Arbres couvrants de poids minimum

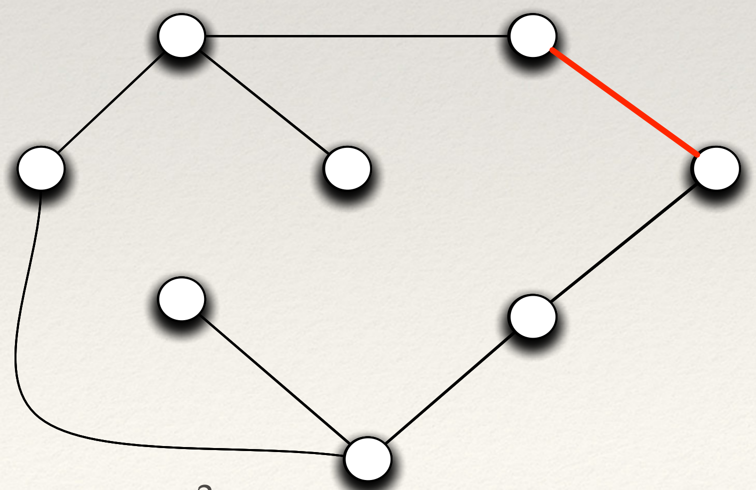
Lélia Blin
3eme année de Licence

Arbre

- Un arbre est une famille particulière de graphe.

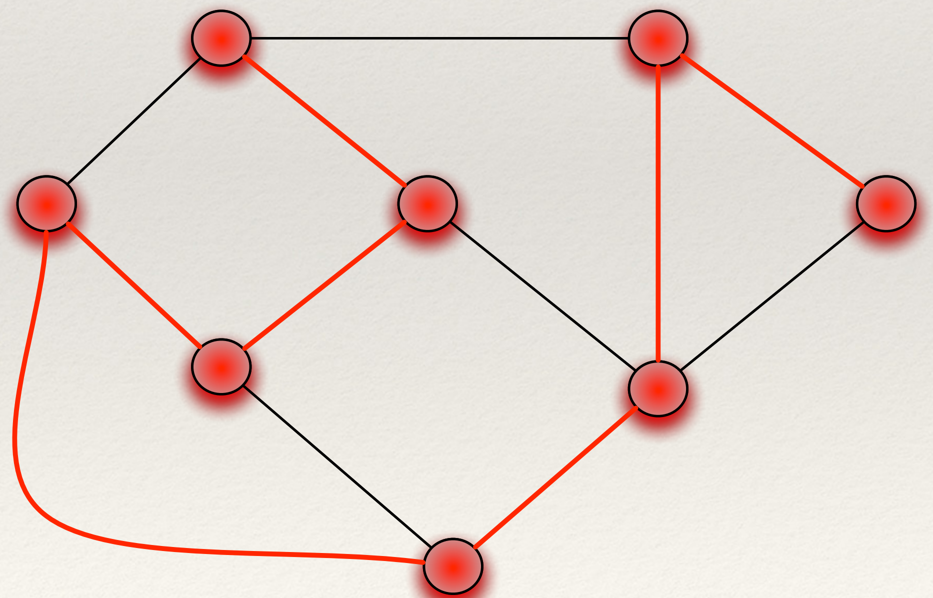
Définitions d'un arbre

- Pour un arbre T a n sommets il y a équivalence entre les définitions suivantes :
  - T est un arbre
  - T est un graphe connexe à n-1 arêtes
  - T est un graphe acyclique à n-1 arêtes
  - T est un graphe connexe et la suppression de toute arête le déconnecte
  - T est un graphe acyclique et l'ajout de toute arête le rend cyclique.



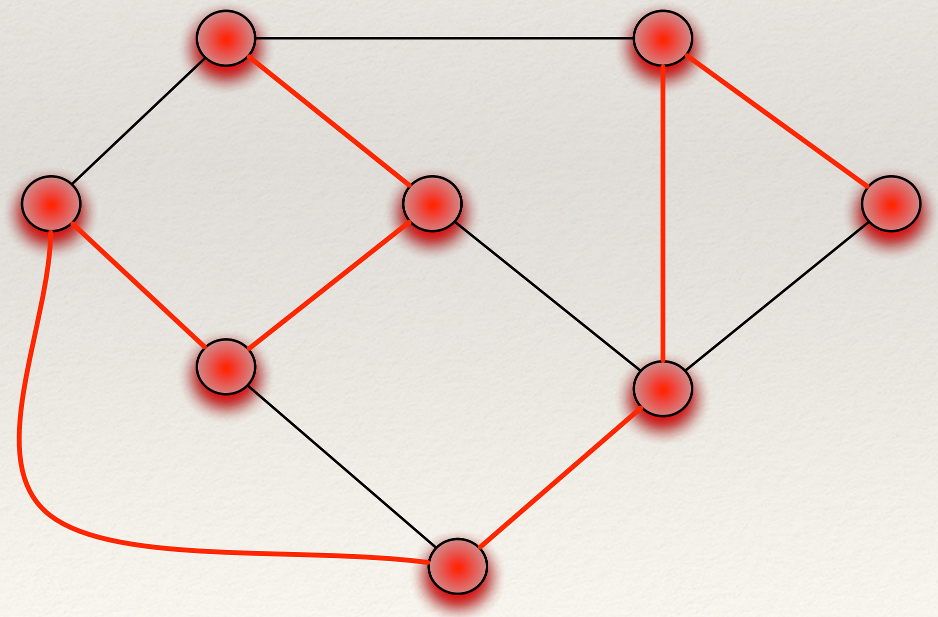
Graphe partiel

- Un graphe partiel G'(V,E') d'un graphe G(V,E) est:
  - Un graphe qui a les mêmes sommets que G.
  - Un graphe dont l'ensemble des arêtes E' est inclus dans E.



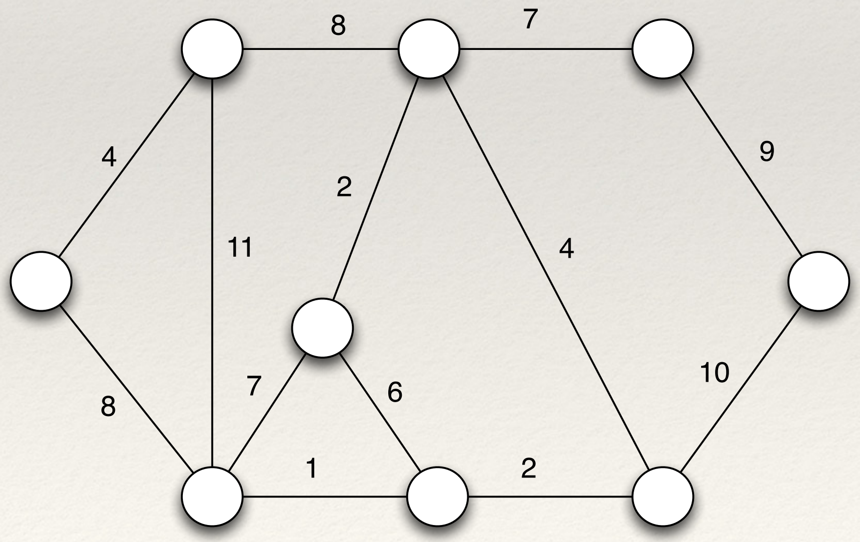
Arbres couvrants

- Un arbre couvrant T d’un graphe G(V,E) est:
  - Un graphe partiel, sans cycle.



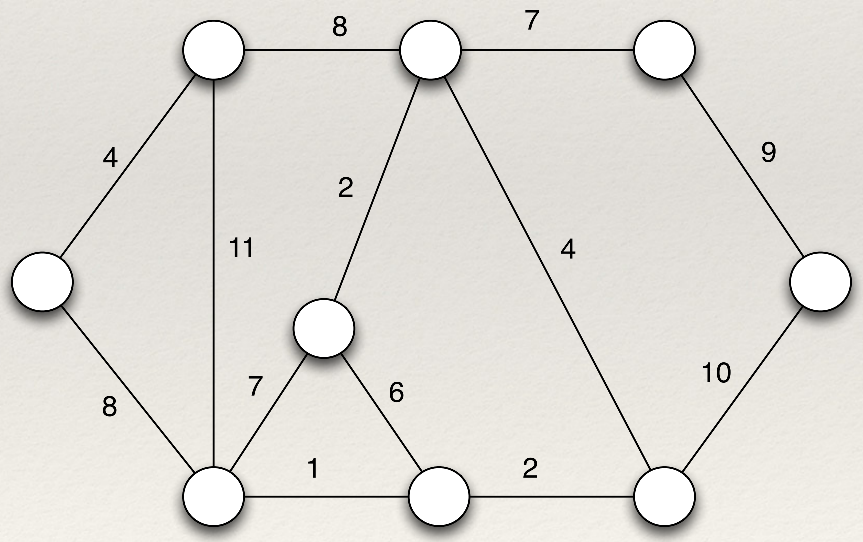
Graphe pondéré

- Un graphe pondéré G(V,E,ω) est un graphe où un entier positif est affecté à chaque arête.
- On appelle cet entier poids de l'arête.



Poids d'un graphe

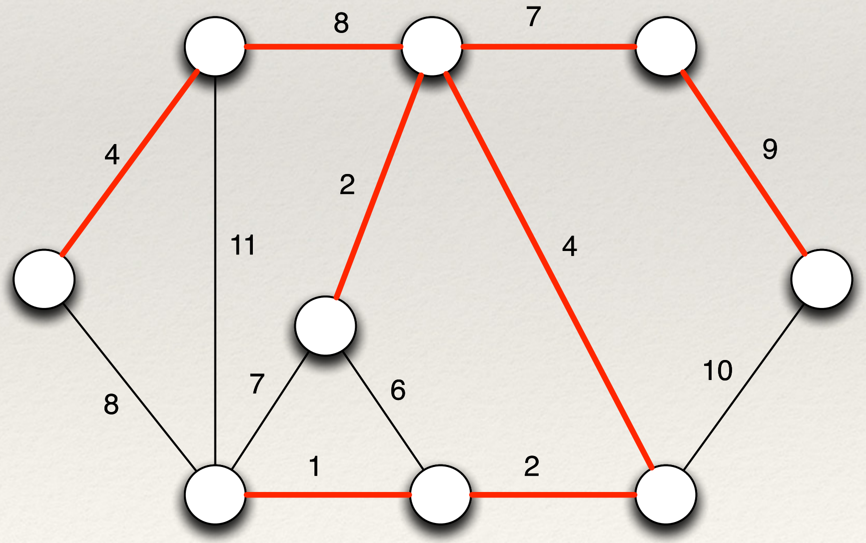
- Le poids (où coût) d'un graphe est la somme des poids des arêtes du graphe.
- On le note $\omega(G)$



$\omega(G)=103$

Arbre couvrant de poids minimum

- Soit un graphe G=(V,E,ω) un graphe non orienté pondéré.
- On appelle arbre couvrant de poids minimum (ou maximum) de G
  - noté ACPM ou MST (minimum Spanning Tree)
  - Tout arbre couvrant dont la somme des poids des arêtes le constituant est minimal (maximal).



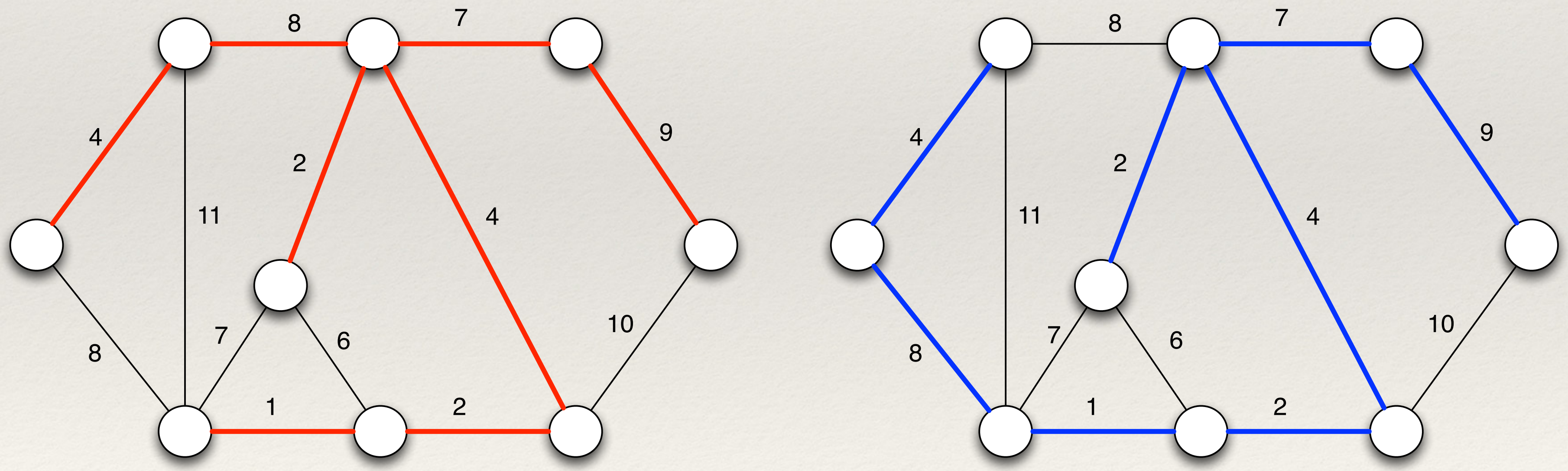
ω(G′)=37

Proposition

- Un graphe admet un arbre couvrant si, et seulement si, il est connexe.

Remarque 1

- L'arbre couvrant de poids minimal n'est pas forcément unique.



Remarque 2

- Un arbre couvrant de poids minimum est unique si et seulement si les poids de ces arêtes sont deux à deux distincts.

Application

- Soit le graphe G qui représente le réseau des rues d'une ville que l'on cherche à cabler.
- Pour cabler de façon optimale cette ville il faut trouver un graphe connexe tel que:
  - Chaque sommet doit avoir accès au réseau.
  - Le coût du câblage doit être minimum.

Application

- En effet le coût d'installation d'un cable peut dépendre de la rue choisie
    - Présence d'obstacles:
        - Tramway
        - Monuments historiques
- Ce coût est modélisé par la pondération ω.
- Il est facile de voir que
    - La pondération ω positive
    - Implique qu'une solution optimale d'un tel problème soit nécessairement un arbre.

Arbre couvrant de poids minimum

- Problématique: On veut
  - Connecter des sommets
  - De la façon la plus économique possible.
- Ce problème admet une solution, car:
  - L'ensemble des arbres couvrants est fini.
  - Il admet au moins un élément de coût minimum.

Arbre couvrant de poids minimum

- Tout le problème est de trouver l'ACPM
  - Sans énumérer tous les arbres couvrants.
- Nombre d'arbres couvrants:
  - Il peut y en avoir beaucoup
  - ex : n^{n-2} pour un graphe complet
    - Théorème de Cayley publié en 1889.

Règles de Tarjan

- Pour ce faire on va construire un schéma d'algorithme basé sur une bi-coloration en rouge ou bleu des arêtes de G.

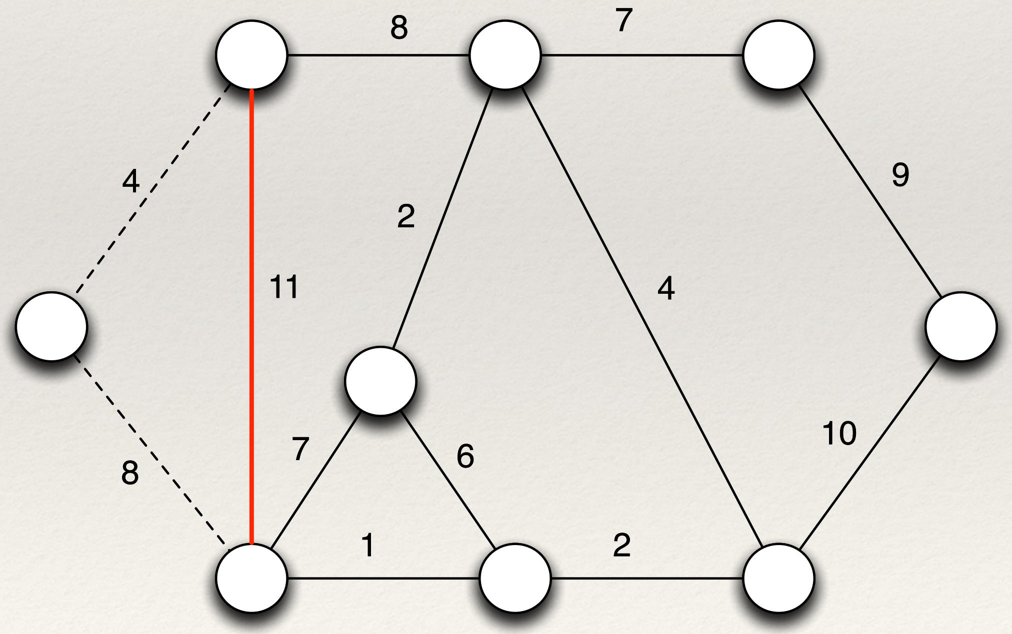
Règles de Tarjan

- Cette présentation très élégante est due à:
  - Tarjan* (Prix Turing 1986)
- Au départ les arêtes ne sont pas colorées
- Pour les colorier (de manière définitive) nous allons utiliser deux règles

*R.E. Tarjan. Data structures and network algorithms, volume 44. SIAM Monography, 1983.

Règle rouge de Tarjan

- Cycle disponible: un cycle μ qui ne contient pas d'arête rouge est dit disponible.
- Règle rouge:
  - Choisir un cycle disponible μ
  - Dans μ choisir une arête e non colorée de coût **maximal** ➤ Coloriée e en rouge

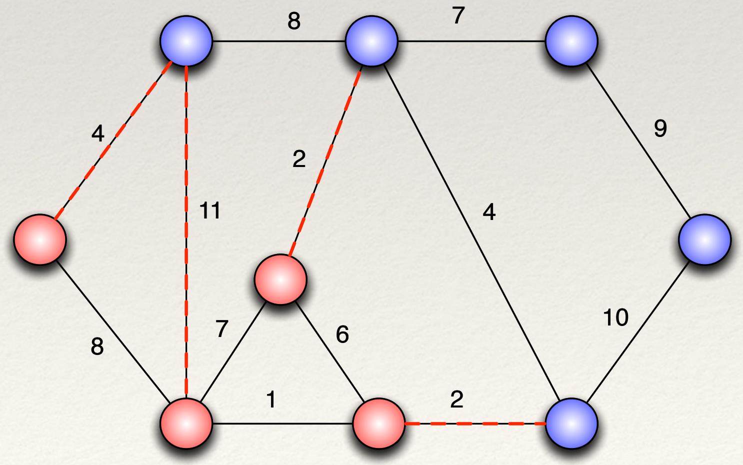


Propriété de la règle rouge

- L'arête de poids maximum d'un cycle ne fait partie d'aucun arbre couvrant de poids minimum.

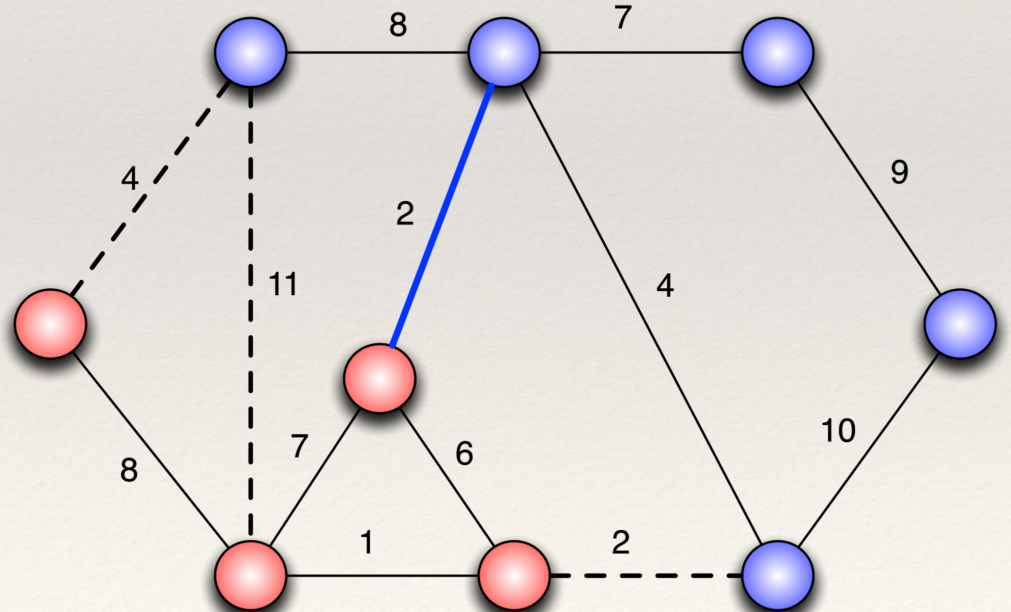
Coupe

- Une coupe (ou cocycle) est un ensemble d'arêtes dont une extrémité est dans un sous-ensemble A de sommets, l'autre extrémité étant dans le complémentaire de A



Règle bleu de Tarjan

- Coupe disponible: une coupe θ qui ne contient pas d'arête bleu est dite disponible.
- Règle bleu:
  - Choisir une coupe disponible θ.
  - Choisir dans θ une arête e non colorée de coût **minimal** ➤ Coloriée e en bleu

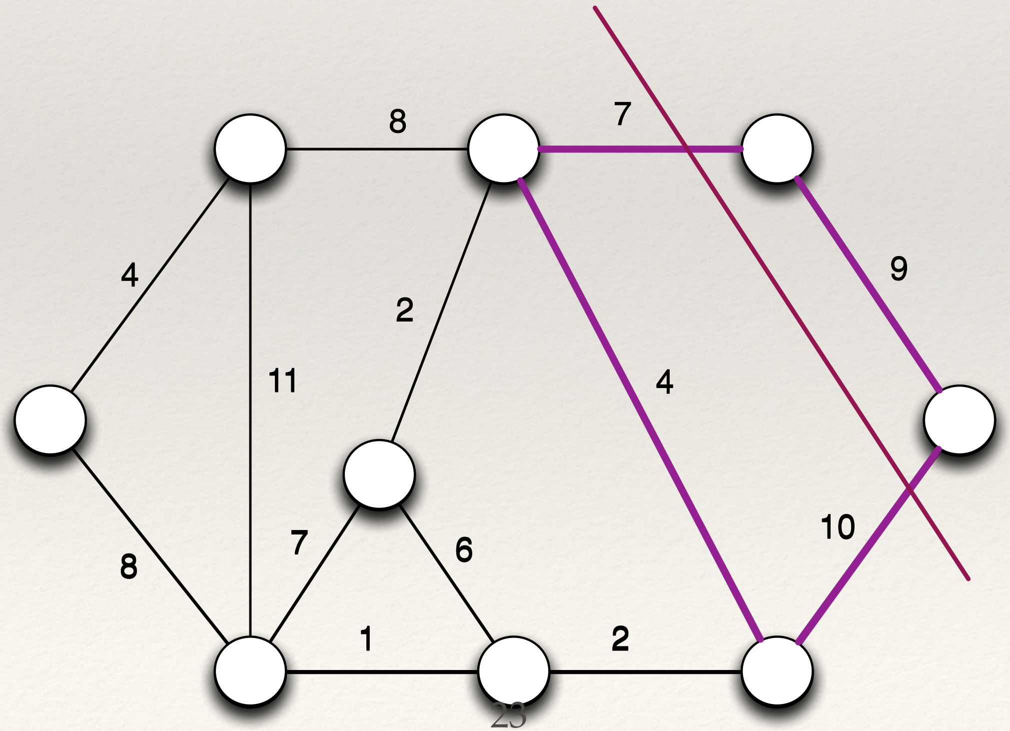


Propriété de la règle bleu

- L'arête de poids minimum d'une coupe fait partie de l'arbre couvrant de poids minimum.

