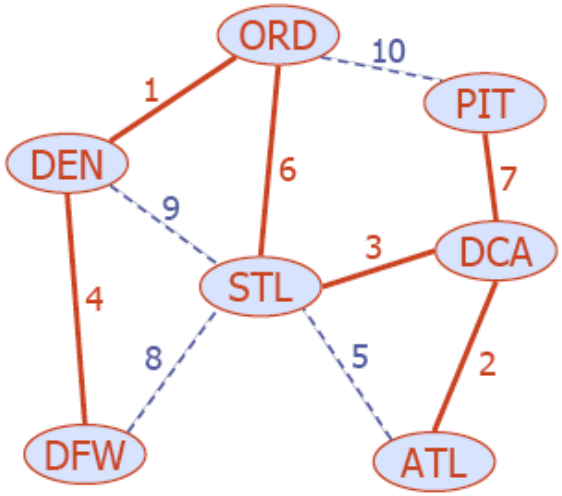
Arbre couvrant minimal d'un graphe

Rappels + définition:

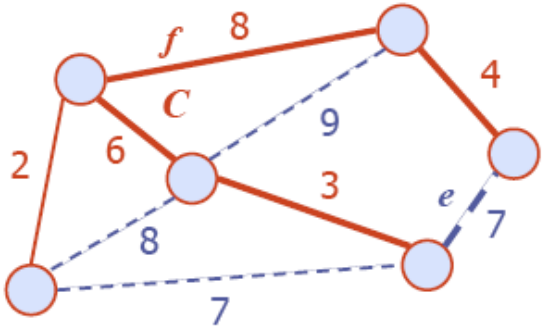
- Un sous-graphe couvrant d'un graphe G est un sous-graphe contenant tous les sommets de G
- Un arbre couvrant d'un graphe est un sousgraphe couvrant qui est un arbre
- Arbre couvrant minimal (minimum spanning tree):
  - Arbre couvrant d'un graphe avec poids dont le poids total des arêtes est minimal



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 :

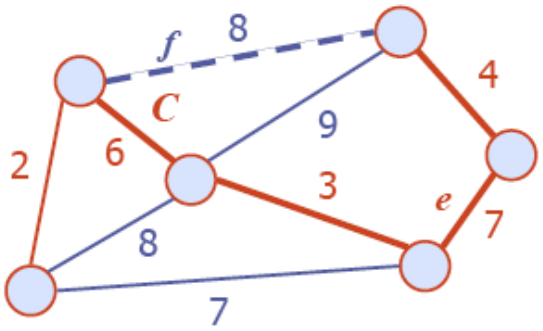
$$poids(f) \leq poids(e)$$



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

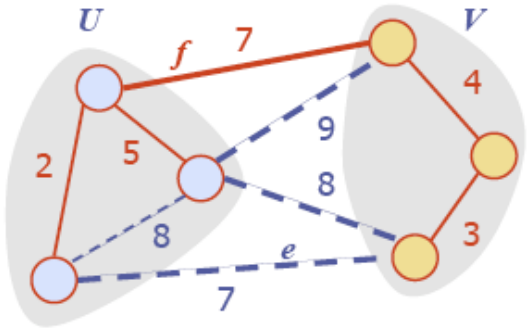
- Preuve:
  - Par contradiction.

