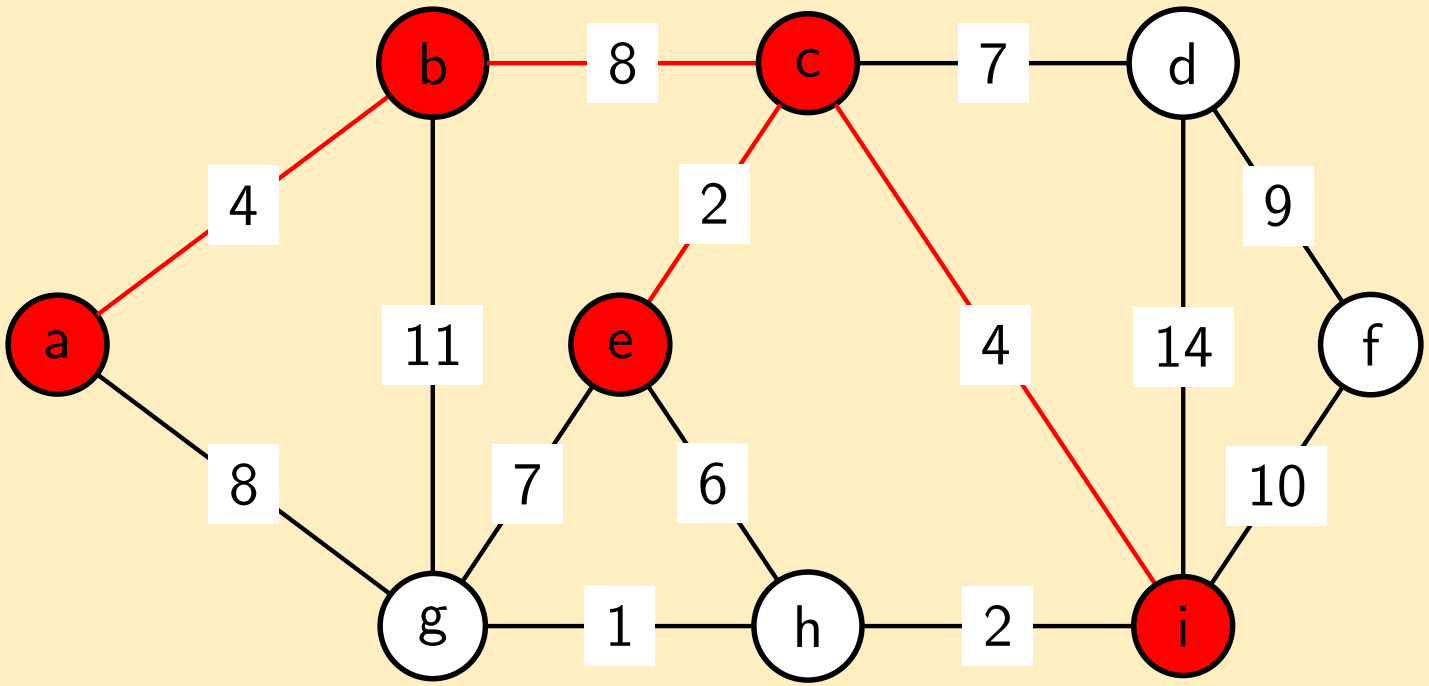
Prim

Exemple



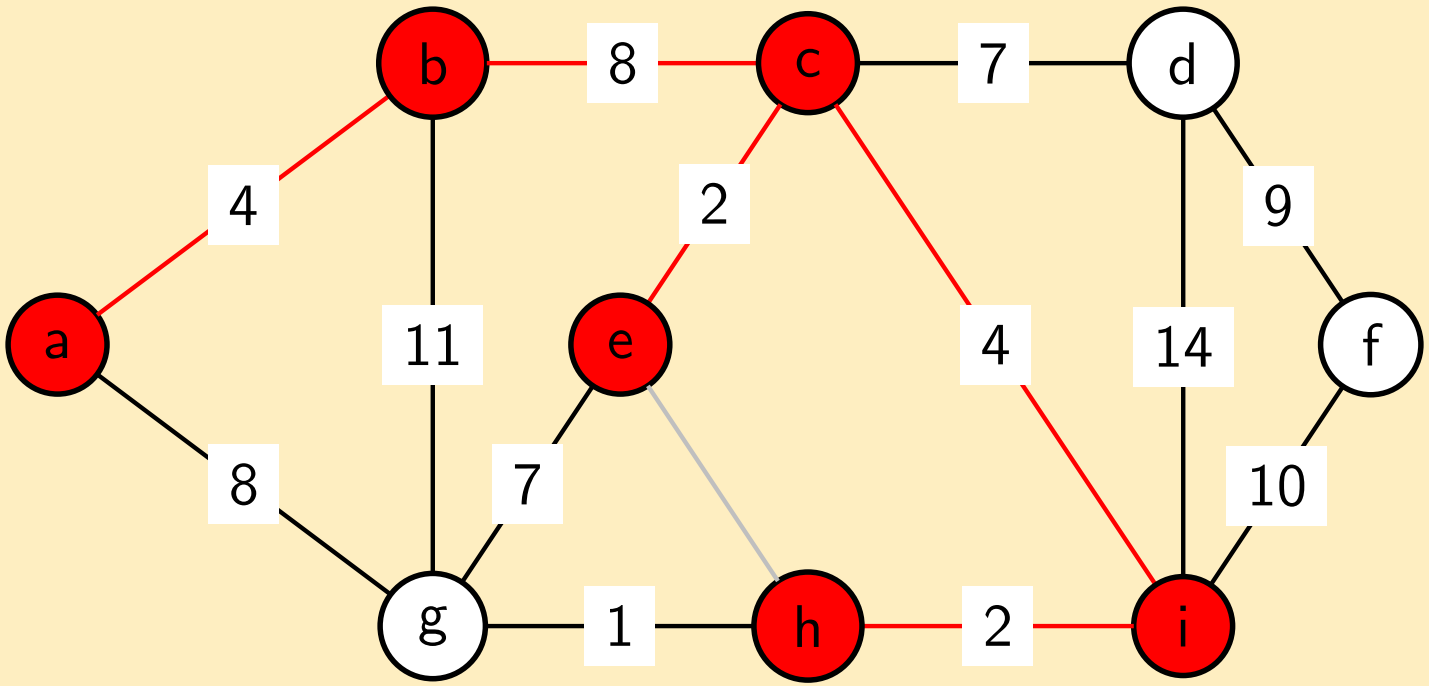
Prim

Exemple



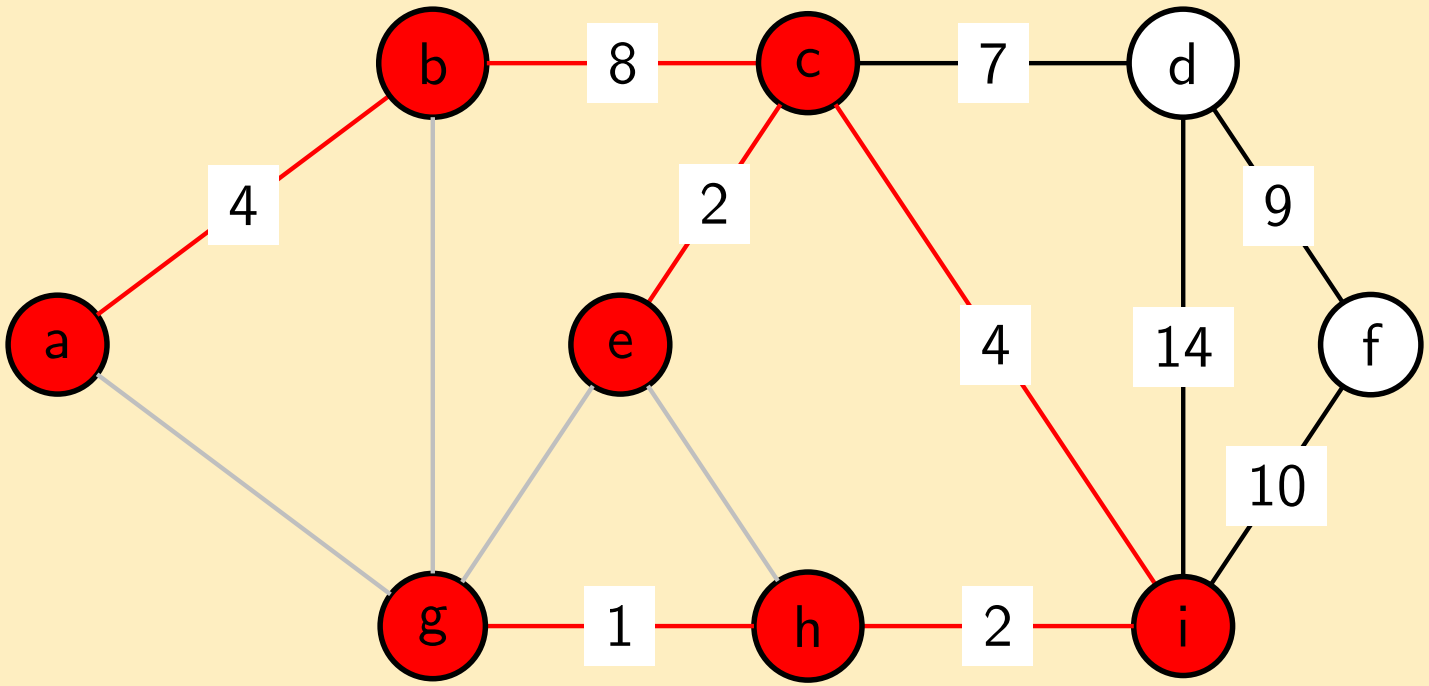
Prim

Exemple



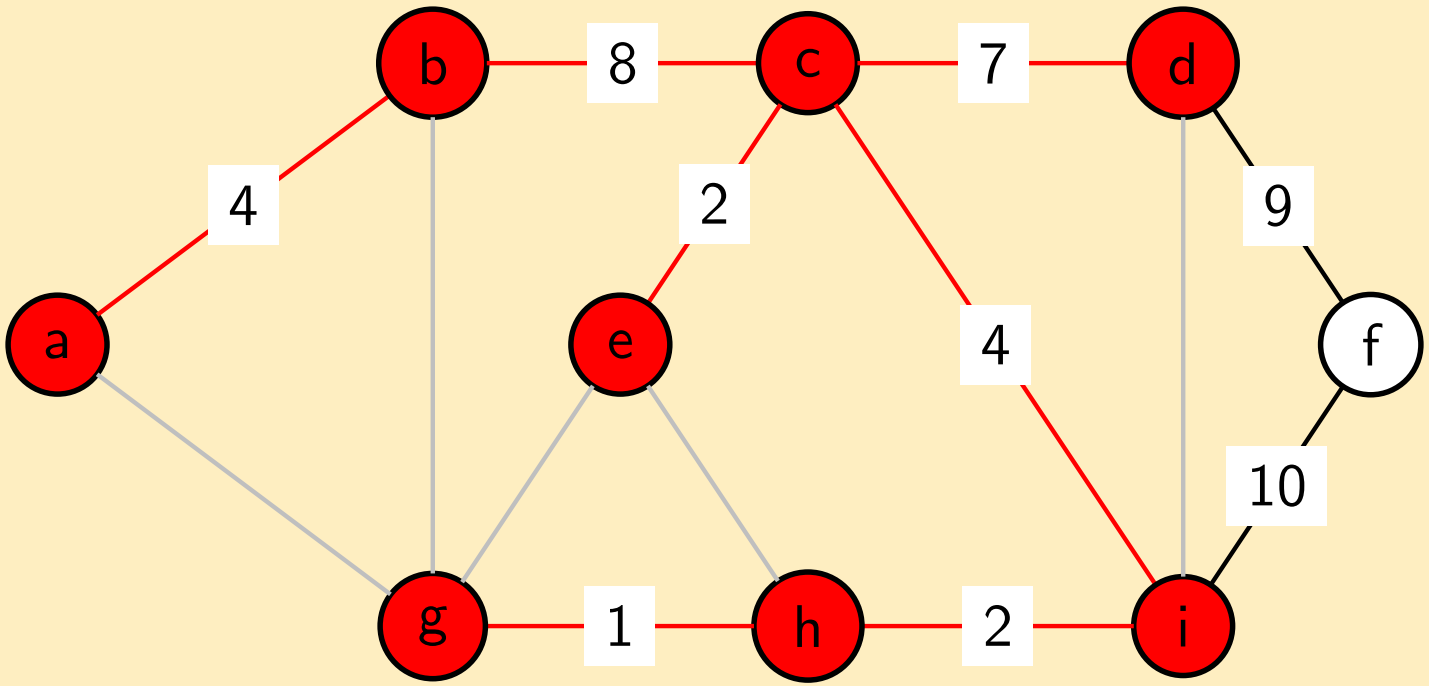
Prim

Exemple



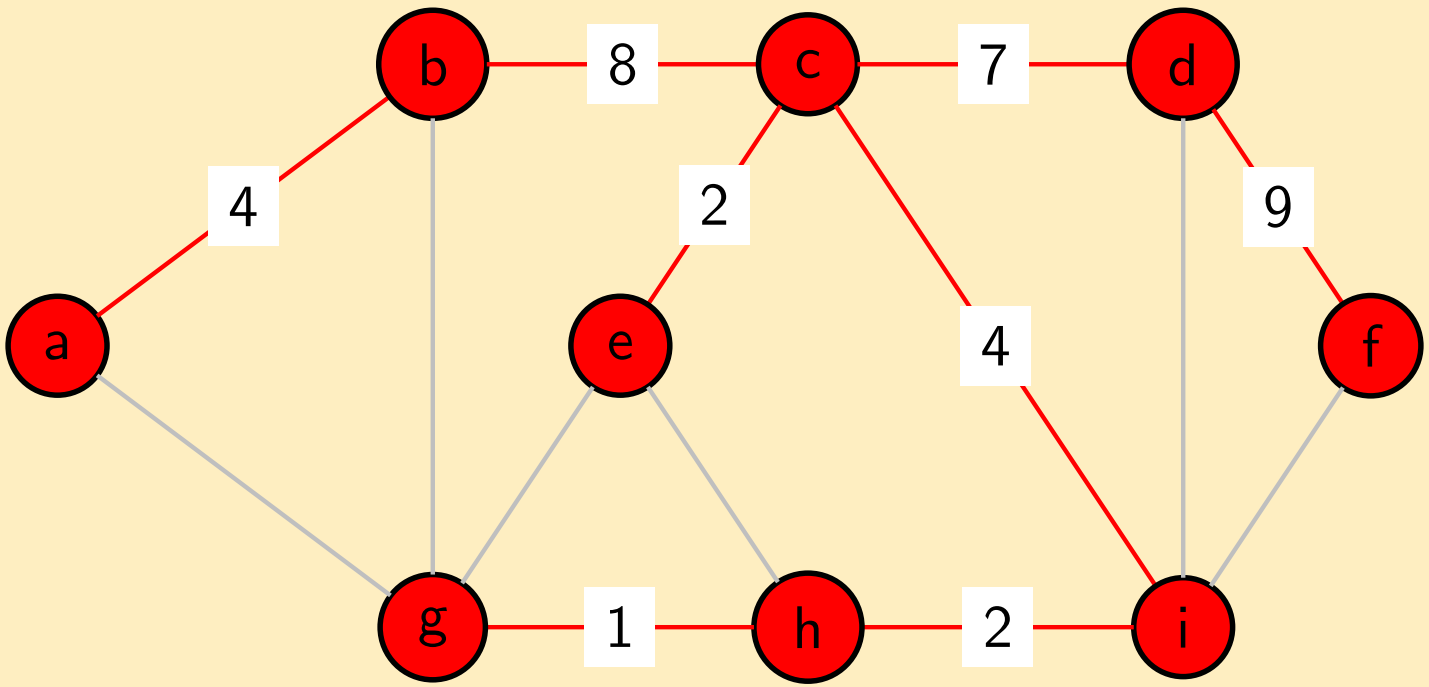
Prim

Exemple



Prim

Exemple



Poids total $= 4 + 8 + 2 + 7 + 9 + 4 + 2 + 1 = 37$

Prim

Justification

On construit un graphe connexe à n sommets et $n-1$ arêtes. Donc c'est un arbre.
Soit T_i l'arbre construit à l'étape i. On va montrer qu'il existe un arbre T^* couvrant de poids minimum tel que T^* contient T_i.
Pour $i = 0$, l'assertion est vraie.
