Algorithmique avancée

Documents autorisés : Une feuille A4 recto-verso de notes personnelles

7 novembre 2022, 13h45 – 15h15 (1h30)

Nom :	Prénom :	Note

1 Preuves et complexité

On considère l'algorithme suivant du tri à bulles :

```
1  def bubble_sort(A, n):
2      done = False
3      k = 0
4      while not done:
5          done = True
6          for i = 0 to n - 2 - k:
7              if A[i] > A[i + 1]:
8                  A[i], A[i + 1] = A[i + 1], A[i] # échange les valeurs
9                  done = False
10         k = k + 1
```

On rappelle qu'une *inversion* est un couple (i, j) tel que $i < j$ et $A[i] > A[j]$.

Q1. (5 min) Donner un invariant pour la boucle `while` et un pour la boucle `for` concernant le nombre d'inversions, qui permettront de prouver la terminaison et la correction de l'algorithme.

while

for

Q2. (5 min) Expliquer brièvement comment ces invariants permettent de prouver la terminaison et la correction de l'algorithme.

Q3. (15 min) Combien d'échanges sont faits en moyenne ? (si A est une permutation aléatoire uniforme de $[1..n]$) ? *Justifier*

Q4. (5 min) Combien de comparaisons sont faites en moyenne ? (si A est une permutation aléatoire uniforme de $[1..n]$) ? *Justifier*

2 Conception d'algorithmes

Q5. (20 min) En utilisant le principe *divide and conquer*, écrire une fonction `maxarr` permettant de calculer la plus grande somme d'éléments contigus dans un tableau d'entiers relatifs ($\mathbb{Z}$). On ne demande pas de renvoyer le sous-tableau. *Expliquer aussi brièvement le principe de l'algorithme.*
Par exemple `maxarr([4,-2,-1,-2,2,-1,3,5,-3]) == 9` (le maximum est réalisé par le sous-tableau `[2,-1,3,5]`).

Code

Explication

Q6. (5 min) Calculer la complexité pire cas de l'algorithme précédent en utilisant le *master theorem*. *Expliciter le cas qui s'applique.*

Q7. (15 min) En utilisant la programmation dynamique, donner un algorithme *bottom-up* permettant de calculer le nombre de coups et la stratégie pour résoudre le problème des tours de Hanoï. *Expliquer aussi brièvement le principe de l'algorithme.*

On rappelle que dans ce problème on dispose de n disques de tailles toutes différentes, troués en leur centre, et que l'on peut empiler sur trois tiges différentes. On a aussi les contraintes suivantes : (i) on ne peut bouger qu'une seul disque à la fois (ii) on ne peut bouger qu'un disque qui est au dessus de sa pile (iii) on ne peut pas empiler un disque au dessus d'un disque plus petit. L'objectif est de déplacer la pile des n disques initialement empilés correctement sur la tige numéro x vers la tige numéro y.

Code

Explication

Q6. (5 min) Donner la complexité de l'algorithme. *Justifier*

3 Structures de données

Q9. (15 min) Écrire la fonction `join` qui étant donnés deux arbres AVL A_1 et A_2 et une valeur x plus grande que tous les éléments de A_1 et plus petite que ceux de A_2 renvoit un arbre AVL contenant A_1, A_2 et x. On pourra utiliser les fonctions `rotate_left` et `rotate_right`, qui renvoient les rotations à gauche et à droite d'un arbre AVL, ainsi que la fonction `height` qui renvoit la hauteur d'un arbre, sans les réécrire. On pourra n'écrire qu'un des deux cas symétriques. *Expliquer aussi brièvement le principe de l'algorithme.*

Code

Explication