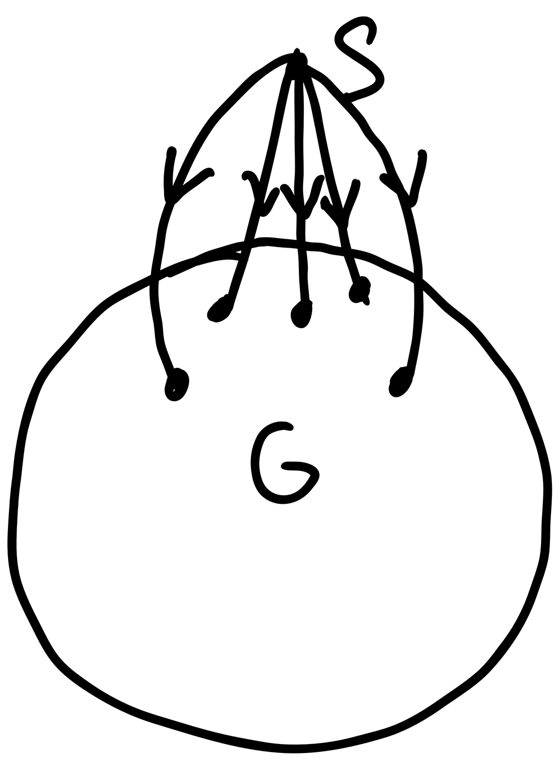
Rpp: $\ell'(u,v) = \ell(u,v) + h(u) - h(v)$

Lemme : P est un plus court chemin de u à v dans G par rapport à ℓ

$\Updownarrow$

P est un plus court chemin de u à v dans G par rapport à ℓ'

Question : Comment trouver une fonction $h : V(G) \to \mathbb{R}$ tq $\nexists$ V negatif par rapport à ℓ'.



Soit G' le graphe obtenu à partir de G en ajoutant un nouveau sommets, et les arcs (s,v) pour tout $v \in V(G)$

Remarque : G' ne contient pas de cycle négatif (on suppose que G n'a pas de cycle nég.)

On définit $h(v) = dist_{G'}(s,v)$

On a $h(v) \leq h(u) + \ell(u,v)$ pour tout $(u,v) \in E(G)$

$$\begin{aligned}\ell'(u,v) &= \ell(u,v) + h(u) - h(v)\\ &\geq h(v) - h(u) + h(u) - h(v)\\ &= 0\end{aligned}$$

Algorithme de Johnson

1. Calculer G'
2. Appliquer Bellman-Ford à G', avec source s, pour calculer $h(v) = dist_{G'}(s,v)$.
3. Repondérer chaque arc $(u,v) \in E(G)$ par $\ell'(u,v) = \ell(u,v) + h(u) - h(v)$
4. Pour chaque $u \in V(G)$, exécuter Dijkstra pour calculer $dist_{\ell'}(u,v)$ pour tout $v \in V(G)$
5. Pour chaque paire $u, v \in V(G)$, on a $dist_{\ell}(u,v) = dist_{\ell'}(u,v) + h(v) - h(u)$

Complexité de l'algo de Johnson

1. $O(n)$

2. $O(mn)$

3. $O(m)$

4. Dijkstra : $O(m + n\log n)$ n fois donc $O(mn + n^2\log n)$

5. $O(n^2)$

Donc, la complexité de l'algo de Johnson est de $O(mn + n^2\log n)$.
Si le graphe est dense ($\Omega(n^2)$ arêtes) alors la complexité revient à $O(n^3)$.
Par contre, pour les graphes peu denses. Johnson est plus efficace que Floyd-Warshall

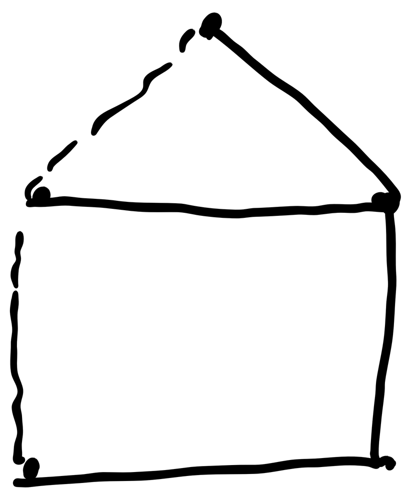
ARBRES COUVRATS DE POIDS MINIMUM

Dans ce chapitre, on revient aux graphes non orientés ! (pondérés)

Rappel : Un graphe G est un arbre ssi G est connexe et acyclique.

Rappel : Un graphe est connexe ssi il contient un arbre couvrant

Exemple



Problème : Étant donné un graphe pondéré G, trouver un arbre couvrant de G de poids minimum.

