

Enseirb, Filière Télécom, Semestre 1

Algorithmes et Structures de Données

Epreuve proposée par Robert Cori

21 janvier 2008, durée 2 heures

Le polycopié, les notes de cours et de travaux dirigés sont autorisées, à l'exclusion de tout autre document.

Exercice 1 : Barème envisagé : 4 points

On considère des suites formées par les 6 caractères suivants : parenthèse ouvrante ou fermante, crochet ouvrant ou fermant, accolade ouvrante ou fermante. Soit :

() [] { }

On note $u_1 u_2 \dots u_p$ une telle suite. On dit qu'elle est cohérente si on peut associer à chaque parenthèse ouvrante une parenthèse fermante située après elle, à chaque crochet ouvrant un crochet fermant situé après lui, et à chaque accolade ouvrante une accolade fermante située après elle ; de façon telle qu'il n'y ait pas de *croisement*.

De manière plus précise si à u_{i_1} on associe u_{j_1} et à u_{i_2} on associe u_{j_2} alors on ne doit pas avoir $i_1 < i_2 < j_1 < j_2$.

Par exemple la suite $(\{ \}) [()] \{ \}$ est cohérente alors que $(\{ \}) [()] \{ \}$ ne l'est pas.

On vous demande de fournir un algorithme utilisant une pile qui vérifie si une suite est cohérente. La suite donnée sera représentée par un tableau u tel que $u[i] = 1$ représente une parenthèse ouvrante, $u[i] = -1$ représente une parenthèse fermante, $u[i] = 2$ représente un crochet ouvrant, $u[i] = -2$ représente un crochet fermant, enfin $u[i] = 3$, -3 représentent une accolade ouvrante ou fermante. Pour cela vous utiliserez les fonctions sur les piles données en cours sans avoir à les décrire en détail.

Exercice 2 : Barème envisagé : 3 points

Soit les 6 symboles a, b, c, d, e, f, g de fréquences respectives 5, 3, 3, 2, 10, 4, 1 (c'est à dire que a apparaît 5 fois, b 2 fois etc.). Comment coder ces symboles par une suite de 0 et de 1 de façon à obtenir un codage de taille minimale. Préciser les différentes étapes de votre calcul.

Exercice 3 : Barème envisagé : 7 points

On se propose de calculer une plus longue sous-suite croissante d'une suite de nombres en utilisant des techniques inspirées de la programmation dynamique.

Pour ceci on commence par un algorithme simple qui sera utilisé par la suite.

Question 1. Soit u un tableau de n nombres entiers ($u[0], u[1], \dots, u[n-1]$) tous distincts. On vous demande d'écrire en détail la fonction `suivant(i)` (où i est un indice compris entre 0 et $n-1$), dont le résultat est j , indice du premier élément du tableau situé après $u[i]$ et supérieur ou égal à $u[i]$. Le résultat doit être -1 si tous les éléments situés après $u[i]$ lui sont inférieurs. Ainsi on doit avoir (si $u[i] = j$ est différent de -1) :

$$j > i \quad u[j] > u[i] \quad u[k] < u[i] \text{ pour } i < k < j$$

Par exemple, pour le tableau 6, 2, 5, 4, 9, 7, 8 on a $u[2] = 5$ ainsi `suivant(2) = 4`, car $u[4] = 9 > 5$ et $u[3] = 4 < 5$. On a aussi `suivant(4) = -1`.

Question 2. Une sous-suite croissante d'une suite u_0, u_1, \dots, u_{n-1} est donnée par $u_{i_1}, u_{i_2}, \dots, u_{i_p}$ tels que

$$i_1 < i_2 \dots < i_p \quad u_{i_1} < u_{i_2} < \dots < u_{i_p}$$

par exemple 2, 5, 9 est une sous-suite croissante de la suite considérée à la Question 1 mais ce n'est pas la plus longue car 2, 5, 7, 8 est plus longue.

On se propose de donner un algorithme qui calcule une sous-suite de longueur maximale. Tout d'abord on calcule cette longueur ; pour cela on note ℓ_i la longueur de la plus longue sous-suite croissante dont le premier élément est u_i , quelle est la valeur de ℓ_i si `suivant[i] = -1` ? On suppose calculé ℓ_j , quelle est la valeur de ℓ_i si `suivant[i] = j` ?

Question 3. Donner un algorithme qui calcule les ℓ_i en utilisant la fonction `suivant` que vous avez donné à la Question 1.

Question 4. Comment compléter l'algorithme précédent afin qu'il affiche une suite de longueur maximale ?

Exercice 4 : Récursivité. Barème envisagé : 6 points

On se propose de trouver un algorithme rapide pour multiplier deux polynômes.

Question 1. Soit deux polynômes de degré 1 : $P = ax + b$ et $Q = cx + d$, le produit de ces deux polynômes est un polynôme de degré 2 : $acx^2 + (ad + bc)x + bd$. Ainsi un calcul direct des trois coefficients de ce polynôme effectue 4 multiplications entre des nombres et une addition. Montrer que l'on peut calculer ces trois coefficients en n'effectuant que 3 multiplications, par contre il faudra aussi utiliser deux additions et deux soustractions . Aide : vous commencerez par calculer : $(a + b)(c + d)$.

Ceci peut être intéressant si on dispose d'une machine qui effectue les additions (et soustractions) beaucoup plus vite que les multiplications.

Question 2. Soit deux polynômes P et Q de même degré égal à $2n + 1$ (impair), montrez que l'on peut calculer le produit PQ en effectuant trois multiplications entre des polynômes de degré n , accompagnées d'additions de soustractions entre polynômes et des multiplications par le monôme x^{n+1} . Vous utiliserez pour cela une technique inspirée de la Question 1 et une décomposition d'un polynôme de degré $2n + 1$ en une somme de deux polynômes l'un contenant les monômes de plus bas degré l'autre ceux de plus haut degré.

Cette remarque est intéressante si l'on considère comme négligeable le nombre d'opérations nécessaires pour additionner des polynômes et multiplier un polynôme par un monôme par rapport au nombre d'opérations pour calculer le produit.

Question 3. Montrer que l'on a un résultat similaire si les deux polynômes sont de même degré pair.

Question 4. On représente un polynôme P de degré n par un tableau `pp` de taille $n+1$ contenant ses coefficients. On suppose données les fonctions suivantes sur les tableaux représentant des polynômes :

- `additionner(pp, qq)` qui calcule le tableau représentant le polynôme somme de deux polynômes représentés par les tableaux `pp` et `qq`.
- `soustraire(pp, qq)` qui calcule le tableau représentant la différence entre le polynôme représenté par le tableau `pp` et celui représenté par le tableau `qq`.
- `multMonome(pp, n)` qui calcule le tableau représentant le produit du polynôme représenté par le tableau `pp` par le monôme x^n .
- `tete(pp, n)` qui calcule le tableau contenant les n premiers éléments du tableau `pp`
- `queue(pp, n)` qui calcule le tableau contenant les n derniers éléments du tableau `pp`

On vous demande de donner en détail un algorithme récursif :

`produit(pp, qq, d)`

de calcul du tableau représentant le produit de deux polynômes de même degré, où `pp`, `qq` sont les tableaux représentant ces polynômes et où `d` est le degré de ces deux polynômes.