Lemme cycle-coupe

- Si un cycle μ et une coupe θ ont une arête en commun, alors ils en ont au moins une deuxième.



Remarque

- Il est possible de préciser le résultat du lemme :
  - un cycle et une coupe ont en commun un nombre pair d'arêtes.

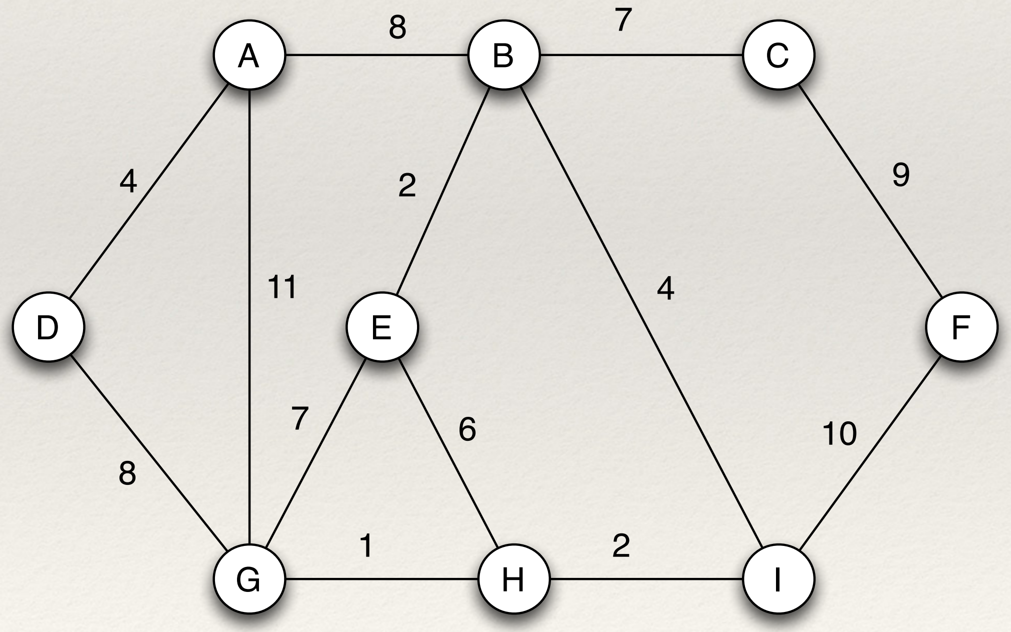
Algorithme de Tarjan

Tant qu'une règle est applicable faire

 appliquer cette règle

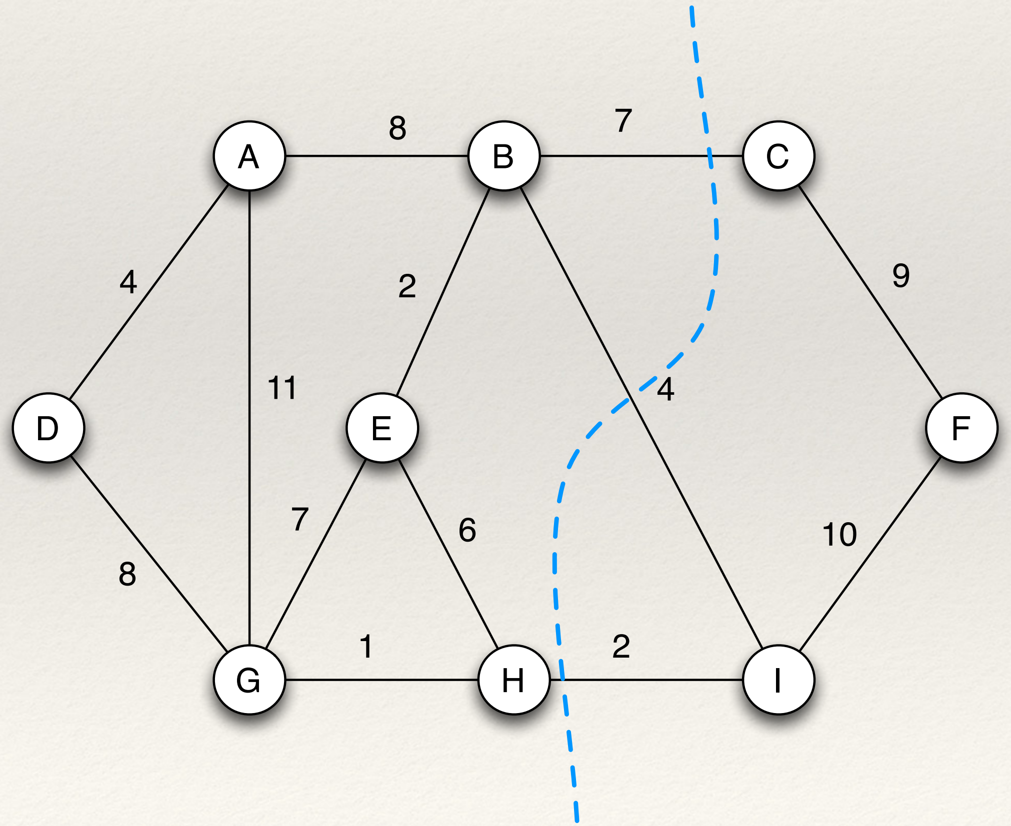
Exemple

- Cycle Maximum
- Coupe Minimum



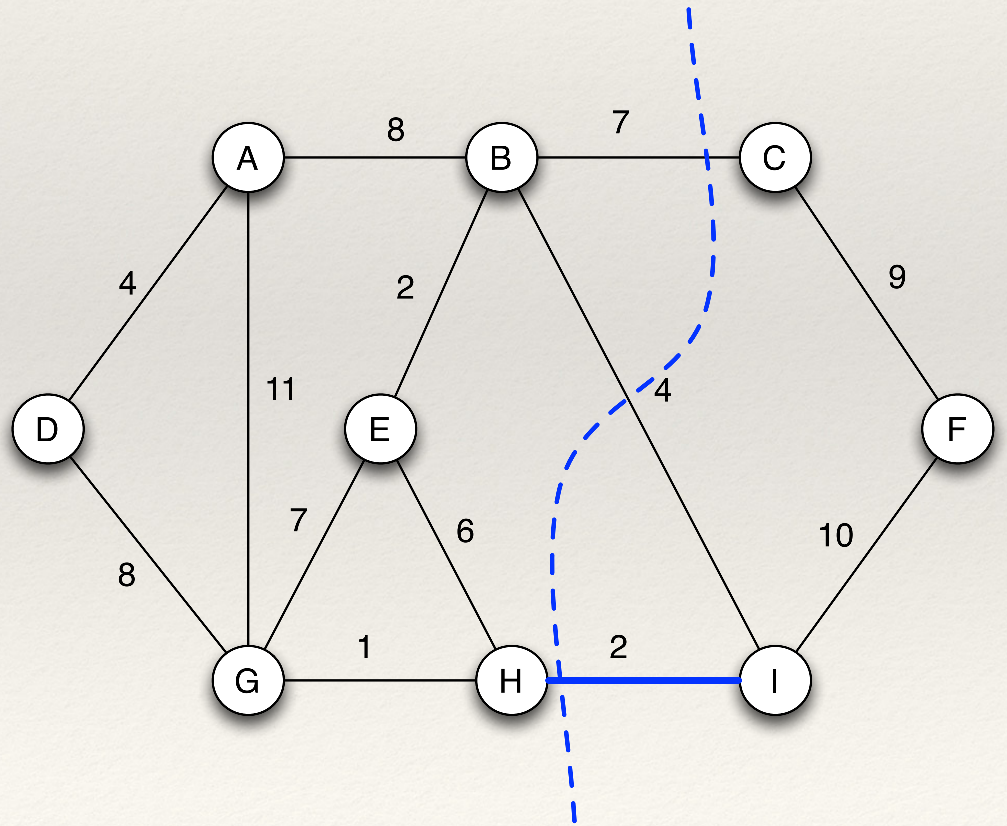
Exemple

- Cycle Maximum
- Coupe Minimum



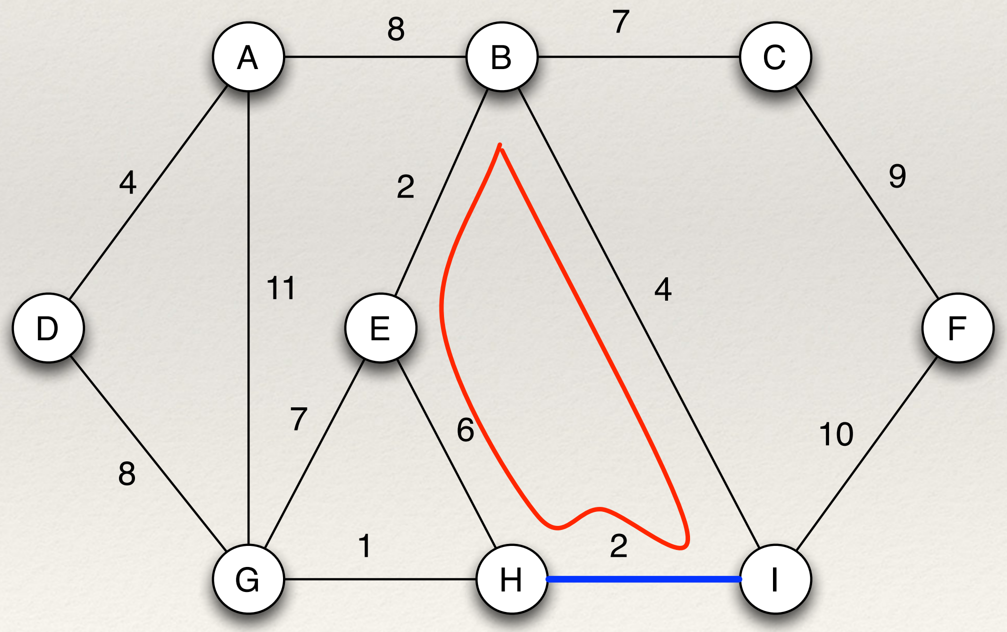
Exemple

- Cycle Maximum
- Coupe Minimum



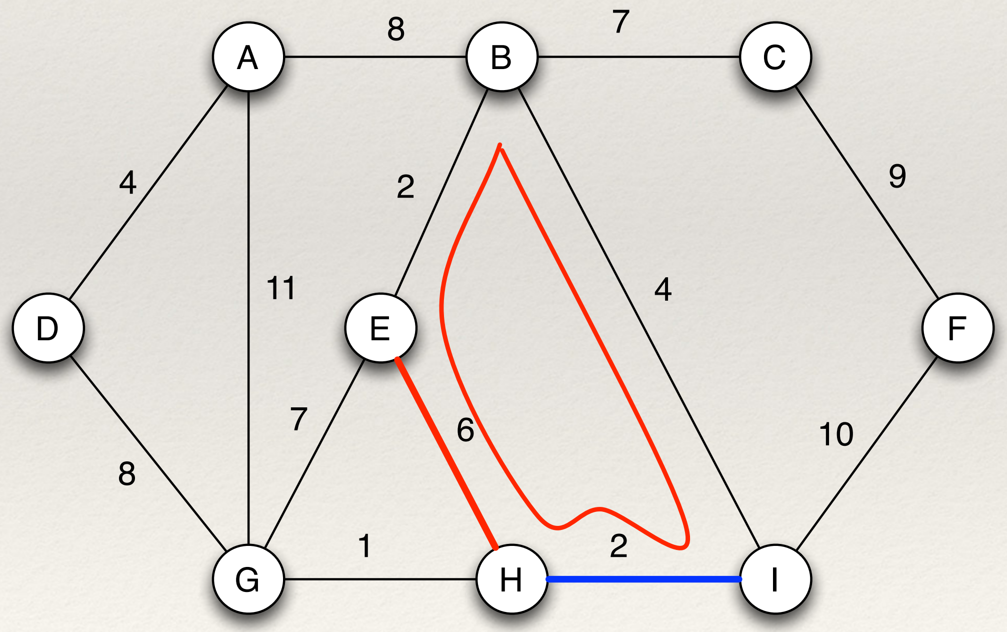
Exemple

- Cycle Maximum
- Coupe Minimum



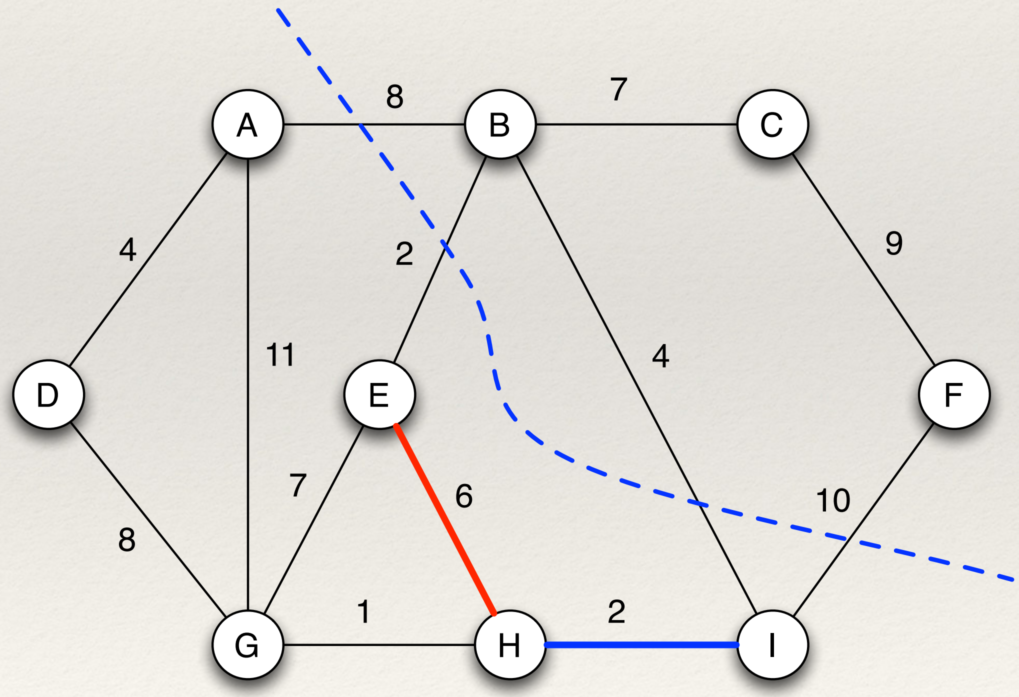
Exemple

- **Cycle Maximum**
- **Coupe Minimum**



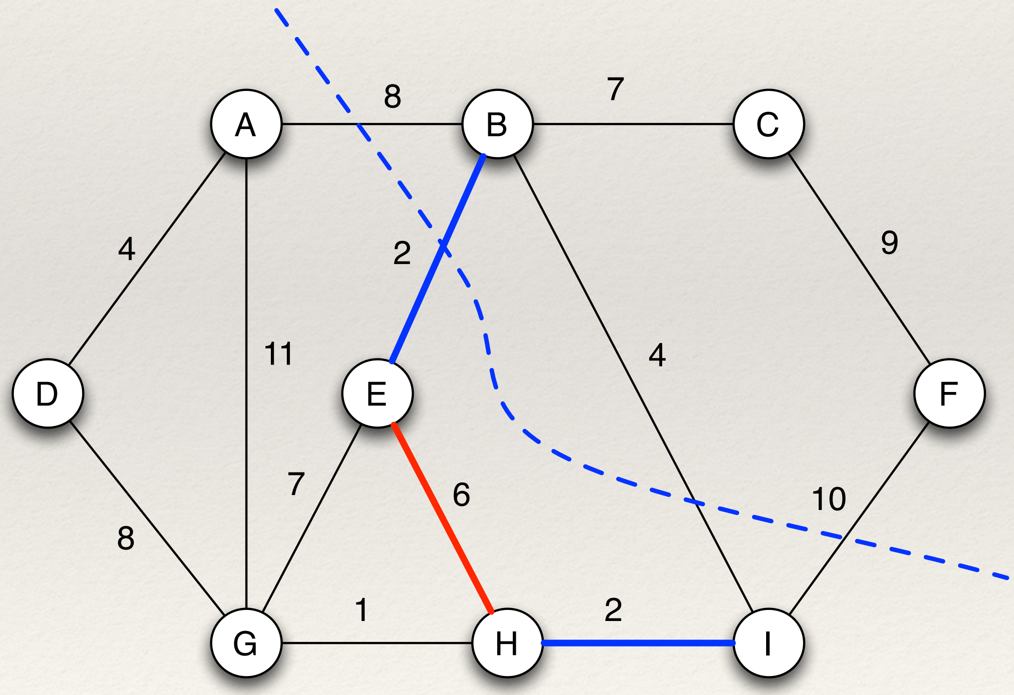
Exemple

- Cycle Maximum
- Coupe Minimum



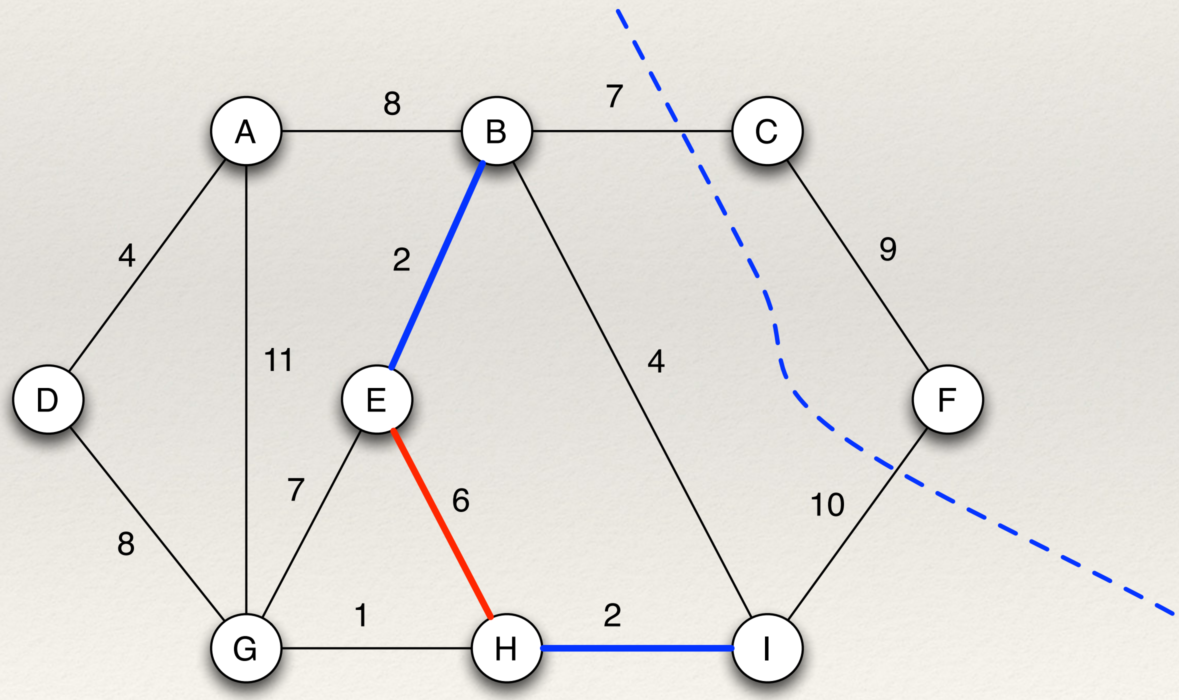
Exemple

- Cycle Maximum
- Coupe Minimum



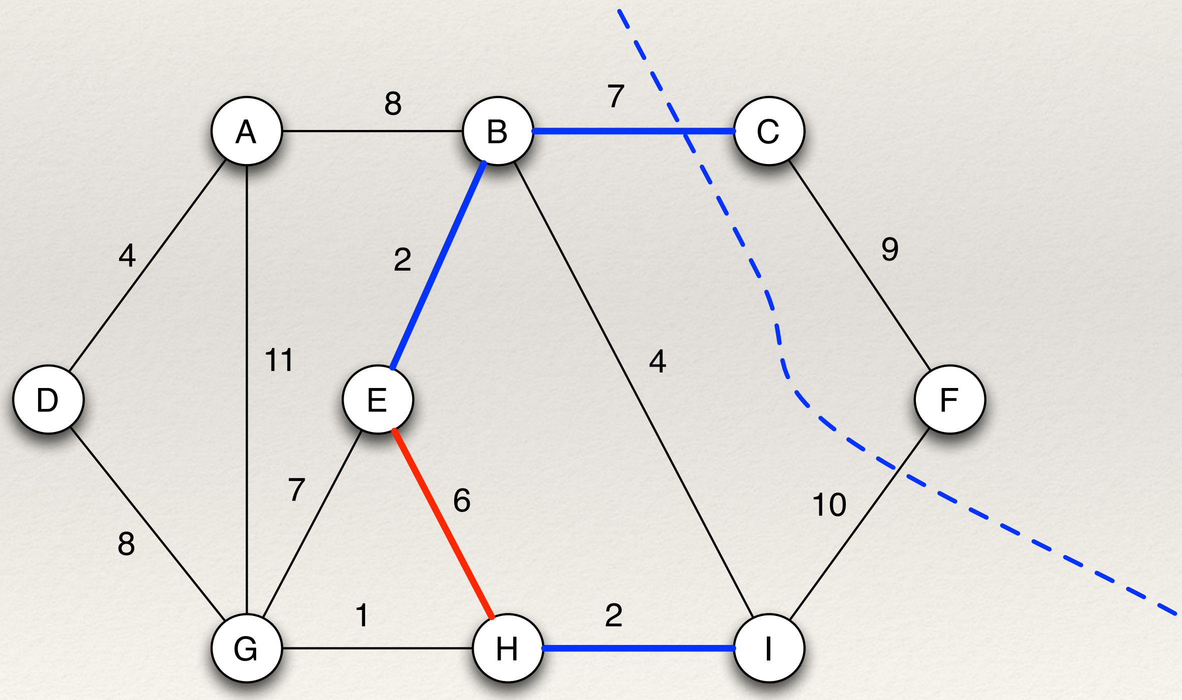
Exemple

- Cycle Maximum
- Coupe Minimum



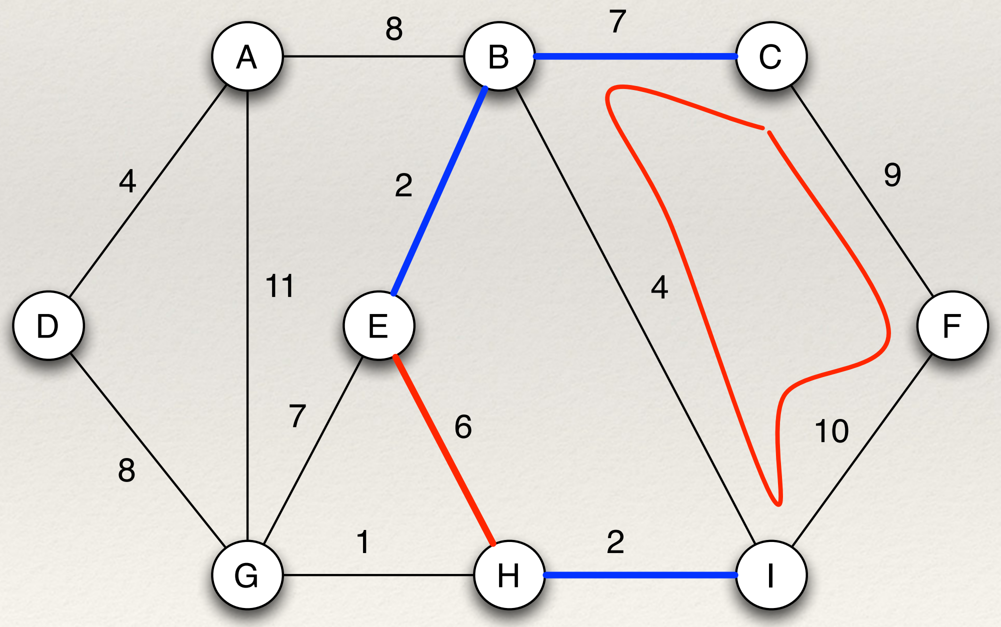
Exemple

- **Cycle Maximum**
- **Coupe Minimum**



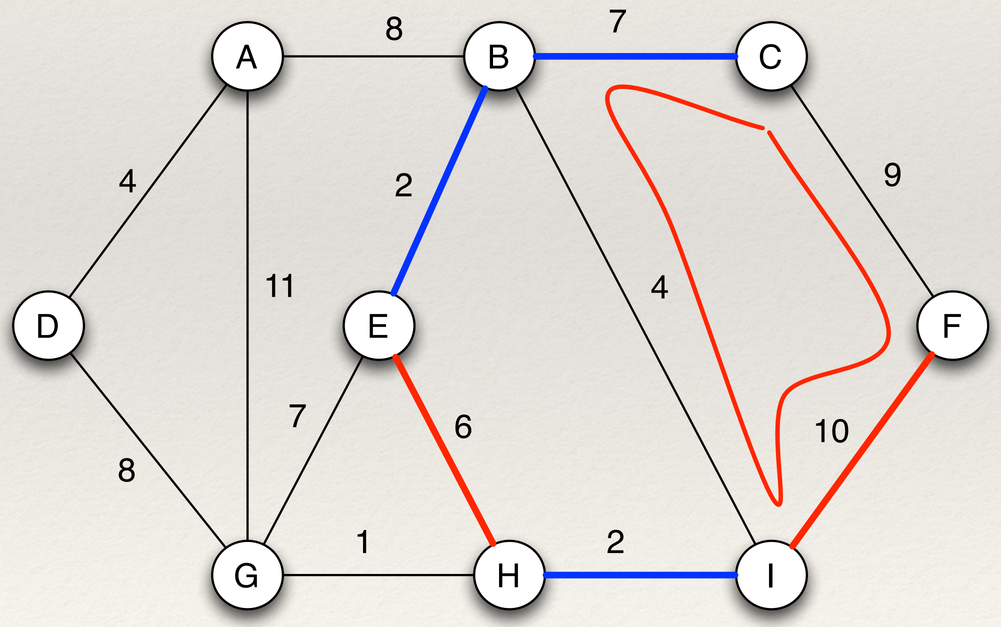
Exemple

- Cycle Maximum
- Coupe Minimum



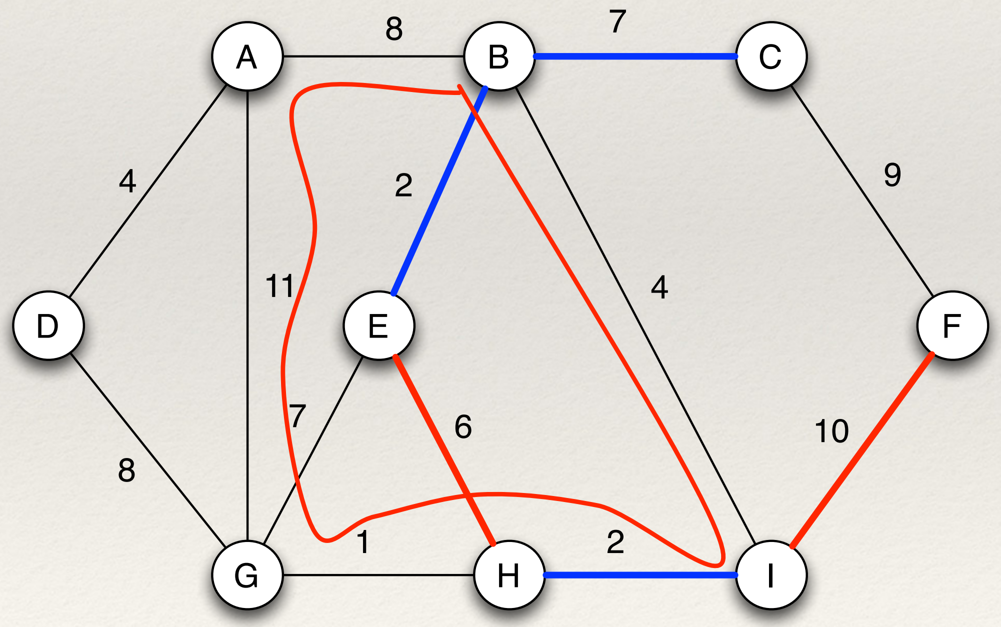
Exemple

