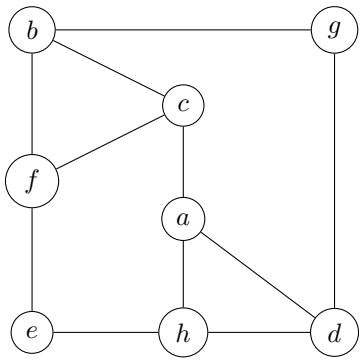
Tutorial 9 : Spanning trees

Graph theory, 1st semester.

2022

Exercise 1 — *Spanning tree and cycle basis*

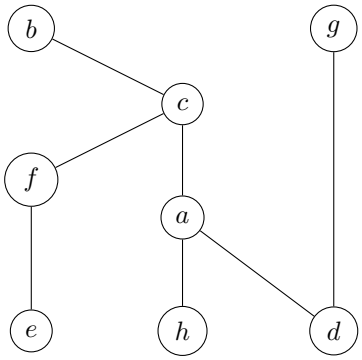
Let G be the following graph :



1. Build a spanning tree T of G such that the associated cycle basis is the set of finite faces of G.

▶ **Correction**

— L'arbre suivant convient :



Exercise 2 — *Build a telecommunication network*

A bank wants to build a telecommunication network linking its main agency, situated at the center of Paris, at *Bourse*, and seven of its secondary agencies. The cost needed to connect two agencies is given in the following array :

	B	O	E	R	SL	L	N
Bourse							
Opera	5						
Etoile	18	17					
République	9	11	27				
St-Lazare	13	7	23	20			
Louvre	7	12	15	15	15		
Neuilly	38	38	20	40	40	35	
Chatelet	22	15	25	25	30	10	45

Model this problem with a graph optimization problem and solve it.

▶ **Correction**

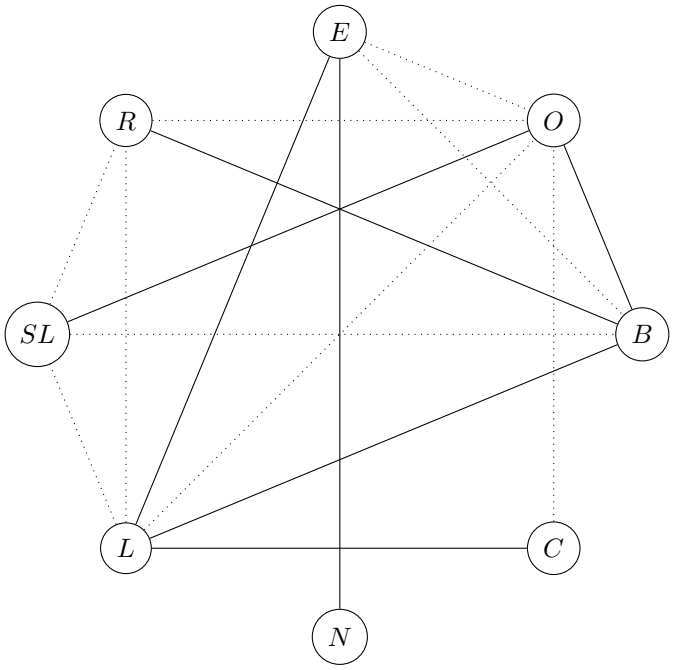
Il peut se modéliser avec un problème d'arbre couvrant de poids minimum. En utilisant l'algorithme de Kruskal (ou de Prim), on peut le résoudre.

Kruskal :

On trie les arêtes par poids :

— $(O, B), 5$
— $(SL, O), 7$
— $(L, B), 7$
— $(R, B), 9$
— $(C, L), 10$
— $(R, O), 11$
— $(L, O), 12$
— $(SL, B), 13$
— $(L, E), 15$
— $(L, R), 15$
— $(L, SL), 15$
— $(C, O), 15$
— $(E, O), 17$
— $(E, B), 18$
— $(Sl, R), 20$
— $(N, E), 20$
— $(C, B), 22$
— $(SL, E), 23$
— $(C, E), 25$
— $(C, R), 25$
— $(R, E), 27$
— $(C, SL), 30$
— $(N, L), 35$
— $(N, B), 38$
— $(N, O), 38$
— $(N, R), 40$
— $(N, SL), 40$
— $(C, N), 45$

On ajoute les arêtes une par une tant qu'elles ne produisent pas de cycle.



Exercise 3 — *Some tree properties*

1. Show that any tree with two nodes has at least two pendant nodes (with degree 1).

▶ **Correction**

Soit G un graphe connexe avec 0 sommet de degré 1. Alors ils sont tous de degré au moins 2. Si G a exactement 1 sommet de degré 1, alors tous les autres sont de degré au moins 2, et un des autres nœuds est de degré 3 (sinon le nombre de nœuds de degré impair serait impair). La somme des degrés vaut alors au moins $2n$, donc le nombre d'arêtes vaut au moins n ce qui exclu le cas où G est un arbre.

