

Date de remise : **jeudi, le 22 octobre 2009, avant 12h**. Vous pouvez remettre votre devoir directement à moi dans la classe ce jour, ou bien vous pouvez l'apporter à mon bureau, SITE 5-064, avant 12 :00h (le mettent sous ma porte si je ne suis pas là). **Notez que les devoirs remis en retard de plus de 24 heures ne seront pas corrigés ; les devoirs remis en retard de moins de 24 heures perdront 9 points (15%).**

Dans votre devoir, assurez-vous s'il vous plaît d'expliquer toutes vos solutions CLAIEMENT, sinon vous perdrez des points. Pour être précis, dans n'importe quelle analyse de cas pire, vous devez exposer la taille d'entrée, les opérations de base étant comptées et une entrée générale de votre taille d'entrée fixée qui donne le cas pire (vous devez expliquer ces choses dans votre solution même si quelque uns d'entre eux sont déjà donnés dans les questions). Dans la mesure du possible chaque fois simplifiez vos réponses en employant les formules appropriées trouvées dans l'Annexe A. Aussi, vous serez noté sur l'efficacité de n'importe quels algorithmes vous concevrez.

Ce devoir sera noté sur 60.

Exercice 1. [10 Points]

Supposons que vous voulez faire une analyse de cas moyenne de la version itérative de la Recherche Binaire pour $n=4$. Considérez que toute l'entrée possible inscrit S de taille 4 et la division de ces entrées comme suit :

Pour $i = 1, 2, 3, 4$, soit I_i la liste de toutes les entrées pour lesquelles le nombre x apparaît dans la position i . Pour les entrées pour lesquelles x n'apparaît pas dans la liste, nous avons:

- I_5 est l'ensemble de toutes les entrées pour lesquelles le nombre x se trouve entre $S[1]$ et $S[2]$,
- I_6 est l'ensemble de toutes les entrées pour lesquelles le nombre x se trouve entre $S[2]$ et $S[3]$,
- I_7 est l'ensemble de toutes les entrées pour lesquelles le nombre x se trouve entre $S[3]$ et $S[4]$,
- I_8 est l'ensemble de toutes les entrées pour lesquelles le nombre x est plus petit que tous les nombres dans S ,
- I_9 est l'ensemble de toutes les entrées pour lesquelles le nombre x est plus grand que tous les nombres dans S .

Faites une analyse du cas moyen de la Recherche Binaire itérative des listes de taille 4 utilisant la division ci-dessus des entrées et comptant les comparaisons de x avec des nombres dans S . Vous devrez supposer que toutes les classes d'entrée ont la même probabilité d'arriver. Aussi, montrez s'il vous plaît votre travail, en incluant des valeurs de $t(I_i)$ et $p(I_i)$ pour $i = 1, \dots, 9$.

Exercice 2. [10 Points]

- a) [4 points] Prouvez en utilisant les limites et la Règle de L'Hôpital :
 $n \log n$ est $\Theta(k n \log(kn))$ pour n'importe quel k constant > 0 .
- b) [4 Points] Prouvez en utilisant les limites et la Règle de L'Hôpital :

$6n^2 + 20n$ est $\Omega(n + 5)$.

c) [2 Points] Prouvez (en employant les définitions formelles) ou réfutez (en donnant un contre exemple)

si $t(n) \in O(g(n))$, alors $g(n) \in \Omega(t(n))$.

Exercice 3. [15 Points]

Le problème de somme maximal continu est comme suit : Étant donné un tableau X d'entiers, trouvez la somme maximale trouvée dans n'importe quel sous-tableau continu de X. Par exemple, pour $X = [31, -41, 59, 26, -53, 58, 97, -93, -23, 84]$ la réponse est 187 (la somme de nombres dans positions 3, 4, 5, 6 et 7).

Considérez l'algorithme récursif suivant pour la résolution de ce problème pour un Tableau X de longueur n (bien sûr une solution vient d'un appel à MaxSum (1, n)) :

```
void MaxSum(int L, int U)
{
int Sum, MaxToLeft, MaxToRight, M, I, MaxCrossing, MaxInA, MaxInB;
if (L >= U) { //cas de base
    if (L > U) // Tableau d'élément Zero
        return 0.0;
    if (L = U) //Tableau d'élément un
        return max(0.0, X[L]); //{***}
}
else { //Cas recursive
    M = (L + U) / 2; //A est X[L..M], B est X[M+1..U]
    //Find the max crossing to the left
    Sum = 0.0; MaxToLeft = 0.0;
    for (I = M; I >= L; I--) {
        Sum = Sum + X[I]; //{***}
        MaxToLeft = max(MaxToLeft, Sum) //{***}
    } //endfor
    // Trouver Max se croisant à droite
    Sum = 0.0; MaxToRight = 0.0;
    for (I = M+1; I <= U; I++) {
        Sum = Sum + X[I]; //{***}
        MaxToRight = max(MaxToRight, Sum); //{***}
    } //endfor
    Maxcrossing = MaxToLeft + MaxToRight; //{***}
    MaxInA = MaxSum(L, M);
    MaxInB = MaxSum(M+1, U);
    return max(MaxCrossing, MaxInA, MaxInB); //{***}
} //endelse
}
```

a) [14 Points]

