

Les énoncés des travaux pratiques seront disponibles sur internet quelques jours avant la séance correspondante, à l'adresse indiquée ci-dessus. Les questions en rapport avec les TP, Caml Light ou l'informatique sont les bienvenues et peuvent être envoyées à victor.nicollet@ens.fr

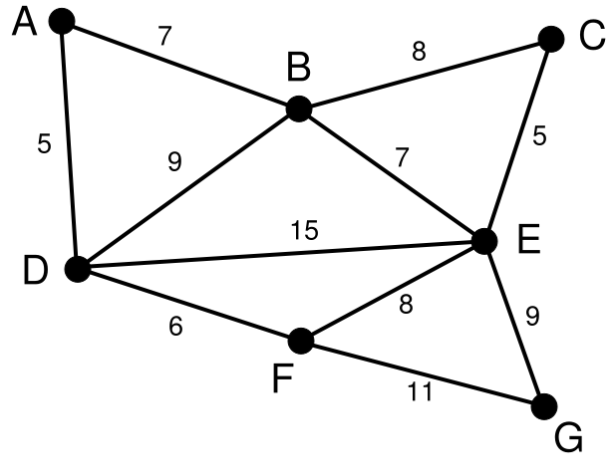
1 DÉFINITIONS

Un *graphe* est un ensemble de *nœuds* pouvant être reliés deux à deux par des *arêtes*. Un *graphe pondéré* est un graphe où une valeur numérique appelée *poids* est associée à chaque arête.

Un *arbre couvrant* d'un graphe est un ensemble d'arêtes de ce graphe tel que, si M et N sont deux nœuds du graphe, alors il existe exactement un chemin constitué d'arêtes de l'arbre joignant M à N .

On appelle *poids* de l'arbre la somme des poids des arêtes qu'il contient. Un *arbre couvrant minimal* est un arbre couvrant dont le poids est minimal parmi tous les arbres couvrants d'un graphe donné.

▷ **Question 1.** À titre d'échauffement, donner un arbre couvrant minimal du graphe ci-contre. Quel est son poids? ◁



2 ALGORITHME DE PRIM

L'algorithme de Prim est un algorithme permettant de construire un arbre couvrant minimal d'un graphe. Pour les besoins de l'exercice, on représentera les nœuds du graphe comme des entiers de $\{0 \dots n - 1\}$ et les arêtes comme des paires d'entiers associées à un poids entier.

Le principe de l'algorithme de Prim est de construire l'arbre couvrant minimal arête par arête: pour ajouter une arête à un arbre partiellement construit, il considère l'ensemble des arêtes dont une extrémité est connectée à l'arbre déjà construit, et l'autre extrémité ne l'est pas, et il choisit dans cet ensemble une arête de poids minimal qu'il ajoute à l'arbre. L'algorithme commence avec un arbre couvrant contenant un nœud et zéro arêtes, et ajoute successivement $n - 1$ arêtes.

▷ **Question 2.** Construire une structure de données permettant de représenter l'ensemble des nœuds qui n'ont pas encore été reliés à l'arbre. On pourra stocker, pour chaque nœud de l'ensemble, le nœud de l'arbre qui en est le plus proche, et la distance correspondante.

Écrire une fonction `trouver_proche ()` qui renvoie l'arête la plus courte reliant un nœud de l'arbre à un nœud de cet ensemble, en temps $O(n)$.

Écrire une fonction `retirer_noeud r a` qui retire le nœud r de cet ensemble, et qui prend en argument la liste a des nœuds reliés à r et la distance qui les sépare, en temps $O(n)$. ◁

▷ **Question 3.** Utiliser cette structure de données pour écrire l'algorithme de Prim. On choisira avec soin le format des données fournies en entrée à l'algorithme. Tester cet algorithme sur le graphe ci-dessus. ◁

▷ **Question 4.** Prouver que l'algorithme de Prim construit bien un arbre couvrant. On pourra distinguer la *connexité* (il existe au moins un chemin) et l'*acyclicité* (il existe au plus un chemin). ◁

▷ **Question 5.** Prouver que l'algorithme de Prim construit bien un arbre couvrant *minimal*. On pourra pour cela considérer un arbre couvrant minimal A , et tout particulièrement la plus petite arête de l'arbre généré par l'algorithme de Prim qui ne se trouve pas dans A . ◁

▷ **Question 6. (difficile)** La complexité de cet algorithme est $O(n^2)$. Proposer une méthode pour améliorer sa complexité, en fonction du nombre d'arêtes a et du nombre de nœuds n . ◁