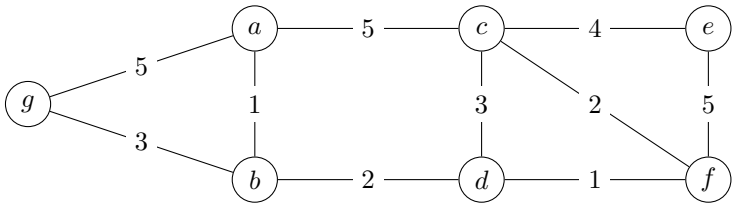
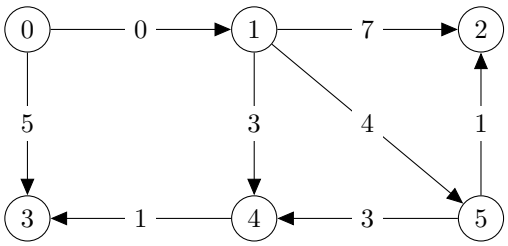
I Application des algorithmes

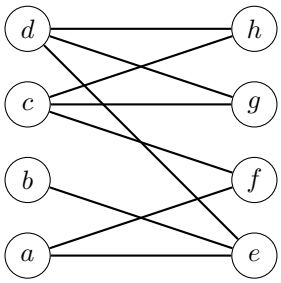
1. Appliquer l'algorithme de Kruskal à la main sur le graphe suivant :



2. Appliquer l'algorithme de Dijkstra à la main pour trouver les distances depuis let sommet 0 vers n'importe quel autre sommet :

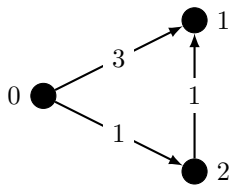


3. Appliquer l'algorithme du cours pour trouver un couplage maximum dans le graphe suivant :



II Plus courts chemin et Dijkstra

1. Trouver un exemple de graphe pour lequel Dijkstra ne fonctionne pas (ne donne pas les plus courts chemins).
2. Change t-on les plus courts chemins si on additionne/multiplie les poids de toutes les arêtes d'un graphe par une constante?

> Solution : Si K est une constante, on définit $w_1(\vec{e}) = w(\vec{e}) + K$ et $w_2(\vec{e}) = w(\vec{e}) \times K$. Soit C un chemin. Alors :
>
> - $w_1(C) = \sum_{\vec{e} \in C} (w(\vec{e}) + K) = w(C) + \ell(C)K$, où $\ell(C)$ est le nombre d'arcs de C. Tous les poids des chemins n'augmentent pas de la même façon... ce qui peut changer les plus courts chemins. Par exemple, si on augmente de 2 les poids sur le graphe suivant, le plus court chemin de 0 à 1 passe de $0 \to 2 \to 1$ à $0 \to 1$:
>
> 
>
>
> - $w_2(C) = \sum_{\vec{e} \in C} (w(\vec{e}) \times K) = w(C) \times K$: si $K > 0$ alors minimiser $w_2(C)$ revient à minimiser $w(C)$, donc les plus courts chemins ne changent pas.

3. Le graphe orienté $\vec{G} = (V, \vec{E})$ d'une chaîne de Markov possède une probabilité sur chaque arc. La probabilité d'un chemin est le produit des probabilités de ses arcs. Comment trouver un chemin le plus probable d'un sommet à un autre?

Solution : 1ère solution : remplacer chaque probabilité $p(u,v)$ par $-\ln(p(u,v)) \geq 0$. Alors la somme des poids des arcs d'un chemin C est $\sum_{(u,v)\in C} -\ln(p(u,v)) = -\ln(\prod_{(u,v)\in C} p(u,v))$ donc maximiser $\prod_{(u,v)\in C} p(u,v)$ revient à minimiser $\sum_{(u,v)\in C} -\ln(p(u,v))$: on peut alors appliquer un algorithme de plus courts chemins classique.
2ème solution : modifier un algorithme de plus court chemin. Par exemple, Dijkstra deviendrait:

Algorithme de Dijkstra pour chemins les plus probables

Initialement: `next` contient tous les sommets
`proba.(r) <- 1` et `proba.(v) <- 0`, $\forall$ `v` $\neq$ `r`
Tant que `next` $\neq \emptyset$:
Extraire `u` de `next` tel que `proba.(u)` soit maximum
Pour tout voisin `v` de `u`:
`proba.(v) <- max proba.(v) (proba.(u) * p(u, v))`

On peut refaire la preuve du cours pour voir que cette version de Dijkstra ne marcherait pas pour des « probabilités » > 1.