Supposons qu'elle soit vraie à l'étape $i-1$. Soit T^* un arbre couvrant de poids minimum contenant T_{i-1} et notons A l'ensemble des sommets de T_{i-1}.
Soit $e = xy$ l'arête choisie par Prim à l'étape i, avec $x \in A$ et $y \notin A$.
Si $e \in T^*$, alors T_i est inclus dans T^*.
Sinon $\exists e' \in T^*$ avec $e' = x'y'$, $x' \in A$ et $y' \notin A$ et $e' \notin T_i$
Soit $T' = T^* + e - e'$. T' est un arbre de poids $\omega(T^*) + \omega(e) - \omega(e')$.
Or e a été choisie comme une arête de poids minimum entre A et $\bar{A}$.
Donc $\omega(e) \leq \omega(e')$ et donc $\omega(T') \leq \omega(T^*)$. Comme T^* est de poids minimum, on a $\omega(T') = \omega(T^*)$ et donc il existe bien un arbre de poids minimum contenant T_i.

Conclusion : T_{n-1} est de poids minimum.

Plus courts chemins

Problématiques

Soit G un graphe orienté et ω une fonction de poids sur les arcs de G.
Si G n'est pas orienté, on peut considérer chaque arête comme une paire d'arcs symétriques, sauf pour les boucles qui seront simplement orientées.

1. Quel est le plus court chemin pour aller de u à v ?
2. Quel est le plus court chemin d'un sommet u à tous les sommets du graphe ?
3. Quel est le plus court chemin entre toute paire de sommets ?

Plus courts chemins

Théorème

