Recherche Opérationnelle - Cours 5

Olivier Baudon

Université de Bordeaux

11 octobre 2022

Composantes fortement connexes

Énoncé de l'algorithme

Voir le document "Algorithmes de graphes" page 4.

Complexité

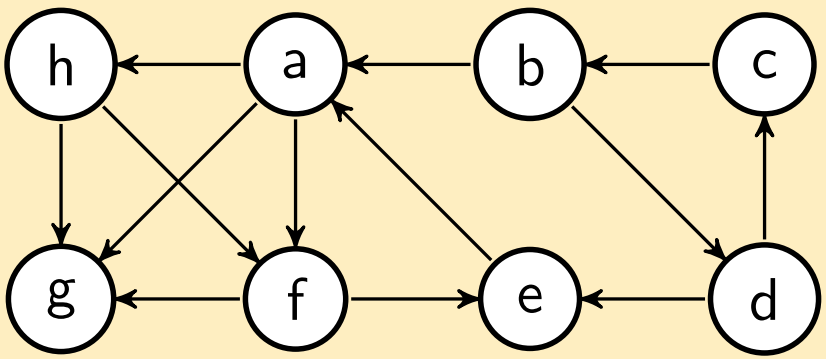
L'algorithme exécute deux parcours en profondeur, soit une complexité de $O(n+m)$. De plus, le calcul du graphe inversé G^{-1} est également en $O(n+m)$ en utilisant par exemple l'algorithme suivant où $Adj_G(v)$ désigne la liste des successeurs du sommet v dans le graphe G.

Inverser(G) :

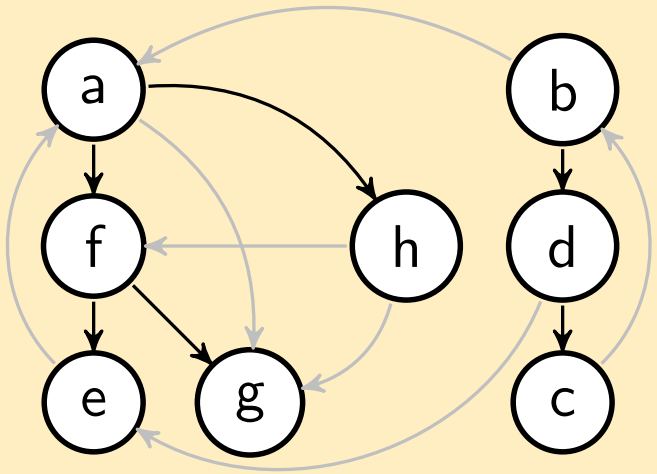
1: $G^{-1} \leftarrow (V(G), \emptyset)$
2: **pour tout** $v \in V(G)$ **faire**
3: **pour tout** $w \in Adj_G(v)$ **faire**
4: $INSERER(v, Adj_{G^{-1}}(w))$
5: **fin pour**
6: **fin pour**
7: **retourner** G^{-1}

Composantes fortement connexes

Exemple



v :	a	b	c	d	e	f	g	h
$d(v)$	1	11	13	12	3	2	5	8
$f(v)$	10	16	14	15	4	7	6	9

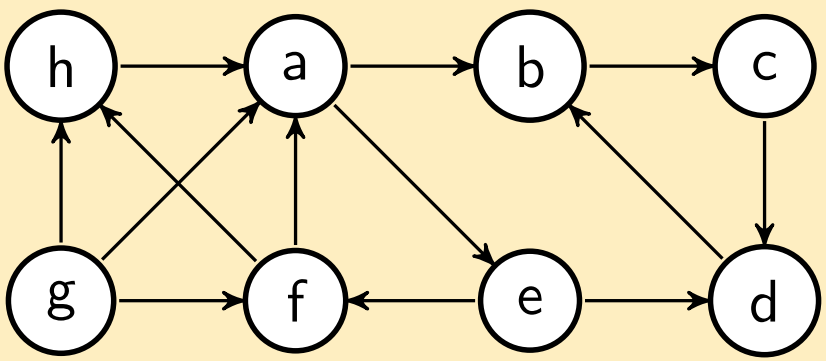


Ordre : b, d, c, a, h, f, g, e

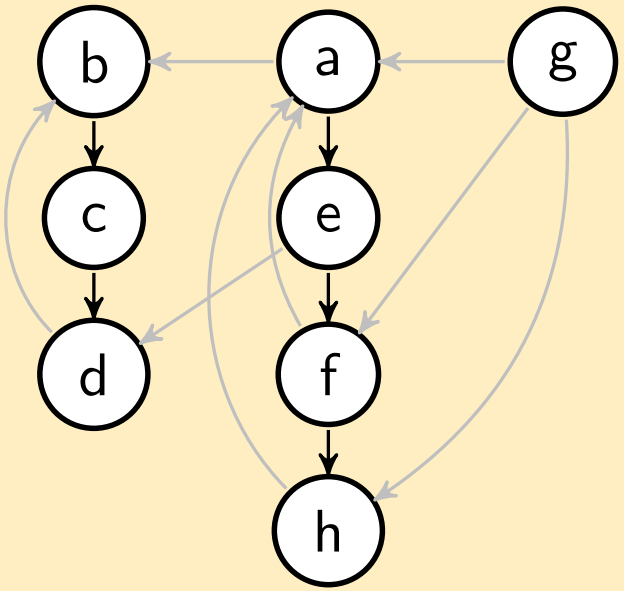
Composantes fortement connexes

Exemple (Suite)

On calcule d'abord le graphe inverse G^{-1}, puis on exécute $PP(G^{-1})$ en traitant dans la boucle 6 de $PP(G^{-1})$ les sommets dans l'ordre inverse de f obtenu lors de $PP(G)$.



Ordre : b, d, c, a, h, f, g, e



CFC : $\{b, c, d\}, \{a, e, f, h\}, \{g\}$.

Composantes fortement connexes

Justification

Lemme 1 : si deux sommets u et v sont dans une même composante fortement connexe, alors aucun chemin entre eux n'utilise de sommet extérieur à la composante.

Preuve : soit C la composante fortement connexe contenant u et v. Soit w un sommet situé sur un chemin P de u à v.

$P = P_{uw}|P_{wv}$ où P_{uw} est un chemin de u à w et P_{wv} un chemin de w à v. D'autre part, comme $v \in C$, il existe un chemin P_{vu} de v à u. Donc il existe un chemin de u à w : P_{uw} et un chemin de w à u : $P_{wv}|P_{vu}$. Donc $w \in C$.

Composantes fortement connexes

Justification

Lemme 2 : lors d'un parcours en profondeur, tous les sommets d'une même composante fortement connexe se trouveront dans la même arborescence.

Preuve : soit C une composante fortement connexe et soit u le premier sommet de C visité lors d'un parcours en profondeur. Lors de la visite de u, tous les autres sommets de C sont blancs et donc pour tout sommet $v \in C$, il existe un chemin de u à v dont tous les sommets sont blancs. D'après le théorème du chemin blanc, v sera un descendant de u. Et donc tous les sommets de C seront dans la même arborescence que u.