4. Soit $\vec{G}$ un graphe sans arc de poids négatif et s, t deux sommets. Montrer que l'on peut trouver tous les arcs utilisés par au moins un plus court chemin de s à t, en utilisant 2 fois l'algorithme de Dijkstra.

Solution : Soit (u,v) un arc. S'il existe un plus court chemin C de s à t passant par (u,v) alors, comme le sous-chemin de C de s à u est un plus court chemin et le sous-chemin de v à t est un plus court chemin, $d(s,t) = d(s,u)+w(u,v)+d(v,t)$. Réciproquement, si $d(s,t) = d(s,u)+w(u,v)+d(v,t)$ alors (u,v) est sur un plus court chemin de s à t (celui constitué d'un plus court chemin de s à u, puis (u,v), puis un plus court chemin de v à t). On peut donc calculer $d(s,u)$, $\forall u \in V$, avec l'algorithme de Dijkstra puis $d(v,t)$, $\forall v \in V$. Pour savoir si (u,v) est sur un plus court chemin, il suffit alors de tester si $d(s,t) = d(s,u) + w(u,v) + d(v,t)$.

5. On suppose avoir un graphe orienté $\vec{G}$ dont les **sommets** sont pondérés par w, au lieu des arêtes. Le poids d'un chemin est alors la somme des poids de ses sommets. Comment modifier $\vec{G}$ pour pouvoir trouver des chemins de plus petit poids de $\vec{G}$, avec les algorithmes du cours?

III Plus large chemin

Soit $G = (V,E)$ un graphe pondéré. La **largeur** $l(C)$ d'un chemin est le minimum des poids de ses arêtes. Soient $u, v \in V$. Un chemin de u à v est un **plus large chemin** (*widest path*) s'il n'existe pas d'autre chemin de u à v de plus grande largeur.

1. Donner un plus large chemin de g à f dans le graphe du I.1.
2. Comment modifier l'algorithme de Kruskal de façon à trouver un arbre couvrant T de poids maximum dans G ?
3. Soit C le chemin de u à v dans T. Montrer que C est un plus large chemin de u à v (l'algorithme de Kruskal permet donc de trouver les plus larges chemins).

Solution : Supposons que C ne soit pas un plus large chemin. Alors, il existe un chemin C' de u à v de plus grande largeur. Soit e une arête de poids minimum de C : on a donc $w(e) < l(C')$. $T - e$ est composé de deux composantes connexes (V_1 contenant u et V_2 contenant v) et C' relie u et v donc il existe une arête e' de C' entre un sommet de V_1 et un sommet de V_2. $T - e + e'$ est un arbre couvrant car est connexe (une seule composante connexe) et contient $n-1$ arêtes (autant que T), où n est le nombre de sommets de G.
S'il y a plusieurs arêtes e de poids minimum, on répète plusieurs fois l'opération précédente.
Ainsi on obtient un arbre $T - e + e'$ tel que $l(T - e + e') > l(T)$, ce qui est absurde.

IV Questions sur les arbres couvrants de poids minimum

Soit $G = (V,E)$ un graphe pondéré.

1. Soit C un cycle de G et $e = \{u,v\}$ une arête de C dont le poids est strictement supérieur au poids des autres arêtes de C. Montrer que e ne peut pas appartenir à un arbre couvrant de de poids minimum de G.