- Cycle Maximum
- Coupe Minimum



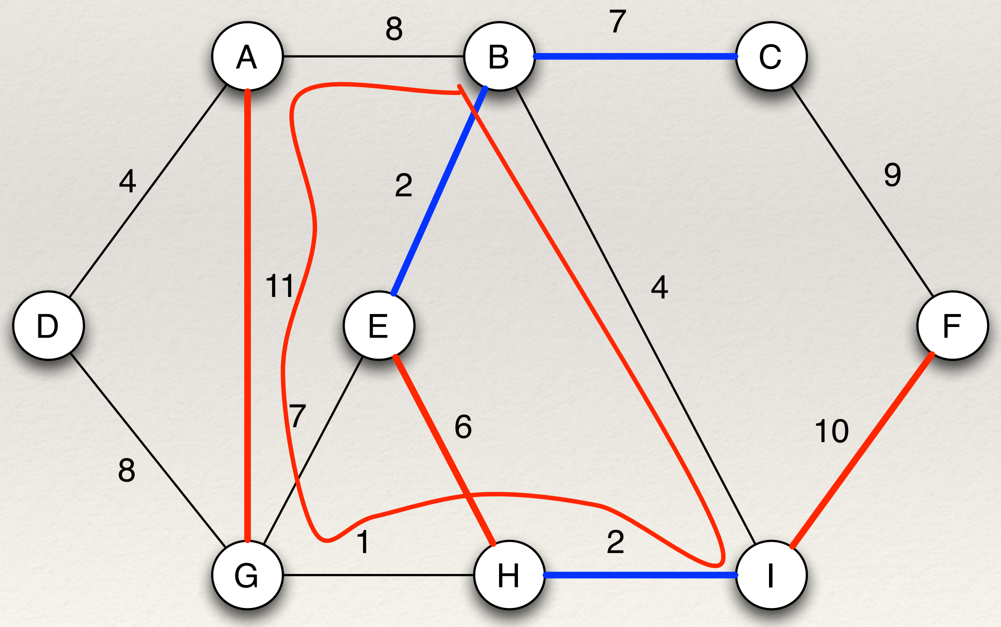
Exemple

- Cycle Maximum
- Coupe Minimum



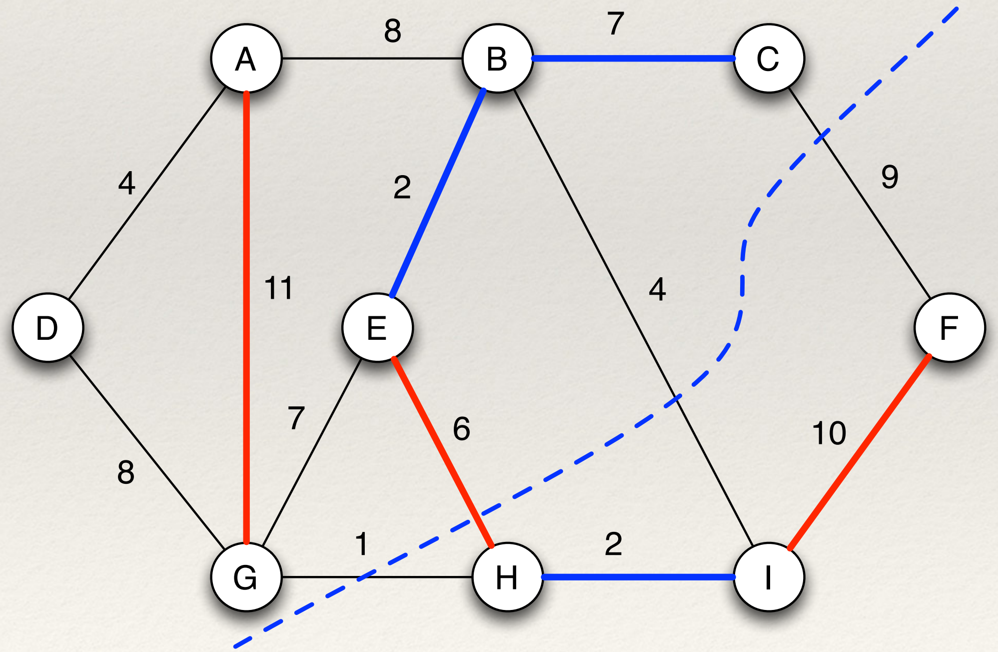
Exemple

- Cycle Maximum
- Coupe Minimum



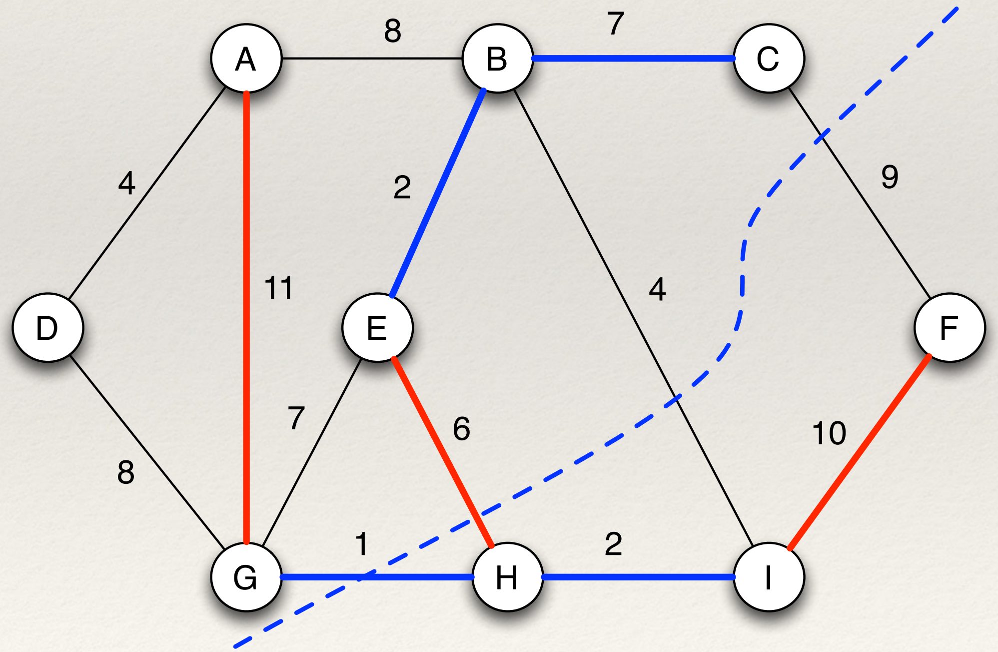
Exemple

- Cycle Maximum
- Coupe Minimum



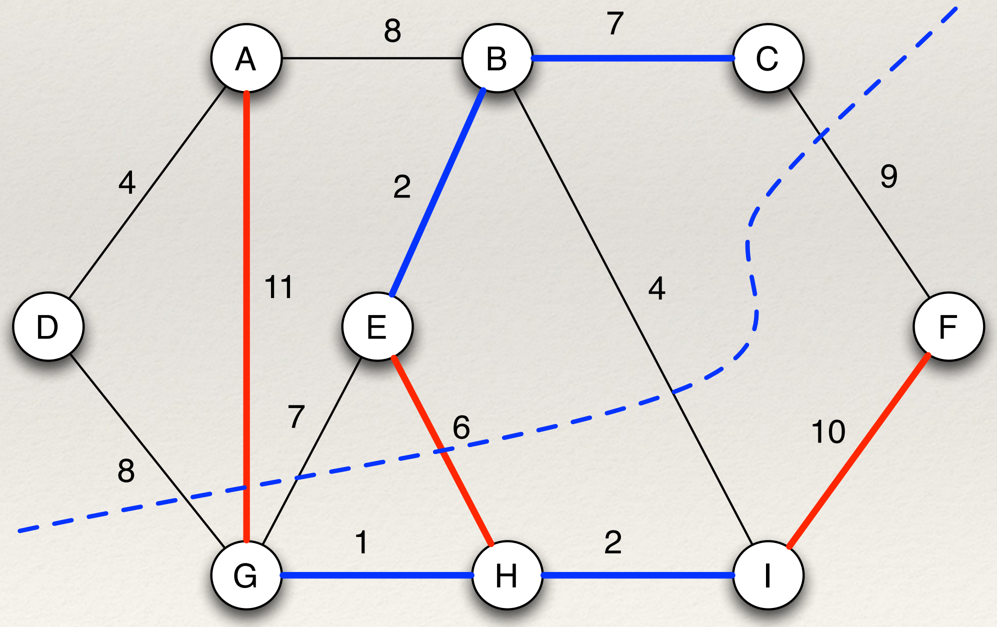
Exemple

- Cycle Maximum
- Coupe Minimum



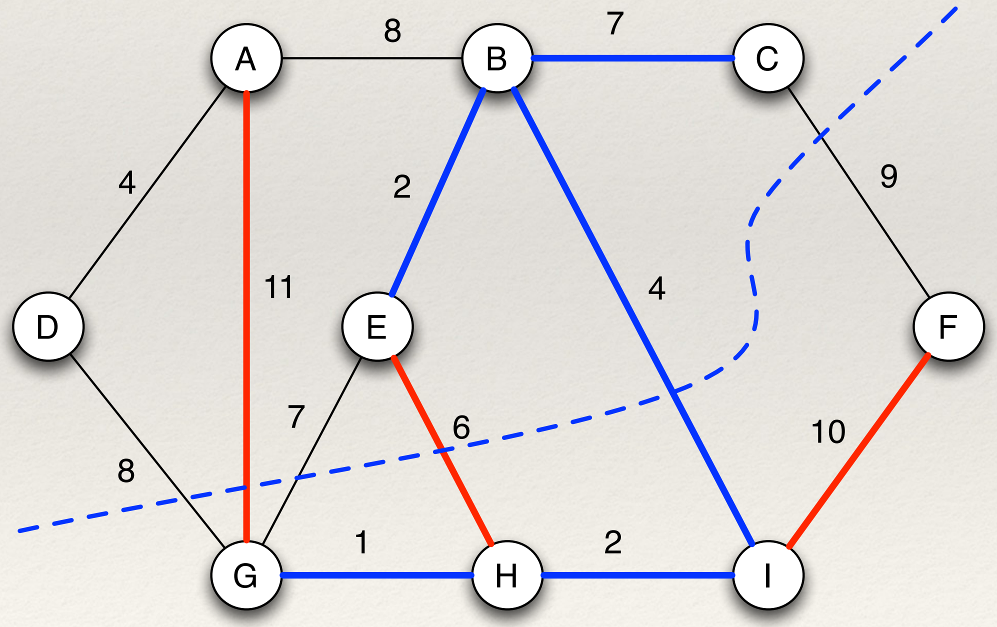
Exemple

- Cycle Maximum
- Coupe Minimum



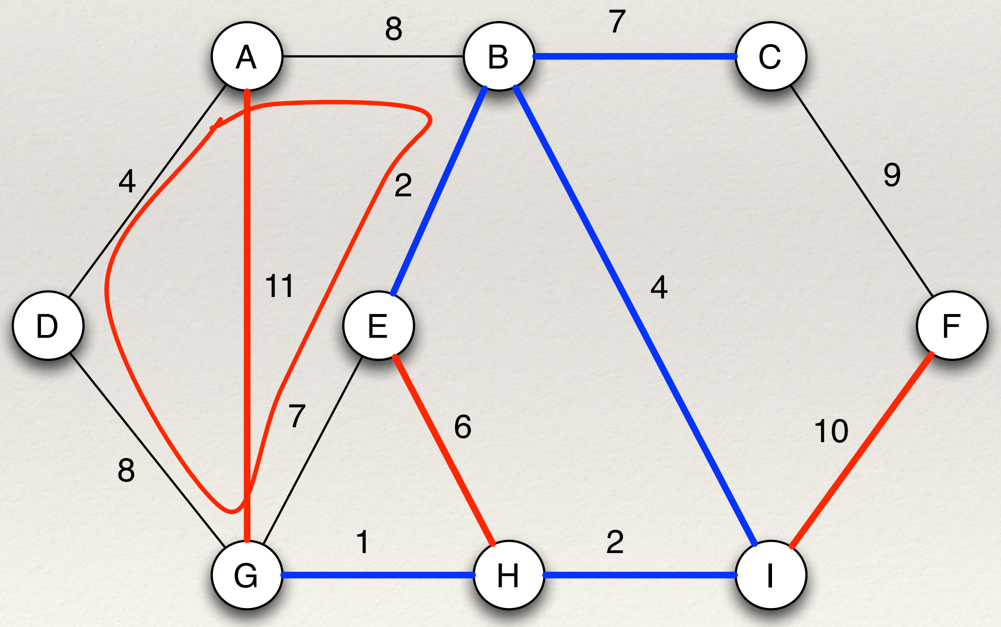
Exemple

- Cycle Maximum
- Coupe Minimum



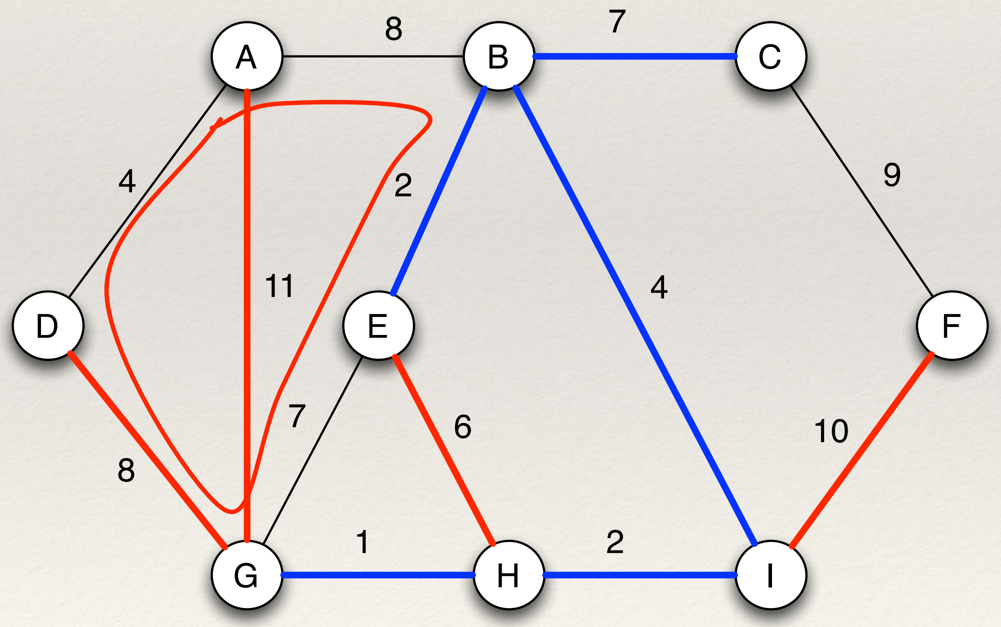
Exemple

- Cycle Maximum
- Coupe Minimum



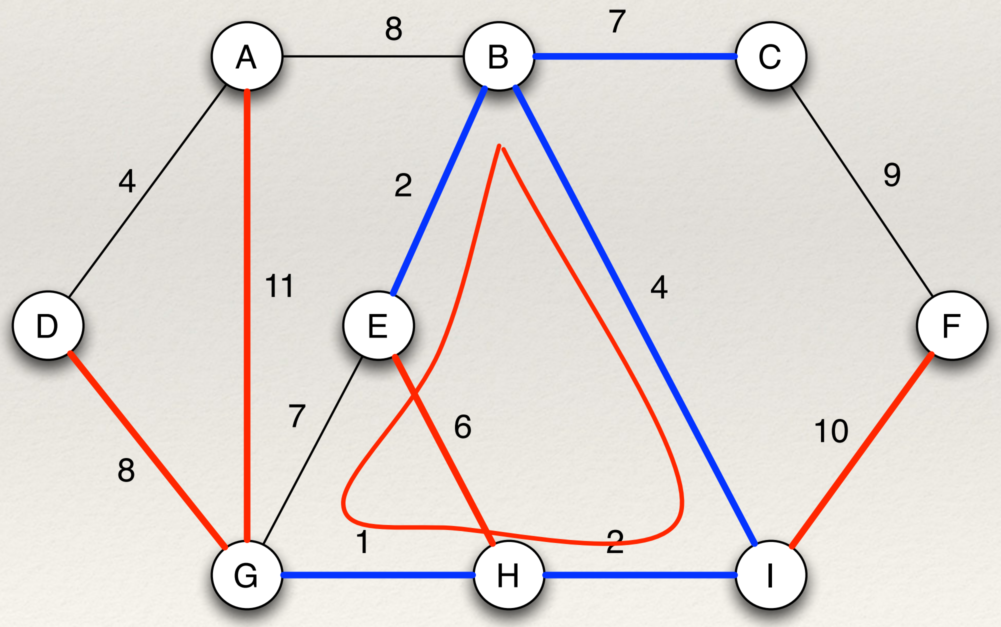
Exemple

- Cycle Maximum
- Coupe Minimum



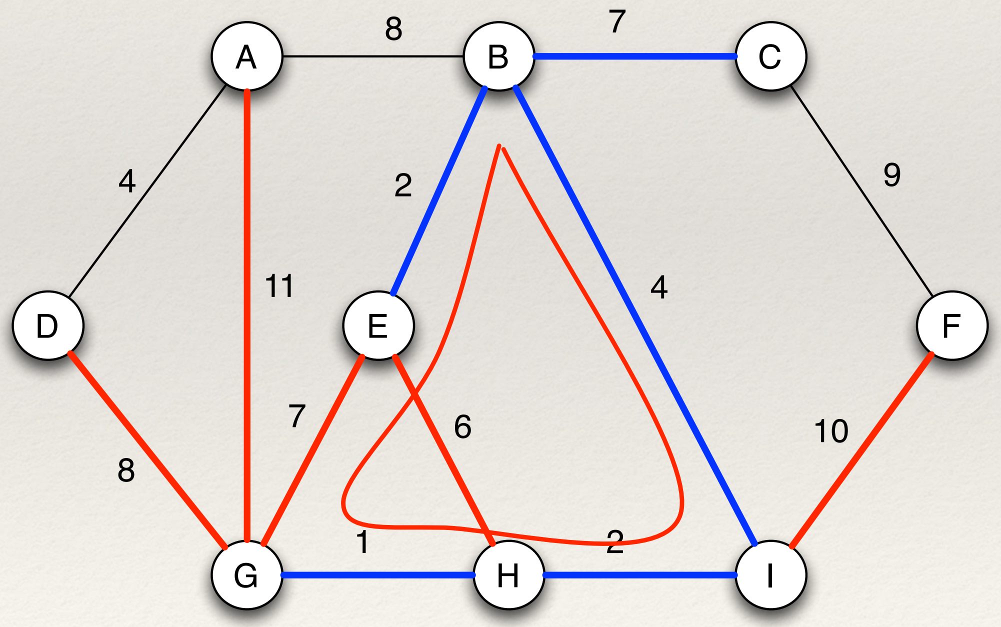
Exemple

- Cycle Maximum
- Coupe Minimum



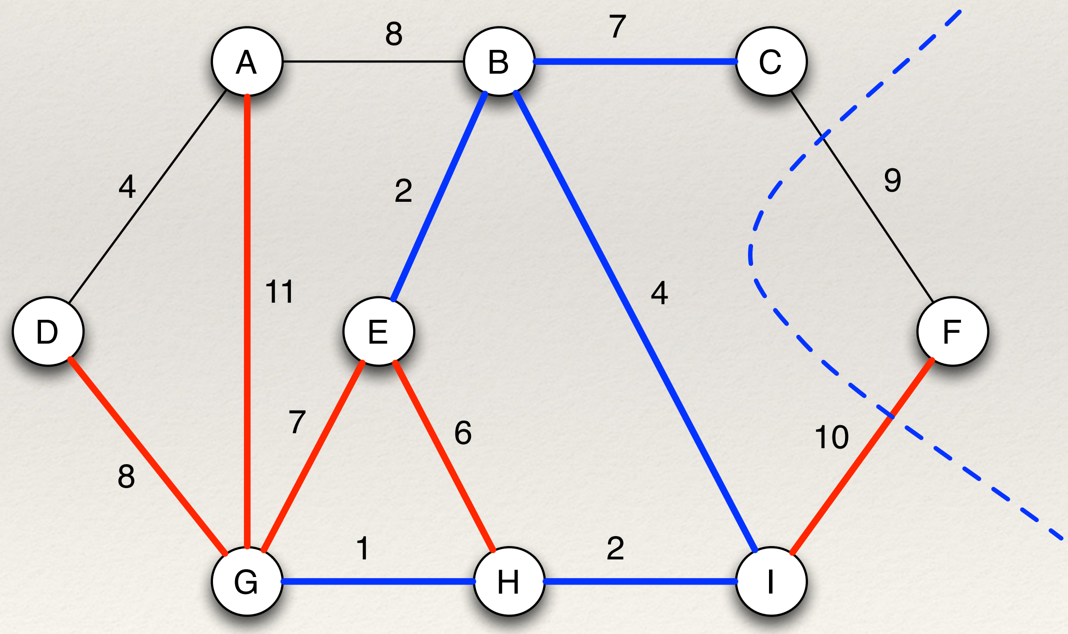
Exemple

- Cycle Maximum
- Coupe Minimum



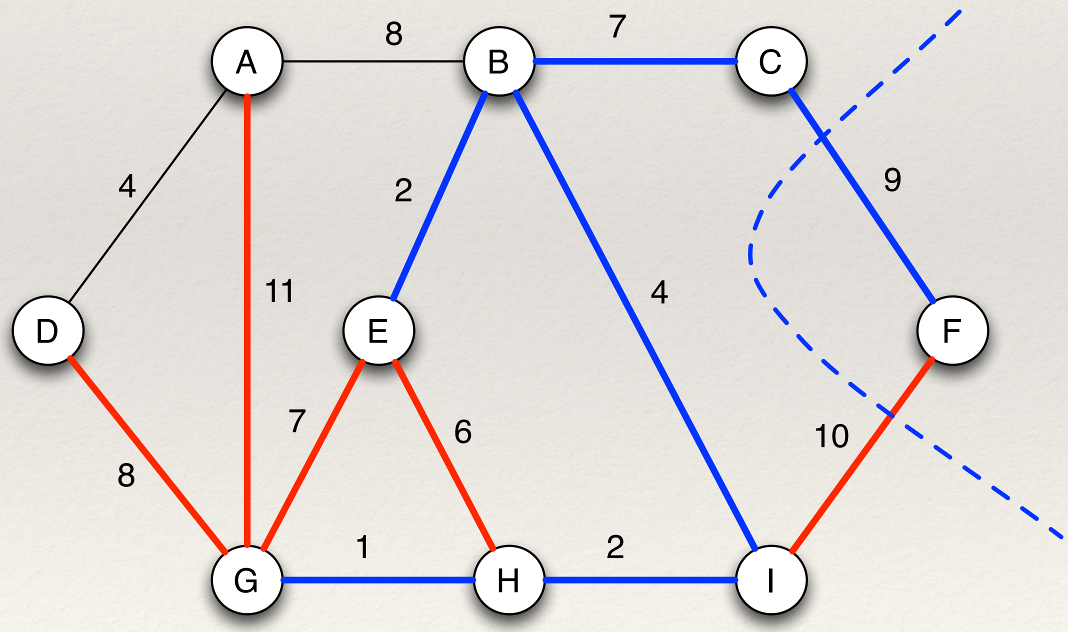
Exemple

- Cycle Maximum
- Coupe Minimum



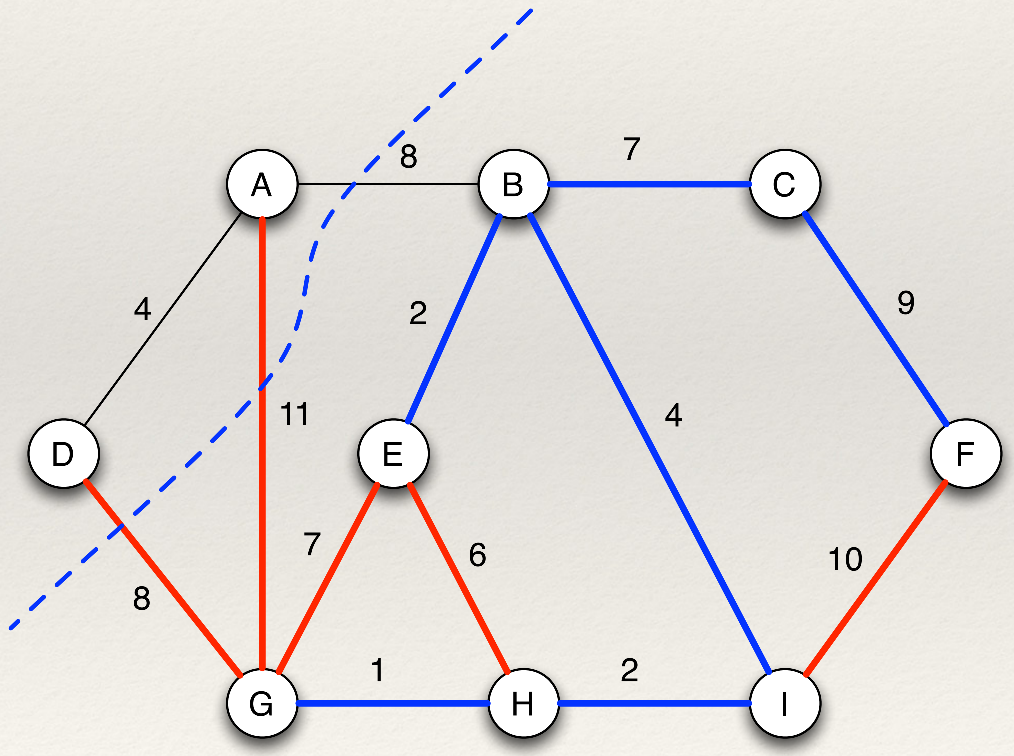
Exemple

- Cycle Maximum
- Coupe Minimum



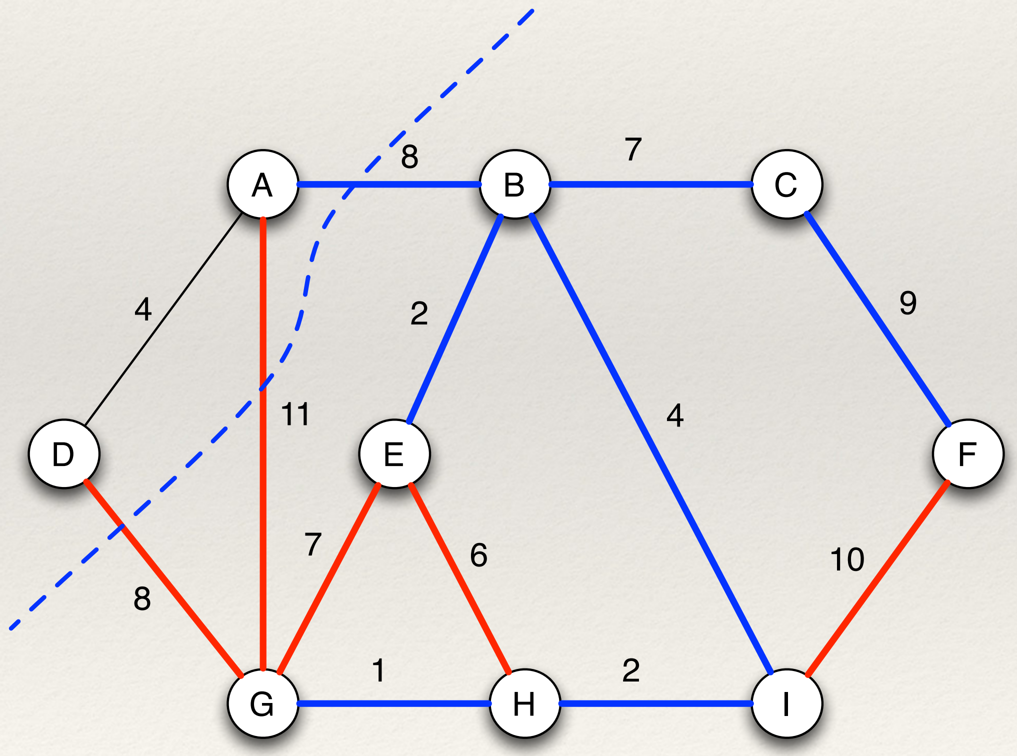
Exemple

- Cycle Maximum
- Coupe Minimum



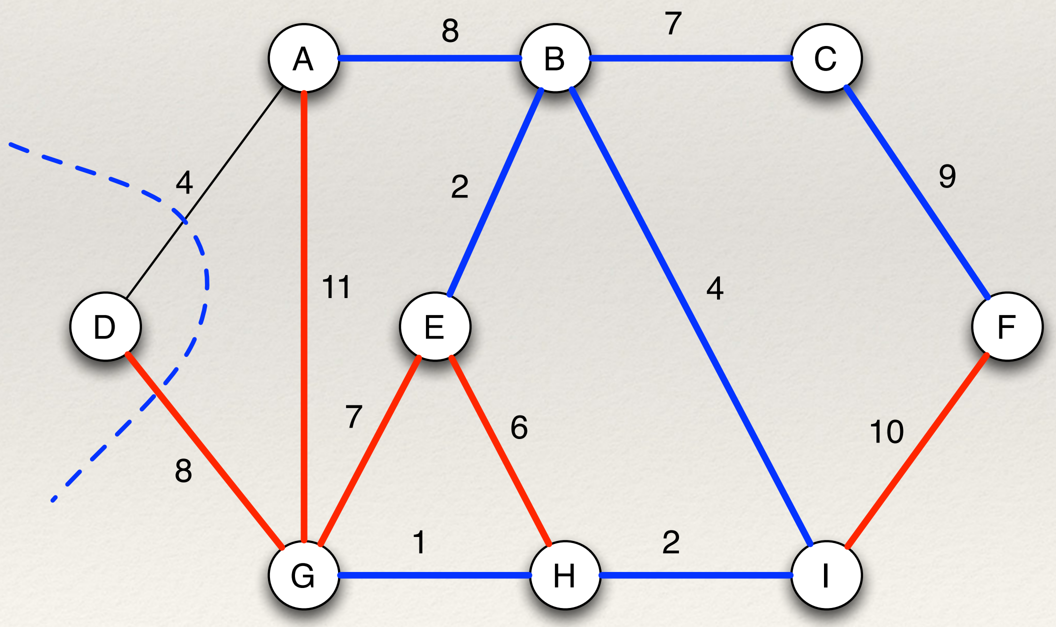
Exemple

- Cycle Maximum
- Coupe Minimum



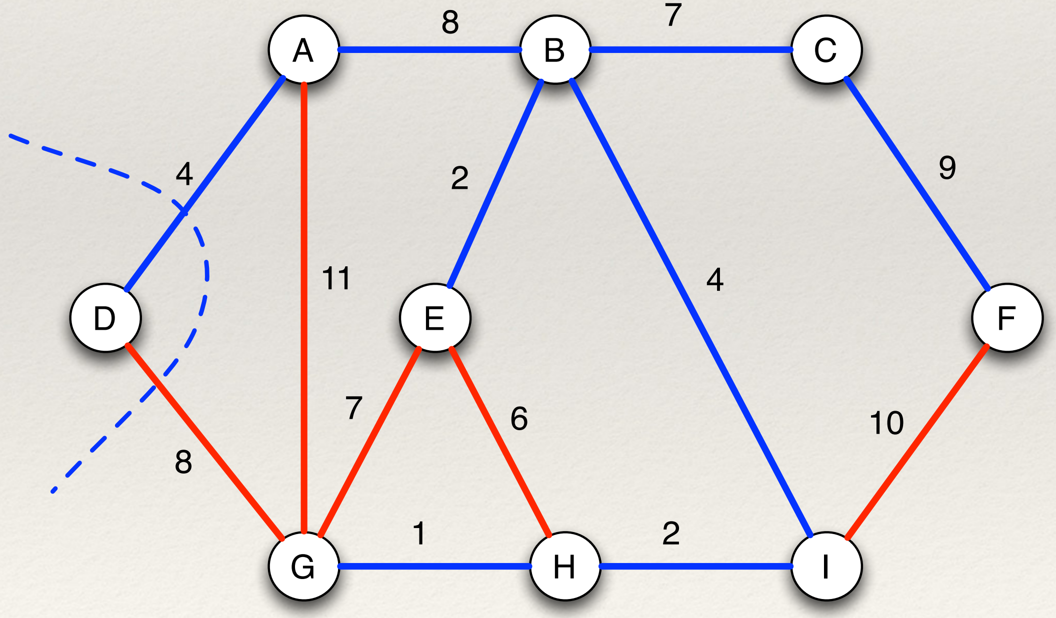
Exemple

- Cycle Maximum