    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

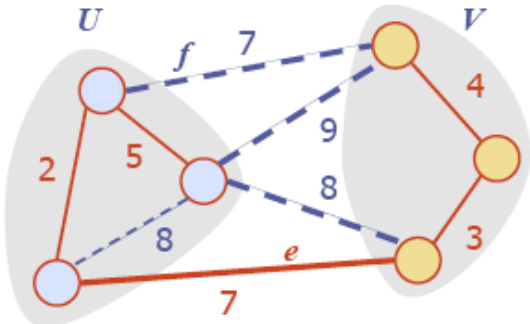


Remplacer f par e nous donne un autre ACM

- 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

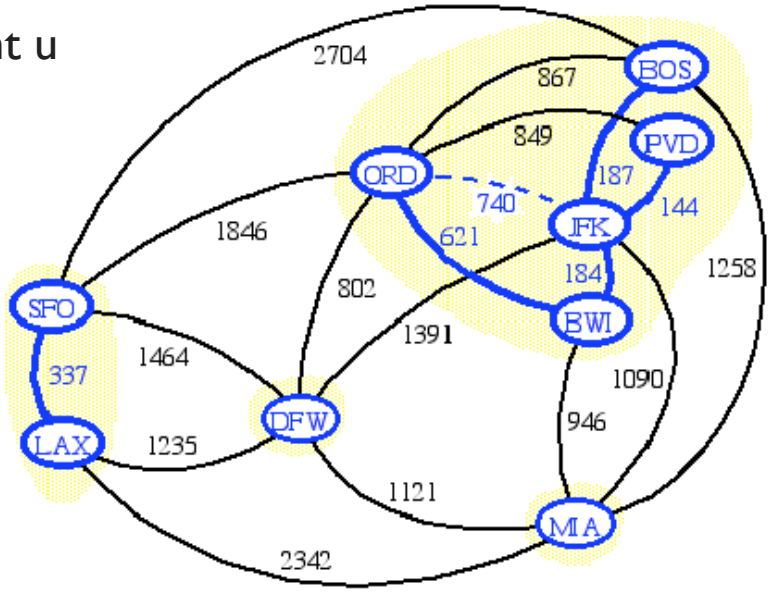
$$poids(f) \leq poids(e)$$

  - 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



Algorithme 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



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**

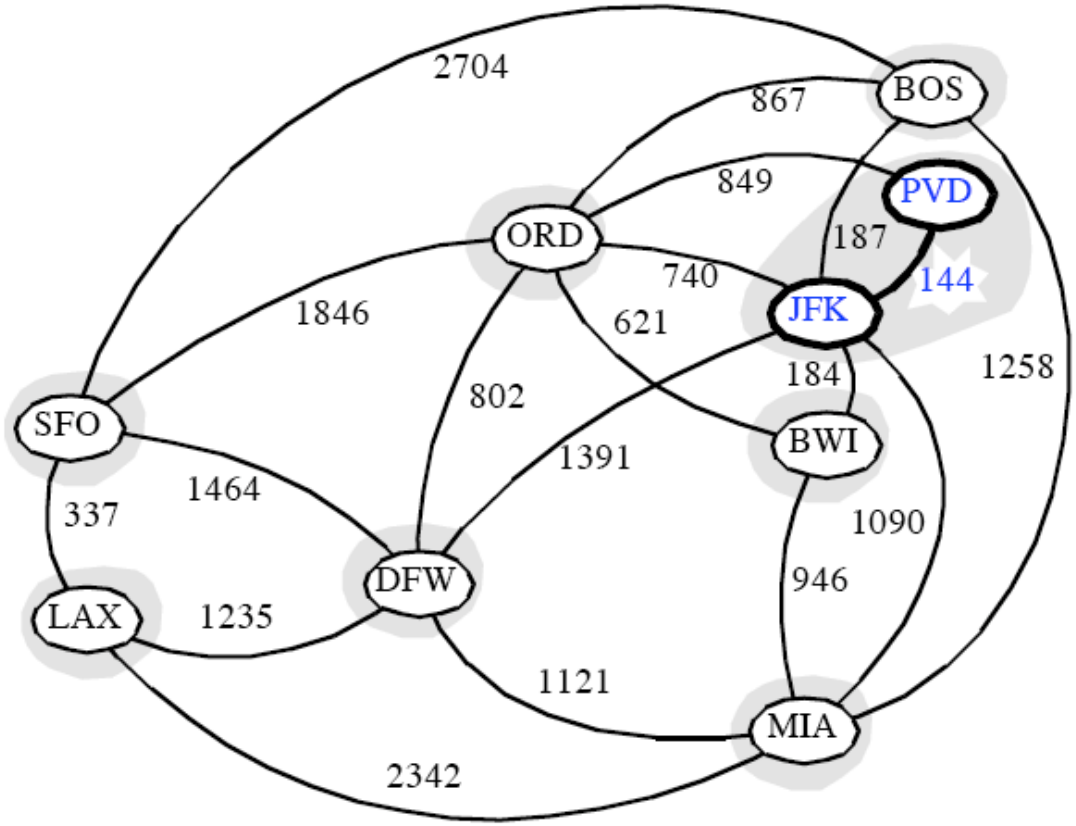
    Ajouter (*u*,*v*) à *T*

    P.union(*u*,*v*)

