Algorithmique - ROB3 - TP2
Diviser pour régner

Le but de ce TP est d'implémenter plusieurs algorithmes de type diviser pour régner pour divers problèmes sur des tableaux d'entiers, et pour chaque problème de comparer expérimentalement les algorithmes proposés.

Exercice 1 (Plus grand et plus petit élément d'un tableau)

Q 1.1 Coder un algorithme qui détermine à la fois le plus grand et le plus petit élément d'un tableau de taille n en réalisant $2(n-1)$ comparaisons en tout.

Q 1.2 Proposer un algorithme de type diviser pour régner pour accomplir la même tâche en $\frac{3}{2}n - 2$ comparaisons si la taille du tableau est une puissance de 2. Coder cet algorithme.

Q 1.3 Comparer expérimentalement le gain de temps obtenu en exécutant les deux algorithmes sur un même tableau de grande taille (plus précisément, on calculera le gain de temps moyen sur 20 tableaux de grande taille tirés aléatoirement).

Exercice 2 (Sous-tableau de poids maximum)

Dans cet exercice, on s'intéresse au *problème du sous-tableau maximum*. Etant donné un tableau $A[1,\ldots,n]$, le problème du sous-tableau maximum de A est de déterminer :

$$\mathrm{stm}(A) = \max\{\sum_{k=i}^{j} A[k] : 1 \leq i \leq j \leq n\}$$

Par exemple, on a $\mathrm{stm}([-3,4,5,-1,2,3,-6,4]) = 13$, valeur que l'on obtient pour le sous-tableau $[4,5,-1,2,3]$. Dans un premier temps, on considère l'algorithme de résolution ci-dessous :

```
max = max{A[1], ..., A[n]}
for i = 1 to n do
    for j = i to n do
        temp = 0
        for k = i to j do
            temp = temp + A[k]
        end for
        if temp > max then
            max = temp
        end if
    end for
end for
return max
```

Q 2.1 Coder cet algorithme. Quelle est sa complexité ? Tracer sa courbe de temps d'exécution.

Q 2.2 Proposer une variante en $\Theta(n^2)$. Coder cette variante et tracer sa courbe de temps d'exécution.

Dans la suite de l'exercice, on s'intéresse à des algorithmes de type diviser pour régner pour ce problème. Soit $m = \lfloor (n+1)/2 \rfloor$, $A_1 = A[1,\ldots,m]$ et $A_2 = A[m+1,\ldots,n]$. Le cas de base est le cas d'un tableau de taille 1, qu'on sait bien sûr résoudre en $\Theta(1)$ (temps constant). Soit $\mathrm{stm}_1 = \mathrm{stm}(A_1)$, $\mathrm{stm}_2 = \mathrm{stm}(A_2)$ et stm_3 la valeur maximum d'un sous-tableau qui contient à la fois $A[m]$ et $A[m+1]$.

Q 2.3 Expliquer comment calculer stm_3 en $\Theta(n)$. Coder l'algorithme correspondant.

Q 2.4 Dans le cadre d'une méthode diviser pour régner, supposons que l'on a calculé $\max_1$ et $\max_2$ (par des appels récursifs). En vous aidant de la réponse à la question précédente, donner le principe d'une opération de fusion des résultats des deux appels récursifs, et indiquer la complexité de cette opération. En déduire la complexité de la méthode diviser pour régner obtenue. Coder la méthode et tracer sa courbe de temps d'exécution.

On s'intéresse désormais à une autre approche, toujours de type diviser pour régner, afin de réduire encore la complexité de l'algorithme. Pour ce faire, on définit :

$$\begin{array}{rcl} \text{pref}(A) & = & \max\{\sum_{k=1}^{i} A[k] : 1 \leq i \leq n\}, \\ \text{suff}(A) & = & \max\{\sum_{k=i}^{n} A[k] : 1 \leq i \leq n\}, \\ \text{tota}(A) & = & \sum_{k=1}^{n} A[k], \end{array}$$

Autrement dit, $\text{pref}(A)$ (resp. $\text{suff}(A)$) est la valeur maximum d'un sous-tableau préfixe (resp. suffixe) de A, et $\text{tota}(A)$ est la somme des cases de A.