2. Show that any connected graph has two non-articulation nodes.

▶ **Correction**

Supposons que tous les nœuds d'un graphe G soient des points d'articulation : si on retire un de ces sommets alors le graphe se déconnecte. Aucun sommet n'appartient à un cycle, donc le graphe est un arbre. Or dans un arbre, retirer les feuilles ne déconnecte pas le graphe, donc tous les nœuds ne sont pas des points d'articulation. L'idée est similaire pour le cas où un seul nœud n'est pas point d'articulation.

3. Give a graph with only two non-articulation nodes.

▶ **Correction**

Le chemin P_3 est un exemple simple.

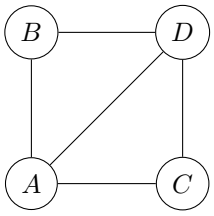
Exercise 4 — *Connectivity of the spanning trees graph*

Let $G = (V, E)$ be a undirected graph and $\mathcal{T}$ be the set of spanning trees of G. Let $H = (\mathcal{T}, E_H)$ be the graph where

— the nodes are spanning trees of G

— An edge links two nodes of H corresponding to spanning trees T_1 and T_2 of G if and only if all the edges of T_1 and T_2 are the same except for one.

1. Draw H when G is the following graph :



► **Correction**

Le graphe H possède 8 nœuds :

— les chemins A, B, D, C ; B, D, C, A ; D, C, A, B ; C, A, B, D ; B, D, A, C et B, A, D, C

— Les deux étoiles enracinées en A et D.

On relie deux nœuds si les arbres correspondent à des arbres avec 2 arêtes en commun. Par exemple A, B, D, C et B, D, C, A mais pas A, B, D, C et B, D, A, C.

2. Show that H is connected.

► **Correction**

On démontre que deux arbres T_1 et T_2 couvrant sont toujours reliés par un chemin dans H. On le fait par récurrence sur le nombre d'arêtes en commun. Deux arbres avec les mêmes arêtes sont reliés puisque c'est le même nœud de H. On suppose que c'est vrai s'ils ont k arêtes en commun. S'ils ont $k-1$ arêtes en commun, il faut montrer qu'on peut enlever une arête de T_1 et ajouter une arête de T_2 pour qu'ils aient k arêtes en commun et utiliser l'hypothèse de récurrence.

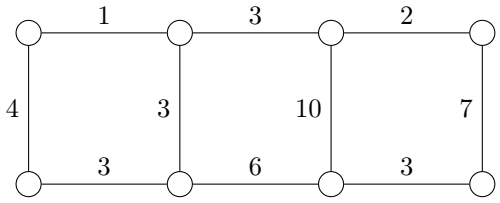
Exercise 5 — *Prim algorithm*

The Prim algorithm finds a minimum spanning tree.

Require: An undirected graph $G = (V, E)$ with weights $\omega : E \to \mathbb{R}^+$.
Ensure: A minimum spanning tree of G
  $T = (V_T, E_T) = (\emptyset, \emptyset)$
  Add an arbitrary node v to V_T
  while $|V_T| \neq |V|$ **do**
    Add to E_T an edge of minimum weight linking $u \in V_T$ to a node $v \in V \backslash V_T$
    Add v to V_T
  return T

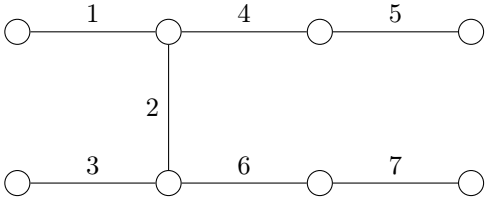
---

1. Apply the algorithm to the following graph :



► **Correction**

On part arbitrairement du nœud en haut à gauche. On ajoute alors les arêtes suivantes, les numéros sont l'ordre dans lequel on les ajoute.



2. Let T_i be the tree T at the beginning of iteration i and e be the edge chosen during that iteration. Let $\mathcal{T}_i$ be the set of spanning trees of G covering T_i. Show that there exists a minimum cost tree of $\mathcal{T}_i$ containing e.

▶ **Correction**

Supposons que tout arbre de poids minimum de $\mathcal{T}_i$ ne contienne pas e. Soit T^* un tel arbre. Soit $f_1, f_2, \ldots, f_p$ les arêtes de T^* sortant de T_i. Or e étant une arête de poids minimum sortant de T_i, on a $\omega(e) \leq \omega(f_j)$. Si on ajoute e à T_i, on forme un cycle, ce cycle passe nécessairement par une arête f_j car e connecte un nœud de T_i et un nœud hors de T_i, ce cycle doit donc posséder une autre arête reliant un nœud de T_i et un nœud hors de T_i. Si on supprime f_j, on obtient un arbre de poids plus petit ou égal ce qui contredit l'hypothèse.

3. Deduce that the Prim algorithm is optimal.

▶ **Correction**

Montrer par récurrence sur i que le sous-arbre construit à l'issue de l'itération i est un sous-arbre d'une solution optimale.