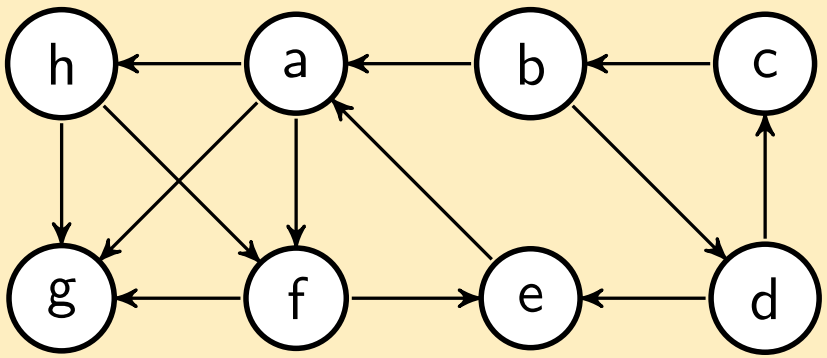
Composantes fortement connexes

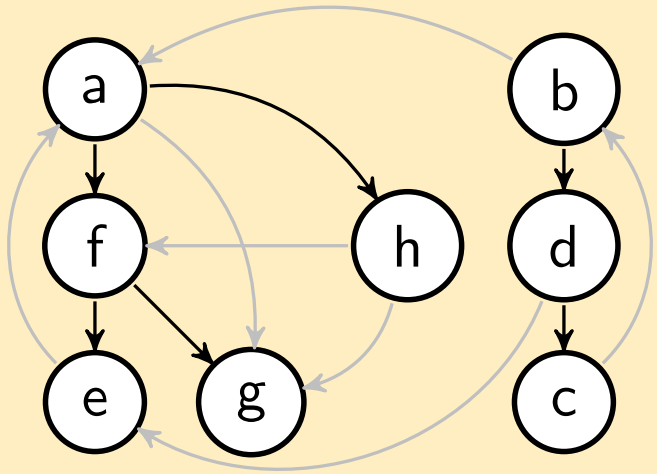
Justification

Lors d'un parcours en profondeur d'un graphe G, on appelle aïeul d'un sommet u, le sommet noté $\phi(u)$ qui est parmi les sommets v accessibles à partir de u celui qui a la plus grande valeur pour f. Noter que l'on peut avoir $\phi(u) = u$.

Composantes fortement connexes

Exemple de calcul de ϕ





v :	a	b	c	d	e	f	g	h
$d(v)$	1	11	13	12	3	2	5	8
$f(v)$	10	16	14	15	4	7	6	9
$\phi(v)$	a	b	b	b	a	a	g	a

Composantes fortement connexes

Justification

Lemme 3 : si deux sommets u et v sont tels qu'il existe un chemin de u à v, alors pour tout parcours en profondeur, on aura $f(\phi(u)) \geq f(\phi(v))$.
Preuve : comme il existe un chemin de u à v et un chemin de v à $\phi(v)$, alors il existe un chemin de u à $\phi(v)$ et donc $f(\phi(u)) \geq f(\phi(v))$.

Justification

Lemme 4 : pour tout sommet u, $\phi(\phi(u)) = \phi(u)$.
Preuve : il existe un chemin de u à $\phi(u)$. Donc d'après le Lemme 3, $f(\phi(u)) \geq f(\phi(\phi(u)))$. Or par définition, $f(\phi(\phi(u))) \geq f(\phi(u))$. Donc $f(\phi(\phi(u))) = f(\phi(u))$ et donc $\phi(\phi(u)) = \phi(u)$.

Composantes fortement connexes

Justification

Lemme 5 : pour tout sommet u, lors d'un parcours en profondeur, $\phi(u)$ est un ancêtre de u.

Preuve :

Il faut montrer que lorsqu'on visite un sommet u pour la première fois, $\phi(u)$ est gris.

- $\phi(u)$ ne peut être noir, sinon on aurait $f(\phi(u)) < d(u)$ et donc $f(\phi(u)) < f(u)$. Ce qui est impossible par définition de ϕ.

Composantes fortement connexes

Justification

Preuve du Lemme 5 (suite et fin) :

- Montrons que $\phi(u)$ ne peut être blanc.
  Si lors de la visite de u, tous les sommets du chemin entre u et $\phi(u)$ sont blancs, alors $\phi(u)$ sera un descendant de u et on aura $f(\phi(u)) < f(u)$, ce qui est impossible.
  S'il existe des sommets non blancs sur le chemin de u à $\phi(u)$, soit v le sommet non blanc sur ce chemin le plus proche de $\phi(u)$. Donc tous les sommets entre v et $\phi(u)$ seront blancs (v non compris). Comme v a un successeur blanc, v ne peut être noir et donc v est gris. Or il existe un chemin blanc de v à $\phi(u)$ et donc $\phi(u)$ sera un descendant de v et donc $f(\phi(u)) < f(v)$. Or v est accessible par un chemin depuis u et $f(v) > f(\phi(u))$, ce qui est contradictoire avec la définition de ϕ.

Composantes fortement connexes

Justification

Lemme 6 : pour tout sommet u, u et $\phi(u)$ appartiennent à la même composante fortement connexe.

Preuve : on a montré dans le Lemme 5 que pout tout sommet u, $\phi(u)$ est un ancêtre de u. Donc il existe un chemin de $\phi(u)$ à u. Par définition de ϕ, il existe également un chemin de u à $\phi(u)$. Donc u et $\phi(u)$ appartiennent à la même composante fortement connexe.

Composantes fortement connexes

Justification

Lemme 7 : deux sommets u et v appartiennent à une même composante fortement connexe si et seulement si ils ont le même aïeul.

Preuve : soient u et v deux sommets appartenant à la même composante fortement connexe C. D'après le Lemme 6, $\phi(u)$ et $\phi(v)$ appartiendront également à C. Donc $\phi(u)$ et $\phi(v)$ sont tous les deux accessibles depuis u et v et donc par définition de ϕ $f(\phi(u)) \geq f(\phi(v))$ car $\phi(u)$ est l'aïeul de u et $f(\phi(v)) \geq f(\phi(u))$ car $\phi(v)$ est l'aïeul de v. Donc $f(\phi(v)) = f(\phi(u))$ et donc $\phi(v) = \phi(u)$.

Inversement, si deux sommets u et v ont le même aïeul, alors ils appartiennent tous les deux à la composante fortement connexe de cet aïeul et donc à la même composante fortement connexe.

Composantes fortement connexes

Justification

Lemme 8 : soit T une arborescence obtenue lors du parcours en profondeur de G^{-1} de racine r. Pour tout sommet u, si $\phi(u)$ désigne l'aïeul de u lors du parcours en profondeur de G (étape 1), alors $\phi(u) = r$ si et seulement si u appartient à T.

Preuve :

r_1 est le sommet ayant la plus grande valeur de f lors de $PP(G)$. Donc il sera l'aïeul de tous les sommets de sa composante fortement connexe, disons C_1. De plus, lors du début de la visite de r_1 dans de $PP(G^{-1})$, tous les sommets de C_1 seront blancs et donc seront des descendants de r_1.

De plus, si un sommet v ne possède pas r_1 comme aïeul, cela signifie qu'il n'existe pas de chemin de v à r_1 dans G et donc pas de chemin de r_1 à v dans G^{-1}. Donc v ne sera pas dans l'arborescence de r_1.

Composantes fortement connexes

Justification

Suite de preuve du Lemme 8 : supposons que l'hypothèse soit vérifiée pour les $k-1$ premières arborescence. Soit T_k l'arborescence suivante de racine r_k, C_k la composante fortement connexe de r_k dans G et $\phi(C_k)$ l'aïeul des sommets de C_k obtenu lors de l'étape 1.
Comme $\phi(C_k)$ est le sommet ayant la plus grande valeur de f parmi les sommets non encore visités, on aura $r_k = \phi(C_k)$.
Si un sommet $v \in C_k$, v est blanc lors de l'examen de r_k et sera donc ajouté à T_k. De même, soit un sommet $w \notin C_1 \cup \ldots \cup C_k$. Comme il n'existe pas de chemin de w à $\phi(C_k)$ dans G, w ne sera pas ajouté à T_k. Donc T_k contiendra exactement les sommets de C_k.
Par induction, le Lemme 8 est donc vrai et justifie l'algorithme CFC.

Arbres

Définition

Un arbre est un graphe (non orienté) connexe sans cycle.
Remarque : un arbre est forcément un graphe simple.

Caractérisations des arbres

Les propriétés suivantes sont équivalentes :

1. G est un arbre.
2. G est sans cycle et possède m arêtes avec $m = n - 1$.
3. G est connexe et possède m arêtes avec $m = n - 1$.
4. G est sans cycle, et en ajoutant une arête on obtient un graphe ayant un cycle (et un seul).
5. G est connexe, et si on supprime une arête quelconque le graphe obtenu n'est plus connexe.
6. Tout couple de sommets est relié par une chaîne élémentaire et une seule.

Lemme 1

Soit G un graphe connexe à n sommets et $m = n - 1$ arêtes, avec $n \geq 2$. Alors G possède au moins un sommet de degré 1.