Il existe un plus court chemin de u à v dans (G, w) si et seulement si il existe un chemin de u à v et aucun chemin de u à v ne contient de circuit de longueur totale strictement négative.

Arborescence

L'ensemble des plus courts chemins à partir d'un sommet s forme une arborescence de racine s.
On notera par $d(v)$ la distance trouvée de s à v et par $\pi(v)$ le prédécesseur de v sur le chemin de s à v de longueur $d(v)$.

Plus courts chemins

Principe du relâchement

Soient
d la valeur du plus court chemin trouvé à l'instant t entre un sommet s et les sommets de G,
π les prédécesseurs de chaque sommet sur les plus courts chemins trouvés à l'instant t depuis le sommet s,
uv un arc de G,
Relacher(G, ω, u, v**)** :

1: **si** $d(u) + \omega(uv) < d(v)$ **alors**
2: $\quad d(v) \leftarrow d(u) + \omega(uv)$
3: $\quad \pi(v) \leftarrow u$
4: **fin si**

Algorithme de Dijkstra

Utilisation

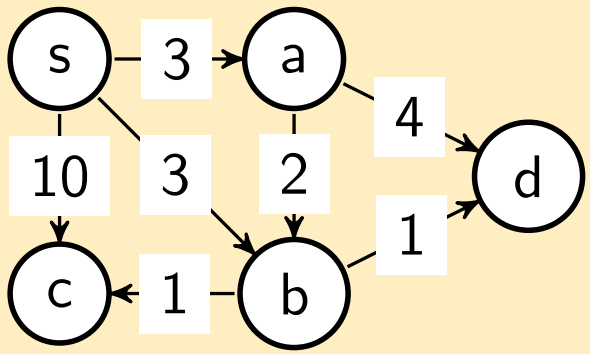
On peut utiliser l'algorithme de Dijkstra à condition que la fonction ω de pondération des arcs soit positive.

Énoncé

Voir document "Algorithmes des graphes" page 6.

Algorithme de Dijkstra

Exemple



Graphe G

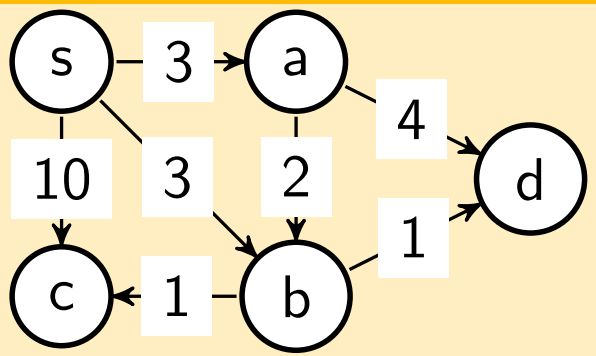
pivot	*s*	*a*	*b*	*c*	*d*
	0	∞	∞	∞	∞
s	*X*	3	3	10	∞
a		*X*	3	10	7
b			*X*	4	4
c				*X*	4
d					*X*