Solution : Soit T un arbre couvrant de de poids minimum. Supposons par l'absurde que $e \in T$. Alors $T - e$ contient deux composants connexes G_u et G_v. Comme $C - e$ est un chemin qui relie u et v, il existe une arête e' de $C - e$ entre un sommet de G_u et un sommet de G_v. $T - e + e'$ est un arbre couvrant car est connexe (une seule composante connexe) et contient $n - 1$ arêtes. Mais $w(T - e + e') = w(T) - w(e) + w(e') < w(T)$, ce qui est absurde.

2. (Propriété d'échange) Soient T_1, T_2 deux arbres couvrants de G et e_2 une arête de $T_2 - T_1$. Montrer qu'il existe une arête e_1 de T_1 telle que $T_1 - e_1 + e_2$ (le graphe obtenu en remplaçant e_1 par e_2 dans T_1) est un arbre couvrant de G.

Solution : Démonstration très similaire à la question précédente.

3. Montrer que si tous les poids des arêtes de G sont différents, alors G admet un unique arbre couvrant de poids minimum.

Solution : Supposons par l'absurde qu'il existe deux arbres couvrants de poids minimum T_1 et T_2. Soit e l'arête de poids minimum qui n'est pas à la fois dans T_1 et T_2. Supposons sans perte de généralité que $e \in T_2$. D'après la question précédente, il existe $e_1 \in T_1 - T_2$ tel que $T_1 - e_1 + e$ est un arbre couvrant. Mais, par définition de e, $w(e) < w(e_1)$ (inférieur strict car tous les poids sont différents) donc $T_1 - e_1 + e$ est un arbre couvrant de poids strictement plus petit que T_2, ce qui est absurde.

4. Le nombre de domination $d(G)$ d'un graphe $G = (V, E)$ est le cardinal minimum d'un ensemble $S \subseteq V$ tel que $\forall v \in V$, $v \in S$ ou v est adjacent à un sommet de V'.
   Montrer que si G est connexe alors $d(G) \geq \dfrac{|V|}{2}$.

5. Soit T_1 un arbre couvrant de poids minimum de G et T_2 le 2ème plus petit arbre couvrant, c'est-à-dire l'arbre couvrant de poids minimum en excluant T_1. Montrer que T_1 et T_2 diffèrent d'une arête et en déduire un algorithme pour trouver T_2.

V Mise à jour d'arbre couvrant de poids minimum

Soit $G = (V, E)$ un graphe pondéré et T un arbre couvrant de poids minimum de G. Soit $e \in E$.

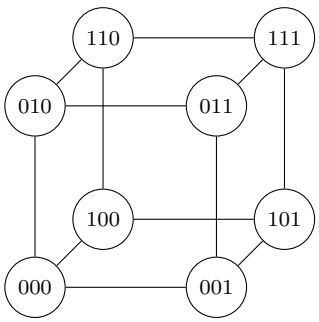
1. On diminue le poids de e. Expliquer comment mettre à jour T pour qu'il soit toujours un arbre couvrant de poids minimum.
2. On augmente le poids de e. Expliquer comment mettre à jour T pour qu'il soit toujours un arbre couvrant de poids minimum.

VI Hypercube

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

1. Dessiner Q_3.

Solution :

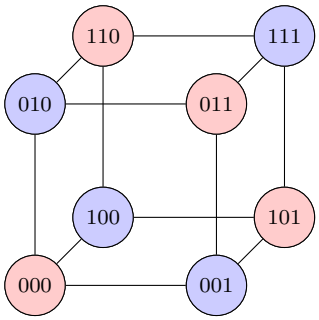


2. Quel est le nombre de sommets et d'arêtes de Q_n?

Solution : Il a 2^n sommets chacun de degré n donc, d'après la formule des degrés, $|E| = \dfrac{\sum \deg(v)}{2} = n2^{n-1}$.

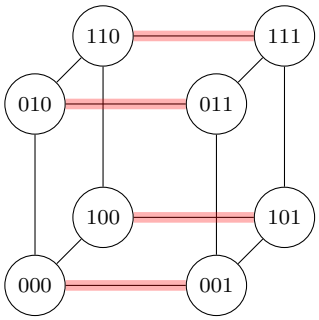
3. Montrer que Q_n est biparti.