Preuve : Comme G est connexe, tous les sommets sont de degré au moins 1. Si tous les sommets sont de degré au moins 2, alors $\sum_{v \in V(G)} deg(v) \geq 2 \times n$, et donc $m \geq n$. Donc il existe au moins un sommet de degré exactement 1.

Lemme 2

Tout arbre ayant au moins deux sommets possède au moins deux sommets de degré 1.

Preuve : Soit T un arbre et $P = v_1, v_2, \dots v_k$ une chaîne de longueur maximale de T.

Si v_1 possède un voisin u hors de $\{v_2, \dots v_k\}$, alors on peut rallonger P en rajoutant u devant v_1. Donc v_1 ne possède aucun voisin hors de $\{v_2, \dots v_k\}$.

Si v_1 possède un voisin parmi $\{v_2, \dots v_k\}$ autre que v_2, alors T contient un cycle. Donc v_2 est le seul voisin de v_1.

Donc v_1 est de degré 1 et par symétrie, v_k également.

Preuve du théorème

On va montrer que $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6 \Rightarrow 1$.

$1 \Rightarrow 2$: Si G est un arbre, alors G est sans cycle et possède $n-1$ arêtes.

Preuve Si G est un arbre, alors il est sans cycle.
Pour le nombre d'arêtes, nous allons procéder par induction.
Les seuls arbres à 1 ou 2 sommets sont respectivement K_1 et K_2 qui possèdent respectivement 1 et 2 sommets et 0 et 1 arête.
Supposons que la propriété soit vraie pour tout arbre ayant au plus $n-1$ sommets avec $n-1 \geq 1$. Soit T un arbre à n sommets.
On sait que T possède au moins deux sommets de degré 1. Soit v l'un de ces sommets.
$T' = T - v$ est connexe et sans cycle, donc un arbre avec $n-1$ sommets et donc $n-2$ arêtes. Si on rajoute v à T' avec son arête, le graphe reste connexe et sans cycle, avec n sommets et $n-1$ arêtes. CQFD.

$2 \Rightarrow 3$: Si G est sans cycle et possède m arêtes avec $m = n - 1$ alors G est connexe et possède m arêtes avec $m = n - 1$.

Preuve : On doit donc montrer que si G est sans cycle avec $n - 1$ arêtes, alors G est connexe.
Soit G sans cycle avec $n - 1$ arêtes. Supposons que G possède k composantes connexes $C_1, \ldots C_k$. Alors chacune des C_i est connexe et sans cycle et donc est un arbre. Donc pour tout i, $1 \leq i \leq k$, $m(C_i) = n(C_i) - 1$.
Le nombre total d'arêtes dans G sera donc $n - k$ et donc $k = 1$ et donc G est connexe.

Arbres

$3 \Rightarrow 4$: Si G est connexe et possède m arêtes avec $m = n - 1$ alors G est sans cycle, et en ajoutant une arête on obtient un graphe ayant un cycle (et un seul).

Preuve : Preuve par récurrence sur n. C'est vrai pour $n = 1$ car $G = K_1$ et en rajoutant une arête, on crée un cycle (une boucle) et un seul.
Supposons que la propriété soit vraie pour tout graphe connexe à $n - 1$ sommets et $n - 2$ arêtes, avec $n - 1 \geq 1$.
Soit G connexe avec n sommets et $n - 1$ arêtes. On sait que G possède au moins un sommet de degré 1, que nous nommerons v. $G - v$ est connexe avec $n - 1$ sommets et $n - 2$ arêtes. Donc $G - v$ est sans cycle et tout ajout d'arête à $G - v$ crée un cycle et un seul.
Si on rajoute v à $G - v$ avec son arête, on ne crée pas de cycle. Si on rajoute à G un arête $e = vw$, comme G est connexe, il existe une chaîne reliant v à w dans G et donc l'ajout de e crée un cycle. De plus, ce cycle est unique car si on enlève v, le graphe est sans cycle. D'autre part, tout ajout d'arête entre deux sommets différents de v crée un cycle unique dans $G - v$ et donc dans G. CQFD.

Arbres

$4 \Rightarrow 5$: Si G est sans cycle, et en ajoutant une arête on obtient un graphe ayant un cycle (et un seul), alors G est connexe, et si on supprime une arête quelconque, le graphe obtenu n'est plus connexe.

Preuve : Si G n'était pas connexe, alors on pourrait rajouter une arête à G sans créer de cycle (en rajoutant l'arête entre deux sommets appartenant à deux composantes connexes distinctes). Si on enlève une arête $e = uv$, alors si le graphe reste connexe, il existe un chaîne dans $G - e$ entre u et v et donc G possède un cycle passant par $uv \rightarrow$ Contradiction.

$5 \Rightarrow 6$: Si G est connexe, et tel que si on supprime une arête quelconque le graphe obtenu n'est plus connexe, alors tout couple de sommets est relié par une chaîne élémentaire et une seule.

Preuve : G est connexe, donc toute paire de sommets est relié par une chaîne.
Si en supprimant une arête quelconque, le graphe n'est plus connexe, cette chaîne est unique.

Arbres

$6 \Rightarrow 1$: Si G est tel que tout couple de sommets est relié par une chaîne élémentaire et une seule, alors G est un arbre.

Preuve : Comme toute paire de sommet est relié par une chaîne, G est connexe. Si G possède un cycle, alors, il y aurait au moins deux chaînes reliant toute paire de sommets appartenant au cycle.

Arbre couvrant de poids minimum

Graphe partiel

Soit $G = (V, E)$ un graphe. Un graphe partiel de G est un graphe $G' = (V, E')$ avec $E' \subseteq E$.

Cycle fondamental

Soit T un arbre. Alors tout ajout d'une arête $e = uv$ crée un cycle unique passant par e. Ce cycle sera appelé le cycle fondamental de e (par rapport à T).
Remarque : si on rajoute e à T, il suffit d'enlever n'importe quelle arête de son cycle fondamental pour obtenir à nouveau un arbre.

Arbre couvrant de poids minimum

Arbre couvrant de poids minimum

Soit (G, ω) un graphe muni d'une fonction de poids sur ses arêtes ω.
Un arbre couvrant de poids minimum de (G, ω) est un graphe partiel de G qui est un arbre et dont le poids total des arêtes est minimum.
Remarque : si ω est positive, alors un arbre couvrant de poids minimum est un graphe partiel connexe de poids minimum.

Algorithmes de calcul d'un arbre couvrant de poids minimum

Il existe essentiellement deux algorithmes pour calculer un arbre couvrant de poids minimum : l'algorithme de Kruskal et l'algorithme de Prim.

Kruskal

Principe

L'algorithme de Kruskal considère les arêtes une à une dans l'ordre croissant de leur poids et ne les garde que si elles ne créent pas un cycle ;