- Coupe Minimum



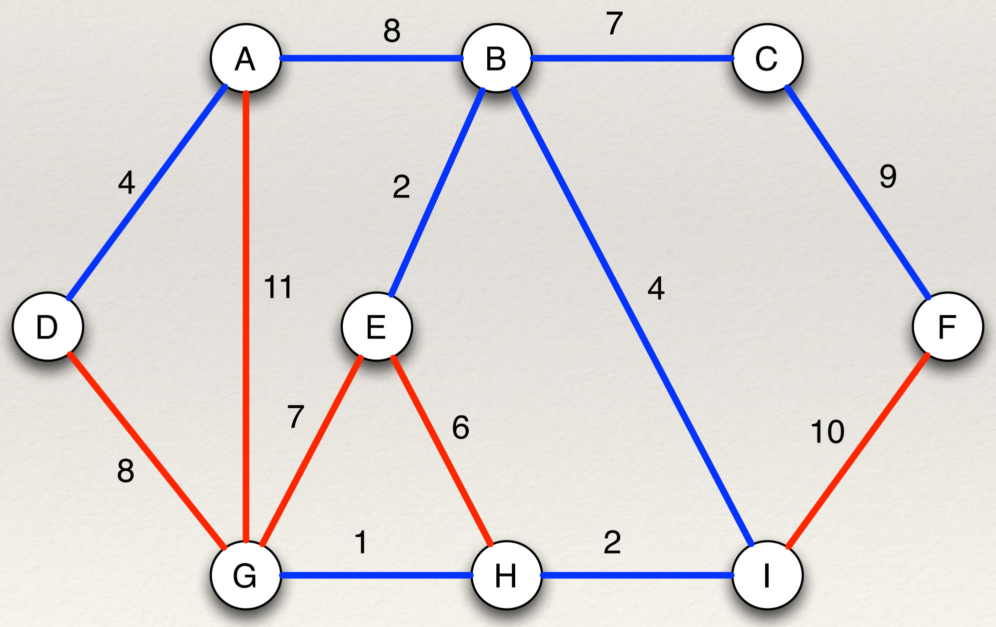
Exemple

- Cycle Maximum
- Coupe Minimum



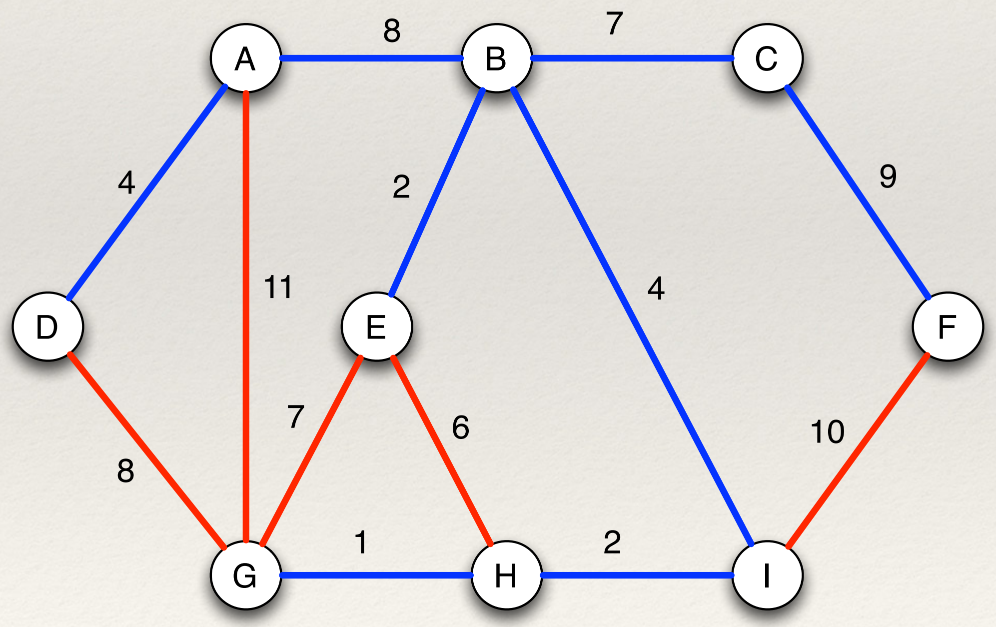
Exemple

- Cycle Maximum
- Coupe Minimum



Exemple

- **En bleu un arbre couvrant de poids minimum T**
- **En rouge un co-arbre à l'arbre T**
  - Un co-arbre est par définition le complémentaire d'un arbre couvrant de G



Remarques

- On peut obtenir un arbre couvrant de poids minimum:
  - En utilisant uniquement la règle bleu.
    - Arrêter l'algorithme après $|V|-1$ boucles.
  - En utilisant uniquement la règle rouge.
    - Arrêter l'algorithme après $|E|-(|V|-1)$ boucles.

Théorème

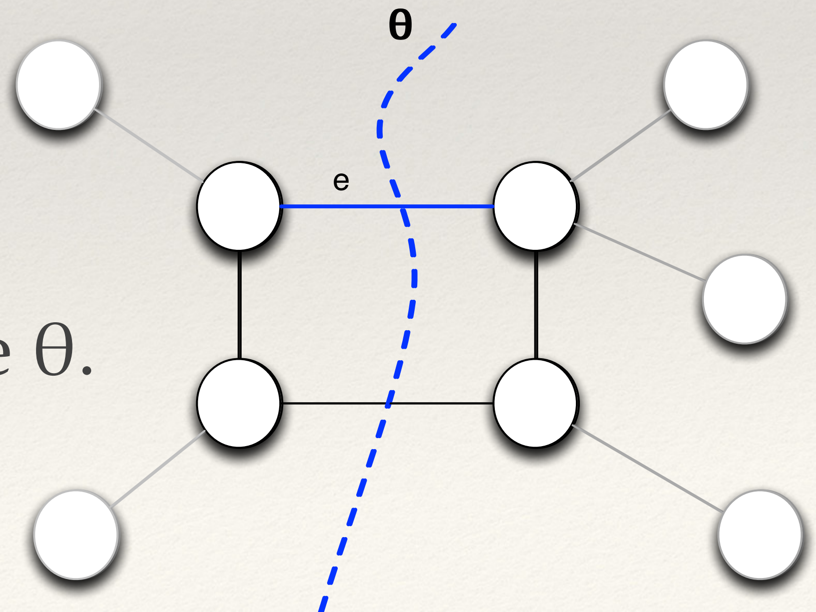
Si G est connexe, l'algorithme de Tarjan calcule un arbre couvrant de poids minimum.

Squelette de la preuve

- Soit un graphe G=(V,E,ω) un graphe non orienté pondéré.
- **Invariant de la boucle:**
  - A la fin de chaque itération de la boucle les arêtes bleues forment:
    - Un graphe-partiel sans-cycle de G (Forêt bleu)
    - Un graphe-partiel sans-cycle de G de poids minimum (Forêt bleu de poids minimum)
- **A la fin de l'algorithme:**
  - Les arêtes en bleues forment un arbre couvrant de poids minimum $T^*(V,E^*)$.
  - Toutes les arêtes sont colorées.
  - Les arêtes en rouges forment un co-arbre de T^*.

