USTHB - Département d'Informatique
Master M1 SSI Complexité

Bab-Ezzouar le 08 Janvier 2012

Epreuve finale : Complexité et algorithmique avancé
(Durée 1h30 mn)

Exercice 1 Soit l'équation de récurrence :

$$T(n) = \begin{cases} 1 & si\ n = 1 \\ \sum_{i=1}^{n-1} T(i) + 1 & si\ n \geq 2 \end{cases}$$

a. Calculez T(n) - T(n-1); pour n >= 2
b. Résoudre T(n)

Solution : **(4 pts)**

a. T(n) – T(n-1) = ($\sum_{i=1}^{n-1} T(i) + 1$) – ($\sum_{i=1}^{n-2} T(i) + 1$) = T(n-1). **(2 pts)**
b. T(n) – T(n-1) = T(n-1) ⇒ T(n) = 2*T(n-1)

Donc on peut réécrire T(n) avec l'expression suivante :

$$T(n) = \begin{cases} 1 & si\ n = 1 \\ 2 * T(n-1) & si\ n \geq 2 \end{cases}$$

Par substitution répétées on obtient :

T(n) = 2^{n-1}. **(2 pts)**

Exercice 2 Soient A(x) = 3x + 4x + 1 et B(x) = x + 2x + 5 deux polynômes de degré n = 2.

a. Calculer l'expression du polynôme C(x) = A(x) * B(x) et donnez la complexité de ce calcul en fonction de n. Quelle est la complexité de ce calcul si les complexités de A(x) et B(x) sont respectivement m et n ?
b. Soit I = [0..4]. Evaluez C(x) dans I (l'ensemble des couples (x, y) tels que x ∊ I et y = C(x)).
c. Donnez la complexité totale de l'algorithme qui calcule et évalue C(x) dans I.
d. Evaluez A(x) dans I et B(x) dans I.
e. Calculer, à partir des évaluations précédentes, A(x) * B(x) dans I. Donnez la complexité de ce calcul et justifiez votre réponse.

Solution : **(5 pts)**

a. C(x) = A(x) * B(x) = (3x + 4x + 1)*(x + 2x + 5) = (3x *x + 3x *2x + 3x *5) + (4x*x + 4x*2x + 4x*5) + (1*x + 1*2x + 1*5). **(0,5)**

Pour n = 2, le produit des deux polynômes de degré 2 nécessite 9 opérations de base (2+1) , on peut déduire donc que de manière générale se calcul s'effectue avec une complexité égal à (N+1) , qui est donc de l'ordre de O(N). **(0,75)**

Dans le cas où les deux polynômes sont de degré différents respectivement m et n alors la complexité du calcul est de l'ordre de O(n*m). **(0,75)**

b. D'après ce qui précède $C(x) = 3x^4 + 10x^3 + 24x + 22x + 5$, donc dans I = [0..4]
C(x) = {(0, 5), (1, 64), (2, 273), (3, 800), (5, 1885)}. **(0,75)**

c. L'évaluation de C(x) dans I est en O(n), donc la complexité totale est égal à
O(n + n) = Max (O(n), O(n)) = O(n) étant donné que l'algorithme générale calcule C(X) puis l'évalue dans I. **(0,75)**

d. A(x) = {(0, 1), (1, 8), (2, 21), (3, 40), (4, 65)}.
Et B(x) = {(0, 5), (1, 8), (2, 13), (3, 20), (4, 29)}. **(0,75)**

e. D'où, A(x) * B(x) = {(0, 5), (1, 64), (2, 273), (3, 800), (5, 1885)} qui est en O(n) puisque toutes les étapes (évaluations) sont chacune en O(n). **(0,75)**

Exercice 3 Soit x un entier en représentation binaire contenu dans un tableau de n cases :

1	0	0	1	0	1	1	1	1	1

a. Ecrire un algorithme qui retourne le tableau qui contient la valeur binaire de x+1 (sans passer par le calcul décimal). Donnez sa complexité.

b. En déduire un algorithme qui calcule x+k, pour k entier donné. Donnez sa complexité.

