Université Paris Diderot – Paris 7
L3 Informatique

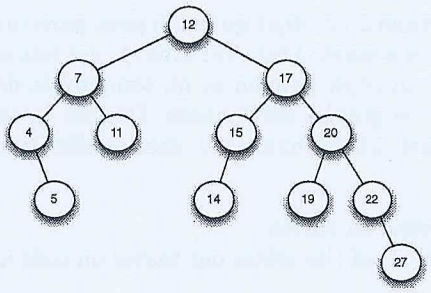
Algorithmique
Année 2014–2015, 1^{er} semestre

AL5 Algorithmique
Examen

Lundi 5 janvier 2015 8h30–11h30
Documentation autorisée : une feuille A4 recto verso manuscrite. Toute autre documentation ou appareil sont interdits.

Indications : Le barème 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. Ne pas négliger les explications courtes qui permettent de comprendre le fonctionnement de vos algorithmes.

Exercice 1 (4 points)



1. Donnez le déséquilibre sur tous les sommets de cet AVL.
2. Insérez l'élément 30 dans l'AVL, sans faire les rotations. Quels sont les sommets déséquilibrés suite à l'insertion de cette valeur ?
3. Combien de rotations sont nécessaires pour rétablir l'équilibre sur tous les sommets ?
4. Donnez l'AVL après avoir effectué toutes les rotations nécessaires. Si vous effectuez plus d'une rotation, donnez l'arbre après chaque rotation. Précisez à chaque fois le type de rotation effectuée.
5. Supprimez la valeur 11 de l'AVL obtenu à la question précédente, sans faire les rotations. Quels sont les sommets déséquilibrés suite à l'insertion de cette valeur ?
6. Combien de rotations sont nécessaires pour rétablir l'équilibre sur tous les sommets ?
7. Donnez l'AVL après avoir effectué toutes les rotations nécessaires. Si vous effectuez plus d'une rotation, donnez l'arbre après chaque rotation. Précisez à chaque fois le type de rotation effectuée.

Exercice 2 (4 points)

Un graphe est k-régulier si tous les sommets du graphe ont exactement k voisins. On souhaite écrire un algorithme qui prend en entrée un graphe et retourne k si le graphe est k-régulier, ou -1 si le graphe n'est k-régulier pour aucun k. L'algorithme et sa complexité varient selon la façon dont est présenté le graphe. Dans les trois cas suivants, donnez un algorithme et analysez sa complexité en fonction de n, le nombre de sommets, et m, le nombre d'arêtes du graphe.

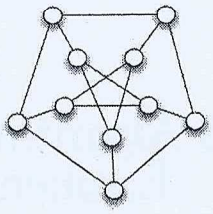


FIGURE 1 – Un graphe 3-régulier

1. Le graphe est donné sous forme de *matrice d'adjacence* `M`, un tableau à deux dimensions de taille $n \times n$. On suppose que les sommets du graphe sont numérotés de 0 à $n-1$. Pour tout `i, j` tel que $0 \leq$ `i` $<$ `n`, $0 \leq$ `j` $<$ `n`, `M[i][j] = 1` s'il existe une arête entre les sommets `i` et `j`, et `M[i][j] = 0` sinon. Vous pouvez supposer que la matrice est symétrique, c'est-à-dire que pour tout `i, j`, `M[i][j] = M[j][i]`. Donnez la complexité en comptant le nombre d'accès aux éléments du tableau.
2. Le graphe est donné sous forme de *liste d'adjacence*. Plus précisément, on aura en entrée un tableau `L` de taille n. On suppose que les sommets du graphe sont numérotés de 0 à $n-1$. Pour tout `i`, $0 \leq$ `i` $<$ `n`, `L[i]` contient la liste (non ordonnée) des sommets voisins de `i`. On considère que si `i` est voisin de `j`, alors `j` est voisin de `i`. Donnez la complexité en comptant le nombre d'accès au tableau, sans négliger le temps de parcours des listes.
3. Le graphe est représenté comme un objet qu'on ne peut parcourir qu'avec des méthodes. La classe `sommet` contient la méthode `sommet.listeVoisins()`, qui retourne une `liste` de ses `sommets` voisins. On donne en entrée un objet `graphe` et un sommet de départ `s` (une instance de la classe `sommet`). On suppose que le graphe est connexe. Donnez la complexité en comptant le nombre d'accès à la méthode `sommet.listeVoisins()`, sans oublier le temps de parcours des listes.

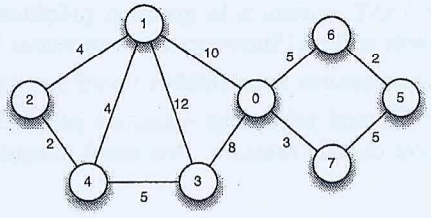
Exercice 3 (4 points)

Donnez un algorithme qui prend en entrée
- un graphe non orienté, non valué (les arêtes ont toutes un coût unitaire), donné sous forme de liste d'ajacence
- un sommet de départ `s`
- un entier `d`,

et qui affiche le nombre de sommets à distance `d` du sommet de départ `s`.

Analyser la complexité de votre algorithme en fonction de n, le nombre de sommets dans le graphe, et m, le nombre d'arêtes du graphe.

Exercice 4 (4 points)



Donnez une exécution de l'algorithme Prim et une exécution de l'algorithme Kruskal sur le graphe valué ci-dessous. Pour chacun des deux algorithmes,

1. préciser à chaque étape
   - l'ensemble des candidats parmi lesquels on choisira le prochain sommet ou la prochaine arête. Si vous utilisez une file de priorité, il suffit d'en donner le contenu (pour chaque élément dans la file, préciser quelle est sa priorité) sans détailler davantage.

- le sommet ou l'arête qui est sélectionné
- le sous-graphe construit jusqu'à cette étape.

2. donner l'arbre couvrant minimal trouvé par l'algorithme ainsi que son coût.

Exercice 5 (4 points)

Un algorithme calcule les plus courts chemins dans un graphe G à partir d'un sommet s retourne un graphe des prédécesseurs qui est donné sous la forme d'un tableau `pred`. Donner un algorithme qui prend en entrée ce tableau ainsi qu'un sommet `v` et retourne le plus court chemin partant de `s` vers `v`. La sortie doit être donnée sous forme de liste de sommets, dont le premier élément est `s` et le dernier élément est `v`. Donnez la complexité de l'algorithme en fonction de n le nombre de sommets du graphe, et m son nombre d'arêtes.