POLYTECH MARSEILLE Aix*Marseille Université **HUGO3**

Algorithmique

TD n°3 : diviser pour régner

Exercice 1 – Application du théorème maître

Déterminez lequel des algorithmes suivants est le plus « rapide » pour résoudre des problèmes où n est grand :

1. un algorithme qui résout un problème de taille n en le divisant en 2 sous-problèmes de taille $n/2$ et qui fusionne les solutions en temps quadratique.
2. un algorithme qui résout un problème de taille n en le divisant en 4 sous-problèmes de taille $n/2$ et qui fusionne les solutions en $\sqrt{n}$ opérations.
3. un algorithme qui résout un problème de taille n en le divisant en 4 sous-problèmes de taille $n/2$ et qui fusionne les solutions en temps quadratique.

Exercice 2 – Application du théorème maître (bis)

Q 2.1 En utilisant le théorème maître, donnez un ordre de grandeur asymptotique pour le nombre d'opérations $T(n)$ dans les cas suivants :

1. $T(n) = 2T\left(\frac{n}{2}\right) + n^2$
2. $T(n) = 4T\left(\frac{n}{4}\right) + n$
3. $T(n) = 3T\left(\frac{n}{2}\right) + n$
4. $T(n) = 4T\left(\frac{n}{4}\right) + c$ avec $c > 0$

Exercice 3 – Recherche d'un élément dans un tableau

Q 3.1 Proposez un algorithme de type « diviser pour régner » qui recherche un élément x dans un tableau `tab` *trié par ordre croissant*. Si l'élément existe, on suppose dans ce cas qu'il est unique et l'algorithme renverra l'indice de cet élément dans le tableau (la première cellule d'un tableau a pour indice 0), sinon il renverra -1.

Q 3.2 Quelle est la complexité de cet algorithme ?

Exercice 4 – Puissance

Q 4.1 Proposez un algorithme de type « diviser pour régner » qui, étant donné deux nombres entiers x et y, renvoie la valeur de x^y (x puissance y).

Q 4.2 Quelle est la complexité de votre algorithme ?

Exercice 5 – Somme d'une suite géométrique

Q 5.1 Proposez un algorithme de type « diviser pour régner » qui, étant donné un nombre x et un entier positif ou nul n, renvoie la valeur de la somme définie par $\sum_{k=0}^{n} x^k$.

Q 5.2 Quelle est la complexité de votre algorithme ?

Exercice 6 – Recherche d'un élément dans un tableau (bis)

Q 6.1 Proposez un algorithme de type « diviser pour régner » qui, étant donné un tableau `tab` de taille n contenant des entiers distincts deux à deux, renvoie un booléen indiquant s'il existe un indice $i < n$ tel que `tab`$[i] = i$. Votre algorithme devra avoir une complexité dans le pire des cas en $O(n \log n)$.

Exercice 7 – Pièce défectueuse

On dispose de n pièces d'or toutes de même poids, sauf une défectueuse qui est plus légère. Le problème consiste à retrouver la pièce défectueuse en utilisant une balance de type Roberval (balance à deux plateaux qui permet de dire uniquement s'il y a un plateau plus lourd que l'autre) :

On part d'un ensemble de $n = 3^p$ pièces, dans lequel on cherche à déterminer la pièce la plus légère.

Q 7.1 On suppose que les n pièces sont réparties en trois ensembles de même taille. Montrez qu'à l'aide d'une balance de type Roberval, on peut décider dans quel ensemble se trouve la pièce la plus légère en une seule pesée.

Q 7.2 Proposez un algorithme de type « diviser pour régner » pour résoudre ce problème.

Q 7.3 On note $T(n)$ le nombre de pesées nécessaires pour trouver la pièce défectueuse dans un ensemble de $n = 3^p$ pièces. Donnez l'expression de la récurrence vérifiée par $T(n)$.

Q 7.4 Résolvez cette récurrence d'une part directement et d'autre part en utilisant le théorème maître.

Exercice 8 – Recherche de l'élément majoritaire

Un élément x d'un tableau T contenant n élément est dit majoritaire si T contient au moins $n/2$ occurrences de x. Pour simplifier, ici, on supposera que n est une puissance de 2.