Énoncé

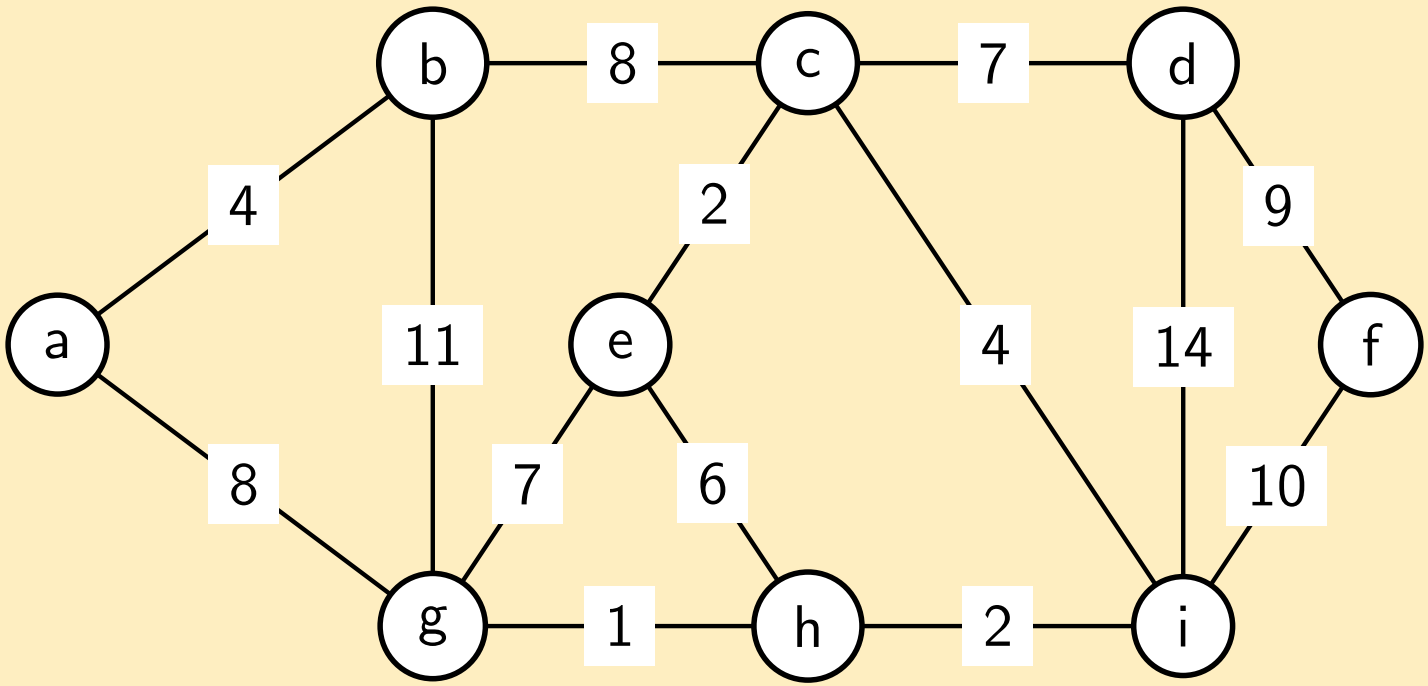
Voir le document "Algorithmes de graphes" page 5.

Complexité

L'algorithme de Kruskal est en $O(m \times log(m))$.