Dijkstra(G, ω, s)

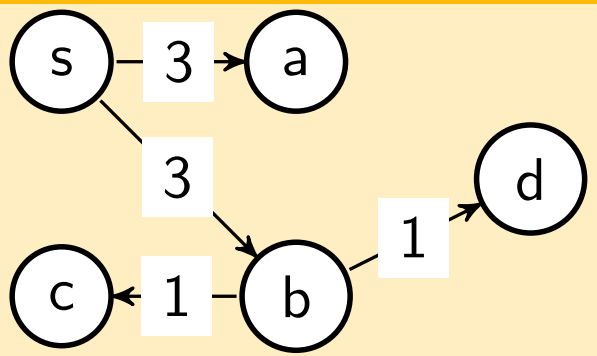
Remarque : $\pi(s) = \text{NIL}$ et $\pi(v)$ est le pivot quand $d(v)$ a été modifié pour la dernière fois.
Ici : $\pi(s) = NIL, \pi(a) = s, \pi(b) = s, \pi(c) = b, \pi(d) = b$.

Algorithme de Dijkstra

Exemple - Arborescence des plus courts chemins



Graphe G



Arborescence

Algorithme de Dijkstra

Complexité

Si la file de priorité est implémentée sous la forme d'une liste, alors la complexité de l'algorithme de Dijkstra est en $O(n^2 + m)$. Si G ne possède pas d'arcs parallèles, cela nous donne du $O(n^2)$.
Si le nombre d'arcs est faible par rapport à n^2, on a intérêt à implémenter la file de priorité à l'aide d'un tas binaire. Dans ce cas, la complexité de Dijkstra devient $O((n + m) \times \log(n))$.

Algorithme de Dijkstra

Validité

On note par $\delta(u)$ la plus courte distance entre s et u dans G.
On va montrer que $d(u) = \delta(u)$ à chaque fois qu'un sommet u est extrait de la file de priorité F.
Supposons que ce ne soit pas le cas. Soit u le premier sommet extrait de F tel que $d(u) \neq \delta(u)$ à cet instant.
u ne peut pas être égal à s, car s est le premier sommet extrait de F et $d(s) = \delta(s) = 0$.
De plus, $d(u) \neq \infty$, sinon, u n'aurait pas été inséré dans F. Donc il existe lors de l'extraction de u un chemin de s à u de longueur $d[u] \neq \infty$ et donc il existe un plus court chemin dans G de longueur $\delta(u) \neq \infty$. Soit p ce chemin et soit y le premier sommet de p non extrait de F quand on extrait u. y existe, sinon on aurait $d(u) = \delta(u)$. Soit x le prédécesseur de y dans p. Comme x a été extrait de F avant u, cela signifie que $d(y) = \delta(y)$ quand on extrait u et comme $\delta(y) < \delta(u) < d[u]$, y aurait du être extrait de F avant u. Donc u n'existe pas !

Algorithme de Bellman

Utilisation

On peut utiliser l'algorithme de Bellman à condition que le graphe G soit sans circuit.
Dans ce cas, on peut obtenir un tri topologique sur les sommets de G, c'est à dire un ordre tel que si un arc va du sommet de rang i vers le sommet de rang j, alors $i < j$.

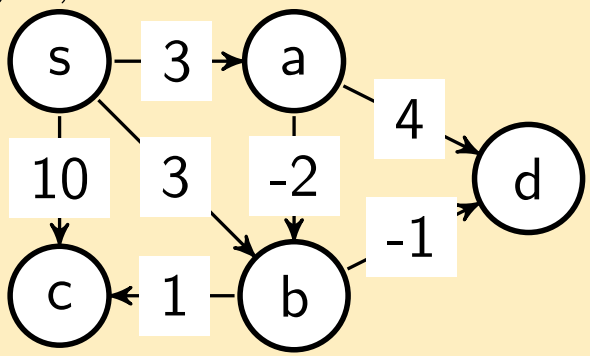
Énoncé

Voir document "Algorithmes des graphes" page 7.

