Université Lille1 Sciences et Technologies

Info 204 - ASD

2012–2013
Licence STS - Semestre 4

DS1 - documents de cours, TD, TP autorisés

Note : lorsque, dans une question, est demandée la complexité, c'est une fonction de la taille de la donnée qui est attendue. Lorsque c'est le comportement asymptotique, la réponse attendue est soit une notation $\mathcal{O}$, soit une notation Θ soit une notation Ω.

Vous pourrez utiliser toute fonction vue en cours, TD, TP à la condition expresse de spécifier correctement les entrées et les sorties.

Exercice 1 : Questions de cours

Q 1.1 Si je prouve qu'un algorithme est en $\Theta(n^2)$, peut-il être en $\mathcal{O}(n)$ sur certains exemplaires ? Répondez par oui ou non.

Q 1.2 Si je prouve qu'un algorithme est en $\Omega(n \log n)$, peut-il être en $\mathcal{O}(n)$ sur certains exemplaires ? Répondez par oui ou non.

Q 1.3 Un algorithme s'exécute dans le meilleur des cas en réalisant $c_m(n) = 2n^2 + 1$ opérations et dans le pire des cas en $c_p(n) = n^4 + 2n + 2$ opérations pour une donnée de taille n. Indiquez et prouvez son comportement asymptotique.

Q 1.4 Le nombre d'appels récursifs d'un algorithme récursif est donné par la formule :

$$c(n) = \begin{cases} 0 & \text{si } n = 0 \text{ ou } 1 \\ 8c\left(\frac{n}{4}\right) + 2n & \text{sinon} \end{cases}$$

Indiquez et prouvez son comportement asymptotique.

Exercice 2 : Meilleur des cas / pire des cas

On propose une stratégie de tri, basée sur le renversement d'une partie d'un tableau. Renverser une partie d'un tableau c'est simplement inverser l'ordre des éléments. L'exemple suivant renverse d'abord les éléments des indices 0 à 3 puis des indices 0 à 5 :

```
(0,1,2,5,4,3) --> renverser(t,0,3) --> (5,2,1,0,4,3) -- renverser(t,0,5) --> (3,4,0,1,2,5)
```

La stratégie de tri repose sur l'identification, à chaque étape, de la position de l'élément maximal, qu'on va replacer, en 2 renversements à sa bonne place, comme on l'a fait pour 5 dans l'exemple ci-dessus.

```
soit t le tableau à trier et n sa longueur
pour i allant de n-1 a 1 faire
  soit p la position de l'élément maximal dans t(0..i)
  renverser les éléments de t entre les positions 0 et p
  // a cette etape t.(0) contient la valeur maximale de t(0..i)
  renverser les éléments de t entre les positions 0 et i
  // a cette etape t.(i) contient la valeur maximale de t(0..i)
fin pour
```

L'opération qui nous importe dans cet exercice est l'opération d'affectation. On suppose que la fonction de recherche de la position où se situe le maximum dans `t(0..m)` s'exécute en $m - 1$ affectations et que le renversement s'exécute en $3 * \lfloor \frac{m}{2} \rfloor$ affectations.

Analyse du meilleur des cas

Q 2.1 Décrivez un meilleur des cas pour le tri proposé.

Q 2.2 Déroulez un exemple de meilleur des cas d'un tableau de taille 6.

Q 2.3 Donnez le comportement asymptotique de l'algorithme de tri pour le meilleur des cas. Justifiez

Analyse du pire des cas

Q 2.4 Décrivez un pire de cas pour le tri proposé.

Q 2.5 Déroulez un exemple de pire des cas d'un tableau de taille 6.

Q 2.6 Donnez le comportement asymptotique de l'algorithme de tri pour le pire des cas. Justifiez.