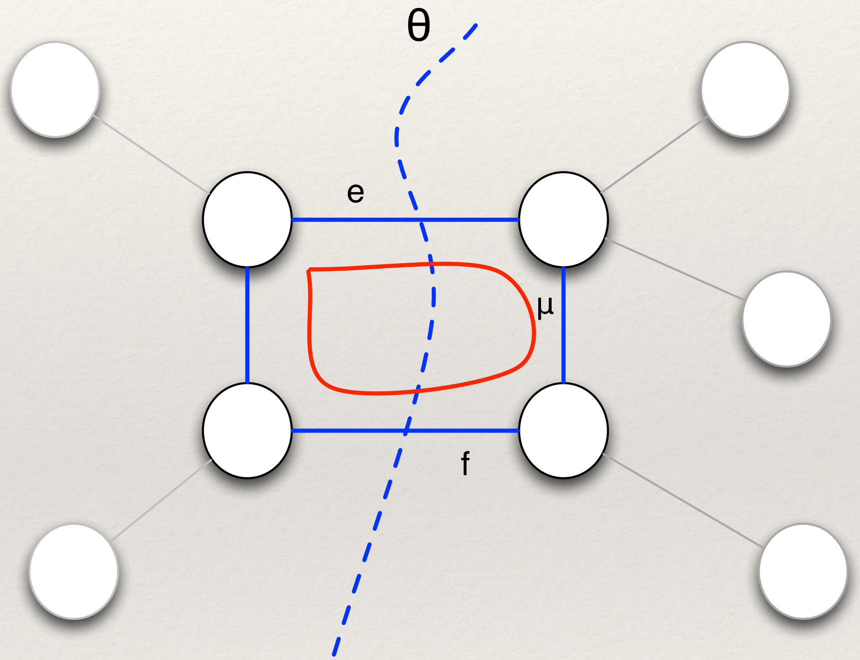
Preuve: Forêt bleu

- Montrons tout d'abord que les arêtes bleues forment une forêt de G
- Au départ il n'y a pas d'arête bleue (donc pas de cycle)
- Intéressons nous à l'application à la $i^{ème}$ boucle de la règle bleue.
  - Soit θ une coupe disponible.
  - Soit e l'arête coloriée en bleu de θ.



Preuve: Forêt bleu

- Supposons que le coloriage de e introduise un cycle entièrement bleu μ.
- D'après le lemme cycle-coupe θ et μ ont au moins deux arêtes en commun.
  - Soit f cette deuxième arête.
- Par définition d'une coupe disponible toutes les arêtes de θ sont:
  - Soient rouges soient non colorées.
  - Contradiction avec le fait que f soit en bleu
- Donc il n'y a pas de cycle totalement bleu.

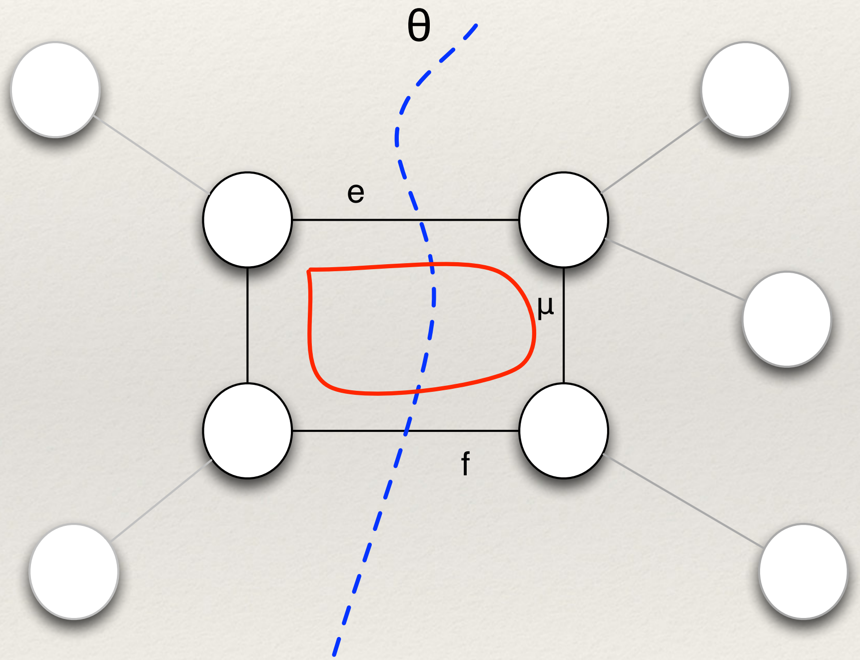


Preuve: Forêt bleu de poids minimum

- Notations:
  - B(i) est l'ensemble des arêtes bleues dans G après la i^{ème} exécution de la boucle.
  - R(i) est l'ensemble des arêtes rouges dans G après la i^{ème} exécution de la boucle.
- Hypothèse de récurrence:
  - $\forall i \leq |E|$, il existe un arbre de poids minimum $T^*(V,E^*)$ tel que $B(i) \subseteq E^*$ et $R(i) \not\subset E^*$.
- Au départ $B(i)=\emptyset$ et $R(i)=\emptyset$ donc l'invariant est vrai.

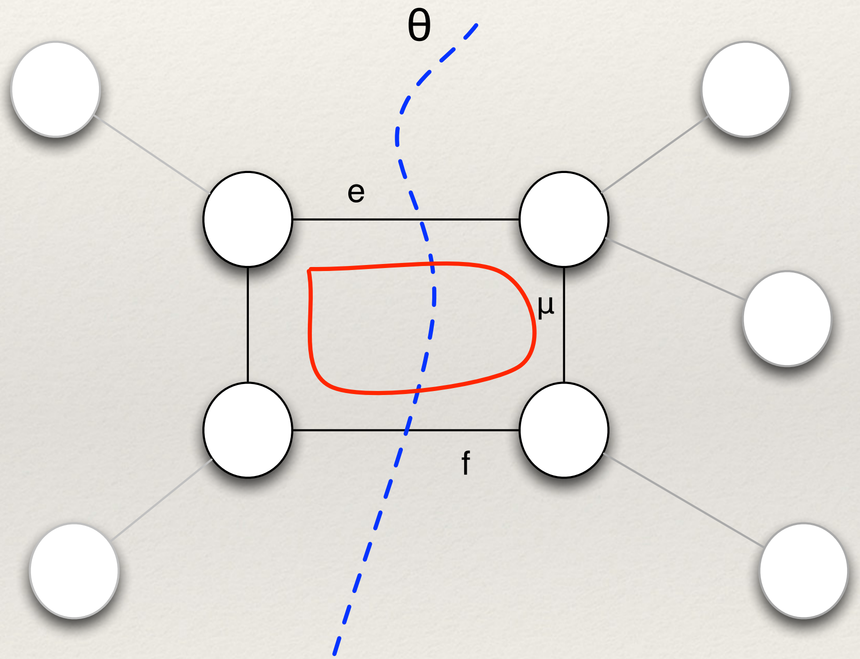
Preuve: Forêt bleu de poids minimum

- Considérons l’exécution de la boucle i+1 avec la règle bleue.
- Soit e l’arête de la coupe θ coloriée en bleu.
- Dans le cas où e∈ E* alors l’hypothèse est trivialement vraie.
- Supposons donc le contraire, e∉E*.
  - Le graphe G’ = (V,E*∪{e}) admet donc un cycle μ.
  - D’après le lemme cycle-coupe un cycle et une coupe ont au moins deux arêtes en commun, notons f une telle arête.
  - Remarque: e et f appartiennent à μ et θ et f∈E*.



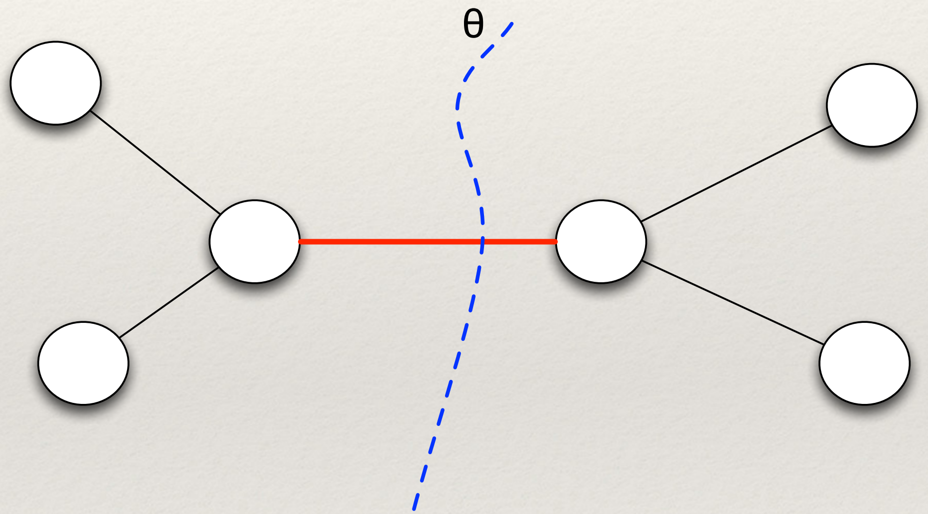
Preuve: Forêt bleu de poids minimum

- Par définition de E*: ω(f) < ω(e).
- Nous obtenons une contradiction car:
  - Soit f était déjà bleue et θ n'est pas une coupe disponible.
  - Soit θ est une coupe disponible et la règle bleu aurait choisit l'arête de la coupe avec le plus petit poids donc f.
- Conclusion l'application de la règle bleu maintient une forêt de poids minimum.
- Le raisonnement est similaire dans le cas d'une application de la règle rouge.



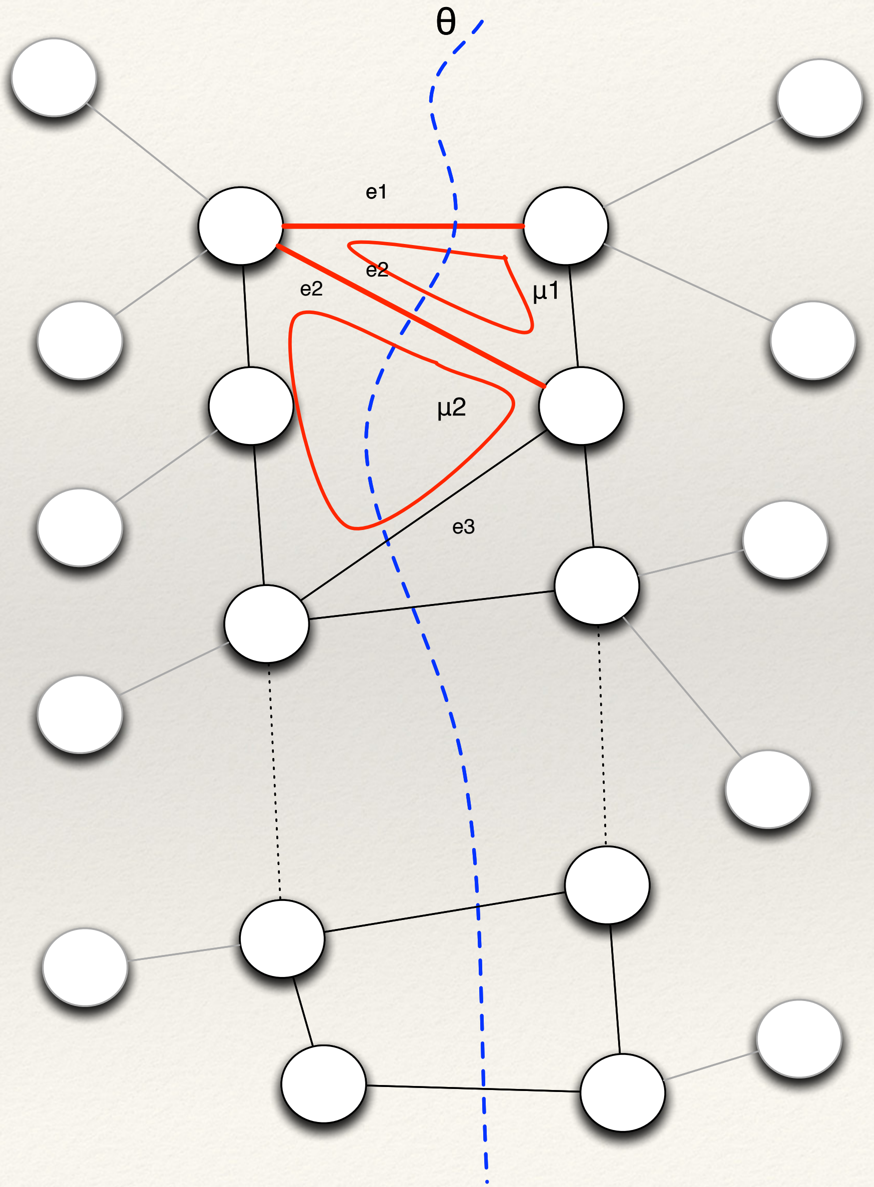
Preuve: Impossibilité d'une coupe rouge

- Supposons qu'il existe une coupe θ qui contient que des arêtes rouges.
- Supposons que:
  - la coupe θ ne coupe aucune arête d'un cycle μ.
  - Contradiction: les arêtes de la coupe ne peuvent pas être rouge car le rouge est réservé au cycle.



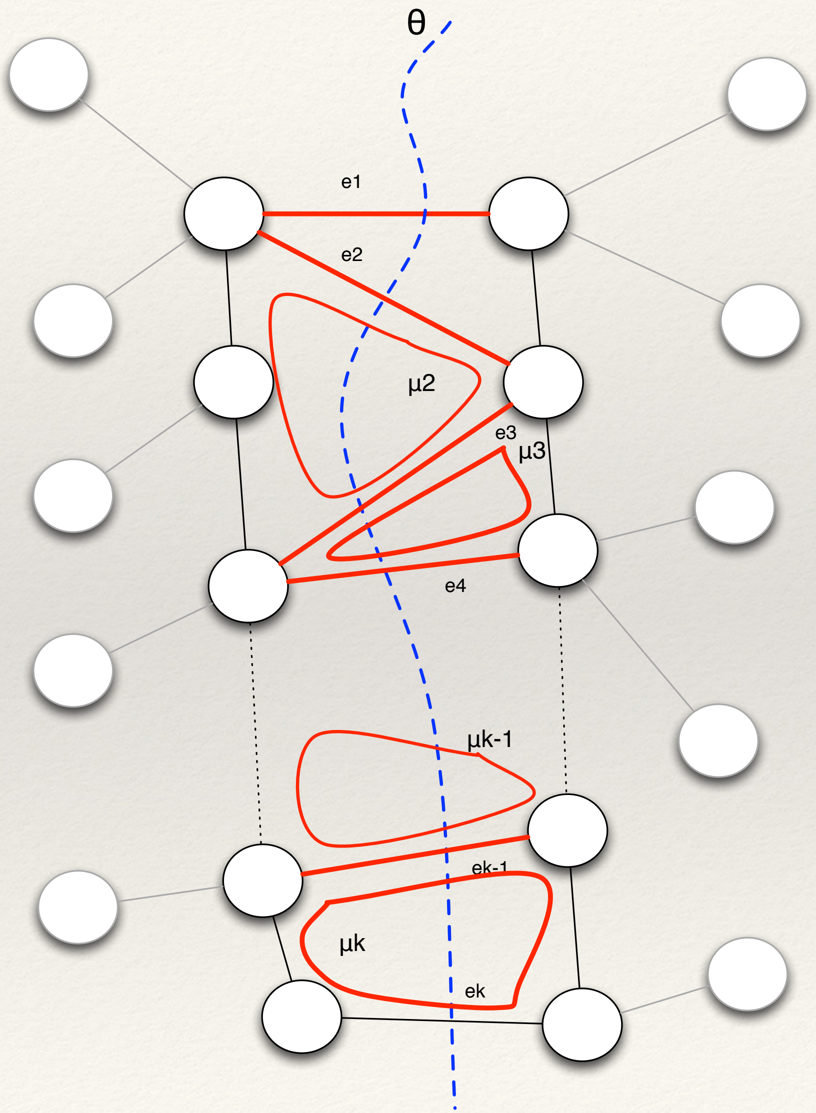
Preuve: Impossibilité d'une coupe rouge

- Supposons que:
  - la coupe θ coupe au moins une arête d'un cycle μ.
- Le lemme cycle-coupe: une coupe et un cycle ont au moins deux arêtes en commun.
  - Supposons que la coupe θ coupe k cycles $\mu_1, \ldots, \mu_k$
  - si $e_1, e_2 \in \theta$ et $e_1, e_2 \in \mu_1$ si $\omega(e_1) > \omega(e_2)$ alors e_1 sera coloriée en rouge.
  - si $e_2, e_3 \in \theta$ et $e_2, e_3 \in \mu_2$ si $\omega(e_2) > \omega(e_3)$ alors e_2 sera coloriée en rouge.
  - …



Preuve: Impossibilité d'une coupe rouge

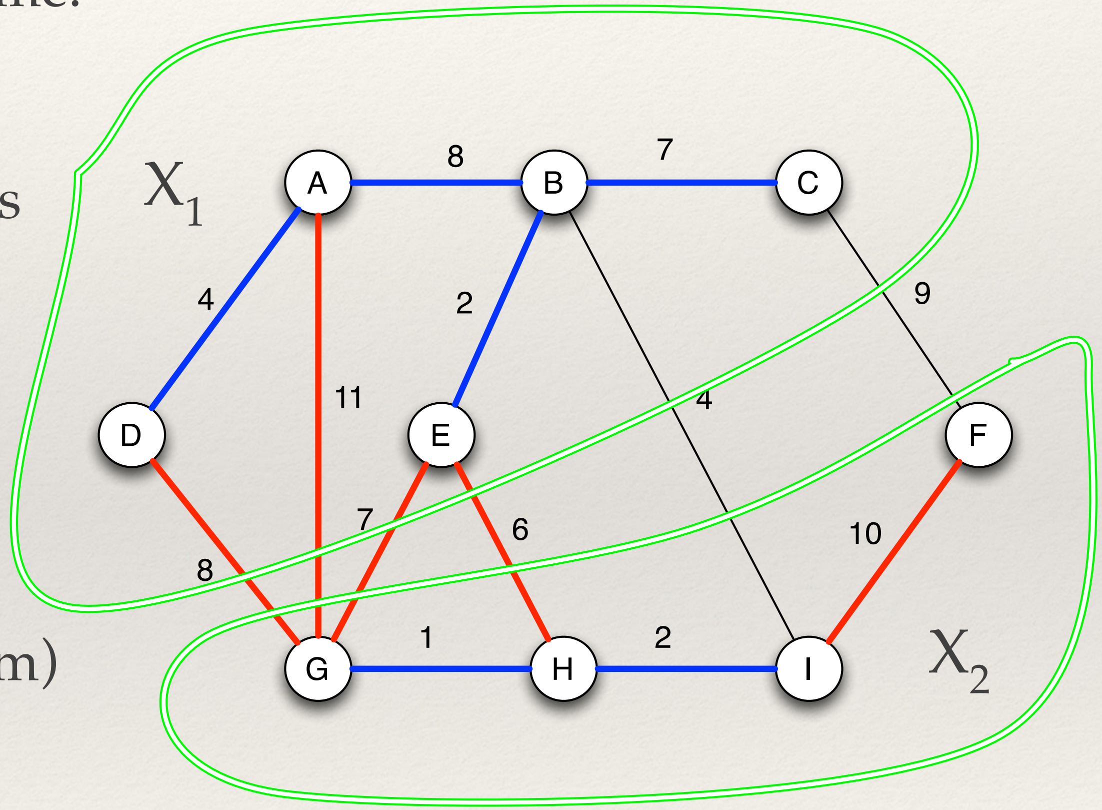
- Soit μ_k le dernier cycle considéré par l'algorithme
- $e_{k-1}, e_k \in \theta$, $e_{k-1}, e_k \in \mu_k$
- L'arête e_{k-1} a été coloriée en rouge quand le cycle μ_{k-1} a été considéré par l'algorithme
- Donc μ_k n'est pas un cycle disponible pour la règle rouge.
- e_k ne peut donc pas être coloriée en rouge



Preuve: Arbre couvrant de poids minimum

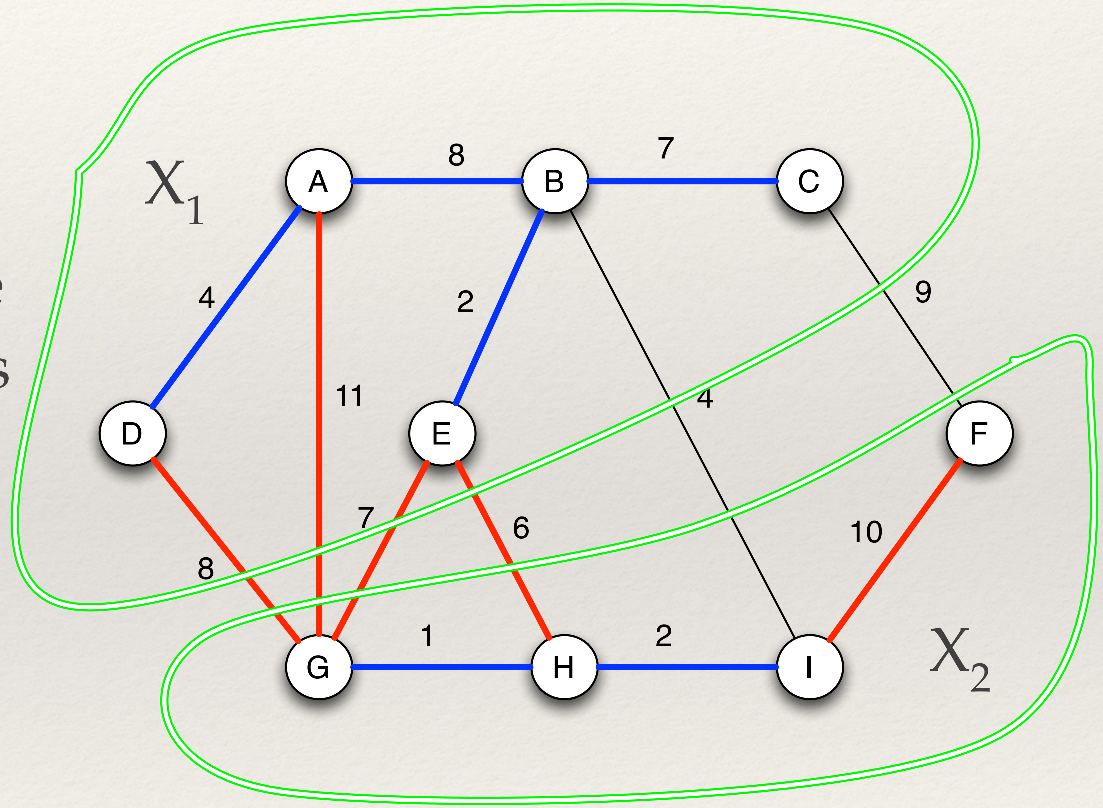
- Considérons la fin de l'algorithme.
    - Aucune règle est applicable.
- Supposons que T* ne forme pas un ACPM.
- Preuve par contradiction:
    - Il existe donc plusieurs composantes connexes

      (Forêt bleu de poids minimum)
    - Sans perte de généralité supposons qu'il en existe deux: X_1 et X_2



Preuve: Arbre couvrant de poids minimum

- Il existe au moins une coupe θ entre X_1 et X_2 qui ne contient aucune arête bleu.
  - Nous avons vu que la coupe ne peut pas contenir que des arêtes rouges.
  - Donc θ contient au moins une arête incolore alors la règle bleu est activable: contradiction avec le fait que l'algorithme est terminé.