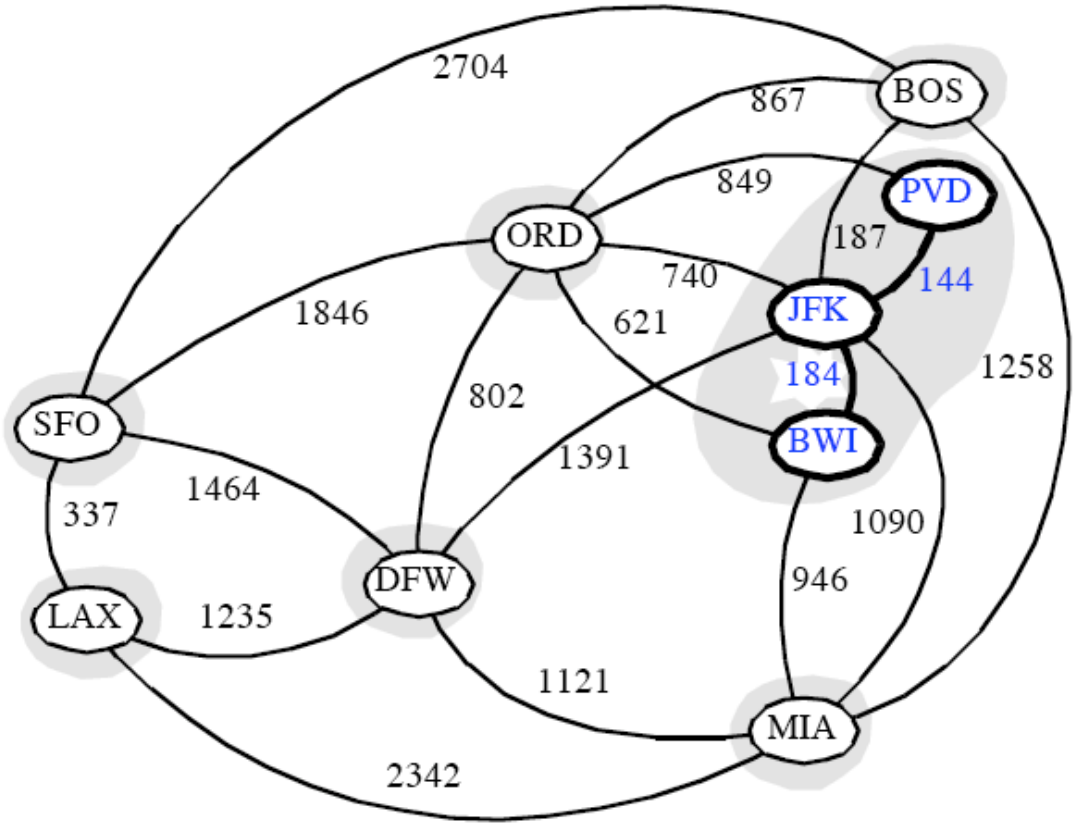
retourner *T*

Complexité en temps:
O((n+m)log n)

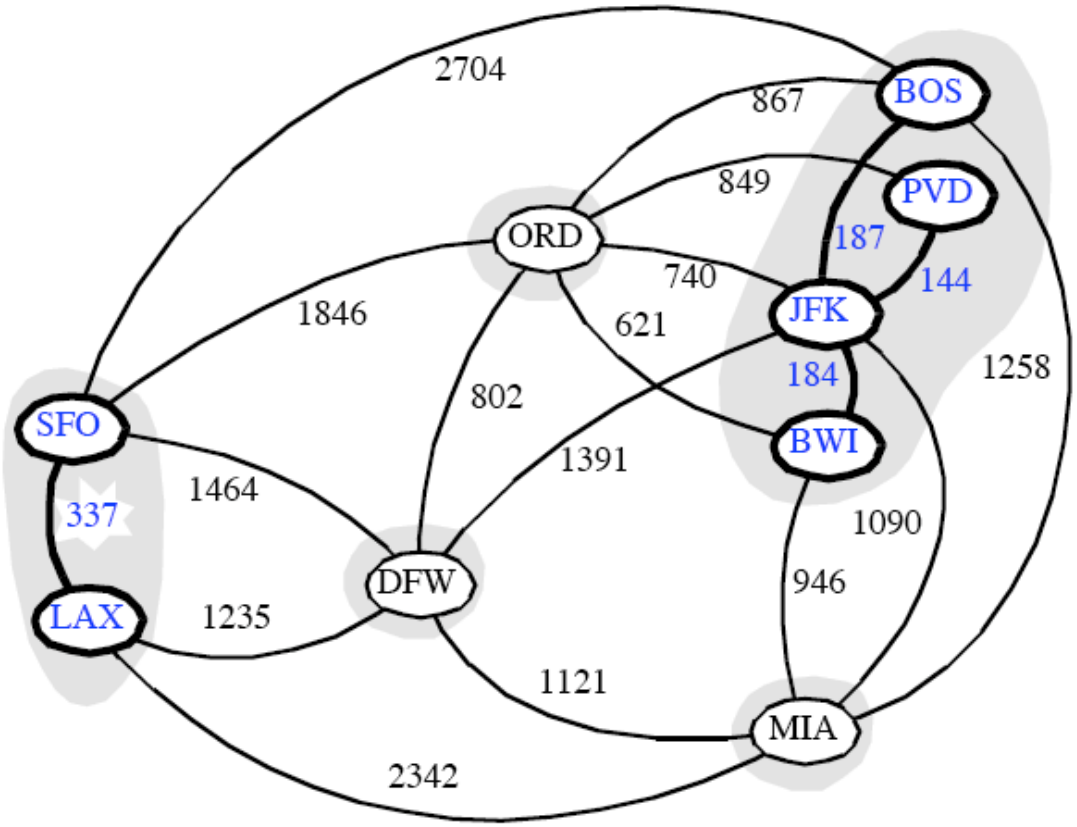
Exemple de Kruskal:



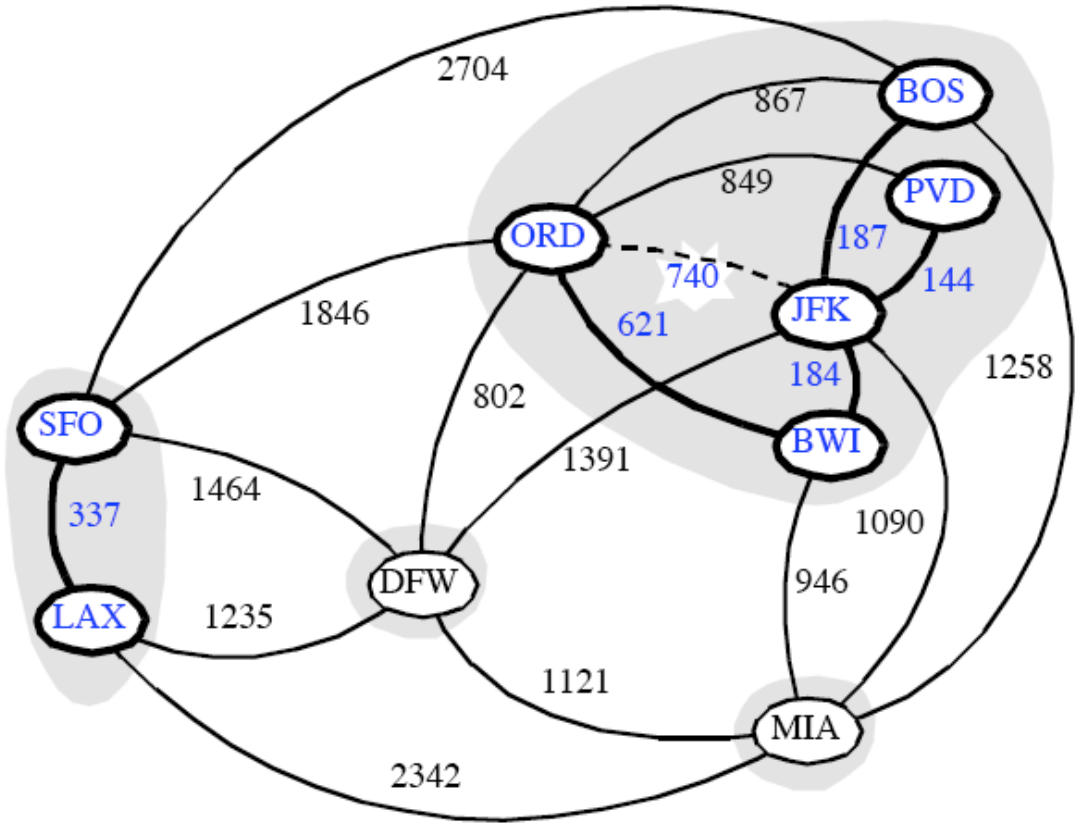
Exemple de Kruskal:



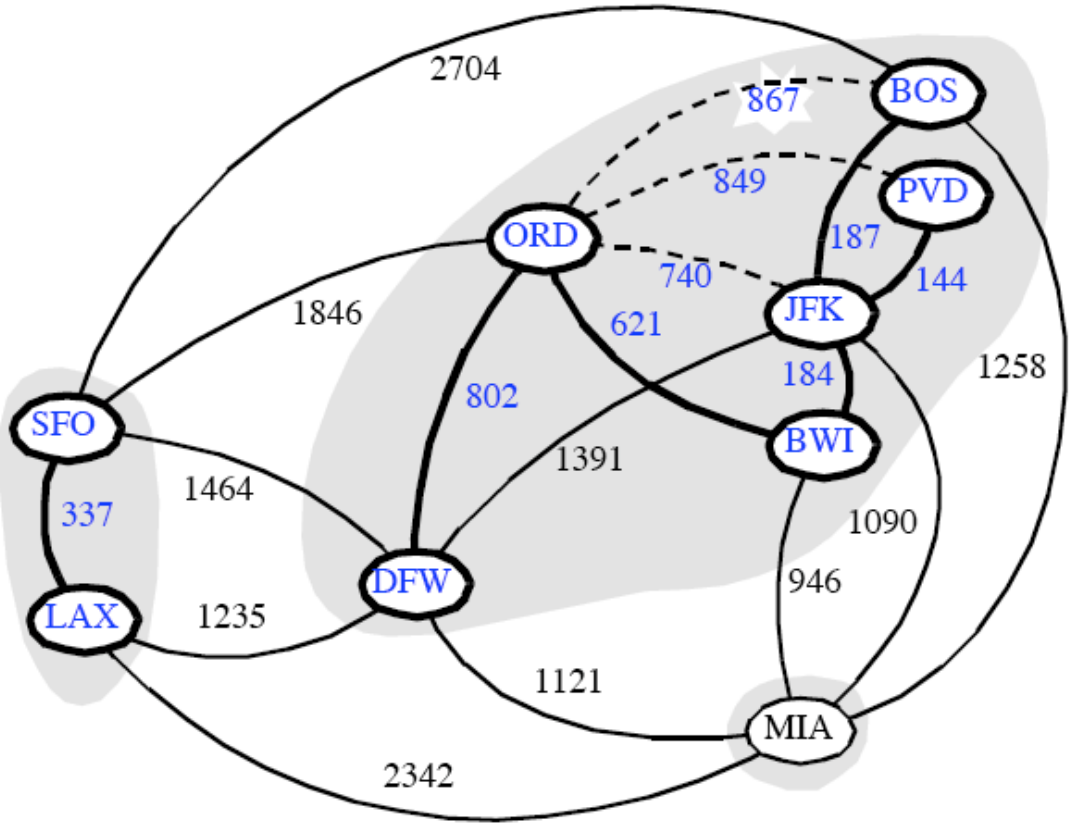
Exemple de Kruskal:



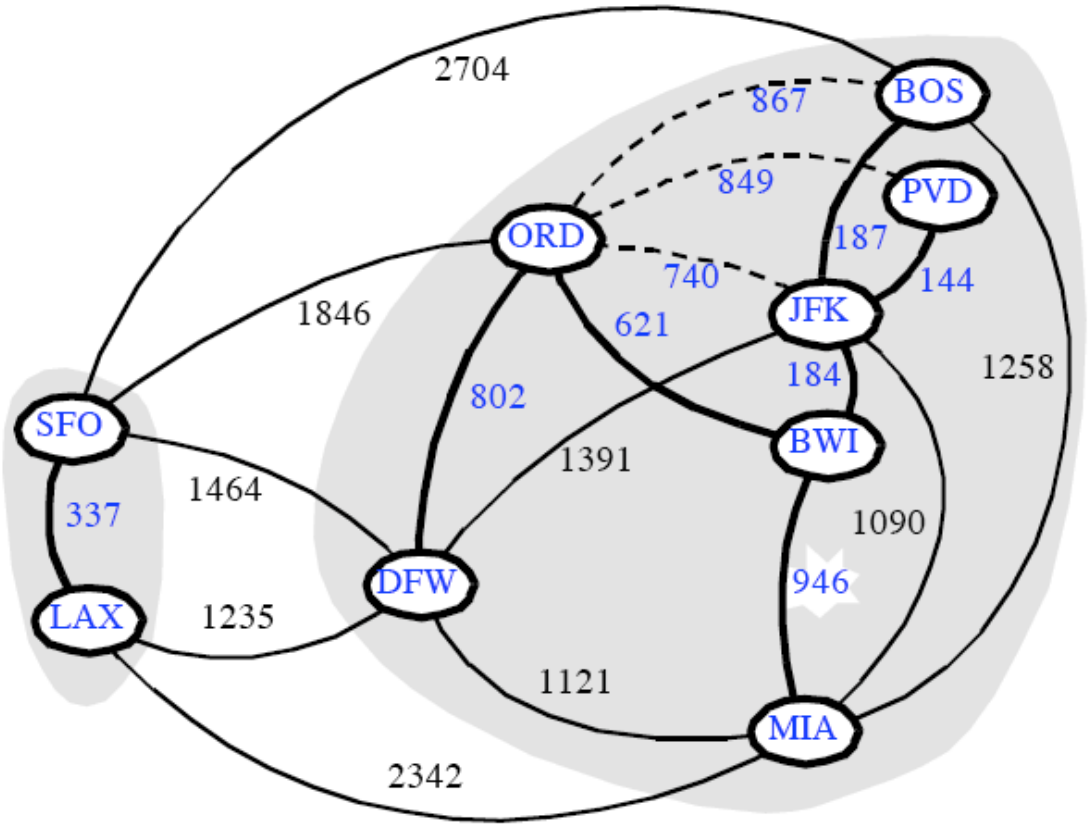
Exemple de Kruskal:



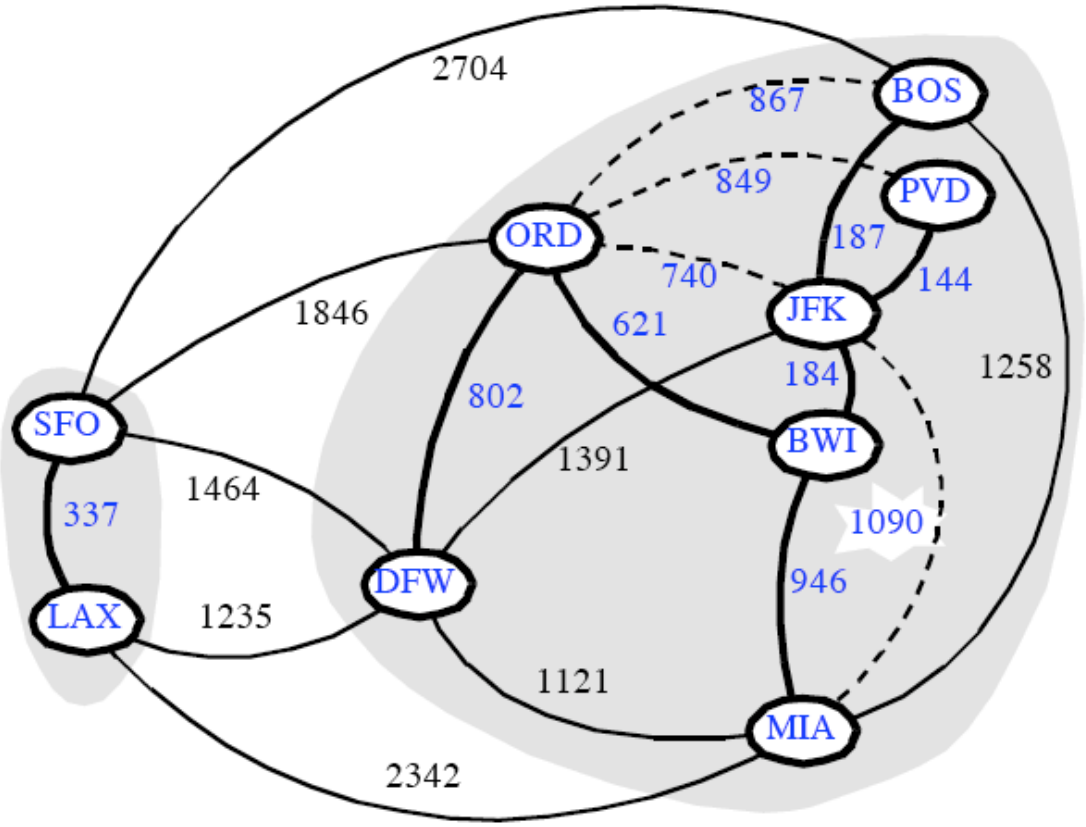
Exemple de Kruskal:



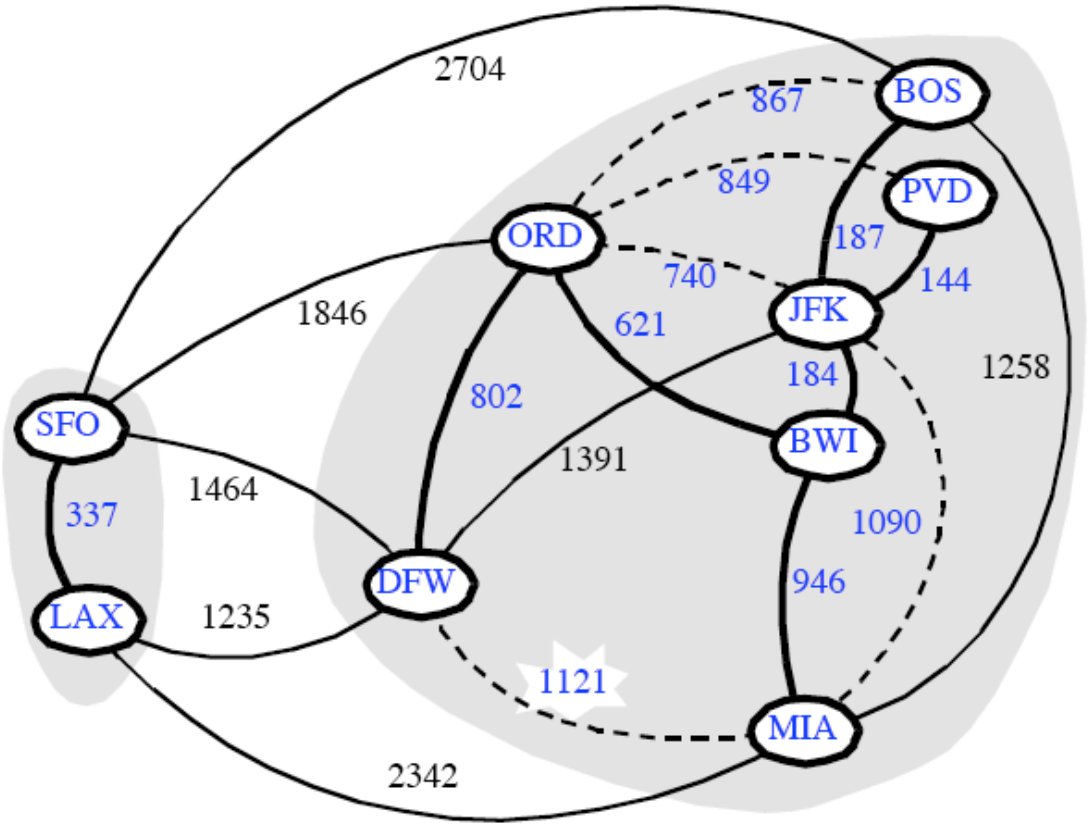
Exemple de Kruskal:



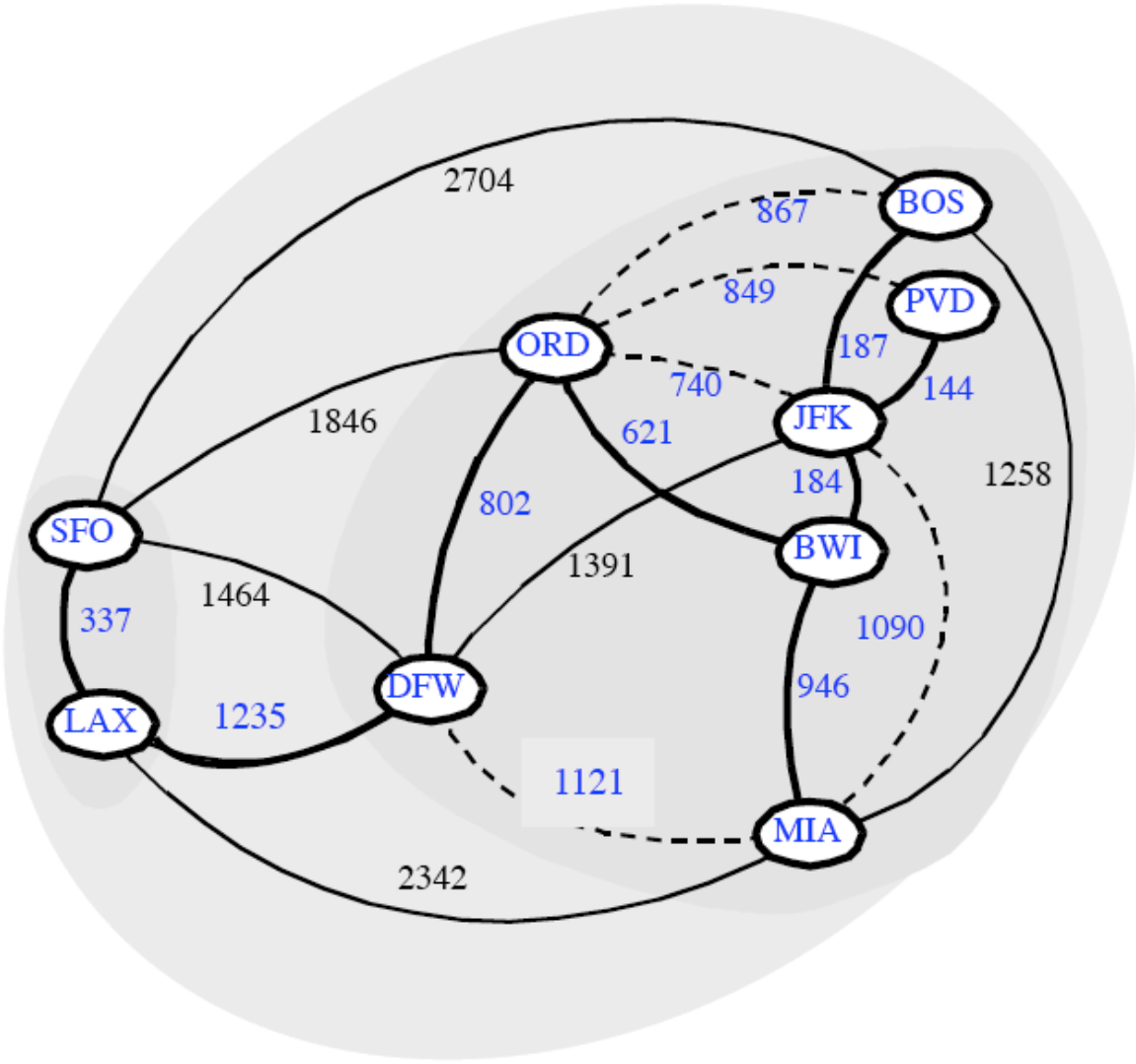
Exemple de Kruskal:



Exemple de Kruskal:



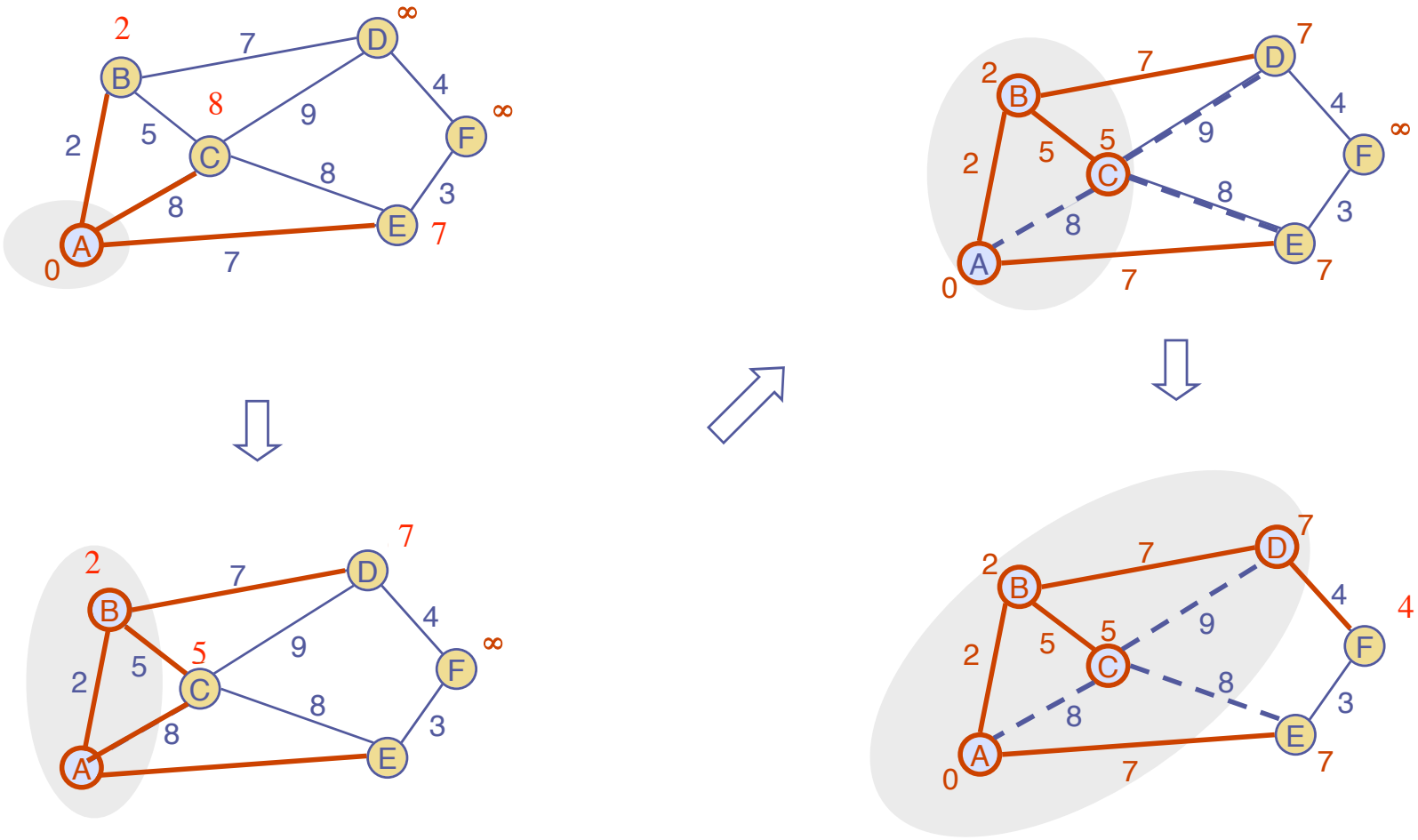
Exemple de Kruskal:



Algorithme de Prim-Jarník

- Algorithme similaire à l''algorithme de Dijkstra (dans le cas des graphes connexes)
- On choisit un sommet s aléatoirement qu'on met dans un "nuage" et on construit l'arbre couvrant minimal en faisant grossir le "nuage" d'un sommet à la fois.
- On garde en mémoire à chaque sommet v, une étiquette d(v) qui ici est égale au poids minimal parmi les poids des arêtes reliant v à un sommet à l'intérieur du nuage.
- À chaque étape:
  - On ajoute au nuage le sommet u extérieur ayant la plus petite étiquette d(u)
  - On met à jour les étiquettes des sommets adjacents à u

Exemple:



Exemple (suite)