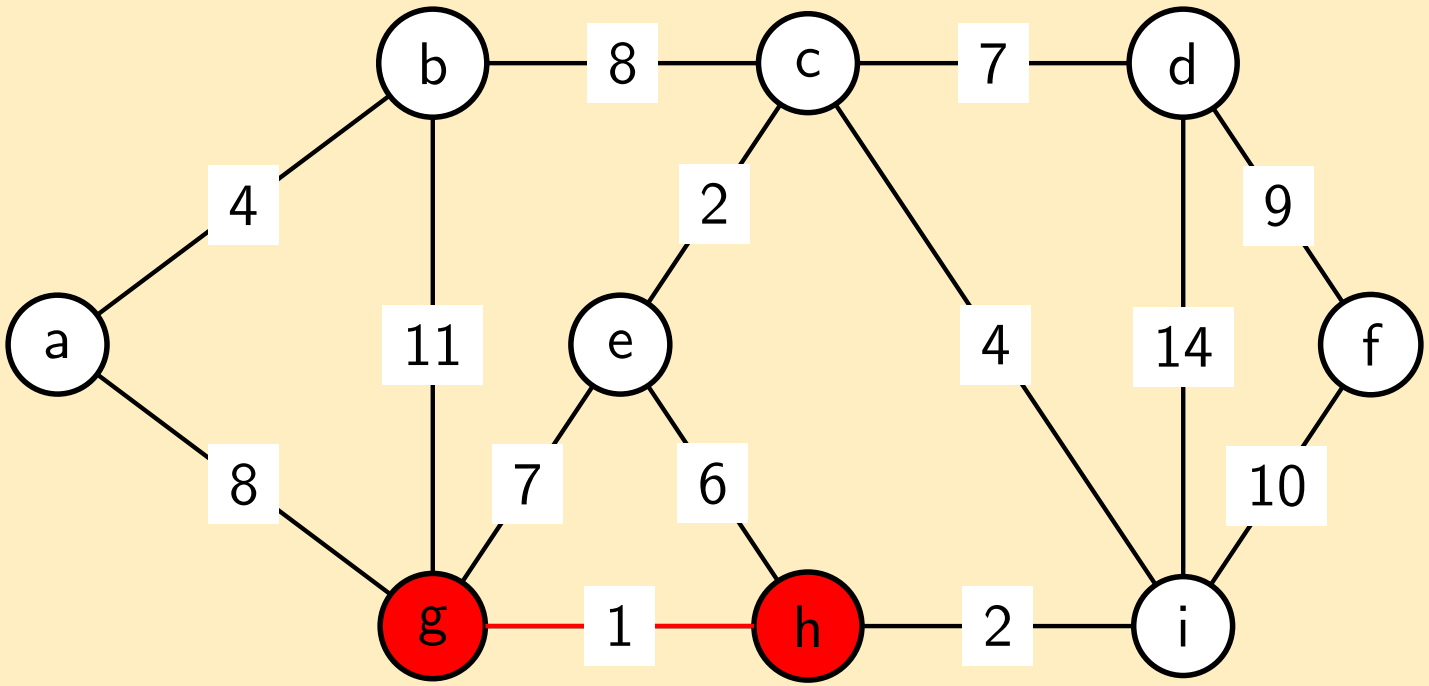
Kruskal

Exemple



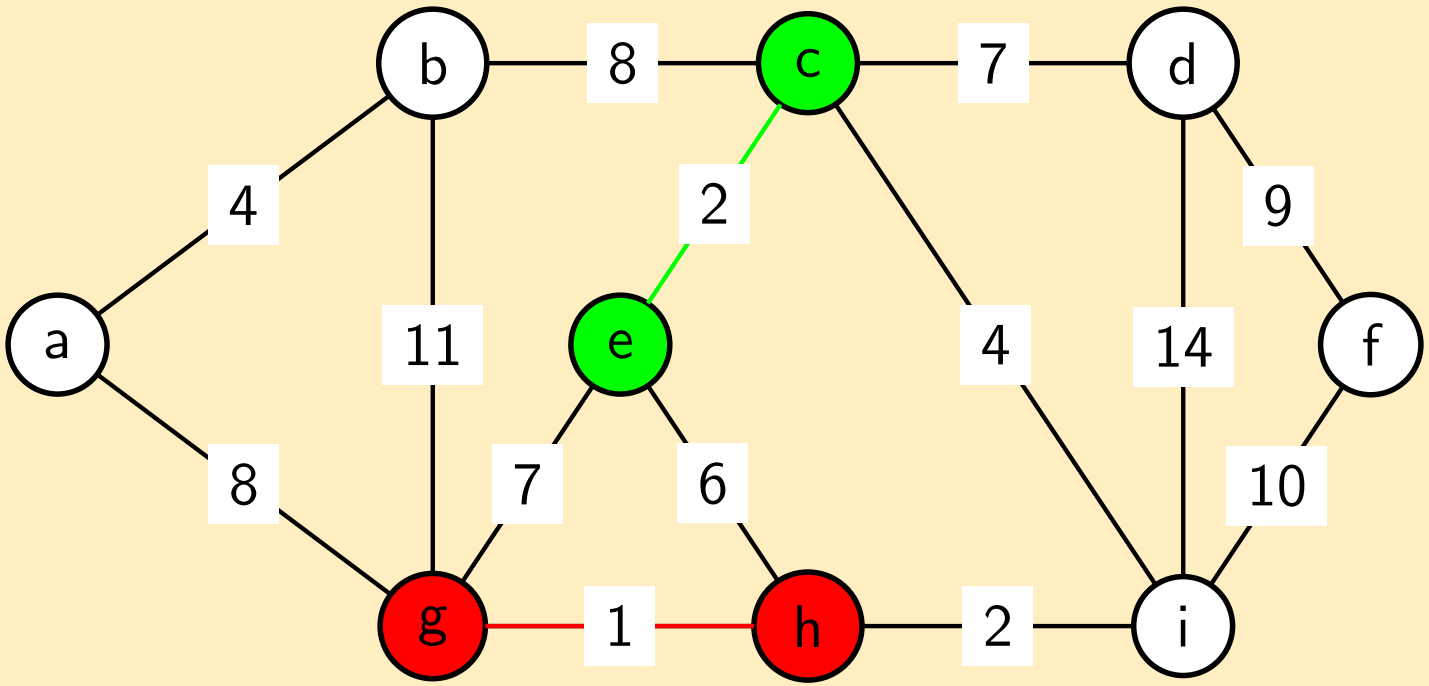
Kruskal

Exemple



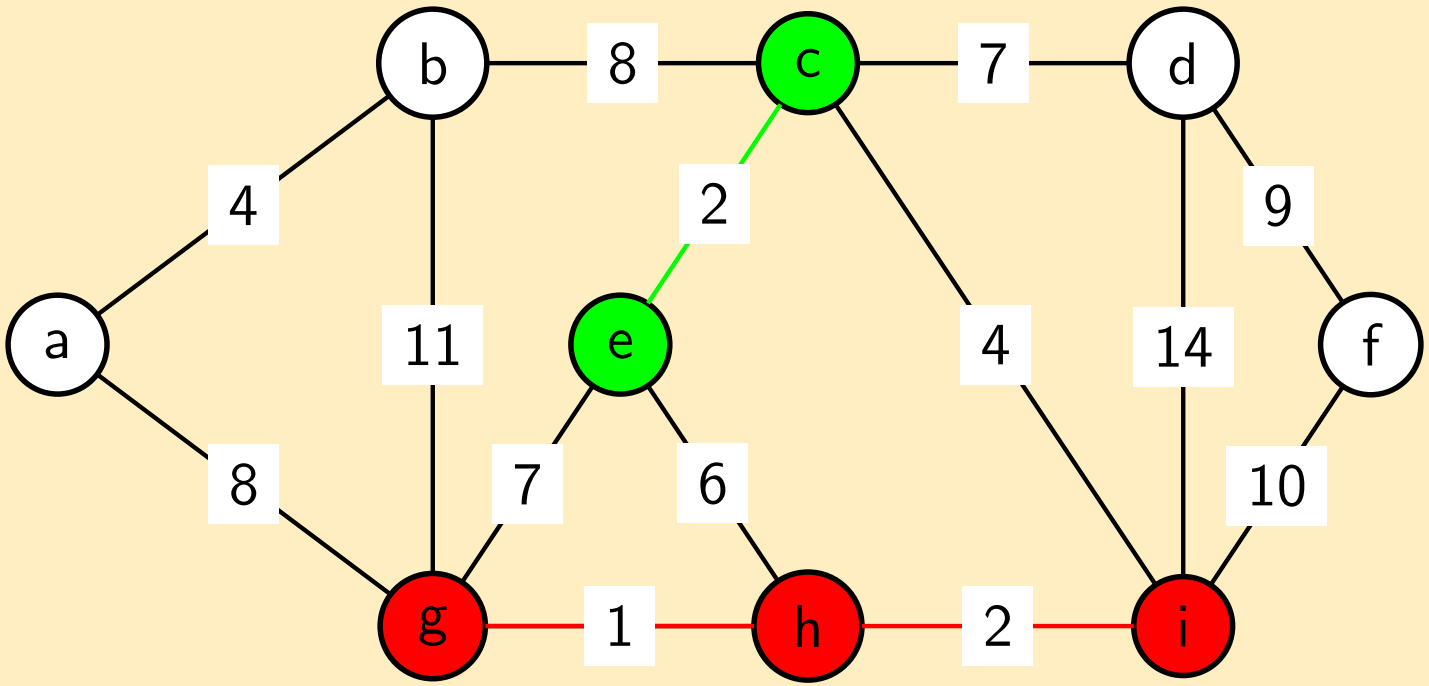
Kruskal

Exemple