Algorithme de Bellman

Exemple

Le tri topologique de G peut donner deux ordres : s, a, b, c, d et s, a, b, d, c. On choisit d'utiliser s, a, b, c, d.



Graphe G

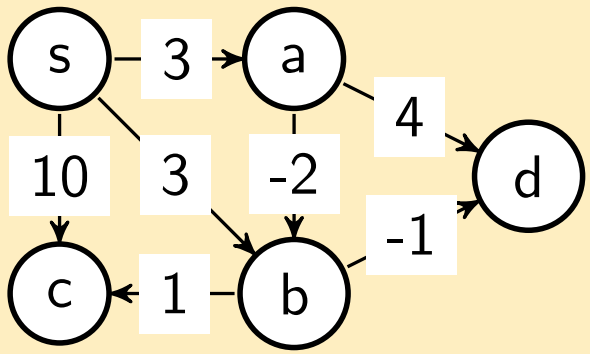
u	s	a	b	c	d
	0	∞	∞	∞	∞

Bellman(G, ω, s)

Algorithme de Bellman

Exemple

Le premier sommet dans le tri topologique s, a, b, c, d est s.



Graphe G

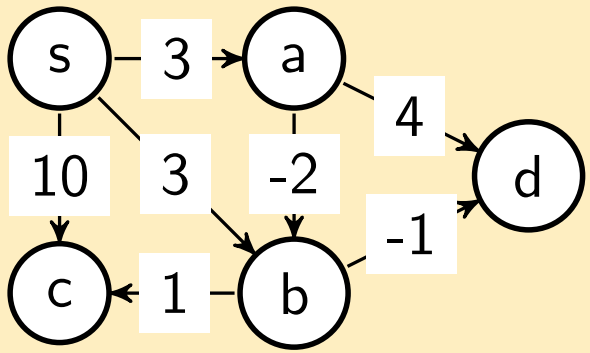
u	s	a	b	c	d
	0	∞	∞	∞	∞
s	0	3	3	10	∞

Bellman(G, ω, s)

Algorithme de Bellman

Exemple

Le second sommet dans le tri topologique s, a, b, c, d est a.



Graphe G

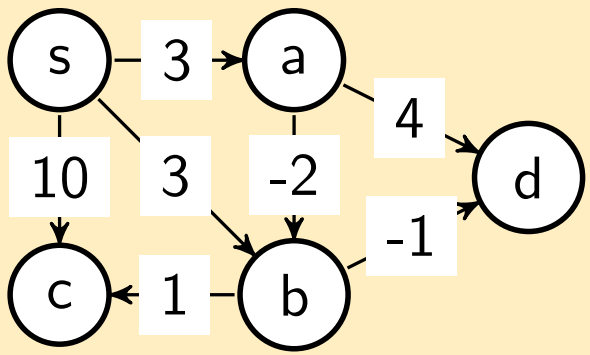
u	s	a	b	c	d
	0	∞	∞	∞	∞
s	0	3	3	10	∞
a	0	3	1	10	7

Bellman(G, ω, s)

Algorithme de Bellman

Exemple

Le troisième sommet dans le tri topologique s, a, b, c, d est b.



Graphe G

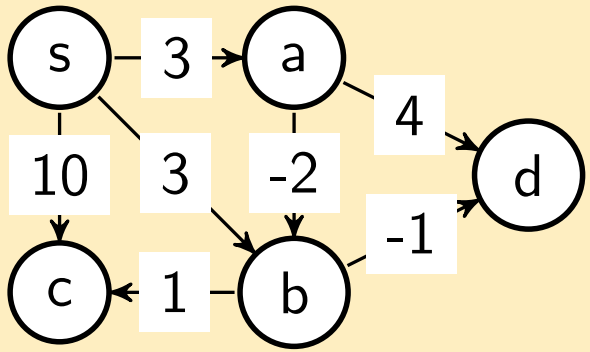
u	s	a	b	c	d
	0	∞	∞	∞	∞
s	0	3	3	10	∞
a	0	3	1	10	7
b	0	3	1	2	0

Bellman(G, ω, s)

Algorithme de Bellman

Exemple

Le quatrième sommet dans le tri topologique s, a, b, c, d est c.