Q 2.5 Supposons qu'on vise à déterminer non plus simplement $\text{stm}(A)$, mais le triplet $(\text{stm}(A), \text{pref}(A), \text{suff}(A))$. Soit $(\text{stm}_1, \text{pref}_1, \text{suff}_1)$ et $(\text{stm}_2, \text{pref}_2, \text{suff}_2)$ les triplets retournés par les appels récursifs sur A_1 et A_2. Donner un petit exemple montrant que ce serait faux de faire la fusion en retournant $(\max\{\text{suff}_1 + \text{pref}_2, \text{stm}_1, \text{stm}_2\}, \text{pref}_1, \text{suff}_2)$ pour A.

Q 2.6 Supposons maintenant qu'on vise à déterminer le quadruplet $(\text{stm}(A), \text{pref}(A), \text{suff}(A), \text{tota}(A))$. Soit $(\text{stm}_1, \text{pref}_1, \text{suff}_1, \text{tota}_1)$ et $(\text{stm}_2, \text{pref}_2, \text{suff}_2, \text{tota}_2)$ les quadruplets retournés par les appels récursifs sur A_1 et A_2. Donner la formule permettant de déterminer le quadruplet pour A à partir de ces quadruplets. Quelle est la complexité de cette opération de fusion ? En déduire la complexité de la méthode diviser pour régner obtenue. Coder la méthode et tracer sa courbe de temps d'exécution.

Exercice 3 (Médiane)

La *médiane* d'une liste de nombres (qu'on suppose stockés dans un tableau dans l'exercice) est son 50ième centile : la moitié des éléments sont plus grands, et la moitié sont plus petits. Par exemple, la médiane de $[45, 1, 10, 30, 25]$ est 25, car c'est l'élément au centre quand les nombres sont triés. Si la liste a une longueur paire, il y a deux éléments qui forment le "centre", auquel cas nous adoptons ici la convention de considérer le plus petit des deux comme l'élément central. Une première méthode pour calculer la médiane de n nombres consiste bien sûr à les trier puis à sélectionner l'élément central.

Q 3.1 Coder un algorithme qui utilise le tri fusion pour déterminer la médiane. Donner sa complexité et tracer sa courbe de temps d'exécution en fonction du nombre n d'éléments.

Remarquons que le tri correspond à beaucoup plus de travail que ce dont on a réellement besoin pour déterminer la médiane : on souhaite uniquement l'élément central et on ne se soucie pas de la façon dont les autres éléments s'ordonnent. Afin de concevoir un algorithme récursif pour ce problème, nous allons nous intéresser à un problème plus général, le problème de sélection : étant donnée une liste S de nombres (non nécessairement tous distincts) et un entier k, déterminer le k^{eme} plus petit élément de S. Par exemple, si $k = 1$, le minimum de S est recherché, tandis que si $k = \lfloor |S|/2 \rfloor$, c'est la médiane.

Q 3.2 On se propose de développer une méthode de type diviser pour régner pour ce problème. Soit $selection(S, k)$ la fonction qui sélectionne le k^{eme} plus petit élément de S. Pour ce faire, on utilise un élément *pivot* v du tableau en fonction duquel on partitionne S en trois sous-ensembles : le sous-ensemble S_G des éléments plus petits que v, le sous-ensemble S_v des éléments égaux à v et le sous-ensemble S_D des éléments plus grands que v. Si $|S_G| < k \leq |S_G| + |S_v|$ alors v est le k^{eme} plus petit élément de S. Identifier l'appel récursif à réaliser après partition de S en S_G, S_v et S_D. Coder la méthode diviser pour régner correspondante si l'on suppose que le pivot v est choisi aléatoirement dans le tableau. *On ne demande pas de calculer la complexité théorique de cet algorithme.* Tracer la courbe de temps d'exécution en fonction de n, et comparer à la courbe précédente.

Exercice 4 (Shadows of the Knight)

Rendez-vous sur le site www.codingame.com.

Une fois rendu(e) sur le site, réalisez les opérations suivantes :

- Ouvrez dans le menu latéral de gauche l'onglet "ENTRAÎNEMENT".
- Dans la section "PUZZLE CLASSIQUE - MOYEN" cherchez et cliquez sur le puzzle "SHADOWS OF THE KNIGHT".
- Cliquez sur le bouton "RÉSOUDRE" en haut à droite de votre écran. Un éditeur de texte s'ouvre.
- Sélectionnez le langage de programmation C.

Pour résoudre le puzzle "Shadows of the Knight", vous devez coder votre réponse dans l'éditeur de texte qui vient de s'ouvrir. La description du puzzle est présente à gauche de votre écran. La réponse attendue est un algorithme qui utilise le principe diviser pour régner pour trouvez le plus rapidement possible le Joker.
Une fois votre réponse codée et le jeu de test passé avec succès, soumettez votre réponse sur le site.