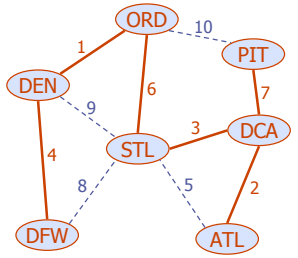
Arbre couvrant minimal d'un graphe

Rappels + définition:

- Un sous-graphe couvrant d'un graphe G est un sous-graphe contenant tous les sommets de G
- Un arbre couvrant d'un graphe est un sous-graphe couvrant qui est un arbre
- Arbre couvrant minimal (minimum spanning tree):
  - Arbre couvrant d'un graphe avec poids dont le poids total des arêtes est minimal

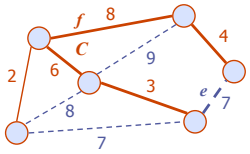


Propriété de cycles des ACM:

Propriété de cycles:

- Soit T un arbre couvrant d'un graphe avec poids G
- Soit e une arête de G n'appartenant pas à T et soit C, le cycle obtenu lorsqu'on ajoute e à T
- Si T est minimal, alors on a que pour toutes arêtes f dans C :

$$poids(f) \leq poids(e)$$

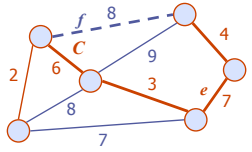


Remplacer f par e nous donne un arbre couvrant de plus petit poids total ...

Preuve:

- Par contradiction.

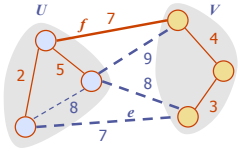
  Si poids(f) > poids(e), on obtient un arbre couvrant de plus petit poids en remplaçant l'arête f par l'arête e dans notre arbre T



Propriété de partition des ACM:

Propriété de partition:

- Considérons une partition des sommets de G en deux ensembles U et V
- Soit e une arête de poids minimal entre U et V
- Alors, il existe un arbre couvrant minimal de G contenant e



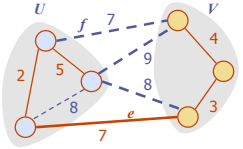
Preuve:

- Soit T un arbre couvrant minimal de G
- Si T ne contient pas e, soit C le cycle formé par l'addition de e à l'arbre T et soit f, une arête entre U et V
- Par la propriété de cycles, on a que

$$poids(f) \leq poids(e)$$

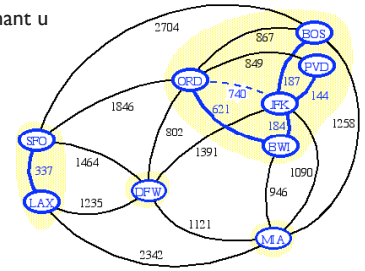
- Comme on avait pris e de poids minimal, on a que poids(f) = poids(e) et alors on obtient un autre ACM en remplaçant f par e

Remplacer f par e nous donne un autre ACM



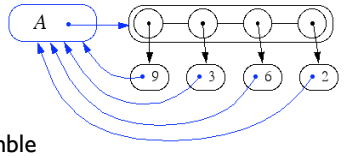
Structure de données pour Kruskal:

- L'algorithme maintient une forêt d'arbres
- Une arête est acceptée, si elle relit deux arbres distincts
- On a besoin d'une structure de données qui maintient une partition i.e une collection d'ensembles disjoints, avec les opérations
  - Trouver(u): retourne l'ensemble contenant u
  - Union(u,v): remplace les ensembles contenant u et v par leur union



Représentation d'une partition:

- Chaque élément d'un ensemble est mis en mémoire dans une séquence (l'ensemble pointe vers la séquence contenant ces éléements)



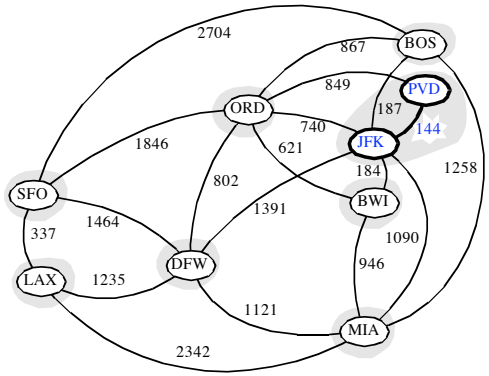
- Chaque élément à un pointeur vers l'ensemble
  - L'opération trouver(u) se fait en $O(1)$ et retourne l'ensemble dont u fait partie
  - Pour l'opération union(u,v), on bouge les éléments du plus petit ensemble dans la séquence du plus grand ensemble et on met à jour leur pointeur
  - La complexité en temps de union(u,v), est $\min(n_u, n_v)$ où n_u est la taille de l'ensemble contenant u et n_v, la taille de l'ensemble contenant v

Kruskal-partition:

