Université Paris Diderot – Paris 7
L3 Informatique

Algorithmique
Année 2013–2014, 1^{er} semestre

AL5 Algorithmique
Examen

Jeudi 9 janvier 2013 12h–15h
Documentation autorisée : deux feuilles A4 recto verso manuscrites. Inscrivez votre nom sur chaque feuille. Toute autre documentation ou appareil sont interdits.

Indications : Le barême est donné à titre indicatif seulement. Justifiez vos réponses et expliquez vos algorithmes. Les algorithmes doivent être aussi efficaces que possible et présentés de façon lisible et claire. Une partie de la note portera sur la clarté de présentation de vos réponses (mais pas la longueur). Un bon choix de noms de variables et de noms d'algorithmes est essentiel mais ne pas négliger les explications courtes qui permettent de comprendre le fonctionnement de vos algorithmes.

Exercice 1 Insertions dans les AVL (3 points)

1. Commençant avec un AVL vide, donnez une séquence d'insertions des éléments 10, 20, 30 qui ne provoque aucune rotation et produit à la fin des 3 insertions un AVL de 3 sommets avec déséquilibre 0 sur tous les sommets. Donnez l'arbre et le déséquilibre de chaque sommet après chaque insertion.

Appelons A l'arbre que vous obtenez après l'insertion des 3 sommets.

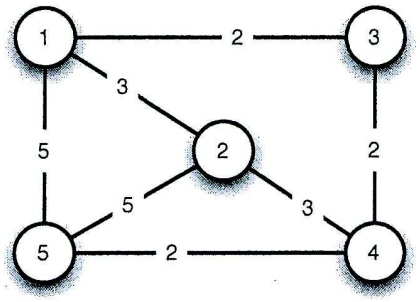
2. Indiquez quelle(s) insertion(s) il faut faire à A pour provoquer :
   (a) Une rotation simple à droite
   (b) Une rotation double à droite

   Donnez le résultat après chaque insertion. Pour la rotation double à droite, reprenez à partir de l'arbre A à 3 sommets. Choisissez judicieusement les valeurs à insérer afin de faire le plus petit nombre d'ajouts qui provoquera uniquement la rotation demandée.

Exercice 2 (2 points) **Opérations sur les tas** On souhaite faire les opérations suivantes sur un tas max, qui est initialement vide. Pour chaque opération, donnez les principales étapes intermédiaires. Donnez à chaque fois le tas sous forme d'arbre et sous forme de tableau.

1. Insérer 10 ;
2. Insérer 30 ;
3. Insérer 20 ;
4. Insérer 40 ;
5. Supprimer l'élément maximum.

Exercice 3 Arbres couvrant minimaux et plus courts chemins (7 points) Pour cette question, reportez-vous au graphe valué ci-dessous.



1. (Arbre couvrant minimum)
   (a) Nommez un algorithme qui trouve un arbre couvrant minimum d'un graphe connexe non-orienté valué.
   (b) Donnez les étapes intermédiaires de l'éxécution de cet algorithme sur le graphe ci-dessus.
   (c) Donnez l'arbre couvrant minimum trouvé par cet algorithme.
2. (Plus courts chemins)
   (a) Nommez un algorithme qui trouve les plus courts chemins à partir d'un sommet donné dans un graphe connexe non-orienté étiqueté.
   (b) Précisez quelles sont les dimensions des tableaux `Pred` et `Dist` utilisés dans cet algorithme. Expliquez brièvement ce que contiennent ces tableaux à chaque itération de l'algorithme.
   (c) Donnez les étapes intermédiaires de l'éxécution de cet algorithme sur le graphe ci-dessus, à partir du sommet 1. Donnez les tableaux `Pred` et `Dist` à chaque itération.
   (d) Quels sont les chemins de 1 aux sommets $\{2, 3, 4, 5\}$ trouvés par l'algorithme ? Pour chaque chemin, donnez sa longueur.
   (e) Donnez un algorithme qui
   - prend en entrée les tableaux `Dist` et `Pred` retournés par cet algorithme sur un graphe G quelconque, ainsi qu'un sommet `i`, et
   - retourne le chemin de `1` à `i` trouvé par l'algorithme, ainsi que la longueur du chemin. Le chemin doit être présenté comme une liste de sommets qui composent le chemin, avec 1 comme premier élément de la liste et i comme dernier élément.

   Donnez la complexité de votre algorithme.

Exercice 4 Composantes connexes (8 points) Pour cet exercice, vous devez proposer deux algorithmes différents pour trouver le nombre de composantes connexes d'un graphe non-orienté. Vos algorithmes doivent
- prendre en entrée une liste de paires représentant des arêtes d'un graphe non-orienté et
- retourner le nombre de composantes connexes du graphe.

Le premier algorithme doit être basé sur un parcours de graphe, et le second doit utiliser le Union Find.

Les sommets sont numérotés par des entiers à partir de 1. Vous pouvez supposer que les algorithmes suivants sont donnés. Pour le parcours, vous pouvez utiliser au choix :
- Une fonction `ParcoursProfondeur(M,u)` qui prend en entrée un tableau `M` qui représente la matrice d'adjacence d'un graphe ainsi qu'un sommet de départ `u`, et qui retourne les tableaux `Pred`, `Debut` et `Fin`.
- Une fonction `ParcoursLargeur(M,u)` qui prend en entrée un tableau `M` qui représente la matrice d'adjacence d'un graphe ainsi qu'un sommet de départ `u`, et qui retourne les tableaux `Pred` et `Dist`.

Pour le Union Find :
- Une fonction `UF()` qui initialise la structure de données Union Find à l'ensemble vide.
- Une fonction `Union(u,v)` qui fait l'union et ne retourne rien.
- Une fonction `Find(u,v)` qui retourne un booléen, `Vrai` si u, v sont dans le même ensemble, `Faux` sinon.
- Une fonction `Root(u)` qui retourne la racine de u.

1 et 2. Pour chacun des deux algorithmes :
   (a) Expliquez en quelques phrases l'idée de votre algorithme et justifiez brièvement pourquoi il est correct.
   (b) Donnez votre algorithme de recherche de nombre de composantes connexes.
   (c) Analysez la complexité en temps de votre algorithme en fonction du nombre de sommets n et du nombre d'arêtes m du graphe. Précisez s'il s'agit de la complexité moyenne, en pire cas, ou amortie.

3. Comparez les deux algorithmes en termes de complexité en temps. Lequel est préférable ?