

Maîtrise d'Informatique

Algorithmes et Complexité – Deuxième partie

examen du Mercredi 28 Janvier 2004, 1h30

Michel Van Caneghem

La Note : La partie de Jean-François Maurras sera comptée sur 9 points et ma partie sur 11 points (Examen sur 8/20*11 points, TP sur 12/20*11 points). Tous les documents sont autorisés.

La fameuse récurrence (2 points – 25 minutes)

Soit l'équation de récurrence suivante :

$$T(N) = 2T(N/2) + N\sqrt{N}, \quad N \geq 0, \quad T(1) = 1$$

1. Résoudre cette équation de récurrence (Attention pour une fois les racines ne sont ni 0, ni 1). Maple8 a trouvé :

$$(2 + \sqrt{2})N\sqrt{N} - (1 + \sqrt{2})N$$

2. Construire un algorithme artificiel qui aurait cette complexité en nombre d'addition par exemple.

Question de TD (2 points – 15 minutes)

Caculez les sommes suivantes :

$$- S_1 = \sum_{k=1}^N \log_2 k$$

$$- S_2 = \sum_{k=1}^N \lfloor \log_2 k \rfloor \text{ pour } N \text{ de la forme } 2^n - 1. \text{ A quelle occasion a-t-on rencontré la deuxième somme ?}$$

Donnez un ordre de grandeur de la différence $S_1 - S_2$. Pour information, voici un tableau que j'ai construit :

N	127	255	511	1023	2047
S_1	709	1675	3866	8759	19569
S_2	642	1538	3586	8194	18434
$S_1 - S_2$	67	137	280	565	1135

Les k plus petits éléments (5 points – 50 minutes)

Soit T un tableau **non trié** de N éléments. On s'intéresse à trouver les k plus petits éléments du tableau, bien sûr, sans trier ce tableau. On supposera que k est petit devant N

A : méthode générale (1 point)

Je vous propose les deux méthodes suivantes :

1. On construit le tas associé au tableau T et on extrait successivement les k plus petits éléments.
2. On prend les k premiers éléments de T , on construit le tas correspondant, puis on parcourt les $N - k$ élément restant de T , en mettant à jour le tas de k éléments chaque fois que l'on a trouvé un élément plus petit que le sommet du tas.

Donnez la complexité de ces deux méthodes. Laquelle est la meilleure ?

B : méthode pour $k = 2$ (2 points)

On va utiliser maintenant un algorithme plus sophistiqué : On divise le tableau en 2 sous-tableaux T_1 et T_2 de taille $N/2$.

1. Montrez que si on connaît les deux plus grands de T_1 (dans l'ordre) et les deux plus grands de T_2 (dans l'ordre) alors on peut trouver les deux plus grands de T (dans l'ordre) avec 2 comparaisons .
2. En déduire que la complexité de ce programme est donné par l'équation : $C(n) = 2 + 2C(n/2)$ (avec $C(2) = 1$). Résoudre cette équation de récurrence et en déduire la complexité de l'algorithme proposé.

C : une autre méthode pour $k = 2$ (2 points)

On peut encore faire mieux ! On cherche l'élément le plus petit, avec un parcours binaire comme précédemment, et en même temps on construit une liste **sans comparaison**, qui va contenir le deuxième plus petit élément. Voici l'esprit de l'algorithme :

```
plusPetit(T[1..n],L)
  x1 = plusPetit(T[1..N/2], L1)
  x2 = plusPetit(T[N/2..N], L2)
  if (x1 < x2) L = x2 + L1; return x1
  else       L = x1 + L2; return x2
```

Voici un exemple de fonctionnement :

```
plusPetit([5,6,7,8,1,2,3,4],L)                                     <= plusPetit = 1, L = [5,3,2]
  plusPetit([5,6,7,8],L)                                           <= plusPetit = 5, L = [7,6]
    plusPetit([5,6],L) <= plusPetit = 5, L = [6]
    plusPetit([7,8],L) <= plusPetit = 7, L = [8]
  plusPetit([1,2,3,4],L)                                           <= plusPetit = 1, L = [3,2]
    plusPetit([1,2],L) <= plusPetit = 1, L = [2]
    plusPetit([3,4],L) <= plusPetit = 3, L = [4]
```

1. Quel est la complexité de l'algorithme en nombre de comparaisons ?
2. Quel est la longueur de la liste L en fonction de N ?
3. Avec cet algorithme on a trouvé le plus petit et une liste L . Pour trouver le second plus petit, il suffit de chercher l'élément le plus petit de L , en parcourant cette liste. En déduire la complexité de l'algorithme qui cherche les deux plus petits (j'ai trouvé : $N + \lceil \log N \rceil - 2$ comparaisons).