Faites une analyse de cas pire pour cet algorithme, assumant que n est une puissance de 2 (c'est-à-dire $n = 2^k$, $k \geq 0$). Dans cette analyse de cas pire, employez n comme la taille d'entrée et comptez le nombre **d'additions** impliquant les éléments de X et l'opération de **trouver le Max** d'un ensemble de nombres (pour être clair, j'ai marqué ces opérations dans l'algorithme par **{***}** - ceux-ci sont exactement ce que vous comptez). Cette analyse sera semblable à celle de Mergesort, mais si vous obtenez la même relation de récurrence, vérifiez de nouveau, cela doit être différent. Soyez sûr que dans votre analyse du cas pire vous faites clairement la chose suivante :

- i) Ecrivez la relation de récurrence $W(n)$ pour cet algorithme
 - ii) "résolvez" cette relation de récurrence en employant la substitution arrière (développez $W(n)$ jusqu'à ce que vous pouvez deviner la solution)
 - iii) Utilisez l'induction pour prouver que votre solution de ii) est correcte.
- b) [1 Point] Trouvez la complexité de cet algorithme, basé sur votre analyse en a).

Exercice 4. [4 Points]

- a) [2 Points] Trouvez le nombre exact total de multiplications, d'additions et soustractions (tous ensemble) pour l'algorithme classique pour la multiplication matricielle, et l'algorithme de Strassen pour la multiplication matricielle pour $n=16$ (montrez les formules que vous employez). Quel est plus rapide pour cette valeur de n ?
- b) [2 Points] Écrivez la formule récursive pour l'algorithme de Strassen, comptant seulement des multiplications, quand le seuil employé pour l'algorithme est $n=4$. Expliquez votre réponse brièvement.

Exercice 5. [5 Points]

Supposons que l'on vous donne une liste de n nombres à trier qui est dans l'ordre décroissant. Vous pouvez supposer que n est une puissance de 2, c'est-à-dire $n=2^k$ pour quelque $k \geq 0$. Écrivez une relation de récurrence pour l'analyse de Mergesort pour ce type d'entrée et expliquez clairement comment vous l'avez obtenu. Vous ne devez pas résoudre cette relation de récurrence.

Exercice 6. [16 Points]

a) [10 Points]

Concevez l'algorithme « diviser pour régner » qui recherche une valeur x dans une table de taille n par m (un tableau bidimensionnel avec n ligne et m colonnes). Cette table est triée le long des lignes et colonnes comme suit :

$$\begin{aligned} \text{table}[i][j] &\leq \text{table}[i][j+1] \\ \text{table}[i][j] &\leq \text{table}[i+1][j] \end{aligned}$$

Votre algorithme sera noté sur sa clarté et son efficacité (en termes de sa classe de complexité). Il doit être écrit en pseudocode semblable à celui du manuel du cours. Pour aider le correcteur, votre description du pseudocode doit être précédée d'une description française claire qui donne l'idée qu'il y a derrière votre algorithme (par exemple expliquez comment il marche et les idées qu'il y a derrière de manière claire).

b) [6 Points]

Faites une analyse de cas pire de votre algorithme de la question a). Pour l'analyse vous pouvez assumer que votre table est n par n , où n est une puissance de 2, $n >= 1$.

Employez n comme votre taille d'entrée et comptez les comparaisons de x avec des nombres dans la table comme l'opération de base. (Note : comptez s'il vous plaît les comparaisons de branche dans votre algorithme comme des comparaisons séparées, par exemple, si vous comparez $x > k$ et $x < k$ dans votre algorithme, vous les comptez comme 2 comparaisons).

Trouvez une formule récursive pour $W(n)$ et résolvez-le employant la substitution en arrière (développez $W(n)$ jusqu'à ce que vous pouvez deviner la solution). Vous n'avez pas besoin de vérifier votre solution utilisant l'induction. Finalement, exposez la complexité de votre algorithme.