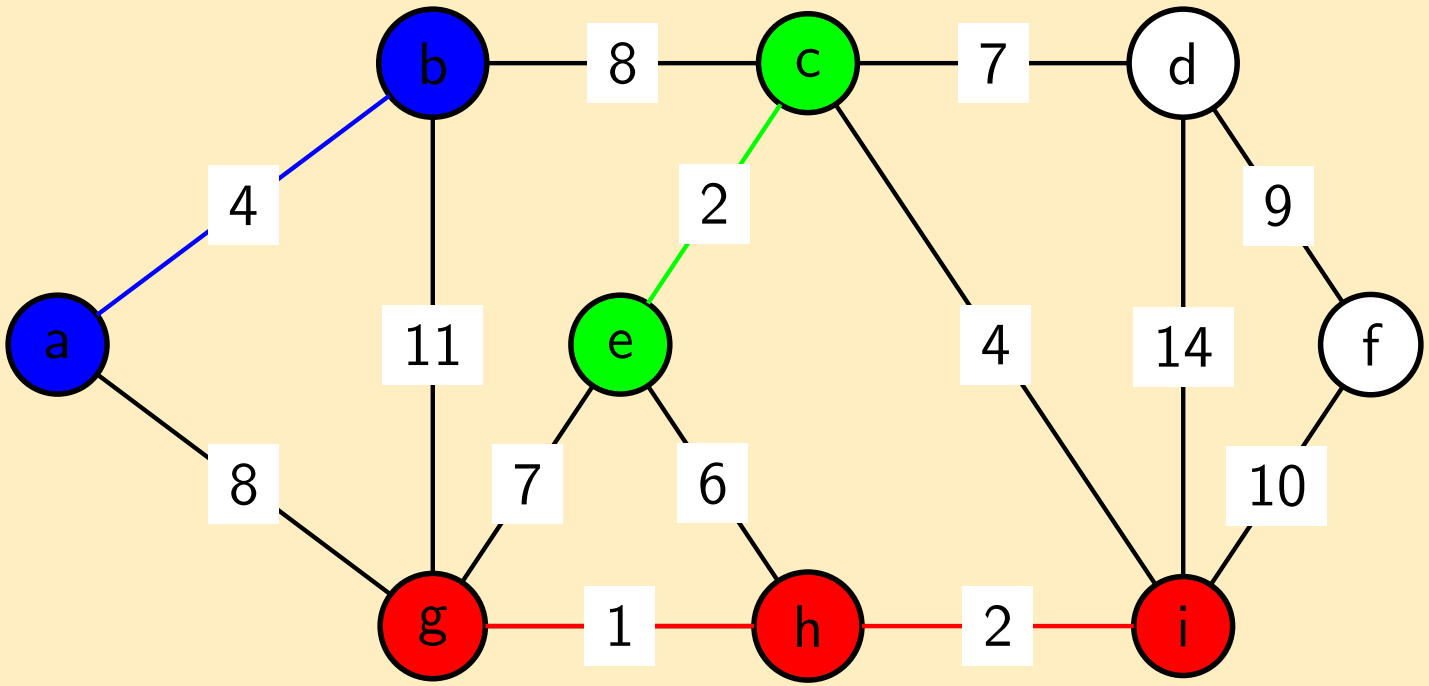
Kruskal

Exemple



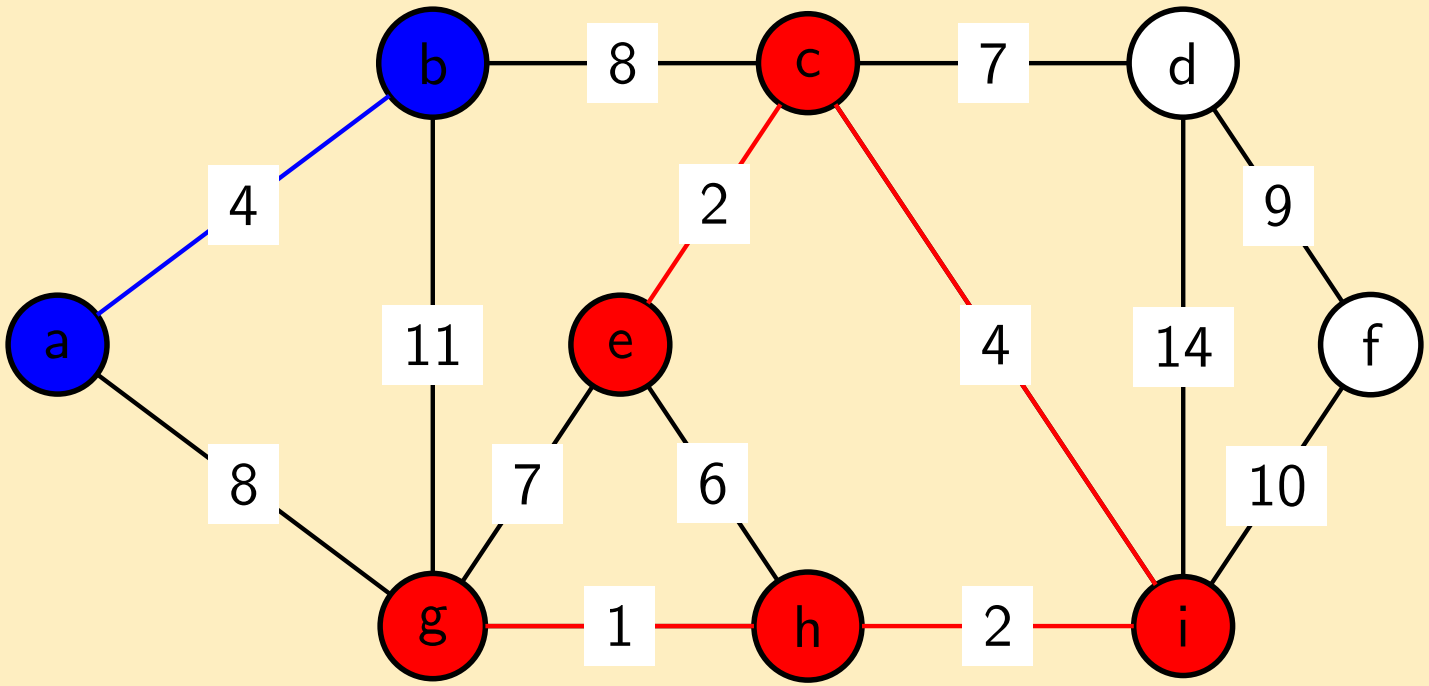
Kruskal

Exemple



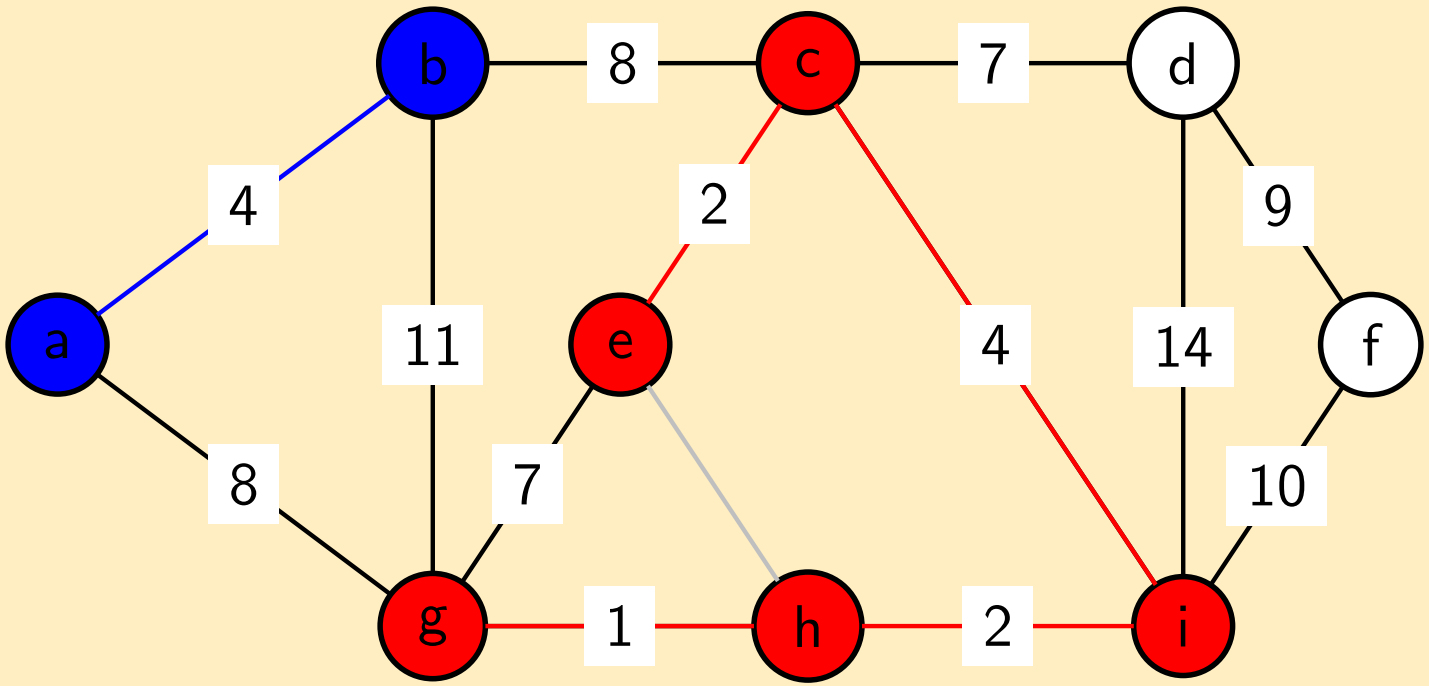
Kruskal

Exemple



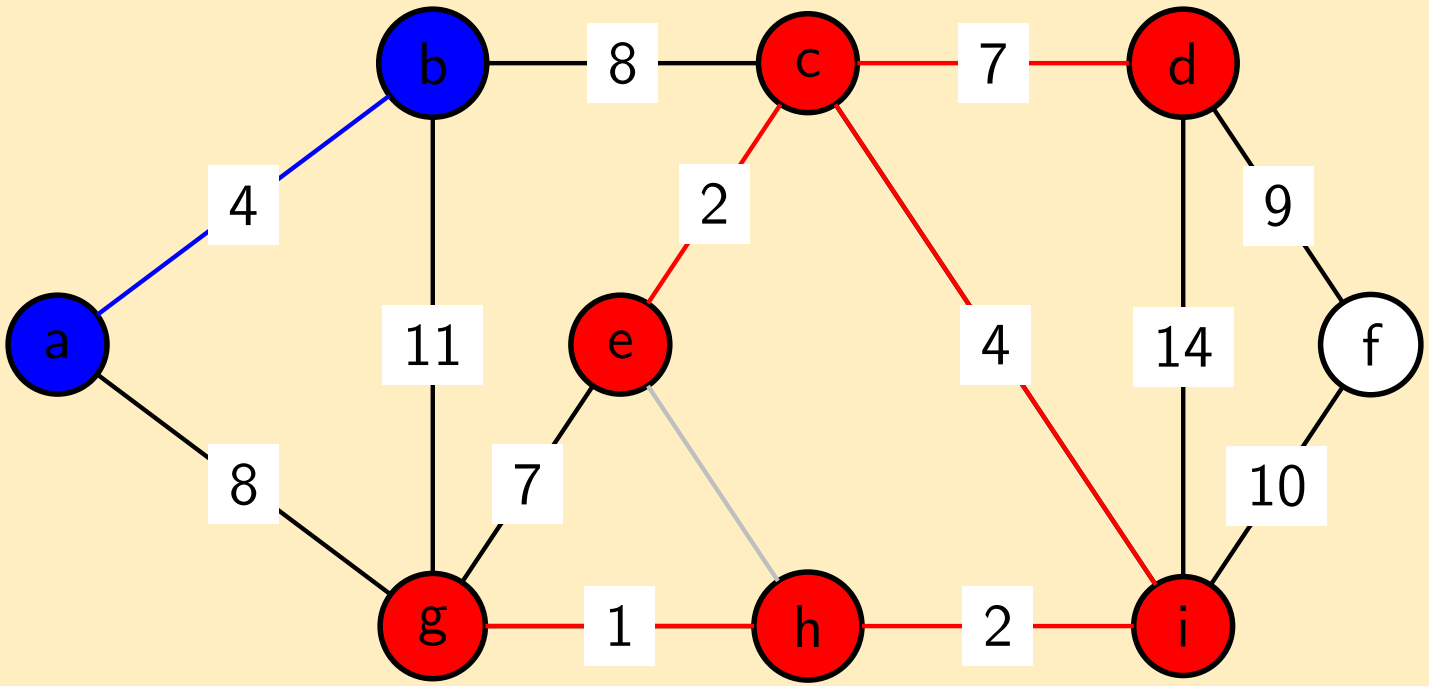
