28/01/2019

Université Internationale de Casablanca

Examen Algorithmique Avancée

Durée : 1h45

Exercice 1 (6 pts)

1- Donner la définition des concepts de complexité et d'optimalité. (2pts)
2- Donner un ordre de grandeur de complexité pour les algorithmes suivants (4pts)
   - Les algorithmes sub-linéaires
   - Les algorithmes linéaires
   - Les algorithmes polynomiaux
   - Exponentiels
   -

Exercice 2 (14 pts)

Soit un tableau d'entiers « T » et un entier « n ». On souhaite rechercher cet entier n dans le tableau T. Si cet entier existe dans le tableau, on souhaite récupérer son indice, sinon, la valeur -1 est renvoyée.

1- Donner l'algorithme de recherche naïf correspondant à une recherche exhaustive de « n » dans le tableau « T ». (2pt)
2- Donner les complexités au meilleur, au pire et en moyenne de cet algorithme. (2pts)

L'algorithme de recherche dichotomique est un algorithme qui peut s'appliquer à des tableaux ordonnés comprenant des éléments possédant un ordre total. Il permet de rechercher un élément donné dans le tableau en profitant du fait qu'il soit trié.

Considérons, ainsi, la fonction algorithmique RechercheDicPartielle(T,d,f,n) avec « T » un tableau, « d » l'indice de début considéré, « f » l'indice de fin considéré, et « n » la valeur recherchée. Si l'intervalle [d,f] est vide alors « n » est absent du tableau « T ». Sinon, soit « m » le milieu de cet intervalle. Si T[m]=n alors l'élément est trouvé en « m ». Sinon, si T[m]<n alors appeler la recherche récursivement en considérant les indice m+1 et f. Sinon, appeler la recherche récursivement sur les indices d et m-1.

3- En supposant que notre tableau est trié de manière ascendante,
   a. Ecrire l'algorithme de la fonction RechercheDicPartielle. (2pts)
   b. En considérant la fonction de la question précédente, écrire l'algorithme de recherche dichotomique d'un tableau « T » de taille « t ». (2pts)

10 → Bien

4- Peut-on classifier ce dernier algorithme comme un algorithme diviser pour régner ? Justifier votre réponse. (2pts)

On rappelle le théorème vu en cours :

Soient a >=1 et b >= 1 deux constantes, soit f (n) une fonction et soit T(n) une fonction définie pour les entiers positifs par la récurrence : T(n) = aT(n/b)+ f (n)

T(n) peut alors être bornée asymptotiquement comme suit :

a. *Si* $f(n) = O(n^{(\log_b a)-\varepsilon})$ *pour epsylon >0 alors* $T(n) = \theta(n^{(\log_b a)})$

b. *Si* $f(n) = \theta(n^{(\log_b a)})$ *alors* $T(n) = \theta(n^{(\log_b a)} \log n)$

5- Donner les complexités au meilleur et au pire de l'algorithme de recherche dichotomique. Comparer avec l'algorithme naïf et conclure. (2pts)

6- En supposant partir d'un tableau non trié, on se propose d'appliquer un tri par fusion avec en séquence un appel à notre algorithme de la recherche dichotomique.

a. Donner la complexité au pire de la séquence en la comparant à une recherche exhaustive naïve. (1pt)

b. Donner votre avis sur l'intérêt d'utiliser cette séquence en précisant le contexte où cela pourrait se justifier. (1pt)