Preuve: arêtes coloriées

- Montrons maintenant que quand plus aucune règle n'est applicable toutes les arêtes sont bien coloriées.
- Preuve par contradiction:
  - Pour ce faire supposons qu'il existe une arête $e \in E$ non coloriée.
  - Par définition $e \notin E^*$ car nous avons vu qu'à la fin de l'algorithme nous avons un arbre couvrant de poids minimum:
  - Il existe un cycle unique dans $E^* \cup \{e\}$ sur lequel nous allons pouvoir appliquer la règle rouge puisqu'il est composé uniquement d'arêtes bleues (E^*) et d'une arête incolore (e).
  - Contradiction avec le fait qu'aucune règle n'est applicable.

Preuve: Conclusion

- L'algorithme s'arrête lorsqu'aucune règle n'est applicable.
- Quand l'algorithme s'arrête toutes les arêtes sont coloriées.
- Les arêtes bleues constituent un arbre couvrant de poids minimum
- Les arêtes rouges un co-arbre

Remarques

- L'approche de Tarjan est gloutonne
- A chaque étape il est choisit la solution qui paraît la meilleure:
  - L'arête de poids minimum pour le rouge
  - L'arête de poids maximum pour le bleu
- Une arête coloriée n'est jamais remise en cause

Arbre couvrant de poids minimum

- Etudes de deux algorithmes:
  - Algorithme de Prim (1957)
  - Algorithme de Kruskal (1956)
- Il en existe d'autres
  - Borukva (1926)
  - Solin (1961)
  - …

L'algorithme de PRIM (1957)

- Principe
- Formalisation de l'algorithme
- Exemple
- Justification de l'algorithme
- Analyse de la complexité

Principe

- L'idée principale
  - Maintenir un sous-graphe partiel connexe
  - A chaque étape
    - Connecter un nouveau sommet au sous graphe partiel existant par l'arête de poids minimum sortante du graphe partiel.
  - L'algorithme de Prim va ainsi faire grossir un arbre jusqu'à ce qu'il couvre tous les sommets du graphe.

Principe

- L'algorithme part d'une constatation simple :
  - Dans un arbre couvrant, il existe nécessairement une arête qui relie l'un des sommets de S avec un sommet en dehors de S.
  - Autrement dit une coupe.
  - Il suffit de choisir parmi les arêtes de la coupe celle de poids minimum.
    - Règle bleu

Algorithme de Prim

PRIM(G,w)

S = {s}; s est un sommet quelconque de V

MST = {}; ensemble des arêtes de l'arbre

tant que (|S| < |V|)

    (u,v) = arête de poids min ayant $u \in S$ et $v \in V-S$;

    $S = S \cup \{v\}$;

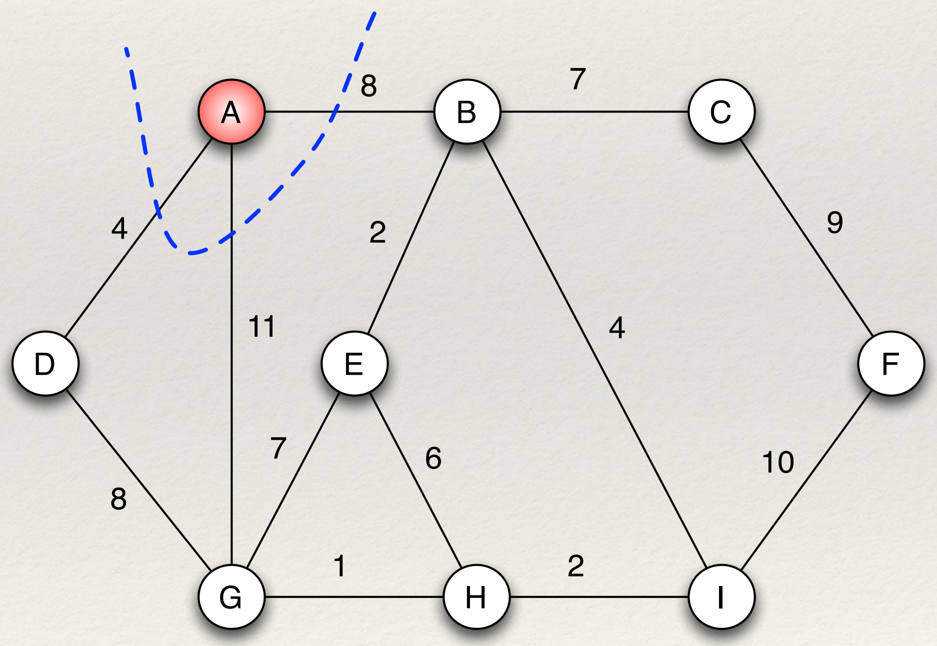
    $MST = MST \cup \{(u,v)\}$;

Fin tant que

Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (| S | < | V |)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

S={A}
MST={}

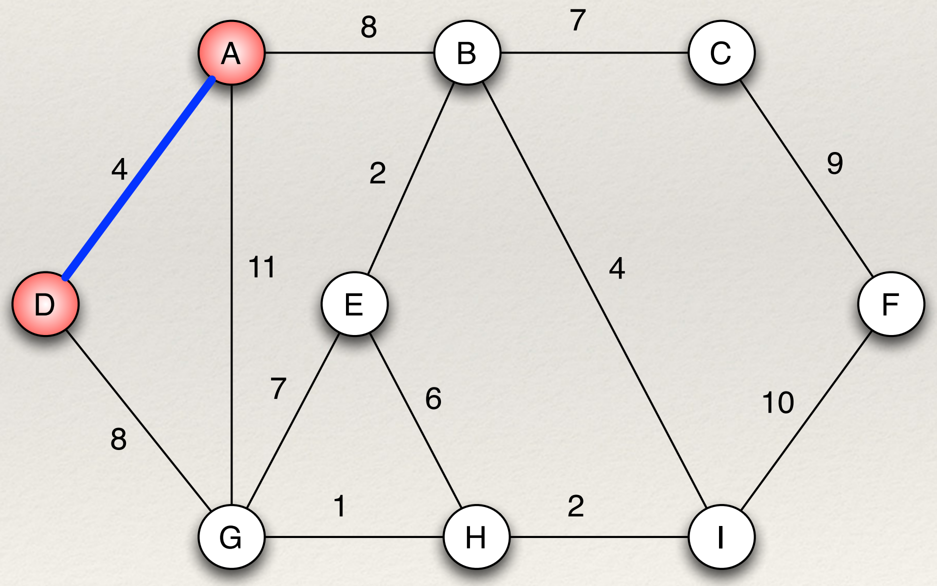


Exemple de Prim

- **ACPM-PRIM(G,w,s)**
- S = {s}; s est un sommet quelconque de V
- MST = {}; ensemble des arêtes de l'arbre
- tant que (| S | < | V |)
-     (u,v) = arête de poids min ayant u∈S et v∈V-S;
-     S = S∪{v};
-     MST = MST ∪{(u,v)};
- Fin tant que

S={A,D}

MST={(A,D)}

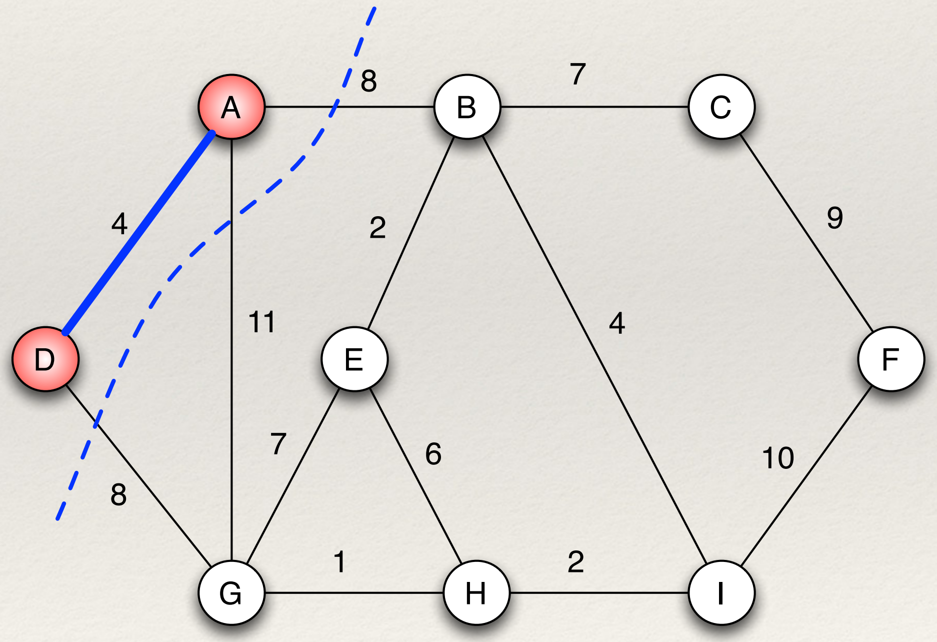


Exemple de Prim

- **ACPM-PRIM(G,w,s)**
- S = {s}; s est un sommet quelconque de V
- MST = {}; ensemble des arêtes de l'arbre
- tant que (| S | < | V |)
-     (u,v) = arête de poids min ayant u∈S et v∈V-S;
-     S = S∪{v};
-     MST = MST ∪{(u,v)};
- Fin tant que

S={A,D}

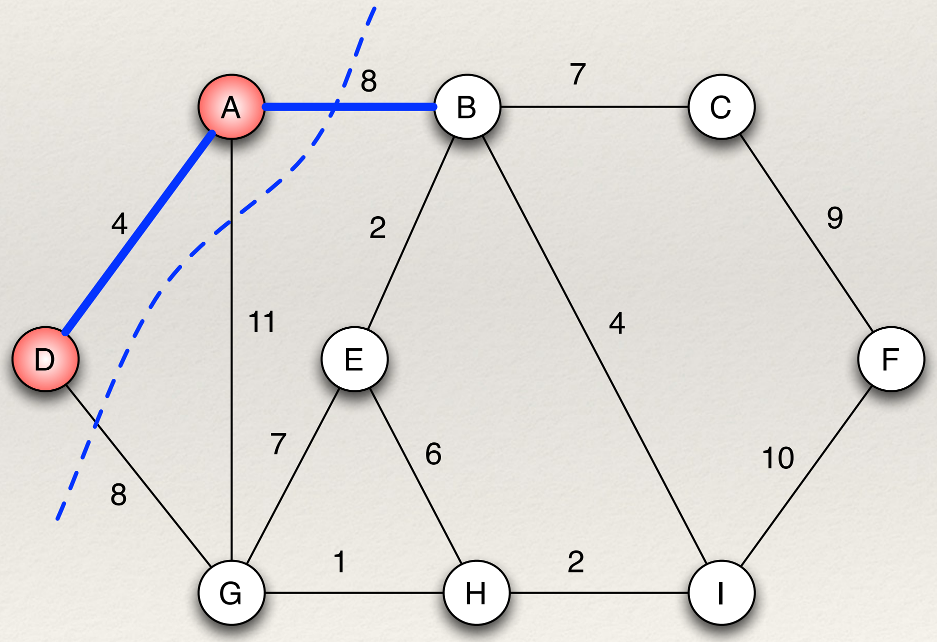
MST={(A,D)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

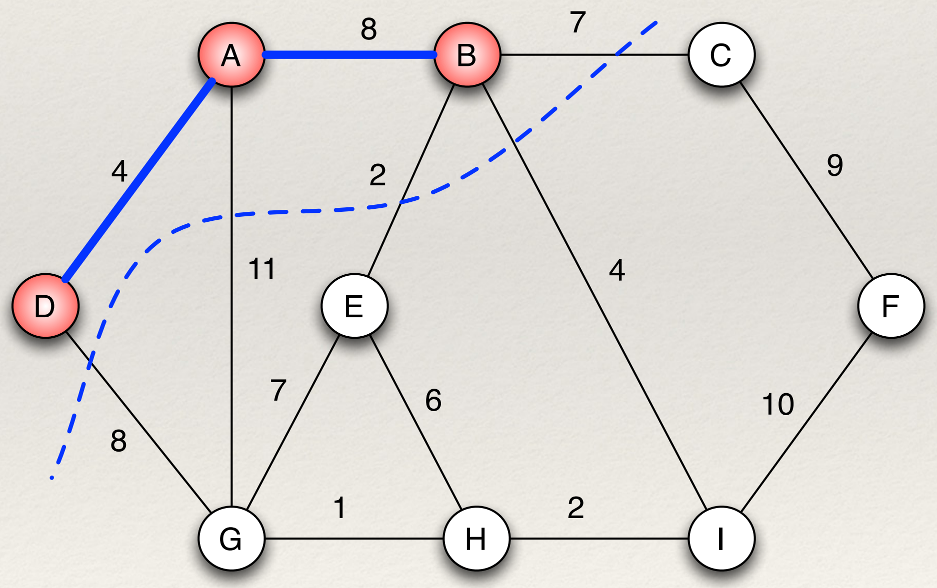
S={A,D,B}
MST={(A,D),(A,B)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

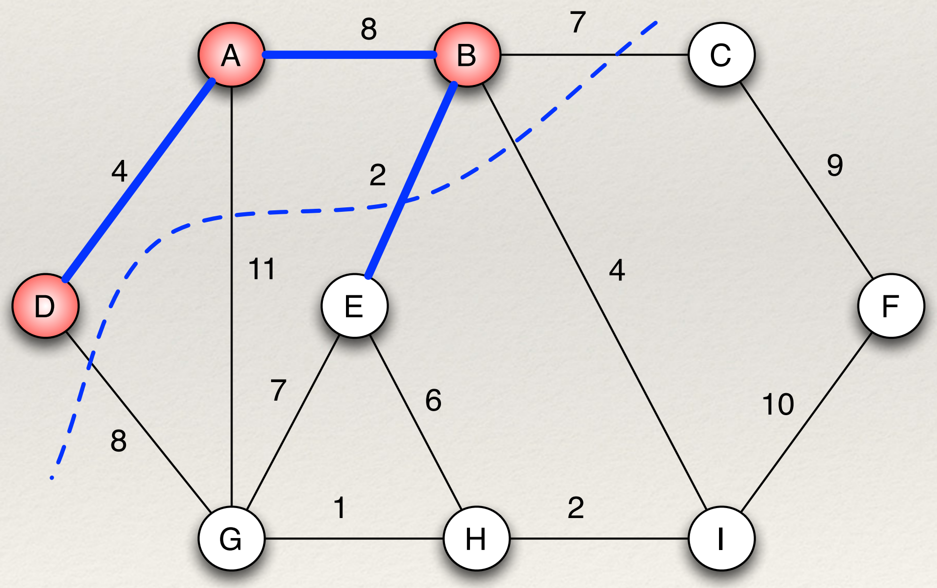
S={A,D,B}
MST={(A,D),(A,B)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

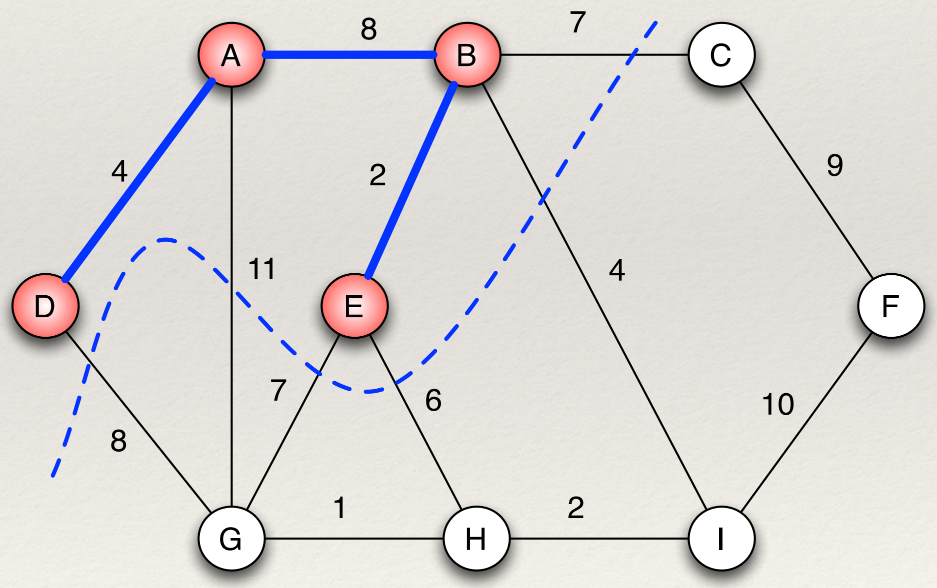
S={A,D,B,E}
MST={(A,D),(A,B),(B,E)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

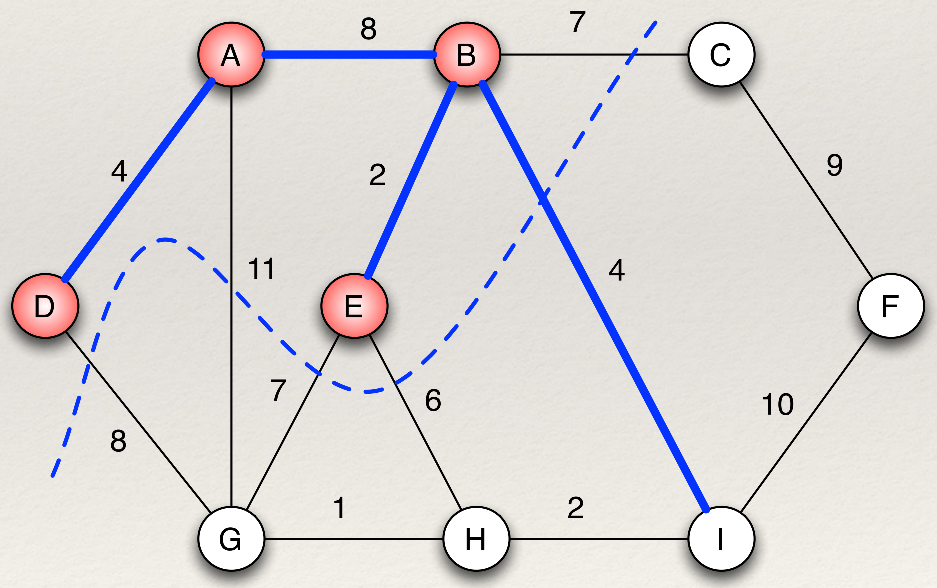
S={A,D,B,E}
MST={(A,D),(A,B),(B,E)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
  (u,v) = arête de poids min ayant u∈S et v∈V-S;
  S = S∪{v};
  MST = MST ∪{(u,v)};
Fin tant que

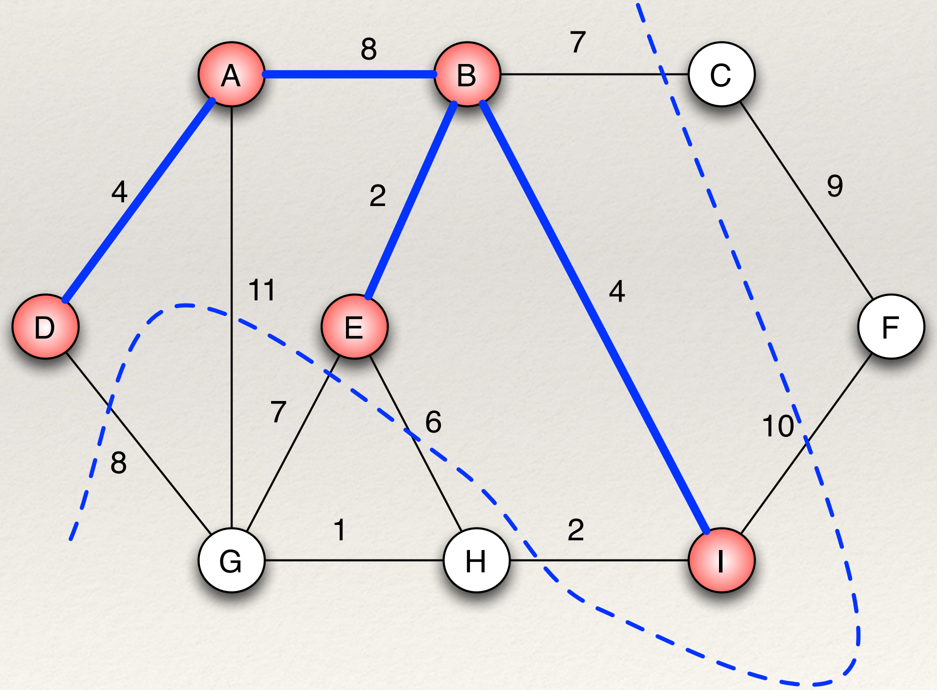
S={A,D,B,E,I}
MST={(A,D),(A,B),(B,E),(B,I)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (| S | < | V |)
  (u,v) = arête de poids min ayant u∈S et v∈V-S;
  S = S∪{v};
  MST = MST ∪{(u,v)};
Fin tant que

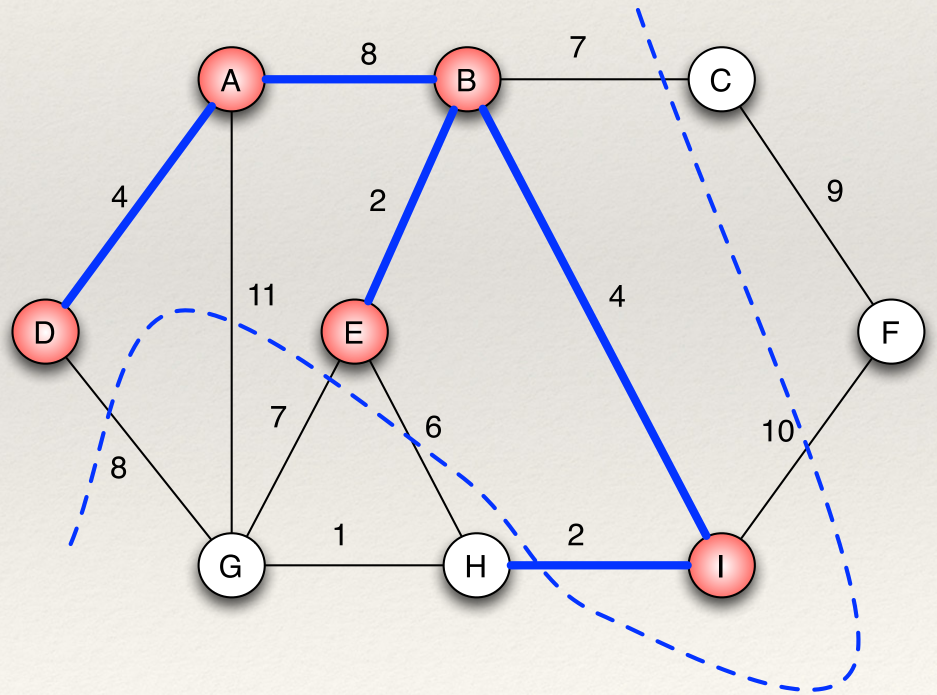
S={A,D,B,E,I}
MST={(A,D),(A,B),(B,E),(B,I)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

