Structures de données et algorithmes

Examen écrit, 5 septembre 2013

Livres fermés. Durée : 3h30.

Remarques

- Répondez à chaque question sur une feuille **séparée**, sur laquelle figurent votre nom et votre section.
- Soyez bref et concis, mais précis.
- Sauf mention explicite, toutes les complexités sont à décrire par rapport au temps d'exécution des opérations concernées. Soyez toujours le plus précis possible dans le choix de votre notation (Ω, O ou Θ).

Question 1

Soit les deux algorithmes suivants (où A est un tableau d'entiers et key un entier) :

```
ALGO-X(A, key)
1  i = 1
2  while i ≤ A.length
3        if A[i] == key
4              ALGO-X-SUB(A, i)
5        else i = i + 1
```

```
ALGO-X-SUB(A, i)
1  while i < A.length
2        A[i] = A[i + 1]
3        i = i + 1
4  A[i] =Null
```

```
ALGO-Y(A)
 1  i = 1
 2  j = 1
 3  maxcount = 0
 4  count = 0
 5  while i ≤ A.length
 6        if A[i] == A[j]
 7              count = count + 1
 8        j = j + 1
 9        if j > A.length
10              if count > maxcount
11                    maxcount = count
12              count = 0
13              i = i + 1
14              j = i
15  return maxcount
```

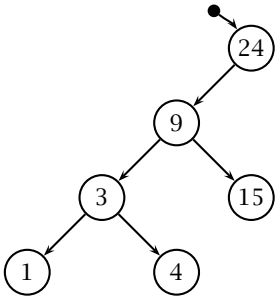
(a) Analysez la complexité dans les meilleur et pire cas de ces algorithmes en fonction de la taille n du tableau A.

(b) Expliquez brièvement ce qu'ils font et écrivez des algorithmes BETTER-ALGO-X et BETTER-ALGO-Y faisant la même chose avec une complexité asymptotique strictement inférieure dans le pire cas. Vous pouvez exploiter des algorithmes ou structures vus au cours.

Question 2

(a) Qu'est-ce qu'un arbre binaire de recherche ?

(b) Soit l'arbre binaire suivant :



Donnez tous les ordres d'insertion des clés qui auraient pu mener à cet arbre.

(c) Donnez la fonction d'insertion d'un nouveau nœud z dans l'arbre (TREE-INSERT(T, z)), en la modifiant de manière à ce qu'elle maintienne un attribut $t.size$ à chaque nœud t contenant le nombre de clés dans le sous-arbre dont t est la racine.

(d) Ecrivez de manière *récursive* et de manière *itérative* une fonction SELECTION(T, i) qui renvoie la $i$ème clé, selon leur ordre croissant, dans T. Par exemple, SELECTION$(T, 4)$ pour l'arbre ci-dessous doit renvoyer 9. Vous pouvez supposer que l'arbre a été construit en utilisant la fonction d'insertion du point précédent.

(e) Donnez la complexité aux meilleur et pire cas de la fonction SELECTION en fonction du nombre de clés n dans l'arbre.

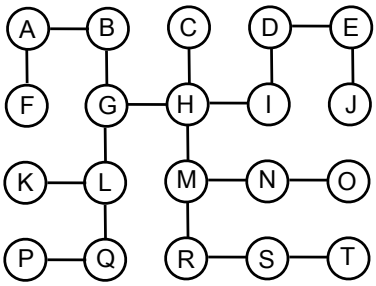
Question 3

(a) Décrivez l'algorithme de parcours de graphe en *largeur d'abord* dans le formalisme de votre choix.

(b) Discutez la complexité de cet algorithme en fonction de l'implémentation du graphe (par liste d'adjacence ou par matrice d'adjacence).

(c) Pour le graphe ci-dessous :
- Donnez le rang *minimum* du nœud R dans un parcours en largeur d'abord à partir de A.
- Donnez le nombre *maximum* de nœuds présents simultanément dans la structure utilisée par l'algorithme lors de son exécution à partir de A.

Justifiez vos réponses.



Question 4

Soit un tableau de nombres entiers $A[1\,..\,n]$. On cherche à partitionner ce tableau en un nombre minimum de sous-séquences contiguës telles que chaque sous-séquence démarre et se termine par le même nombre. Par exemple, la séquence $[8, 2, 6, 7, 1, 2, 3, 4, 5, 4, 5, 3, 1]$ peut se découper en 4 sous-séquences commençant et se terminant par le même nombre de la manière suivante $([8], [2, 6, 7, 1, 2], [3, 4, 5, 4, 5, 3], [1])$ et il n'est pas possible de découper ce tableau avec moins de sous-séquences.

(a) Donnez une formulation récursive pour le nombre minimum $S(i)$ de sous-séquences pour couper le sous-tableau $A[1\,..\,i]$.

(b) En déduire le pseudo-code d'un algorithme *efficace* MINCUT(A) renvoyant le nombre minimum de sous-séquences pour couper le tableau A.

(c) Analysez la complexité de votre algorithme.

(d) Complétez cet algorithme pour qu'il affiche la position du premier élément de chaque sous-séquence d'une solution.