Quelques exercices plus ≪ avancés ≫

1 Parcours

Etant donné un graphe $G = (V, E)$ connexe orienté, on appelle *point d'articulation* tout sommet x tel que, si l'on supprime x, le nouveau graphe $G - x$ devient non connexe. Un graphe est dit 2-*connexe* s'il est connexe et sans point d'articulation. On appelle *composante* 2-*connexe* du graphe G chaque sous-ensemble maximal de sommets qui induit une composante 2-connexe.

1. Observer sur un exemple que les composantes 2-connexes d'un graphe forment une structure arborescente (et qu'elles ne sont pas disjointes).
2. Adapter l'algorithme de parcours en profondeur pour qu'il calcule, toujours en temps linéaire, les points d'articulation du graphe en entrée.
3. Adapter le parcours en profondeur pour obtenir, en temps linéaire, les composantes 2-connexes du graphe en entrée.

2 Ordres, ordonnancement

Concevoir une heuristique qui prend en entrée un ensemble de tâches avec leur temps d'exécution et le graphe de précédence entre taches, et qui ordonnance ces tâches sur p processeurs. On peut même ≪ corser ≫ le problème en supposant que le graphe représente les communication entre tâches (dans un modèle par envoi de messages) et chaque arc (xy) indique le coût de communication de x vers y si les deux tâches ne sont pas sur le même processeur ; si x et y sont ordonnancées sur le même processeur, ce coût de communication devient nul.

3 Chemins courts

L'algorithme de Dijkstra donne les plus courts chemins d'un sommet à tous les autres, dans un graphe pondéré (sur les arêtes) sans poids négatifs.

(Re)découvrir l'algorithme de Bellman, qui fonctionne également dans le cas avec poids négatifs. On fera en sorte qu'à chaque étape k, $d[x]$ représente la longueur du plus court chemin de la source s au sommet x passant par au plus k arcs.

Même question pour calculer le plus court chemin entre chaque paire de sommets (algorithme de Floyd-Warshall). Pour ce faire, à chaque étape k on calculera $d[x, y]$, le plus court chemin de x à y qui ne passe que par les sommets $\{1, \ldots, k\}$. On peut faire une implémentation très simple de cet algorithme, avec une structure de données type matrice d'adjacente, et en temps $O(n^3)$.

4 Flots

Adapter l'algorithme de coupe minimum pour résoudre les problèmes suivants :

1. Etant donné un graphe non orienté $G = (V, E)$ et deux sommets s, t, trouver le plus petit ensemble F d'*arêtes* qui déconnectent s et t.
2. Etant donné un graphe non orienté $G = (V, E)$ et deux sommets s, t, trouver le plus petit ensemble W de *sommets* qui déconnectent s et t.

En déduire des algorithmes calculant l'arête-connexité et la sommet-connexité d'un graphe. Quelles sont leur complexités respectives ?