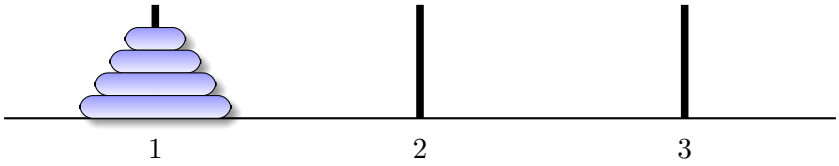
Q 8.1 Montrez que s'il existe un élément majoritaire dans T, alors x est également majoritaire dans au moins l'une des deux moitiés de T.

Q 8.2 Déduisez-en un algorithme de type « diviser pour régner » afin de trouver l'élément majoritaire (s'il existe).

Q 8.3 Quelle est la complexité de votre algorithme ?

Exercice 9 – Les tours de Hanoï

Le problème des tours de Hanoï est le suivant : il s'agit d'un jeu constitué de 3 piquets verticaux, que l'on nommera 1, 2 et 3. Sur l'un de ces piquets sont enfilés des disques de tailles décroissantes, comme le montre la figure ci-dessous :



Le jeu consiste à déplacer les disques du piquet 1 vers le piquet 3, en respectant les règles suivantes :

1. on ne peut déplacer les disques qu'un par un ;
2. un disque ne doit pas se retrouver au-dessus d'un disque plus petit que lui.

On souhaite ici afficher l'ensemble des déplacements permettant de résoudre un tel problème. Par exemple, pour un jeu avec 3 disques, on souhaite obtenir l'affichage suivant :

```
déplacer un disque du piquet 1 vers le piquet 3
déplacer un disque du piquet 1 vers le piquet 2
déplacer un disque du piquet 3 vers le piquet 2
déplacer un disque du piquet 1 vers le piquet 3
déplacer un disque du piquet 2 vers le piquet 1
déplacer un disque du piquet 2 vers le piquet 3
déplacer un disque du piquet 1 vers le piquet 3
```

Pour résoudre aisément ce problème, on peut utiliser une fonction récursive fondée sur le principe suivant : pour déplacer un tas de n disques d'un piquet i vers le piquet $j \neq i$, il suffit de :

1. déplacer le tas de $n-1$ disques se trouvant sur le dessus du piquet i vers le piquet $k \neq i, j$ (on notera que l'on a toujours $i + j + k = 6$ puisque les 3 piquets sont numérotés de 1 à 3, donc $k = 6 - i - j$) ;
2. déplacer le dernier disque du piquet i vers le piquet j ;
3. déplacer le tas de $n-1$ disques du piquet k vers le piquet j.

Q 9.1 Écrivez une fonction récursive `Hanoi` qui, étant donné un nombre entier `n` de disques, et deux entiers `i` et `j`, affiche les déplacements à effectuer pour déplacer les `n` disques du piquet `i` vers le piquet `j`.

Q 9.2 Écrivez le code en langage C correspondant.

Q 9.3 Quelle est la complexité de cette fonction ?

Exercice 10 – Recherche du sous-tableau maximum

Étant donné un tableau `tab` de n nombres, le problème du sous-tableau maximum (ou « maximum segment sum » en anglais) consiste à trouver une suite d'éléments contigus dont la somme est maximale. On ne s'intéresse pas aux indices des éléments de cette suite, mais seulement à la valeur de sa somme. En effet, à l'origine, ce problème était une modélisation simplifiée de l'estimation du maximum de vraisemblance de motifs dans des images numérisées. Dans ce problème, on cherche donc à résoudre :

$$S(\texttt{tab}) = \max \left\{ \sum_{k=i}^{j} \texttt{tab}[i] : 0 \leq i \leq j < n \right\}.$$

Q 10.1 Proposez un algorithme « brute-force » pour résoudre ce problème.

Q 10.2 Indiquez la complexité de votre algorithme.

Q 10.3 Pour $n \geq 2$, soit $m = \lfloor n/2 \rfloor$. Proposez un algorithme pour calculer la valeur maximum d'un sous-tableau contenant à la fois $\texttt{tab}[m]$ et $\texttt{tab}[m+1]$.

Q 10.4 Notons `tab1` et `tab2` les sous-tableaux de `tab` contenant respectivement les éléments d'indices 0 à m et $m+1$ à $n-1$. Supposons que l'on connaisse $S(\texttt{tab1})$ et $S(\texttt{tab2})$. En vous servant de la question précédente, déduisez-en comment calculer $S(\texttt{tab})$.

Q 10.5 Proposez un algorithme de type « diviser pour régner » afin de résoudre le problème du sous-tableau maximum.

Q 10.6 Quelle est la complexité de votre algorithme ?