Solution : Soit A l'ensemble des sommets de Q_n ayant un nombre pair de 1, et B l'ensemble des sommets de Q_n ayant un nombre impair de 1. Alors, comme il n'y a pas d'arêtes entre deux sommets de A ni entre deux sommets de B, Q_n est biparti.



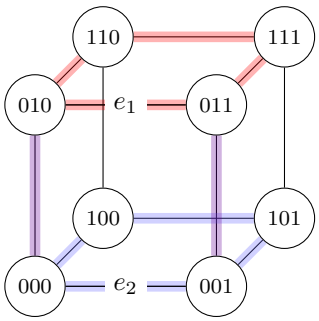
4. Montrer que Q_n possède un couplage parfait.

Solution : Soit $M = \{\{u0, u1\} \mid u \text{ est un mot de taille } n-1\}$. En d'autres termes, M contient toutes les arêtes d'un sommet v vers le sommet obtenu en échangeant le dernier bit de v. Chaque sommet est adjacent à une unique arête de M, par définition. Donc M est un couplage parfait.



5. Soit $n \geq 2$. Montrer que Q_n est **hamiltonien** : il existe un cycle qui visite tous les sommets exactement une fois. Dessiner un tel cycle de Q_3.

Solution : On commence par remarquer qu'on peut obtenir Q_n à partir de deux copies de Q_{n-1} en reliant chaque sommet à sa « copie ».
On peut alors démontrer que Q_n est hamiltonien, par récurrence sur n. C'est évident pour $n = 1$. Supposons que Q_n soit Hamiltonien. Q_{n+1} peut être construit à partir de deux Q_n qui possèdent des cycles hamiltoniens C_1 et C_2, qui sont des copies l'un de l'autre. Soient $e_1 = (u_1, v_1)$ et $e_2 = (u_2, v_2)$ deux arêtes copies l'une de l'autre, dans C_1 et C_2. Alors on peut remplacer, dans $C_1 \cup C_2$, e_1 et e_2 par (u_1, u_2) et (v_1, v_2) afin d'obtenir un cycle Hamiltonien de Q_{n+1}, ce qui achève la récurrence.



VII Questions sur les couplages

1. Soit G un graphe. Montrer que si G a un couplage parfait alors G possède un nombre pair de sommets. La réciproque est-elle vraie ?
2. Soit M_1 et M_2 deux couplages d'un graphe G, avec M_2 maximal. Montrer que $|M_1| \leq 2|M_2|$, puis donner un cas d'égalité.

Solution : Soit V_2 l'ensemble des sommets couverts par M_2. Chaque arête couvre deux sommets différents donc $|V_2| = 2|M_2|$.
Pour toute arête $e = su, v$ de M_1, au moins l'un des sommets u ou v est couvert par M_2 (sinon, on pourrait ajouter e à M_2).
Donc $\boxed{|M_1| \leq |V_2| = 2|M_2|}$.

3. Soit $G = (V, E)$ un graphe. Une couverture par sommets (*vertex cover*) de G est un ensemble C de sommets tels que chaque arête de G est adjacente à au moins un sommet de C. L'objectif est de trouver une couverture par sommets C^* de cardinal minimum.
On propose l'algorithme suivant :

2-approximation de vertex cover

$M \longleftarrow$ couplage maximal de G
$C \longleftarrow$ ensemble des sommets couverts par M

Montrer que C est bien un couplage et que $|C| \leq 2|C^*|$.

4. Soit $M = (m_{i,j})$ une matrice de taille $n \times n$. On définit le permanent de M :

$$\text{per}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}$$

où S_n est l'ensemble des permutations de $\{1, \ldots, n\}$.
Définir un graphe G dont le nombre de couplages parfaits est égal à $\text{per}(M)$.
Remarque : il n'existe pas d'algorithme efficace pour calculer le permanent d'une matrice, contrairement au déterminant qui peut être calculé en $O(n^3)$ avec l'algorithme du pivot de Gauss.

VIII Questions sur les graphes bipartis

1. Montrer qu'un arbre est un graphe biparti.
2. Montrer qu'un graphe est biparti si et seulement s'il n'a pas de cycle impair.