Examen de 2nde session - 24 juin 2010
Documents de cours, TD, TP autorisés
Ouvrages interdits

Exercice 1 : Cours (4 points)

Pour cet exercice, notation QCM : réponse fausse = des points en moins.

Q 1.1 Quelle est la complexité de la recherche fructueuse d'un élément dans une table de hachage en adressage ouvert contenant n éléments pour laquelle il y a eu une et une seule collision à chaque insertion (sauf pour la première, évidemment) ?

Q 1.2 Quelle est la complexité de la recherche fructueuse d'un élément dans une table de hachage à résolution des collisions par chaînage contenant n éléments pour laquelle il y a eu collision à chaque insertion (sauf pour la première, évidemment) ?

Exercice 2 : Ensembles (8 points)

On souhaite manipuler des ensembles disjoints d'entiers compris entre 1 et une valeur maximale `MAX`, c'est-à-dire des ensembles tels que, étant donné deux ensembles e_1 et e_2, l'intersection de e_1 et e_2 est vide. De plus, parmi les éléments de chaque ensemble l'un se voit attribué un rôle particulier, on le nomme le représentant.

La structure de données adoptée pour stocker ces ensembles est un tableau, indicé de 1 à `MAX` :

```
1          type
2            TABLEAU_D_ENSEMBLES : array [1..MAX] of INTEGER;
```

Si `t` est un `TABLEAU_D_ENSEMBLES`, alors :

- `t[i]` vaut -1 si `i` n'appartient à aucun ensemble,
- `t[i]` vaut 0 si `i` est le représentant d'un des ensembles,
- `t[i]` vaut `j` si `i` est dans l'ensemble ayant pour représentant `j`.

Q 2.1 Donner le contenu du tableau si `MAX` vaut 10 et qu'il existe deux ensembles : $\{5, 6, 2\}$ ayant pour représentant 2, et $\{3, 9, 1\}$ ayant pour représentant 9.

Q 2.2 Ecrire en PASCAL un algorithme capable d'afficher la liste des ensembles d'un `TABLEAU_D_ENSEMBLES` passé en paramètre dont la complexité en espace est $\Theta(1)$:

```
procedure afficher (t : TABLEAU_D_ENSEMBLES);
```

L'élément représentant sera précédé d'un '$\star$'. On ne se soucie ni de l'ordre d'affichage des éléments dans un ensemble, ni de l'ordre d'affichage des ensembles.

Q 2.3 Justifier le fait que la complexité en espace de votre algorithme est $\Theta(1)$.

Q 2.4 Quelle est la complexité asymptotique en nombre de comparaisons dans le pire des cas et dans le meilleur des cas ? Justifier très clairement.

Q 2.5 Ecrire en PASCAL un algorithme capable d'afficher la liste des ensembles d'un `TABLEAU_D_ENSEMBLES` passé en paramètre dont la complexité en espace est $\Theta(\texttt{MAX})$ et dont la complexité en nombre de comparaisons est inférieure à celle de la question 2.2 :

```
procedure afficher_bis (t : TABLEAU_D_ENSEMBLES);
```

L'élément représentant sera précédé d'un '$\star$'. On ne se soucie ni de l'ordre d'affichage des éléments dans un ensemble, ni de l'ordre d'affichage des ensembles.

Q 2.6 Justifier le fait que la complexité en espace de votre algorithme est $\Theta(\texttt{MAX})$.

Q 2.7 Quelle est la complexité asymptotique en nombre de comparaisons dans le pire des cas et dans le meilleur des cas ? Justifier très clairement.

Un ensemble est sous *forme normale* si son représentant est la plus petite valeur de l'ensemble.

Q 2.8 Ecrire en PASCAL un algorithme qui met les ensembles contenus dans un tableau t sous forme normale :

```
procedure forme_normale (var t : TABLEAU_D_ENSEMBLES);
```

Q 2.9 Quelle est la complexité asymptotique dans le pire des cas en nombre de comparaisons de la mise en forme normale ?

On souhaite maintenant disposer d'un algorithme capable de décider si un tableau d'ensembles est inclus dans un autre. Etant donnés deux tableaux d'ensembles t_1 et t_2, on dira que t_1 est inclus dans t_2 si chaque ensemble de t_2 s'exprime comme l'union d'ensembles de t_1.

Par exemple, si $t_1 = \{\{\star 1\}, \{\star 3\}, \{\star 4, 5, 6\}\}$ et $t_2 = \{\{\star 1, 3\}, \{4, \star 5, 6\}\}$, alors t_1 est inclus dans t_2 puisque $\{\star 1\} \cup \{\star 3\} = \{\star 1, 3\}$ et $\{\star 4, 5, 6\} = \{4, \star 5, 6\}$. On remarquera que l'élément représentant n'a pas d'importance.

Q 2.10 On considère dans cette question que t_1 et t_2 sont sous forme normale.

a. Si `t1[i] = -1`, que doit vérifier `t2[i]` pour que `t1` soit inclus dans `t2` ?

b. Si `t1[i] = 0`, expliquer brièvement pourquoi l'inclusion de `t1` dans `t2` est possible.

c. Si `t1[i] > 0` et `t1[i] < t2 [i]`, expliquer brièvement pourquoi l'inclusion de `t1` dans `t2` est impossible.

d. Si `t1[i] > 0` et `t1[i] >= t2[i]`, expliquer brièvement pourquoi on doit avoir `t2[t1[i]] = t2[i]` pour que `t1` soit inclus dans `t2` ?