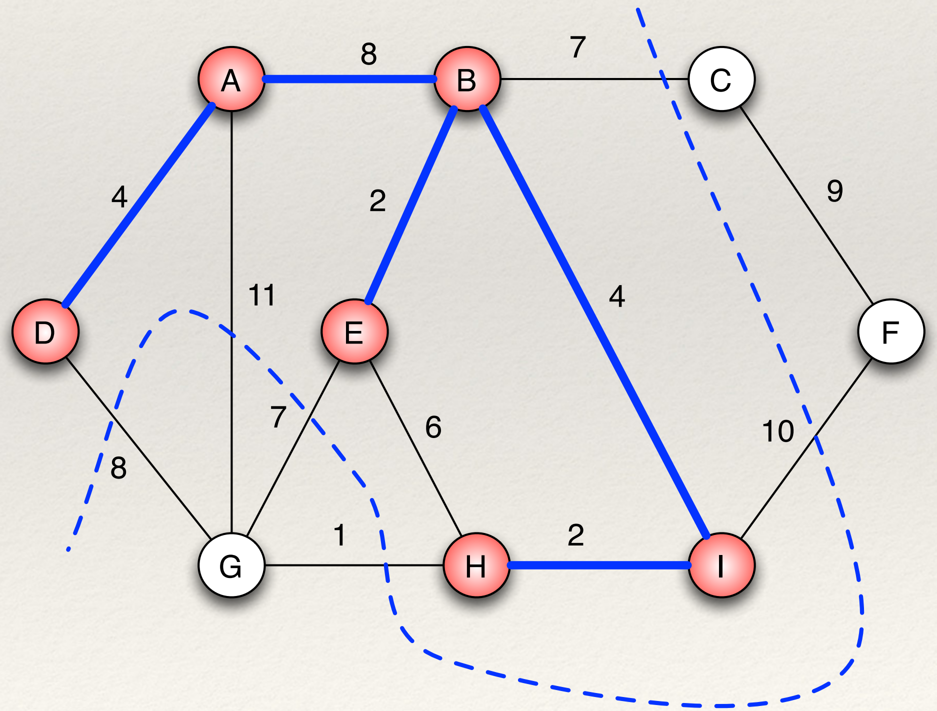
S={A,D,B,E,I,H}
MST={(A,D),(A,B),(B,E),(B,I),(I,H)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

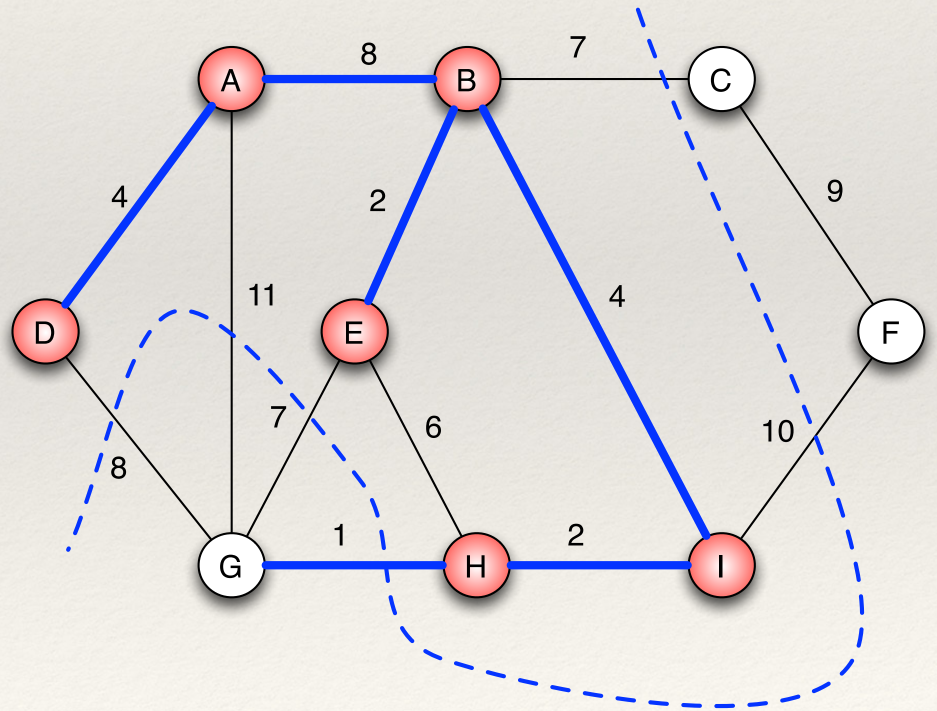
S={A,D,B,E,I,H}
MST={(A,D),(A,B),(B,E),(B,I),(I,H)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
  (u,v) = arête de poids min ayant u∈S et v∈V-S;
  S = S∪{v};
  MST = MST ∪{(u,v)};
Fin tant que

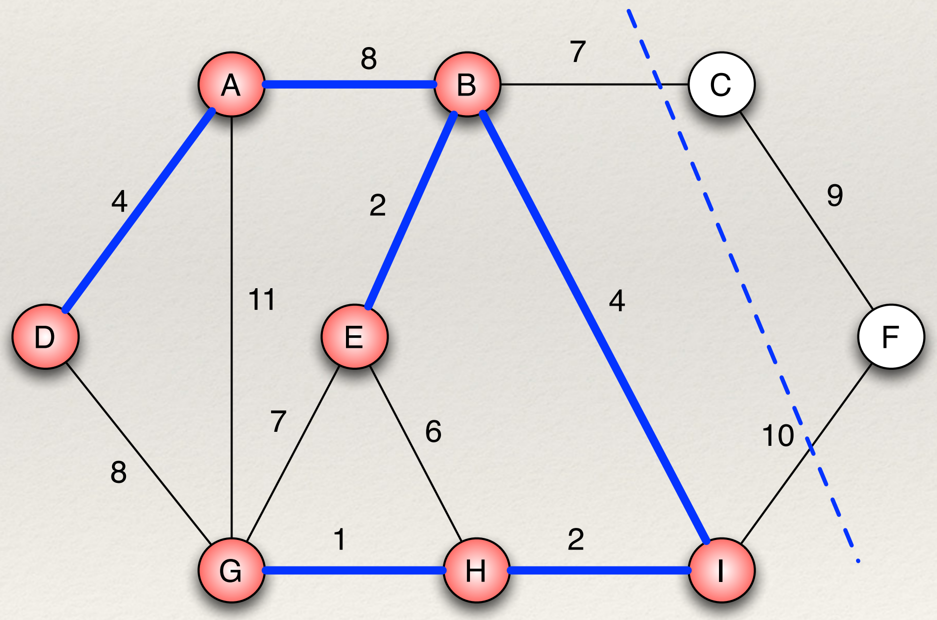
S={A,D,B,E,I,H,G}
MST={(A,D),(A,B),(B,E),(B,I),(I,H),(H,G)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

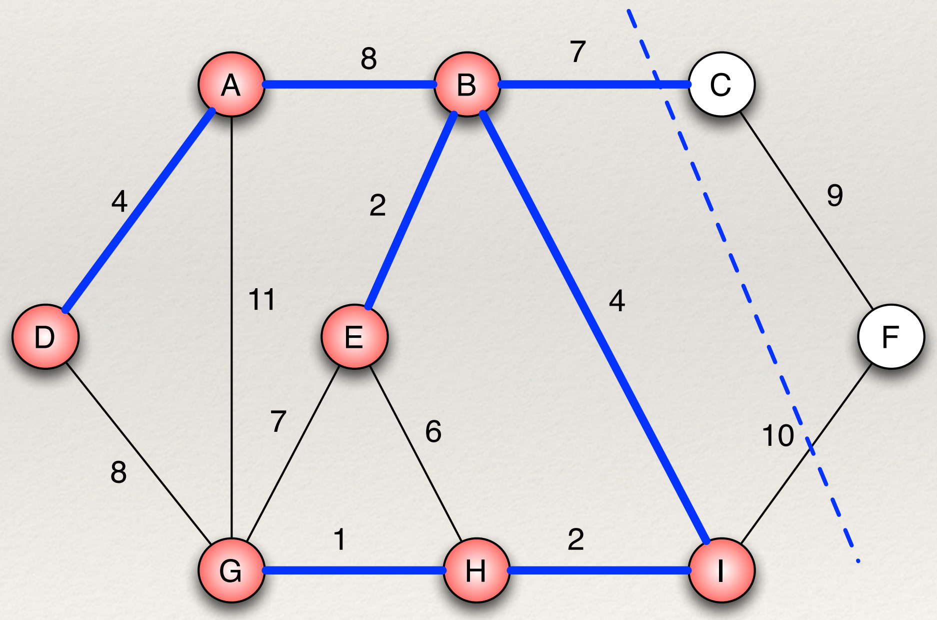
S={A,D,B,E,I,H,G}
MST={(A,D),(A,B),(B,E),(B,I),(I,H),(H,G)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

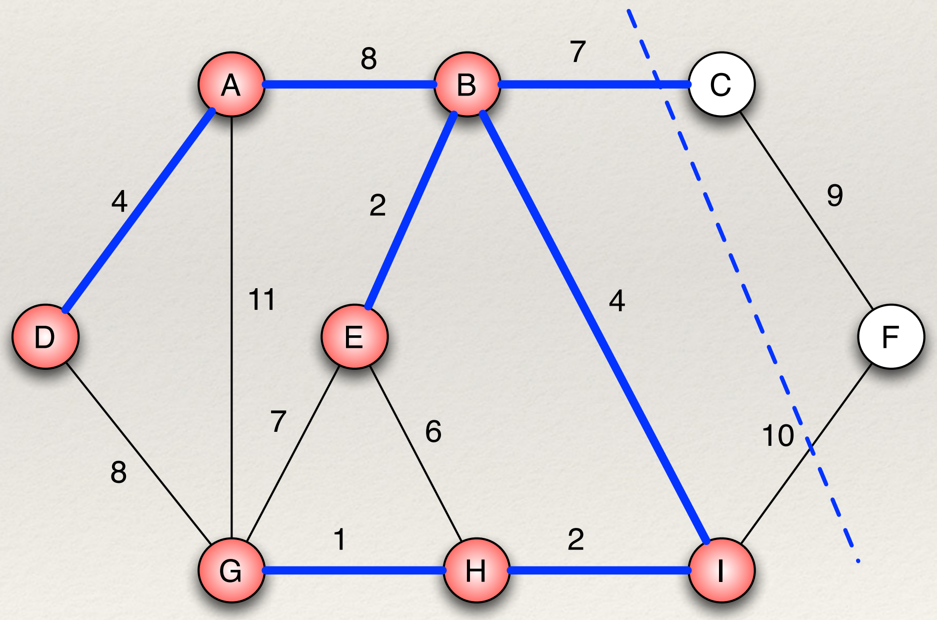
S={A,D,B,E,I,H,G,C}
MST={(A,D),(A,B),(B,E),(B,I),(I,H),(H,G),(B,C)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

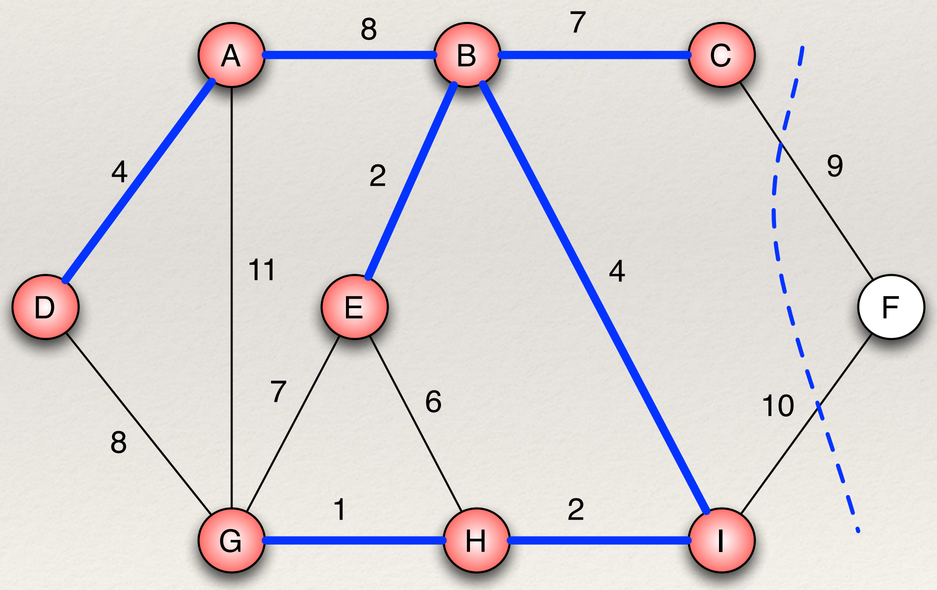
S={A,D,B,E,I,H,G,C}
MST={(A,D),(A,B),(B,E),(B,I),(I,H),(H,G),(B,C)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

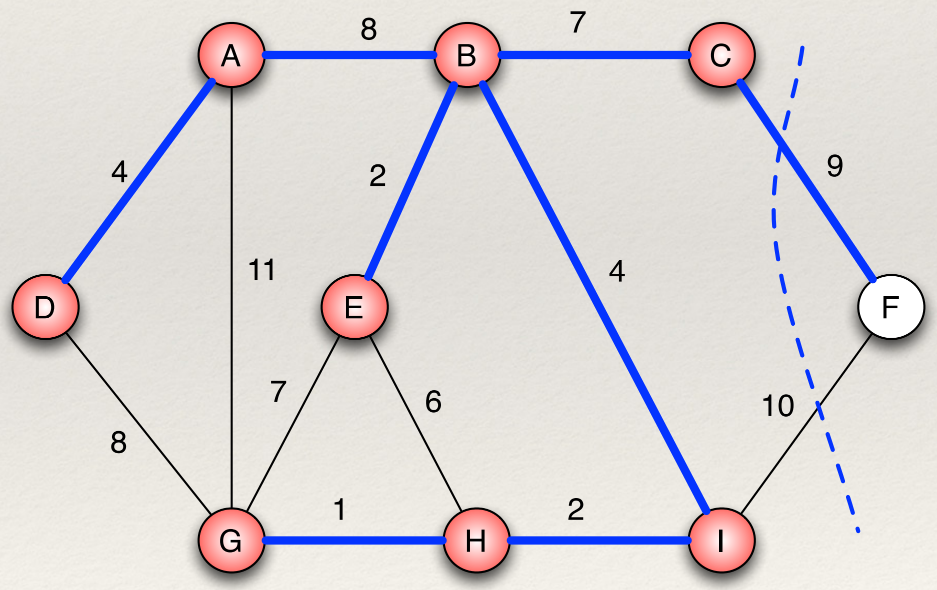
S={A,D,B,E,I,H,G,C}
MST={(A,D),(A,B),(B,E),(B,I),(I,H),(H,G),(B,C)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (|S| < |V|)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

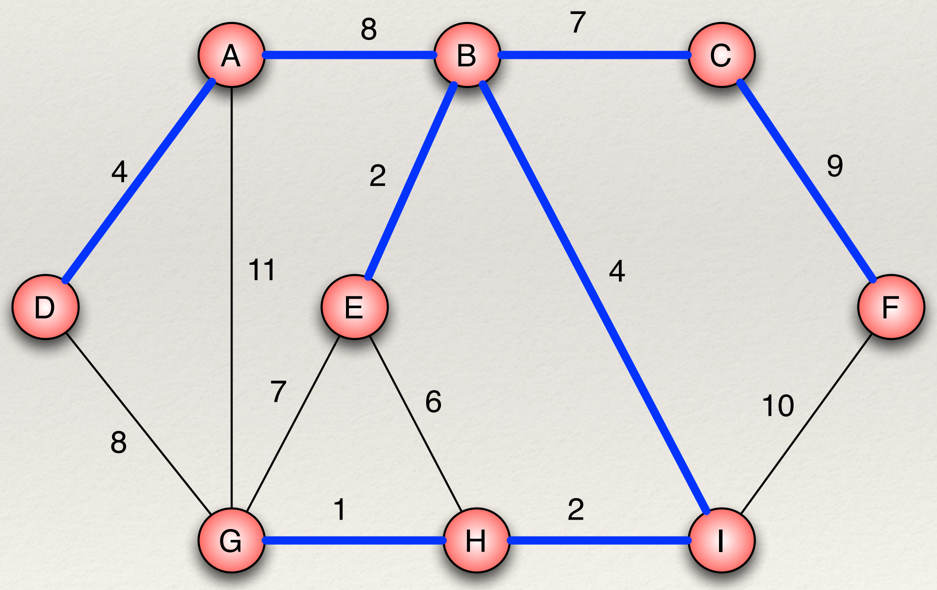
S={A,D,B,E,I,H,G,C,F}
MST={(A,D),(A,B),(B,E),(B,I),(I,H),(H,G),(B,C),(C,F)}



Exemple de Prim

ACPM-PRIM(G,w,s)
S = {s}; s est un sommet quelconque de V
MST = {}; ensemble des arêtes de l'arbre
tant que (| S | < | V |)
    (u,v) = arête de poids min ayant u∈S et v∈V-S;
    S = S∪{v};
    MST = MST ∪{(u,v)};
Fin tant que

S={A,D,B,E,I,H,G,C,F}
MST={(A,D),(A,B),(B,E),(B,I),(I,H),(H,G),(B,C),(C,F)}



Théorème

L'algorithme de Prim construit un arbre couvrant de poids minimum sur tout graphe connexe.

Preuve de correction

- On peut l'obtenir directement de la preuve de Tarjan
- Prim est l'application de la règle bleu
- Seul l'ordre de choix des coupes est imposé par Prim

Preuve de complexité

- L'algorithme comporte |V| étapes (Entrée des sommets dans S).
- A chaque étape il faut déterminer:
  - l'arête sortante de poids minimum.
- Une implémentation possible consiste à chaque étape:
  - Vérifier toutes les arêtes incidentes à un sommet de S,
  - Déterminer si elles sont sortantes
  - Garder celle de poids minimum.

Preuve complexité

- Cette implémentation conduit à une complexité de l'ordre de la somme des degrés pour chaque étape:
- Propriété des degrés: La somme des degrés des sommets d'un graphe est égal à 2 fois son nombre d'arêtes.
- Complexité totale $O(|V|.|E|)$
- Remarque: En utilisant un tas de Fibonacci, on peut obtenir :
  - $O(|E| + |V| \times \log|V|)$.

L'algorithme de KRUSKAL (1956)

- Principe
- Formalisation de l'algorithme
- Exemple
- Justification de l'algorithme
- Analyse de la complexité

Principe

- L'algorithme de Kruskal se base sur
  - La caractérisation des arbres comme des graphes acycliques maximaux
  - On maintient l’absence de cycle

Principe

- L'algorithme de Kruskal va ainsi ajouter au fur et à mesure des arêtes en s'assurant que le graphe partiel reste à chaque étape une forêt
- L’algorithme s’arête quand la forêt devienne un arbre !

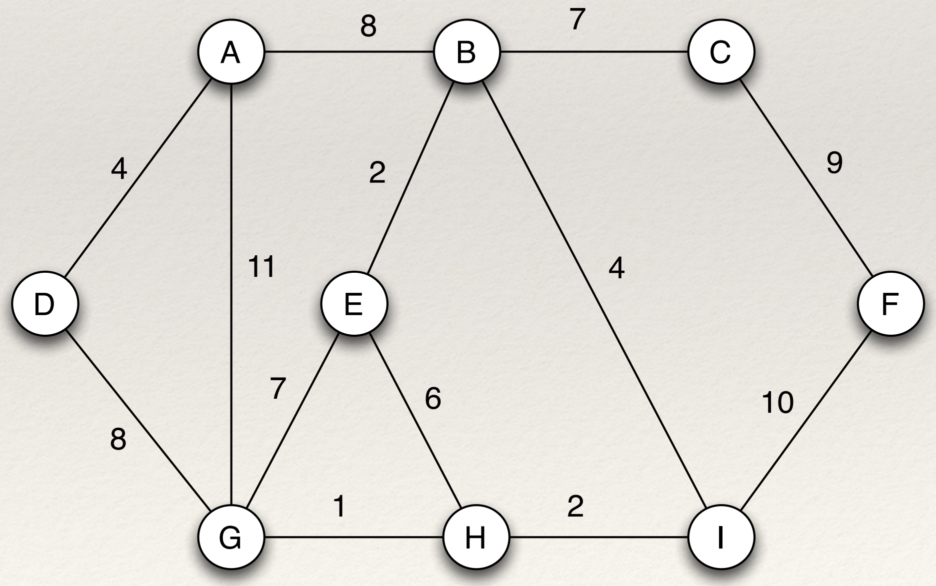
Algorithme de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
 Fin pour
```

Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
Fin pour
```

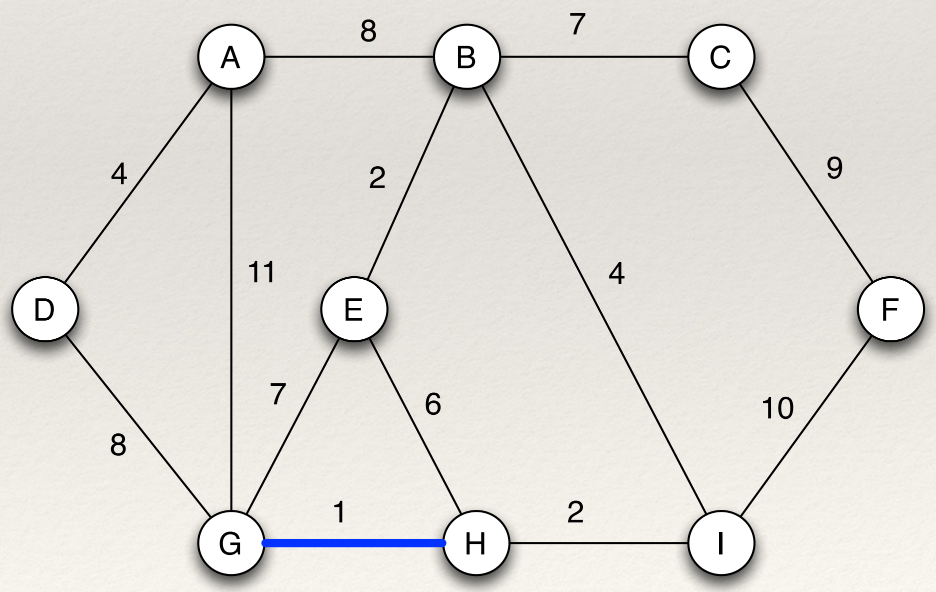
- $\omega\{G,H\}=1$
- $\omega\{B,E\}=2$
- $\omega\{H,I\}=2$
- $\omega\{A,D\}=4$
- $\omega\{B,I\}=4$
- $\omega\{E,H\}=6$
- $\omega\{B,C\}=7$
- $\omega\{E,G\}=7$
- $\omega\{A,B\}=8$
- $\omega\{D,G\}=8$
- $\omega\{C,F\}=9$
- $\omega\{F,I\}=10$
- $\omega\{A,G\}=11$



Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
 Fin pour
```

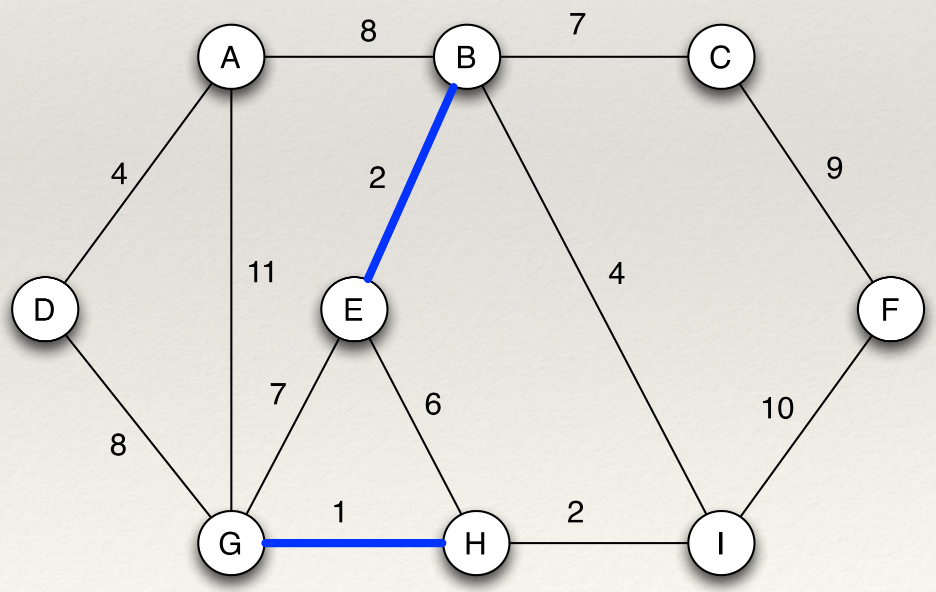
- **$\omega\{G,H\}=1$**
- $\omega\{B,E\}=2$
- $\omega\{H,I\}=2$
- $\omega\{A,D\}=4$
- $\omega\{B,I\}=4$
- $\omega\{E,H\}=6$
- $\omega\{B,C\}=7$
- $\omega\{E,G\}=7$
- $\omega\{A,B\}=8$
- $\omega\{D,G\}=8$
- $\omega\{C,F\}=9$
- $\omega\{F,I\}=10$
- $\omega\{A,G\}=11$



Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
Fin pour
```

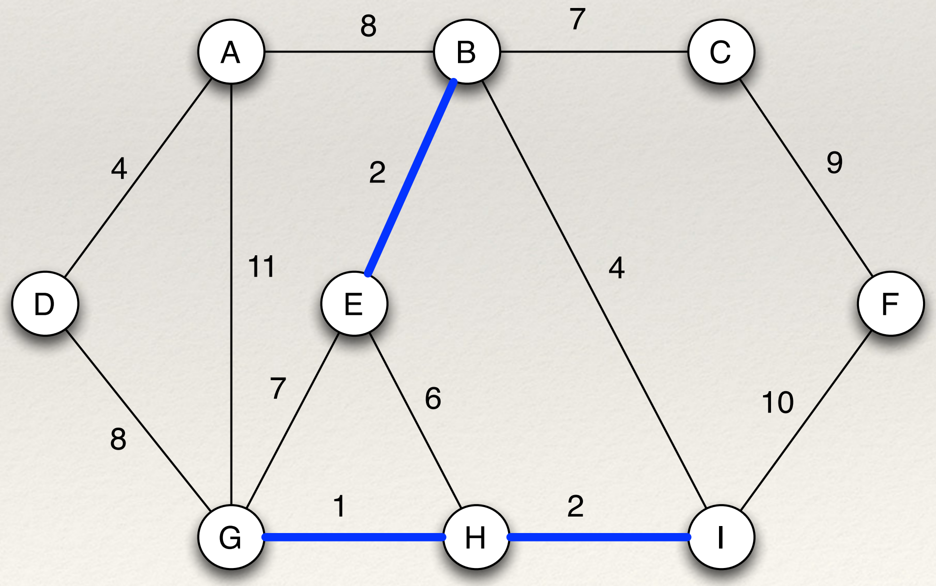
- **ω{G,H}=1**
- **ω{B,E}=2**
- ω{H,I}=2
- ω{A,D}=4
- ω{B,I}=4
- ω{E,H}=6
- ω{B,C}=7
- ω{E,G}=7
- ω{A,B}=8
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11



Exemple de Kruskal

```
   ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
 Fin pour
```

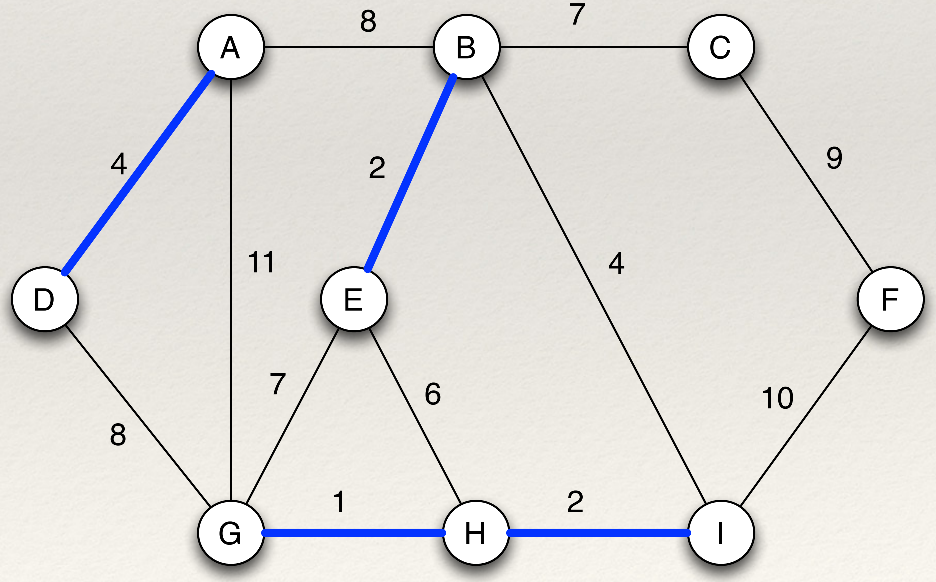
- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- ω{A,D}=4
- ω{B,I}=4
- ω{E,H}=6
- ω{B,C}=7
- ω{E,G}=7
- ω{A,B}=8
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11



Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
Fin pour
```

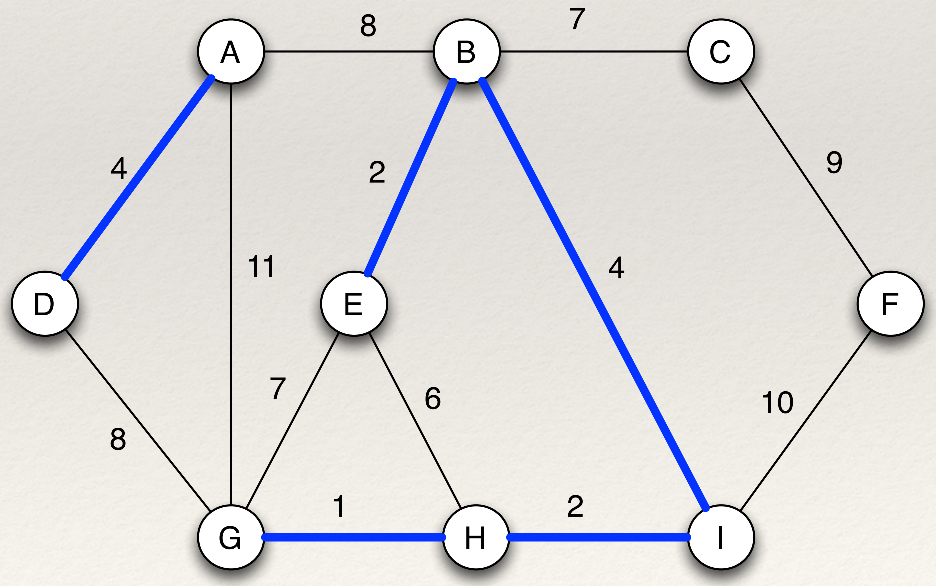
- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- ω{B,I}=4
- ω{E,H}=6
- ω{B,C}=7
- ω{E,G}=7
- ω{A,B}=8
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11



Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
 Fin pour
```

- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- **ω{B,I}=4**
- ω{E,H}=6
- ω{B,C}=7
- ω{E,G}=7
- ω{A,B}=8
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11

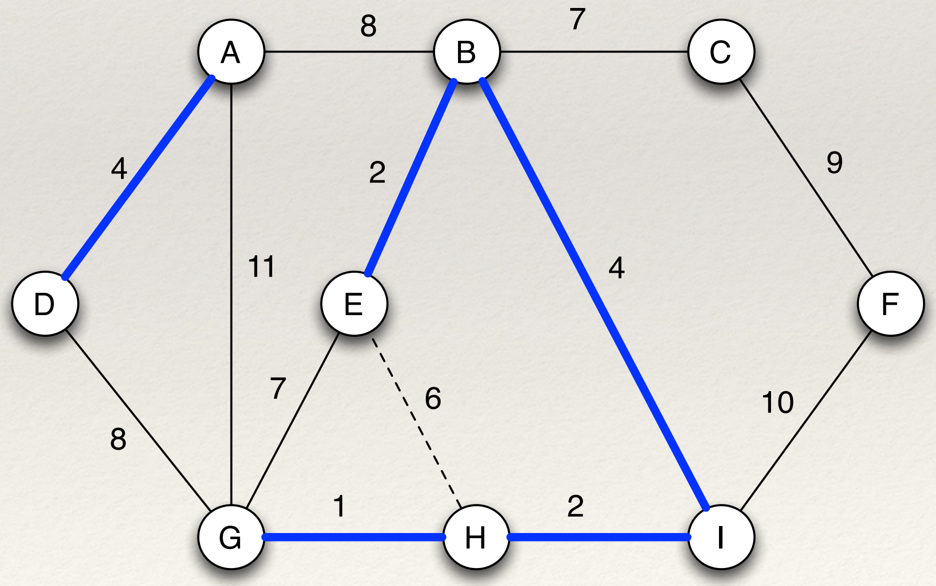


Exemple de Kruskal

ACPM-KRUSKAL(G,w,s)

1. E ensemble des arêtes de G triées dans l'ordre croissant
2. F = {}; ensemble des arêtes de l'arbre couvrant
3. Pour chaque élément e de E et tant que |F|<|V|-1 faire
4. Si F ∪ e ne forme pas de cycle alors
5. F = F ∪ e;
6. Fin Si
7. Fin pour

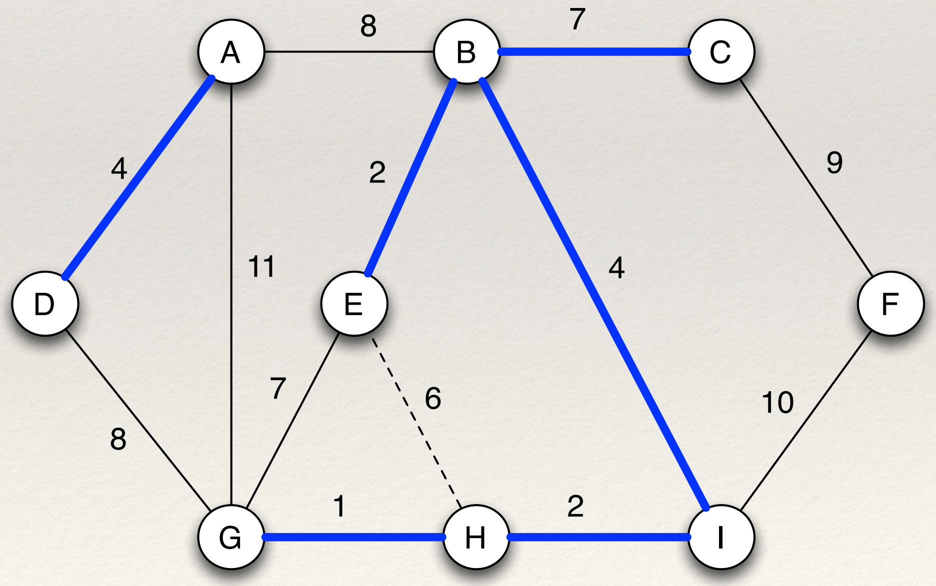
- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- **ω{B,I}=4**
- ω{E,H}=6
- ω{B,C}=7
- ω{E,G}=7
- ω{A,B}=8
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11



Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
Fin pour
```

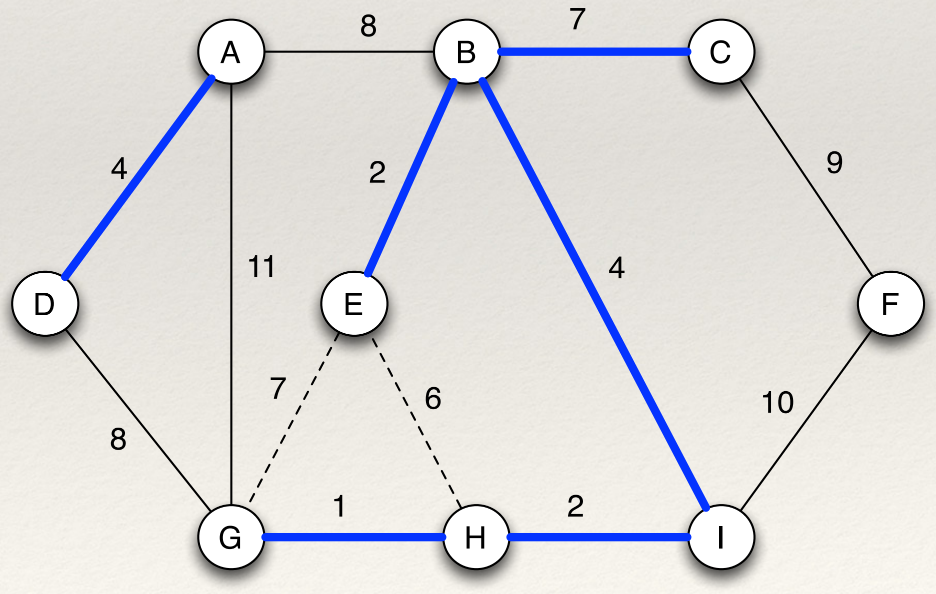
- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- **ω{B,I}=4**
- ω{E,H}=6
- ω{B,C}=7
- ω{E,G}=7
- ω{A,B}=8
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11



Exemple de Kruskal

ACPM-KRUSKAL(G,w,s)
1. E ensemble des arêtes de G triées dans l'ordre croissant
2. F = {}; ensemble des arêtes de l'arbre couvrant
3. Pour chaque élément e de E et tant que |F|<|V|-1 faire
4. Si F ∪ e ne forme pas de cycle alors
5. F = F ∪ e;
6. Fin Si
7. Fin pour

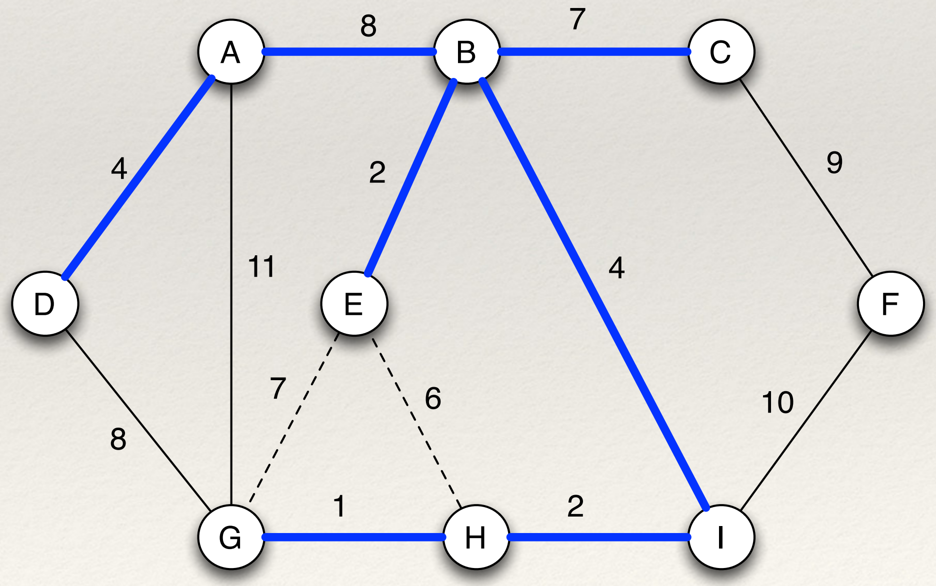
- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- **ω{B,I}=4**
- ω{E,H}=6
- **ω{B,C}=7**
- ω{E,G}=7
- ω{A,B}=8
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11



Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
Fin pour
```

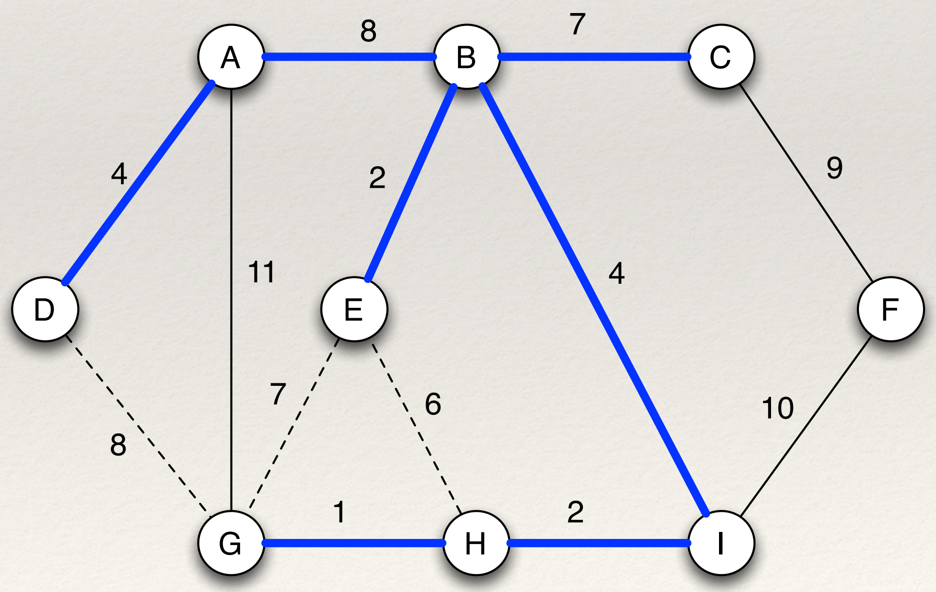
- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- **ω{B,I}=4**
- ω{E,H}=6
- **ω{B,C}=7**
- ω{E,G}=7
- **ω{A,B}=8**
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11



Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
         F = F ∪ e;
    Fin Si
 Fin pour
```

- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- **ω{B,I}=4**
- ω{E,H}=6
- **ω{B,C}=7**
- ω{E,G}=7
- **ω{A,B}=8**
- ω{D,G}=8
- ω{C,F}=9
- ω{F,I}=10
- ω{A,G}=11

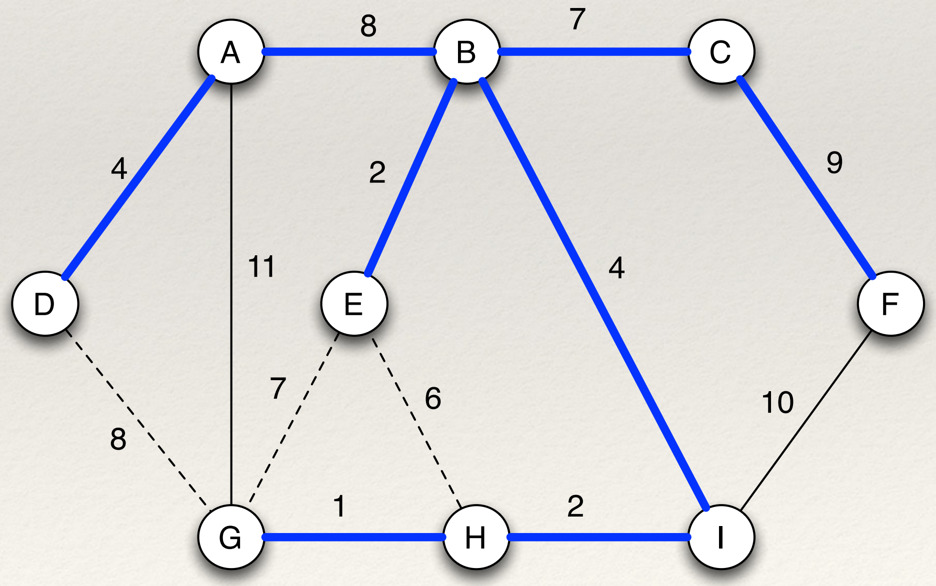


Exemple de Kruskal

```
ACPM-KRUSKAL(G,w,s)
E  ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
   Si F ∪ e ne forme pas de cycle alors
          F = F ∪ e;
    Fin Si
Fin pour
```

- **ω{G,H}=1**
- **ω{B,E}=2**
- **ω{H,I}=2**
- **ω{A,D}=4**
- **ω{B,I}=4**
- ω{E,H}=6
- **ω{B,C}=7**
- ω{E,G}=7
- **ω{A,B}=8**
- ω{D,G}=8
- **ω{C,F}=9**
- ω{F,I}=10
- ω{A,G}=11

Fin car |F| = |V| -1



Théorème

L'algorithme de Kruskal calcule en temps $O(|E| \log |E|)$ une forêt couvrante maximale de poids minimum.

Preuve

- Remarquons tout d'abord que l'algorithme de Kruskal délivre:
  - une forêt couvrante maximale :
- Le graphe T=(S,F) retourné par l'algorithme est:
  - Par construction une forêt couvrante,
  - Puisque l'on ajoute des arêtes de E
  - Sans jamais créer de cycle.

Preuve de correction

- On peut l'obtenir directement de la preuve de Tarjan
- Kruskal peut être vu comme:
  - L'application de la règle rouge.
  - L'arête de poids maximum d'un cycle n'est pas retenue.

Complexité

- La première phase de l'algorithme
  - Le tri des arêtes
    - Temps $O(|E|\log |E|)$.
  - C'est la phase la plus coûteuse de l'algorithme !

Complexité

- La seconde phase nécessite de pouvoir:
  - Détecter si l'ajout d'une arête crée un cycle.
  - Si ce n'est pas le cas fusionner deux composantes connexes.

Complexité

- Pour fusionner de façon efficace, il suffit de modifier les étiquettes des sommets de la plus petite des 2 composantes.
- Avec une structure de donnée appropriée pour les composantes
  - Par exemple une LISTE de ses sommets,
- Les opérations de mises à jour se font au total en temps $O(|V| \log |V|)$ pour tout l'algorithme.

Conclusion

- Nous avons vu deux algorithmes pour le problème de l'arbre couvrant de poids minimum
- L'algorithme de Prim
  - Maintient une composante connexe
  - Temps: $O(|E| + |V| \times \log |V|)$
- L'algorithme de Kruskal
  - Maintient l'absence de cycle
  - Temps: $O(|E| \log |E|)$