Exercice 3 : Stratégie diviser pour régner

On considère un problème dont la donnée est un tableau et la solution un tableau de même taille. Pour résoudre ce problème sur des tableaux de grande taille, on propose les deux algorithmes suivants :

Algo A un algorithme qui traite un tableau de taille n en le divisant en 2 sous-problèmes de taille $\frac{n}{2}$ et qui combine les solutions en temps n^2,

Algo B un algorithme qui traite un tableau de taille n en le divisant en 4 sous-problèmes de taille $\frac{n}{4}$ et qui combine les solutions en temps $\sqrt{n}$.

Q 3.1 Donnez les équations de récurrence des deux algorithmes.

Q 3.2 En justifiant votre réponse, indiquez lequel des deux il faut choisir.

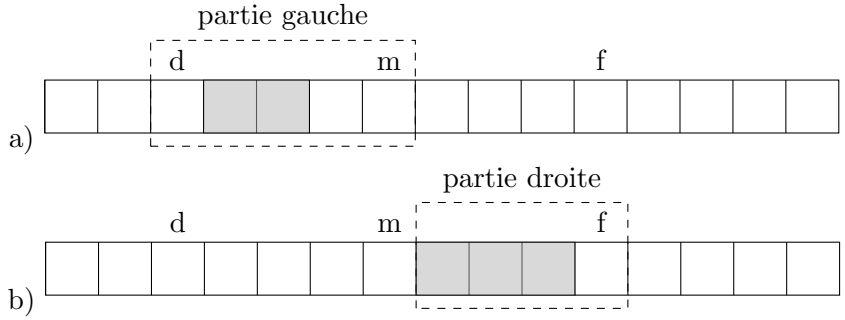
Exercice 4 : Elaborer et comprendre un algorithme

On dispose d'un tableau d'entiers relatifs de taille n. On cherche à déterminer la somme maximale d'une suite d'entiers consécutifs du tableau. Par exemple, pour le tableau `[|-1; 9; -3; 12; -5; 4|]`, la solution est 18 (somme des éléments de `T` entre les positions 1 et 3 : `[|9; -3; 12|]`).

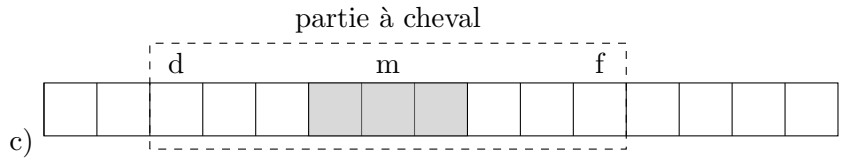
Q 4.1 Ecrivez en CAML un algorithme simple, non récursif, qui calcule la somme maximale d'une suite d'entiers consécutifs d'un tableau t.

Q 4.2 Donnez et justifez le comportement asymptotique de votre algorithme.

On se propose d'adopter une stratégie diviser pour régner pour traiter ce problème de manière un peu plus efficace. Admettons être en train de rechercher la somme maximale dans la tranche du tableau comprise entre les indices d (début) et f (fin). Si on divise la tranche en deux, la somme maximale se trouve : a) soit dans la partie gauche, b) soit dans la partie droite, c) soit dans la partie qui se trouve à cheval.



c)



Il suffit donc de calculer ces trois sommes puis de garder le maximum des trois.

Les sommes maximales sur les parties gauche et droite sont obtenues récursivement en réappliquant la même stratégie.

La somme maximale sur la partie à cheval est obtenue en utilisant une fonction prédéfinie `somme_a_cheval` qui prend en entrée un tableau t et les positions d (début) et f (fin) qui déterminent la tranche sur laquelle le calcul va être effectué. Le coût de cette fonction est $f - d$ additions d'éléments du tableau.

Q 4.3 Ecrivez le code CAML de la fonction récursive `somme_maximale` à trois arguments : le tableau t et les indices d et f définissant la tranche.

Q 4.4 Donnez l'équation de récurrence qui compte le nombre d'additions d'éléments du tableau.

Q 4.5 Quel est le comportement asymptotique de ce nouvel algorithme ?

On rappelle que :

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

et que

$$\log_b a = \frac{\log_c a}{\log_c b}$$

et enfin que si $c = \log_b a$ alors $b^c = a$