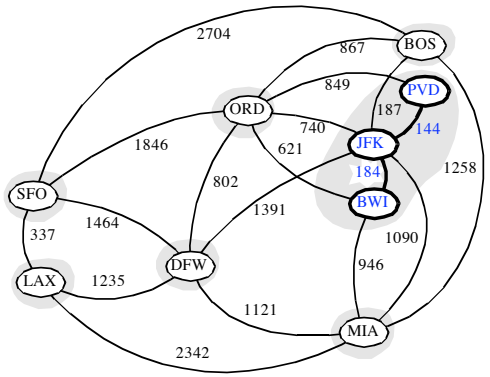
Algorithme Kruskal(*G*):
  Entrée: un graphe avec poids ***G***.
  Sortie: Un ACM ***T*** pour ***G***.
Soit ***P*** une partition des sommets de ***G***, où chaque sommet est dans un ensemble séparé.
Soit ***Q*** une liste avec priorités gardant en mémoire les arêtes de ***G***, ordonnées selon leur poids
Soit ***T*** un arbre initialement vide
Tant que ***Q*** n'est pas vide **faire**
  (***u***,***v***) ← ***Q***.enleverMin()
  si ***P***.trouver(***u***) != ***P***.trouver(***v***) **alors**
    Ajouter (***u***,***v***) à ***T***
    P.union(***u***,***v***)
retourner ***T***

Complexité en temps: O((n+m)log n)

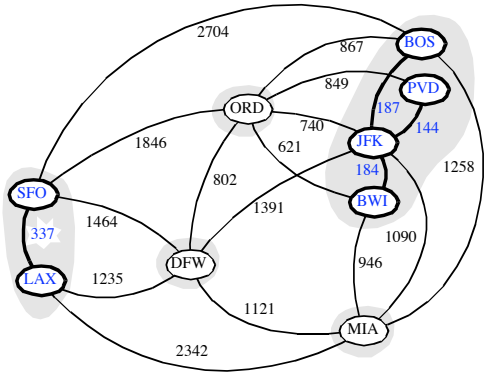
Exemple de Kruskal:



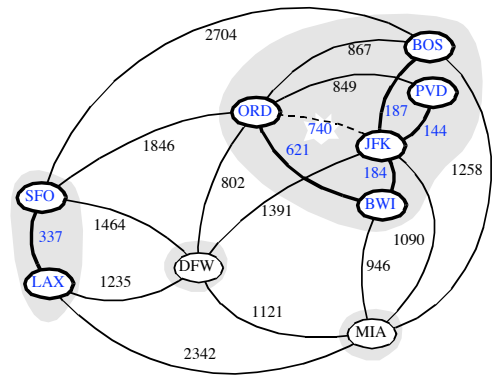
Exemple de Kruskal (suite):



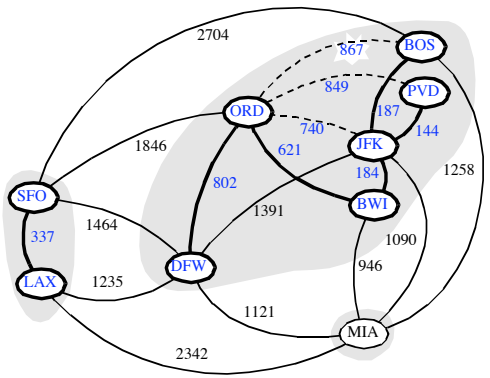
Exemple de Kruskal (suite):



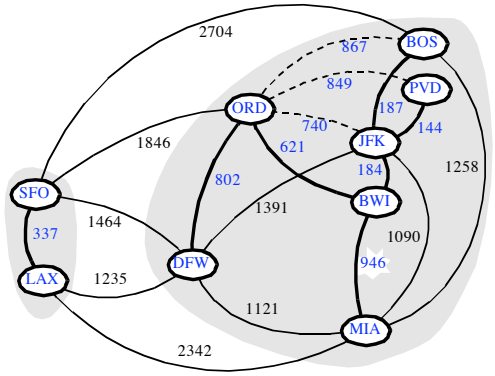
Exemple de Kruskal (suite):



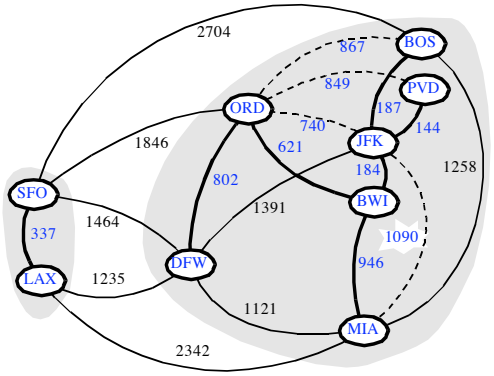
Exemple de Kruskal (suite):



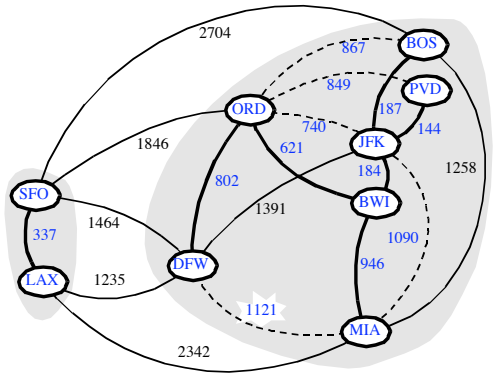
Exemple de Kruskal (suite):



Exemple de Kruskal (suite):



Exemple de Kruskal (suite):



Exemple de Kruskal (suite):