Plus précisément, soit $G = (V, E)$ un graphe ~~pondéré tq~~ connexe, avec pondération $w \in \mathbb{R}^{|E|}$ des arêtes. Trouver un arbre couvrant T de G tq $W(T) = \sum_{e \in E(T)} w(e)$ est minimum.

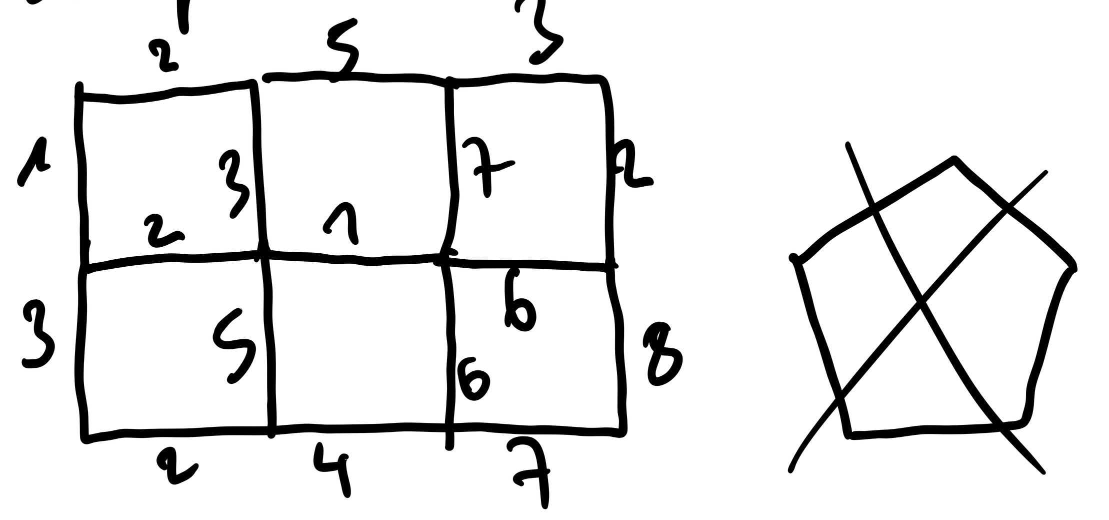
On verra deux algorithmes != pour résoudre ce problème.

- L'algo de Kruskal
- L'algo de Prim

Il s'agit d'algorithmes "gloutons" - l'idée est de construire une solution en ajoutant les arêtes une par une, toujours prenant celle qui donne les meilleurs bénéfices.

L'essence de l'algo de Kruskal
Ajouter l'arête la plus légère qui ne crée pas de cycle

Exemple



L'algorithme de Kruskal

Entrées : Graphe connexe $G = (V; E)$ avec pondération $w \in \mathbb{R}^{|E|}$

Sortie : Ensemble d'arêtes $X \subseteq E$ d'un arbre couvrant de G.

$X \leftarrow \emptyset$

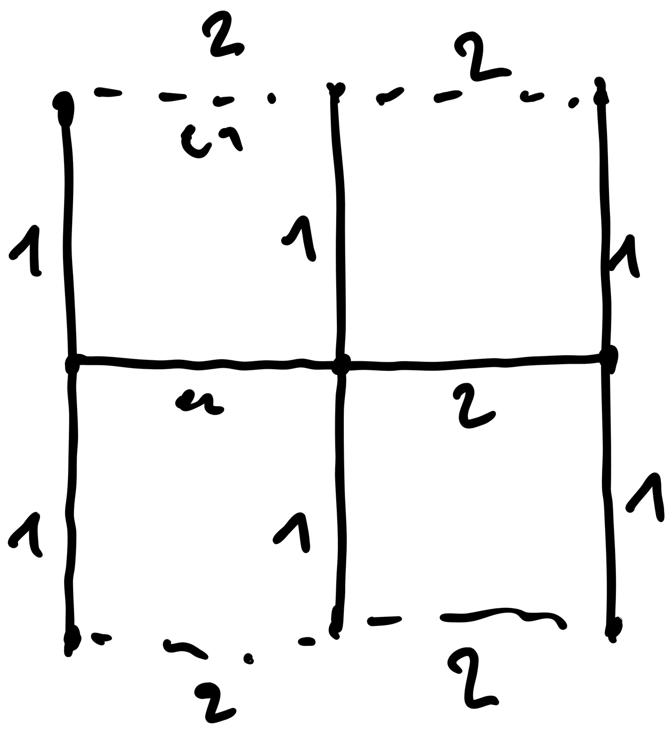
Trier les arêtes E par poids croissant pour tous les $e \in E$ (dans l'ordre) :
    si $(V, X \cup \{e\})$ est acyclique :
        $X \leftarrow X \cup \{e\}$

