Solution du devoir 3

Auteur : Alexandre Blondin Massé

Question	1	2	Total
Sur	80	20	100
Note			

1. (80 points) En 1972, Tarjan a proposé un algorithme permettant de calculer les composantes fortement connexes d'un graphe orienté en temps linéaire, en une seule passe du graphe (voir l'entrée Wikipedia sur le sujet). Dans cette question, nous nous intéressons à modifier l'implémentation de son algorithme pour qu'il retourne l'ensemble des composantes fortement connexes sous forme d'ensembles disjoints plutôt que de simplement afficher le contenu des composantes, comme c'est typiquement présenté.
   (a) (20 points) Donnez le pseudocode de l'algorithme de Tarjan pour que celui-ci retourne des ensembles disjoints des sommets appartenant à chacune des composantes fortement connexes d'un graphe orienté quelconque G.
   (b) En utilisant votre langage de programmation préféré parmi C, C++, Java et Python, implémentez votre algorithme. *Contraintes :* Vous n'avez pas le droit d'utiliser de bibliothèques de graphes ou d'ensembles disjoints. De plus, la structure de données d'ensembles disjoints doit être implémentée telle que vue classe, c'est-à-dire que la fusion doit se faire par rang et le calcul d'un représentant par compression de chemin. La répartition des points se fera comme suit :
      i. (20 points) Structure d'ensembles disjoints;
      ii. (20 points) Structure de graphe orienté;
      iii. (20 points) Algorithme de Tarjan.

   Proposez votre propre implémentation à partir du pseudocode, sans recopier une autre implémentation que vous pourriez trouver en ligne. Dans tous les cas, soyez bien prudents de citer toutes vos sources si vous vous basez fortement sur une d'entre elles.

Solution: À venir...

2. (20 points) Soit F une file binomiale. Pour chacun des deux énoncés suivants, indiquez s'il est vrai ou faux. Appuyez votre réponse à l'aide d'une courte démonstration.
   (a) (10 points) Si F est initialement vide et qu'on y insère n éléments quelconques, alors le coût total des insertions est $\mathcal{O}(n)$.

Solution: C'est vrai.

Nous avons vu en classe que le nombre d'éléments présents dans une file binomiale détermine de façon unique quels arbres binomiaux sont présents ou non dans la structure. En effet, on a que l'arbre B_i est présent dans une file binomiale de n éléments si et seulement si le i-ème bit (en partant de la droite et en commençant à 0) de la représentation binaire de n est égal à 1.

Par ailleurs, on sait que le coût d'une insertion est égal au nombre d'arbres qui sont fusionnés lors de cette insertion, c'est-à-dire au nombre de bits qui passent de 0 à 1 plus le nombre de bits qui passent de 1 à 0. Nous avons déjà vu un exemple similaire dans le cours ! Il s'agit de l'exemple du compteur binaire.

Bref, le coût amorti d'une insertion dans une file binomiale initialement vide est égal au coût amorti de l'incrémentation d'un compteur binaire initialisé à 0. Or, on sait que ce coût amorti est $\mathcal{O}(1)$. On en conclut que le coût total de n insertions dans une file binomiale est $\Theta(n)$.

(b) (10 points) Si F contient n éléments et qu'on effectue m extractions du minimum, où $m \leq n$, alors le coût total des extractions est $\mathcal{O}(m)$.

Solution: C'est faux.

Si c'était vrai, on aurait un algorithme qui trie un ensemble de valeurs quelconques en temps linéaire. En effet, soit E un ensemble de n éléments. Alors on peut construire une file binomiale contenant les éléments de E en $\mathcal{O}(n)$, par la partie (a). Si on extrait ensuite le minimum n fois et qu'on le place en queue d'une liste L (initialement vide) à chaque fois, on obtiendrait que L est la liste des valeurs triées de E. On aurait donc trié l'ensemble E en temps linéaire, ce qui est impossible.