Kruskal

Exemple



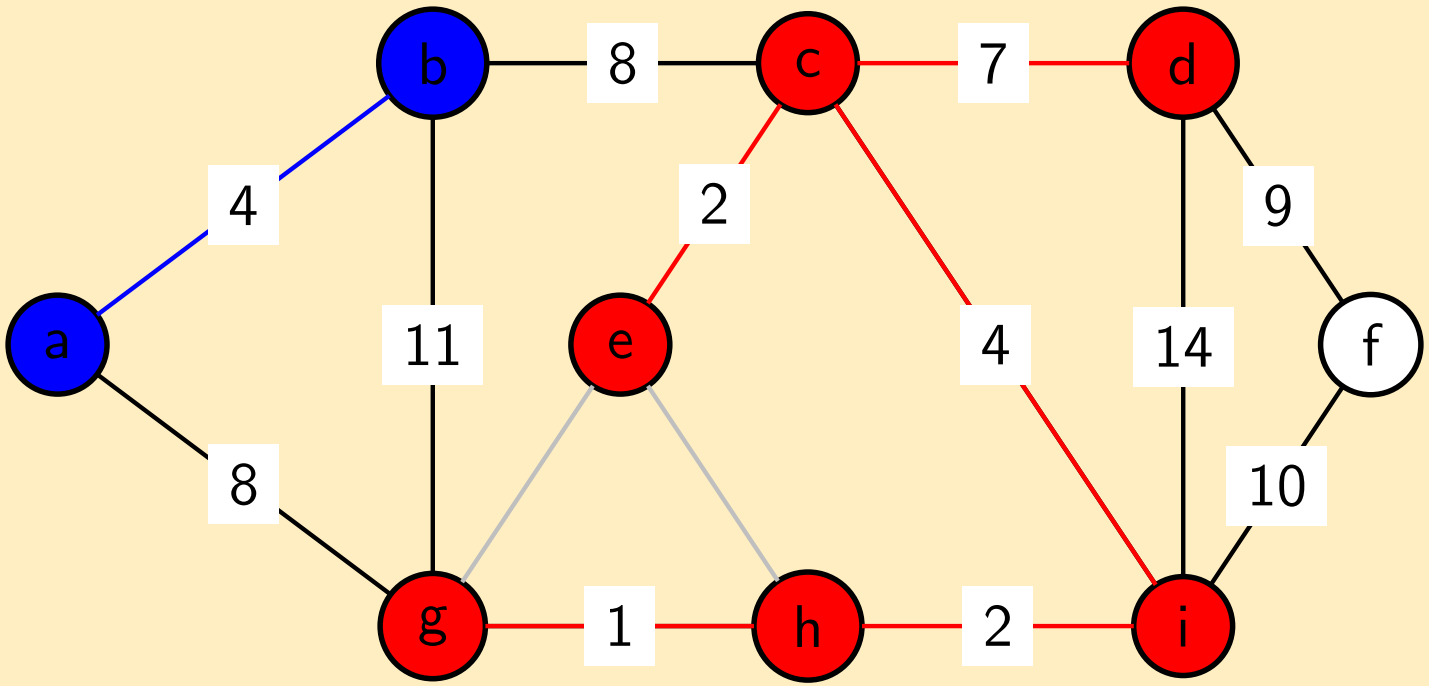
Kruskal

Exemple



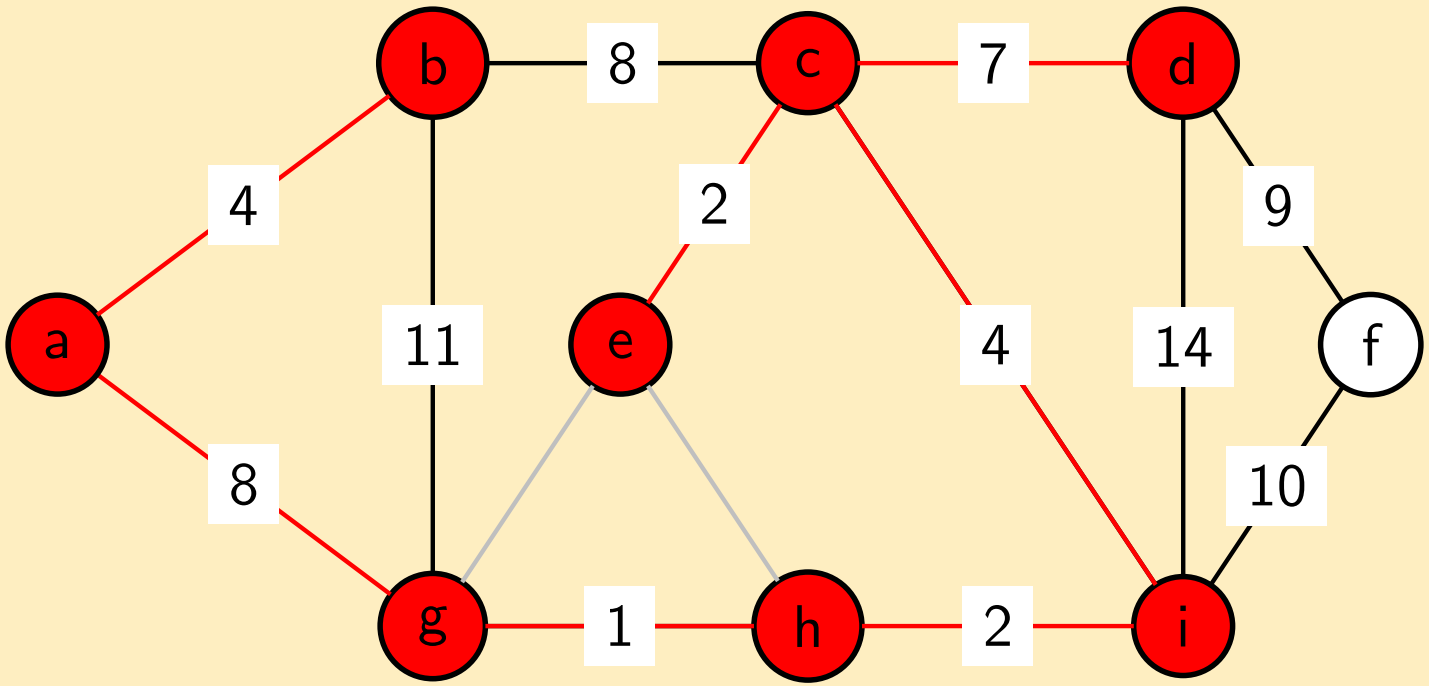
Kruskal

Exemple



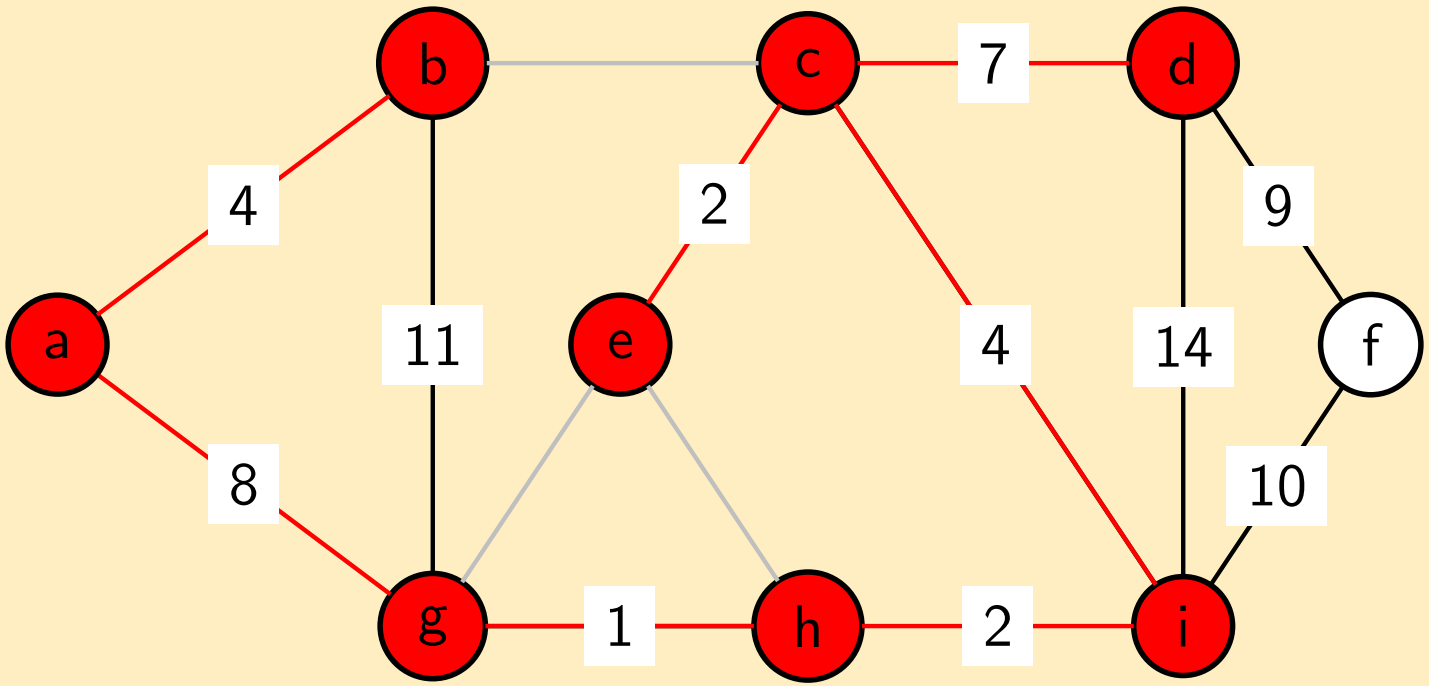
Kruskal