Solution : **(5 pts)**

a. Fonction incrementation (E/S : A :tableau de [1 .. n] entier ; n : entier ;) **(3 pts)**

```
Début
i := n ;
Tant que A[i] = 1 faire A[i] = 0 ;
                        i := i+1 ;
                    Fait ;
A[i] = 1 ;
Fin ;
```

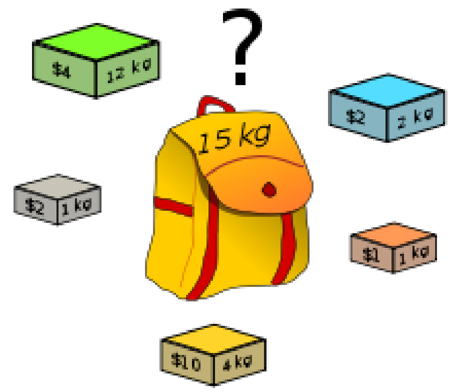
Cet algorithme est en O(n).

b. Pour calculer x+k on exécute l'équivalence de x+1+1+1+ …+ 1 l'appel suivant :

Pour i := 1 à k faire incrementation(A, n) ; Fait ;

Qui est en O(k*n) **(2 pts).**

Exercice 4 Le **problème du sac à dos** modélise la situation du remplissage du sac à dos d'un alpiniste, ne pouvant supporter plus d'un certain poids W, avec tout ou une partie d'un ensemble d'aliments noté A={a_1, …, a_n}, nécessaires pour son assomption, ayant chacun un poids w et une valeur nutritive v. Les objets mis dans le sac à dos doivent maximiser la valeur totale V, sans dépasser le poids maximum W.



- Comment peut-on représenté une solution à ce problème ?
- Proposez un algorithme de validation pour démontrer que le problème du sac à dos est NP ?

Exemple : Le sac à dos illustré possède un poids maximal W = 15 kg, et l'alpiniste nécessite une valeur nutritive pour son assomption supérieur à V= 10.

Solution : **(6 pts)**

Représentation possible pour la solution du problème du sac à dos **(1 pt)**

1. Un vecteur binaire de taille N (même taille que l'ensemble des aliments) tel que S[i] = 1 si le $i^{ème}$ aliment est dans le sac à dos et 0 sinon.
2. Un vecteur d'enregistrements qui regroupe l'aliment, son poids et sa valeur nutritive en même temps.

Structure { aliment : chaine de caractère ou entier ;
Poids, valeur_nutritive : entier ;} boite_aliment ;

3. Vecteur d'entier représentant l'indice des éléments de A qui figurent dans le sac à dos.

Les première et dernière représentations, nécessitent de faire appel à deux autres tableaux pour la représentation du poids (Poids[i]) et de la valeur nutritive (V_nutritive[i]) de chaque aliment.

Algorithme de validation **(4 pts)**

```
Début
    i :=1 ;
    p_sac :=0 ; // poids totale du contenu du sac
    v_sac :=0 ; //valeur nutritive totale du contenu du sac
    Valide = vrai ;
    tant que (i<=n et valide = vrai) faire
                si S[i] = 1 alors p_sac := p_sac + Poids[i] ;
                        v_sac := v_sac + V_nutritive[i];
                        Si p_sac > P alors valide = faux;
                        Sinon i := i+1 ;
                        Fsi ;
                Fsi ;
    Fait ;
```

Si valide = vrai et v_sac > V alors écrire (‘solution valide ‘) ;
    Sinon écrire(‘solution non valide’) ;
Fsi ;
Fin.

Cet algorithme est de complexité O(n) donc polynomiale de degré k = 1, ce qui permet de déduire que la problème du sac à dos est de classe NP. **(1 pt)**