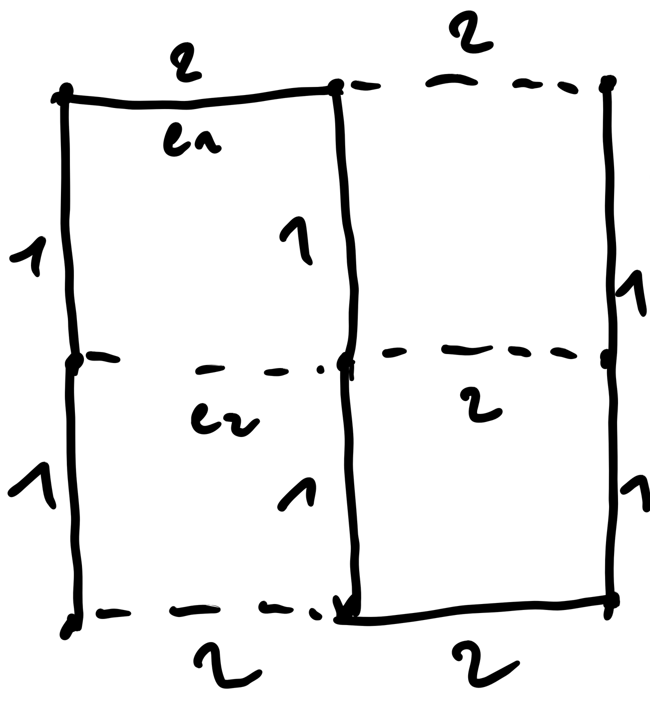
Correction de l'algorithme.
L'algo s'arrête au bout d'au plus $|E|$ itérations.
Fini

Soit $X \subseteq E$ la sortie de l'algo. $T = (V, X)$ est sous-graphe couvrant de G.
Supposons par l'absurde que T n'est pas connexe.
Comme G est connexe, il doit y avoir une arête de G qui relie les deux

composantes connexes de T.

Soit e une telle arête de poids minimum.
Comme $T \cup \{e\}$ est acyclique, l'algo de Kruskal aurait ~~été~~ ajouté e à X à une certaine étape. Contradiction.
Donc, T est connexe.. De plus, T est acyclique par la condition dans l'algo.
Donc, T est un arbre couvrant de G.
Il reste à montrer que T est un arbre couvrant de poids minimum.
Soit $T_0 = (V, X_0)$ un arbre couvrant de poids minimum qui maximise $|X \cap X_0|$.
C à d, T_0 a le plus d'arêtes en commun avec T. Nous allons prouver que $T = T_0$.


T


T_0

Supposons par l'absurde que $|X \cap X_0| < |X|$, et soit e_1 l'arête la plus légère dans $X_0 \setminus X$.

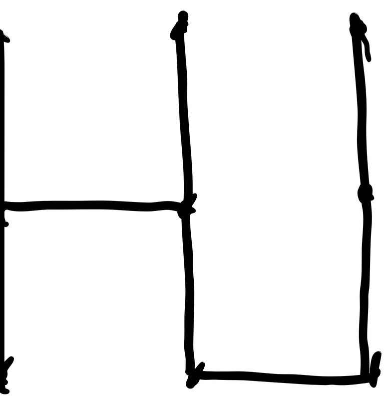
Ce graphe $T \cup \{e_1\}$ contient un cycle C. Comme T_0 est acyclique au moins une arête e_2 de $C \not\in T_0$.

Si $w(e_1) < w(e_2)$, l'algo de Kruskal aurait choisi l'arête e_1 avant l'arête e_2.

Donc, $w(e_1) \geqslant w(e_2) \Rightarrow w(e_1) - w(e_2) \geqslant 0$

Soit $X_1 = (X_0 \setminus \{e_1\}) \cup \{e_2\}$

Comme $w(X_1) = \sum_{e \in X_1} w(e) = \underbrace{\left(\sum_{e \in X_0} w(e) \right) - w(e_1) + w(e_2)}_{\leqslant w X_0}$



$T_1(V, X_1)$

T_1 est donc un ACM avec plus d'arêtes en commun avec T que T_1 :

$|X_1 \cap X| = |X_0 \cap X| + 1$ Contradiction avec le choix de $T_0 = (V_1, X_0)$.

Donc, $|X \cap X_0| = |X|$. Autrement dit $X = X_0$

$T = (V, X)$ est donc bien un ACM. ■

Question : Comment déterminer si $(V, X \cup \{e\})$ est acyclique ?
Une manière naïve de le faire est d'appliquer BFS ou DFS - complexité $O(n+m)$

Il faut le faire m fois, donc complexité $O(nm+m^2)$.
Pour les graphes denses : $O(n^4)$