L3-INFO Algorithmique et Modélisation

UNIVERSITÉ Grenoble Alpes

Quick – 2 mars 2015 – durée 1 h

Sont interdits : les documents, les ordinateurs, les téléphones (incluant smartphone, tablettes,... tout ce qui contient un dispositif électronique).
Seuls les dictionnaires papier pour les personnes de langue étrangère sont autorisés.
Il sera tenu compte de la qualité de la rédaction et de la clarté de la présentation.

Exercice 1 : Calcul du pic

On dispose d'un tableau T de taille n d'éléments comparables ayant un pic, c'est à dire qu'il existe $p \in \{1, \ldots, n\}$ tel que

$$T[1] < T[2] < \cdots < T[p] > T[p+1] > \cdots > T[n].$$

1. Proposer un algorithme naïf pour calculer la position du pic et calculer sa complexité.
2. Proposer un algorithme de type diviser pour régner pour calculer la position du pic et calculer sa complexité.

Exercice 2 : Opérations sur les ensembles

On se donne deux tableaux T et U de tailles n et m, on note $N = \max\{m, n\}$. On veut produire un tableau V contenant tous les éléments apparaissant dans T mais pas dans U.

1. Proposer un algorithme de coût $\mathcal{O}(N^2)$ pour effectuer cette opération.
2. On peut améliorer l'opération précédente en commençant par trier les tableaux, calculer la complexité d'une telle solution ?
3. Peut-on calculer le tableau V en coût $\mathcal{O}(N)$? Si oui, écrire l'algorithme.

(suite au verso)

Problème : Élément de rang k

On se donne un tableau T comprenant n éléments distincts. On appelle élément de rang k du tableau T, le $k$ième plus grand élément du tableau T (l'élément de rang 1 est donc le plus petit élément du tableau tandis que l'élément de rang n est le plus grand élément du tableau).

1. Écrire un algorithme de complexité $\mathcal{O}(n)$ qui calcule les éléments de rang 1 et de rang n.
2. Écrire un algorithme naïf permettant de calculer l'élément de rang k, calculer la complexité de votre algorithme et, si ce n'est pas le cas, modifier votre algorithme pour obtenir une complexité $\mathcal{O}(n \log n)$ dans le pire cas.

On rappelle l'algorithme *quicksort* randomisé :
- Choisir un élément au hasard (uniformément) parmi $T[1] \dots T[n]$ comme pivot
- Partitionner les éléments en plaçant les éléments plus petit que le pivot en début du tableau et les éléments plus grand que le pivot en fin de tableau.
- Appeler récursivement *quicksort* sur les deux sous-tableaux.

3. En s'inspirant de l'algorithme de *quicksort*, écrire un algorithme `QuickSelect`(T,k) qui calcule l'élément de rang k (*indication :* après avoir partitionné le tableau en deux, il facile de tester si l'élément de rang k est plus petit ou plus grand que le pivot.)

Soit $C(n,k)$ le nombre moyen de comparaisons qu'effectue `Quickselect` pour trouver l'élément de rang k d'un tableau de taille n

4. Montrer que (il est fortement suggéré de faire des dessins)

$$C(n,k) = (n-1) + \frac{1}{n}\sum_{i=1}^{k-1} C(n-i,k-i) + \frac{1}{n}\sum_{i=k+1}^{n} C(i,k)$$

On notera $M(n)$ et on admettra (question bonus) que

$$M(n) \leq \frac{2}{n}\sum_{p=n/2}^{n-1} M(p) + \mathcal{O}(n).$$

5. En déduire que $M(n) = \mathcal{O}(n)$.