Exemple



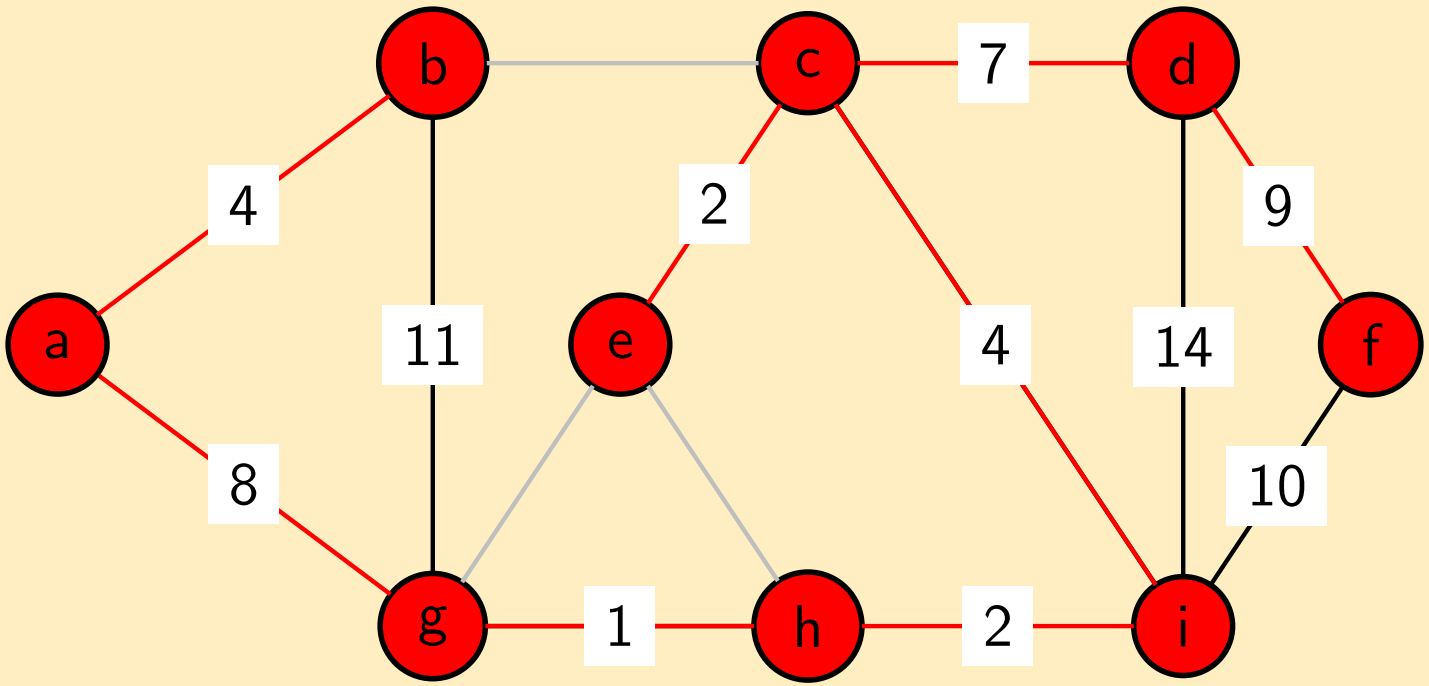
Kruskal

Exemple



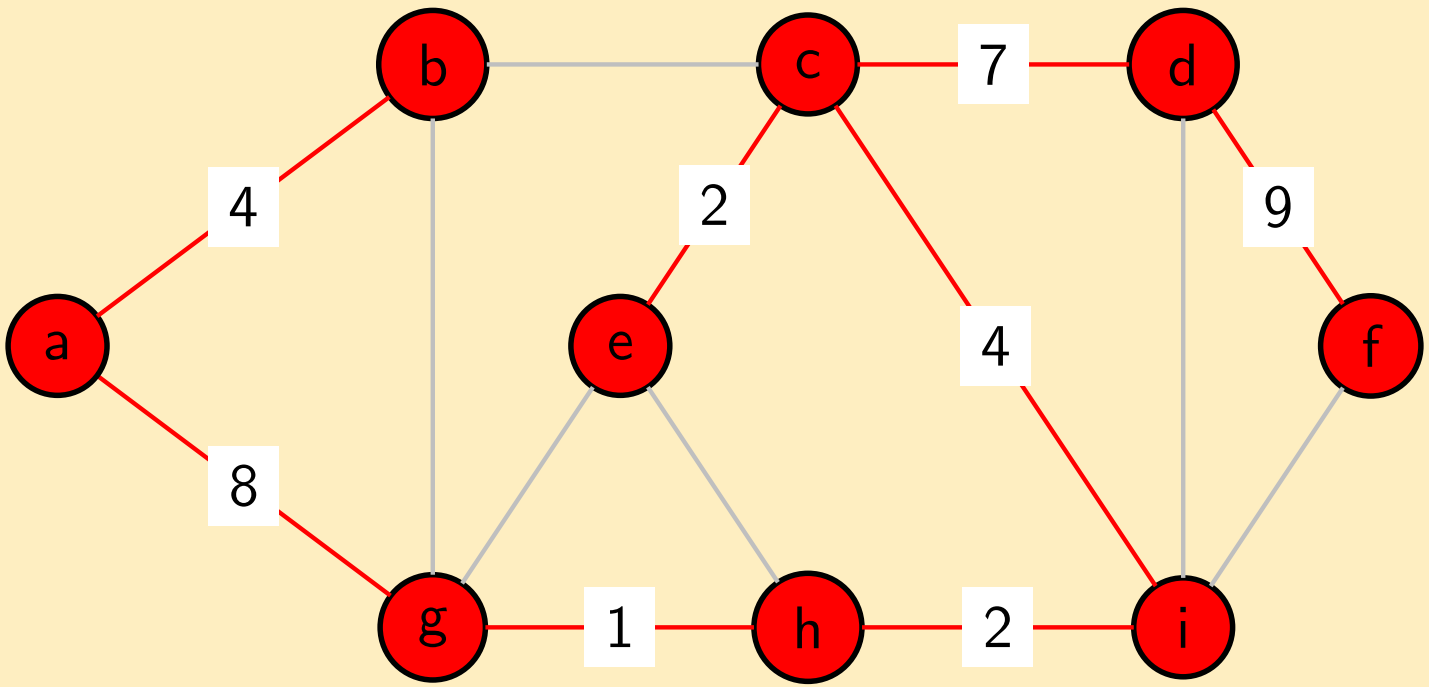
Kruskal

Exemple



Kruskal

Exemple



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