Examen 2020
Analyse d'Algorithmes et Programmation

Yann Rotella

Lundi 25 mai 2020

La durée de cet examen est de 2 heures. Il est possible de communiquer si nécessaire par mail ou par téléphone (yann.rotella@uvsq.fr) ou 06.16.41.00.19. Il est impératif d'envoyer un e-mail entre 9h55 et 10h10, indiquant la bonne réception du sujet. La fin de l'épreuve est à 12h. Toute copie reçue après 12h20 ne sera pas corrigée. Le format de la copie doit être en PDF. Pour les étudiant.e.s ayant droit à un tiers temps, i.e. jusqu'à 12h40, la copie doit être envoyée par mail avant 12h50.

Le barème est donné à titre indicatif. Toute erreur dans le sujet sera prise en compte dans la correction. La ressemblance entre les copies sera examinée et pourra être sanctionnée. Toute ressemblance avec des informations trouvées sur le Web sera aussi examinée. La qualité de la rédaction sera aussi prise en compte dans la correction.

Questions de cours

1 (1 point) Soit $f, g : \mathbb{N} \to \mathbb{R}$, telles que $f \in O(n)$ et $g \in O(n \log(n))$. Est-ce que $f \in O(g)$? Est-ce que $g \in O(f)$? Justifiez brièvement.

2 (1 point) Soit $f : \mathbb{N} \to \mathbb{R}$ définie par $f(n) = n \log(n) \log(\log(n))$. Est-ce que $f(n) = O(n^2)$? Justifiez brièvement.

3 (1 point) Donnez trois inconvénients et un avantage à utiliser une liste chaînée. Expliquez brièvement.

4 (1 point) Donnez le principal avantage à utiliser un arbre binaire de recherche. Expliquez brièvement.

5 (4 points) On considère un algorithme qui utilise une technique de type "diviser pour mieux régner". Pour cet algorithme, on montre que la fonction de coût $C(n)$ qui à une instance de taille n donne le coût en temps de l'algorithme vérifie

$$C(n) \leq 4C\left(\frac{n}{3}\right) + 3cn + d\,,$$

où c et d sont deux constantes strictement positives, avec égalité quand 3 divise exactement n. Donnez le **représentant** de la classe de complexité

de l'algorithme O et Ω. Prouvez le résultat. On supposera que la fonction de coût est croissante.

6 (1 point) On considère la procédure d'insertion dans un arbre binaire de recherche classique (qui ne laisse pas nécessairement l'arbre équilibré). Dessinez l'arbre binaire de recherche obtenu par insertions successives de la suite d'entiers suivants (on utilise naturellement la relation d'ordre classique sur les entiers).

$$3, 14, 5, 18, 16, 12, 17, 1, 10, 6$$

Le tri de la crêpière

Pour tout tableau T, on note a_i pour i allant de 1 à n l'élément à la position i dans T. On note aussi $T[i:j]$ le sous-tableau de T, composé des éléments de T allant de la i-ième position à la j-ième position incluse. Pour tout i allant de 1 à n, on peut aussi noter $T[i]$ l'élément à la position i, c'est à dire a_i si les éléments a_i n'ont pas changé de position.

On définit les algorithmes suivants.

Algorithm 1 indiceMax(T,k)

```
m = 1
j = 2
while j ≤ k do
    if T[m] < T[j] then
        m = j
    end if
    j = j + 1
end while
return m
```

Algorithm 2 echanger(T,i,j)

```
tmp = T[i]
T[i] = T[j]
T[j] = tmp
```

Algorithm 3 retourner(T,i,j)

```
if i < j then
    echanger(T, i, j)
    retourner(T, i + 1, j − 1)
end if
```

7 (1 point) Soit $T = [5, 8, 2, 0, 3, 6, 5, 2, 1]$. Donnez la valeur de T après l'appel de retourner$(T, 2, 8)$.

8 (1 point) Montrez par récurrence que retourner(T, i, j) se termine pour tout couple d'entiers i, j compris entre 1 et la taille du tableau T.

9 (2 points) Le miroir de tout tableau $T = [a_1, a_2, \ldots, a_{n-1}, a_n]$ est le tableau $T' = [a_n, a_{n-1}, \ldots, a_2, a_1]$. Montrez par récurrence que l'algorithme retourner(T, i, j) transforme le sous-tableau $T[i : j]$ par son miroir et laisse les autres éléments inchangés, pour tout $1 \leq i \leq j \leq n$ où n donne la taille du tableau T.

On définit maintenant la procédure suivante.

Algorithm 4 placer(T,k)

```
ind = indiceMax(T, k)
if ind ≠ k then
    retourner(T, 1, ind)
    retourner(T, 1, k)
end if
```

10 (2 points) Soit $1 \leq k \leq n$. On note T' le tableau en sortie de placer(T, k). Montrez que $\forall i \leq k$, $T'[i] \leq T'[k]$.

On définit aussi la procédure de tri suivante.

Algorithm 5 tri(T)

```
k = taille(T)
while k ≥ 1 do
    placer(T, k)
    k = k − 1
end while
```

11 (5 points) En utilisant un invariant de boucle, montrez par récurrence que la procédure tri définie ci-dessus se termine et qu'après l'appel à tri, le tableau est trié par ordre croissant.

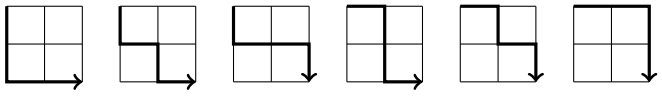
12 (2 points) Donnez les classes de complexité de toutes les procédures précédemment vues (O, Ω, Θ).

Un algorithme de tri très rapide

13 (4 points) On veut trier un tableau T de taille n, dont on sait que tous les éléments sont des entiers compris entre 1 et $4n$. Décrire alors un algorithme, et les structures de données que vous utilisez, afin de trier le tableau T en un temps linéaire en n.

Un algorithme à proposer

On considère des grilles, et on part du coin en haut à gauche, pour aller au coin en bas à droite. On ne peut se déplacer que vers le bas ou vers la droite. On cherche alors le nombre de chemins possibles de cette forme dans une grille de taille $n \times m$, où n désigne le nombre de colonnes et m désigne le nombre de lignes. Dans l'exemple suivant, on liste tous les chemins possibles pour une grille de taille 2×2. Ils sont au nombre de 6.



14 (1 point) Donnez une relation de récurrence qui lie le nombre de chemins possibles pour des grilles de tailles différentes, et qui permettrait de calculer ce nombre algorithmiquement.

15 (4 points) Donnez un algorithme (en pseudo-code ou syntaxe Python), qui permettrait à un ordinateur actuel de calculer le nombre de chemins possibles (définis au-dessus) pour une grille de taille 40×40. Justifiez pourquoi votre algorithme fonctionnerait en pratique.

Le simplexe

On considère le problème linéaire suivant.

$$\begin{aligned} x + y &\leq 3 \\ x + 4y &\geq 1 \end{aligned}$$

avec x et y positives. On cherche à maximiser l'expression $2x + y$.

16 (2 points) Écrire le problème sous sa forme standard.

17 (3 points) Appliquer l'initialisation de simplexe, et obtenez une solution réalisable de base.

18 (4 points) Résoudre le problème à l'aide de la méthode du simplexe.