# I Hypercube

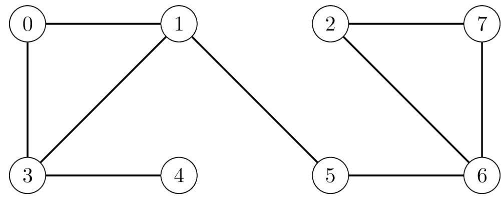
Un **hypercube** $Q_n$ a pour sommets les mots binaires de taille $n$, 2 sommets étant reliés s'ils diffèrent d'un bit.

1. Dessiner $Q_3$.
2. Quel est le nombre de sommets et d'arêtes de $Q_n$?
3. Montrer que $Q_n$ est biparti.
4. Montrer que $Q_n$ possède un couplage parfait.
5. Montrer que $Q_n$ est hamiltonien : il existe un cycle (**hamiltonien**) qui visite tous les sommets exactement une fois. Dessiner un tel cycle de $Q_3$.

# II Pont et algorithme de Tarjan

Soit $G = (V, E)$ un graphe connexe non orienté. Un **pont** de $G$ est une arête telle que $G - e$ n'est pas connexe.

1. Donner un algorithme en complexité quadratique pour trouver tous les ponts d'un graphe $G$.
2. Donner tous les ponts du graphe ci-dessous.



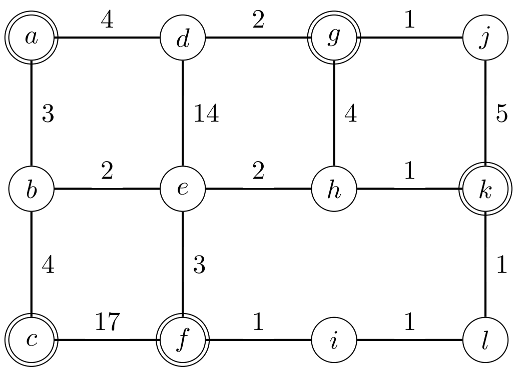

Soit $r \in V$. Soit $T$ une arborescence de parcours en profondeur depuis $r$.

On définit un graphe orienté $\overrightarrow{G}$ en orientant les arcs de $T$ dans le sens du parcours et les arcs arrières en direction de l'ancêtre. On définit $p(v)$ comme étant la profondeur du sommet $v$ dans $T$. On définit $l(v)$ comme étant la plus petite valeur de $p(u)$ pour un sommet $u$ accessible depuis $v$ dans $\overrightarrow{G}$.

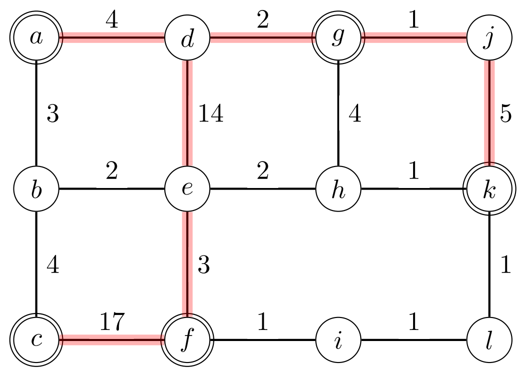
3. On effectue un parcours en profondeur depuis 0 dans le graphe de la question 2, en visitant à chaque fois le plus petit sommet en priorité lorsqu'il y a plusieurs choix. Dessiner $\overrightarrow{G}$ et les valeurs de $p(v)$ et $l(v)$ pour chaque sommet $v$.
4. À quelle condition sur $l$ et $t$ un arc $(u, v)$ de $T$ est-il un pont ?
5. Donner une relation de récurrence permettant de calculer $l(v)$.
6. En déduire un algorithme en OCaml permettant de trouver tous les ponts d'un graphe $G$. On choisira des structures de données adaptées. Quelle est la complexité de cet algorithme ?

## III Arbre de Steiner

Soit $G = (V, E)$ un graphe non-orienté pondéré par $w$ et $R \subseteq V$ un sous-ensemble de sommets.
Un **arbre de Steiner** est un arbre $T$ de $G$ qui contient tous les sommets de $R$.
Remarque : $T$ peut contenir des sommets qui ne sont pas dans $R$, mais tous les sommets de $R$ doivent être dans $T$.
Le poids de $T$, noté $w(T)$, est la somme des poids des arêtes de $T$.
L'objectif est de trouver un arbre de Steiner de poids minimum.
Dans tout l'exercice, on utilise le graphe $G_1$ suivant comme exemple, avec $R = \{a, c, f, g, k\}$ (sommets doublement entourés) :



Un graphe $G_1$ non-orienté pondéré



Un arbre de Steiner $T_1$ (surligné)

À droite surligné, on a représenté un exemple d'arbre de Steiner $T_1$ de $G_1$.