Q 2.11 En déduire un algorithme en PASCAL qui teste si t_1 est inclus dans t_2 :

```
function est_inclus(t1,t2 : TABLEAU_D_ENSEMBLES) : BOOLEAN;
```

Q 2.12 Donner un exemple de pire des cas, en nombre de comparaisons, pour l'algorithme écrit à la question précédente.

Q 2.13 Quelle est la complexité asymptotique en nombre de comparaisons dans le pire des cas ?

Exercice 3 : Complexité de la multiplication de matrices (4 points)

Dans cet exercice, on s'intéresse à des algorithmes permettant le calcul de produit de matrices. Par exemple, étant données A et B deux matrices carrées de taille 2, le produit $C = A \times B$ est obtenu ainsi :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}, B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}, C = A \times B = \begin{pmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{pmatrix}$$

avec $a_{i,j}, b_{i,j}$ et $c_{i,j}$ les coefficients des matrices A, B et C. Pour simplifier, nous ne considérerons que des matrices carrées dont la taille n est une puissance de 2. Ainsi, pour des matrices plus grandes, de taille n, on pourra découper le problème récursivement :

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}, B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}, C = A \times B = \begin{pmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{pmatrix}$$

avec $A_{i,j}, B_{i,j}$ et $C_{i,j}$ des sous-matrices des matrices A, B et C de taille $\frac{n}{2}$.

Q 3.1 Si on applique la méthode récursivement comme décrit ci-dessus, combien d'opérations de multiplications sont réalisées à chaque étape ?

Q 3.2 Etablir l'équation de récurrence donnant le nombre de multiplications nécessaires au calcul de la multiplication de deux matrices de taille n.

Q 3.3 En déduire la complexité asymptotique en nombre de multiplications de la méthode présentée.

Nous présentons maintenant une autre méthode, proposée par Strassen en 1969. Celle-ci est basée sur le calcul de sept matrices intermédiaires définies ainsi :

$$\begin{aligned} M_1 &= (A_{1,1} + A_{1,2})(B_{1,1} + B_{1,2}) \\ M_2 &= (A_{2,1} + A_{2,2})B_{1,1} \\ M_3 &= A_{1,1}(B_{1,2} - B_{2,1}) \\ M_4 &= A_{2,2}(B_{2,1} - B_{1,1}) \\ M_5 &= (A_{1,1} + A_{1,2})B_{2,2} \\ M_6 &= (A_{2,1} - A_{1,1})(B_{1,1} + B_{2,1}) \\ M_7 &= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}) \end{aligned}$$

Et les coefficients de C sont obtenus ainsi :

$$C = A \times B = \begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{pmatrix}$$

Q 3.4 A chaque étape, combien d'opérations de multiplications sont réalisées ?

Q 3.5 Etablir l'équation de récurrence donnant le nombre de multiplications nécessaires au calcul de la multiplication de deux matrices de taille n.

Q 3.6 En déduire la complexité asymptotique en nombre de multiplications de l'algorithme de Strassen.

<u>Exercice 4</u> : Palindrome (4 points)

Un mot $u = u_1 u_2 \dots u_n$ de longueur n est un palindrome si pour tout $1 \leq k \leq \frac{n}{2}$, $u_k = u_{n-k+1}$. Par exemple, bob, kayak, selles sont des palindromes.

On propose d'écrire un algorithme de programmation dynamique permettant de localiser, s'il en existe, les palindromes présents dans une phrase $v = v_1 v_2 \dots v_m$ de longueur m. La récurrence donnant la réponse à ce problème est la suivante :

$$\begin{aligned} p(i,i) &= \text{vrai} && 1 \leq i \leq m \\ p(m+1,m+1) &= \text{faux} && \\ p(i,i+1) &= \begin{cases} \text{vrai} & \text{si } v_i = v_{i+1} \\ \text{faux} & \text{sinon} \end{cases} && 1 \leq i \leq m-1 \\ p(i,j) &= \begin{cases} p(i+1,j-1) & \text{si } v_i = v_j \\ \text{faux} & \text{sinon} \end{cases} && 1 \leq i \leq m, 1 \leq j \leq m, j \geq i+2 \end{aligned}$$

On supposera que la longueur de la phrase est contenue dans une variable `m` et que la phrase elle-même est contenue dans une variable `v`.

Q 4.1 Quelle propriété le mot v vérifie si $p(i, j)$ est vrai (avec $i \leq j$) ?

Q 4.2 Construire le tableau p si $v =$`mon kayak`. Dessiner le tableau et indiquer 'V' ou 'F' dans chacune des cases.

Q 4.3 Donner la déclaration en PASCAL de la table de programmation dynamique.

Q 4.4 Donner le code en PASCAL nécessaire au remplissage de la table (bien réfléchir aux dépendances de cases pour définir le sens de remplissage).

Q 4.5 Donner le code permettant d'afficher tous les couples de positions (début,fin) dans la phrase donnant les bornes des palindromes s'il en existe, n'affiche rien sinon. On prendra garde s'il existe un palindrome en positions (i, j) à ne pas afficher qu'il en existe également un en positions $(i + 1, j - 1)$, $(i + 2, j - 2)$, etc, ainsi que de ne pas afficher les palindromes à une seule lettre.