Graphe G

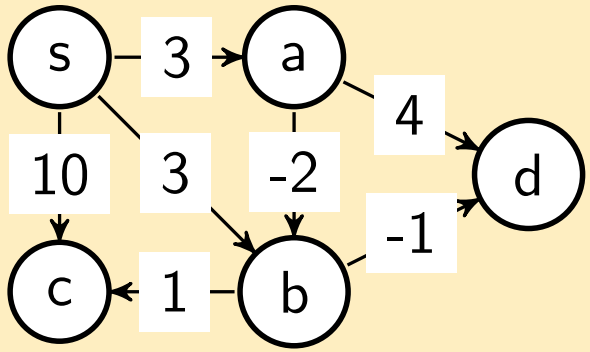
u	s	a	b	c	d
	0	∞	∞	∞	∞
s	0	3	3	10	∞
a	0	3	1	10	7
b	0	3	1	2	0
c	0	3	1	2	0

Bellman(G, ω, s)

Algorithme de Bellman

Exemple

Le cinquième (et dernier) sommet dans le tri topologique s, a, b, c, d est d.



Graphe G

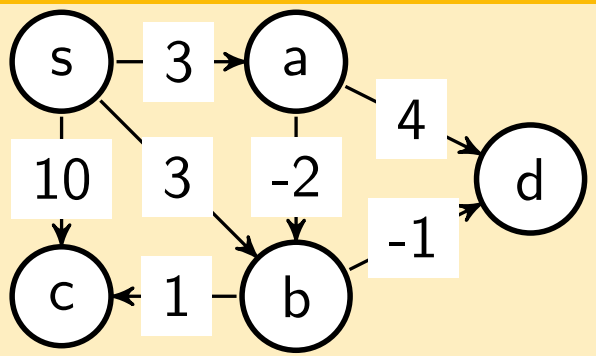
u	s	a	b	c	d
	0	∞	∞	∞	∞
s	0	3	3	10	∞
a	0	3	1	10	7
b	0	3	1	2	0
c	0	3	1	2	0
d	0	3	1	2	0

Bellman(G, ω, s)

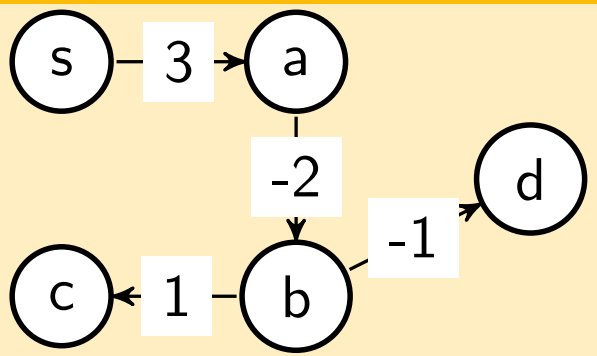
Remarque : $\pi(s) = \text{NIL}$ et $\pi(v)$ est le sommet traité dans l'ordre topologique quand $d(v)$ a été modifié pour la dernière fois.
Ici : $\pi(s) = NIL, \pi(a) = s, \pi(b) = a, \pi(c) = b, \pi(d) = b$.

Algorithme de Bellman

Exemple - Arborescence des plus courts chemins



Graphe G



Arborescence

Algorithme de Bellman

Complexité

La complexité de l'algorithme de Bellman est en $O(n+m)$.

Validité

On a bien $d(s) = \delta(s)$
Si $v_1 = s, v_2, \dots v_n$ est l'ordre obtenu par le tri topologique, il est clair que si pour tout $j < i$, $d(v_j) = \delta(v_j)$, alors $d(v_i) = \delta(v_i)$.
Donc par récurrence, on a bien $\forall i, 1 \leq i \leq n, d(v_i) = \delta(v_i)$.

Algorithme de Bellman

Remarque : cas particulier

Si $v_1, \dots v_n$ est l'ordre obtenu lors du tri topologique et que $s = v_i$ avec $i > 1$, alors $v_1, \dots v_{i-1}$ ne sont pas accessibles à partir de s et donc on aura $\forall j < i, d(v_j) = \infty$.