1. $T_1$ est-il un arbre de Steiner de poids minimum ?

On peut montrer que trouver un arbre de Steiner de poids minimum est un problème NP-complet.
Dans la suite, on s'intéresse à une 2-approximation d'un arbre de Steiner de poids minimum.
La première étape consiste à construire un graphe complet (c'est-à-dire : contenant toutes les arêtes possibles) $\widetilde{G}$ dont l'ensemble des sommets est $R$. De plus, le poids d'une arête $\{u, v\}$ dans $\widetilde{G}$ est égal au poids d'un plus court chemin entre $u$ et $v$ dans $G$.

2. Dessiner $\widetilde{G_1}$.
3. Quel algorithme peut-on utiliser pour construire $\widetilde{G}$ ?

La deuxième partie consiste à trouver un arbre couvrant $\widetilde{T^*}$ de poids minimum de $\widetilde{G}$.

4. Donner un arbre couvrant $\widetilde{T_1^*}$ de poids minimum de $\widetilde{G_1}$, en utilisant l'algorithme de Kruskal.

Enfin, on revient dans $G$ en remplaçant chaque arête $\{u, v\}$ de $\widetilde{T^*}$ par le plus court chemin de $u$ à $v$ correspondant, dans $G$.

5. Dessiner $G_1$ en faisant apparaître les arêtes obtenues à partir des plus courts chemins de $\widetilde{T_1^*}$.
6. Si les arêtes obtenues à la question précédente ne forment pas un arbre, on supprime des arêtes dans des cycles jusqu'à obtenir un arbre. Est-ce que l'arbre $T$ obtenu est un arbre de Steiner ? Est-il forcément de poids minimum ?

Soit $T_{opt}$ un arbre de Steiner de $G$. On veut montrer que $T$ est une 2-approximation de $T_{opt}$, c'est à dire :

$$w(T) \leq 2w(T_{opt})$$

7. On traverse $T_{opt}$ depuis un sommet fixé en faisant, par exemple, un parcours en profondeur.
   On obtient ainsi un cycle $C$ de $T_{opt}$ qui passe exactement 2 fois par chaque arête.
   Que vaut $w(C)$, en fonction de $w(T_{opt})$ ?
8. On considère un cycle $\widetilde{C}$ dans $\widetilde{G}$ obtenu à partir de $C$ en remplaçant les portions de chemins entre deux sommets $r_1$, $r_2$, de $R$ par l'arête $\{r_1, r_2\}$. Par exemple, si $C$ utilise les sommets $r_1, v_1, ..., v_k, r_2$ avec $v_1, ..., v_k \notin R$, $\widetilde{C}$ utilisera seulement l'arête de $r_1$ à $r_2$.
   Montrer que $w(\widetilde{C}) \leq w(C)$.
9. En déduire que $w(T) \leq 2w(T_{opt})$.

10. Quelle est la complexité totale de cette méthode pour trouver une 2-approximation de l'arbre de Steiner de poids minimum ?

# IV Algorithme de Johnson

Soit $\overrightarrow{G} = (V, \overrightarrow{E})$ un graphe orienté pondéré par $w : \overrightarrow{E} \longrightarrow \mathbb{R}$ (des poids peuvent être négatifs).
Comme les poids de $\overrightarrow{G}$ peuvent être négatifs, l'algorithme de Dijkstra ne peut pas être utilisé pour trouver tous les plus courts chemins.
L'idée de l'algorithme de Johnson est de modifier les poids de $\overrightarrow{G}$ pour les rendre positifs, sans modifier les plus courts chemins. On peut alors appliquer $|V|$ fois l'algorithme de Dijkstra (une fois depuis chaque sommet) pour obtenir tous les plus courts chemins.

1. Soit $h : V \longrightarrow \mathbb{R}$. On définit $w_h : (u, v) \longmapsto w(u, v) + h(u) - h(v)$. Montrer que, dans $\overrightarrow{G}$, les plus courts chemins pour $w$ et $w_h$ sont les mêmes et qu'il existe un cycle de poids négatif pour $w$ ssi il en existe un pour $w_h$.

2. Trouver $h : V \longrightarrow \mathbb{R}$ telle que $w_h \geq 0$. On pourra supposer dans un premier temps que tous les sommets de $\overrightarrow{G}$ sont atteignables depuis un sommet $r$.

3. En utilisant plusieurs fois l'algorithme de Dijkstra, écrire une fonction `johnson` renvoyant la matrice des distances entre toute paire de sommets, dans $\overrightarrow{G}$ pondéré par $w$.

4. Comparer la complexité de `johnson` avec celle de Floyd-Warshall.