Graphes : algorithmes et modélisation

---

1 Parcours de graphe par un agent

Considérons un graphe représentant un bâtiment, pour faire simple on va dire que les nœuds représentent les salles et les arêtes représentent les couloirs reliant ces salles. Un agent (par exemple un robot de nettoyage, un agent de sécurité qui vérifie que personne n'a été oublié dans les salles ou tout ce que vous voulez imaginer) entre dans le bâtiment par la porte principale, passe dans toutes les salles er ressort par la même porte. Nous allons organiser le *tour* de cet agent dans le bâtiment.

1.1 Le cas de l'arbre

Supposons d'abord que le bâtiment a une structure d'arbre – c'est par exemple le cas de la partie enseignement du bâtiment IIIA. L'agent se déplacera dans l'arbre par un parcours en profondeur.

1. Modifier l'algorithme de parcours en profondeur pour qu'il affiche les nœuds par lesquels passe l'agent, dans l'ordre du parcours.

2. Montrer que l'agent parcourt chaque arête de l'arbre exactement deux fois (une fois dans chaque sens).

1.2 Le cas général

Revenons au cas ou le graphe est quelconque, et supposons de plus qu'on associe à chaque arête une longueur positive (la longueur du couloir respectif). Le problème qui consiste à trouver le tour le plus court est connu pour sa difficulté. Nous allons néanmoins proposer une solution convenable, même si elle n'est pas forcément optimale.

1. On appelle *tour* d'un graphe un cycle passant au moins une fois par chaque sommet (on peut passer plusieurs fois par un même sommet ou par une même arête). La *longueur* du tour est la somme des longueurs de ses arêtes. Soit T un arbre recouvrant de poids minimum du graphe G et $Tour$ un tour de G. Montrer que le poids de l'arbre est inférieur à la longueur du tour.

2. Pour faire le tour du graphe G, on procède de la façon suivante : on calcule d'abord un arbre recouvrant de poids minimum T de G, puis le tour $Tour_T$ qui consiste à faire le tour de T par un parcours en profondeur. Montrer que $Tour_T$ est au pire deux fois plus long que le tour le plus court.

3. Trouver un exemple ou ce pire cas est (presque) atteint.

2 Routage

Considérons un graphe qui représente un réseau de communication entre ordinateurs. Lors de l'envoi d'un message par une arête xy, cet envoi a un coût $c(x, y)$.

1. **Broadcast.** Un nœud s souhaite envoyer un message à tout le monde, tout en minimisant le coût global de la transmission. Comment résoudre ce problème ?

2. **Envoi entre deux deux nœuds**. Un nœud s souhaite envoyer un message à un unique autre nœud t (bien entendu le message peut passer par d'autres nœuds). Comment résoudre ce problème ?

3. **Multicast.** Un nœud s souhaite envoyer un message à un ensemble de nœuds W. Comment modéliser ce problème ?

4. **Connexité.** Donner un algorithme qui calcule le nombre minimum d'arêtes qu'il faut supprimer pour déconnecter les sommets s et t, ainsi que l'ensemble d'arêtes correspondant.

3 Affectation de tâches

Un atelier d'usinage doit affecter un ensemble T de tâches à un ensemble M de machines. Pour chaque tâche on connaît l'ensemble des machines qui peuvent l'effectuer, voir ci-dessous un exemple avec 5 tâches et 5 machines :

Tâche	Machines compatibles
t_1	m_1, m_2, m_3
t_2	m_2, m_4, m_5
t_3	m_1, m_5
t_4	m_4
t_5	m_2, m_3, m_5

Proposer une modélisation ainsi qu'un algorithme qui permet de calculer le nombre maximum de tâches que l'on peut effectuer en parallèle (et sur quelles machines).

4 Pavages

1. Question pour vos petits frères, cousins etc. : dans un échiquier, on découpe deux coins opposés. Peut-on paver cet échiquier tronqué par des dominos de taille 2×1 ?

2. Dans une feuille à carreaux, on découpe une forme arbitraire, en suivant les lignes. Ces formes sont savamment appelées *polyominos*. Proposer un algorithme qui prend en entrée un polyomino et qui vérifie s'il peut être pavé par des dominos.