Université Lille1 Sciences et Technologies

Info 204 - ASD

2012–2013
Licence STS - Semestre 4

DS2 - 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 : Calcul avec les valeurs d'un ABR

On donne la définition d'abre suivante :

```
type abr = Vide | Cons of noeud
and noeud = {
  valeur : int;
  filsg : abr;
  filsd : abr;
}
```

et une fonction récursive qui calcule le produit de toutes les valeurs des nœuds de l'arbre :

```
let rec produit_rec a =
  if a = Vide then
    1
  else
    let n = match a with | Cons n -> n | Vide -> failwith "arbre␣vide" in
    n.valeur * (produit_rec n.filsg) * (produit_rec n.filsd)
```

Q 1.1 Ecrire en CAML une version non récursive de cette fonction. Vous pourrez utiliser toute structure de données annexe.

Exercice 2 : Suppression dans les tables de hachage

On suppose manipuler une table de hachage avec les définitions de type suivantes (les mêmes que celles du cours) :

```
type couple = {
  cle : string;
  val : int;
};
let table = couple array;
let occupe = bool array;
```

On suppose également disposer d'une fonction `h` dont la spécification est la suivante :

```
/*
   h [k] : retourne un entier positif ou nul calcule a partir de la
           chaine de caracteres [k]
   Note : cet entier n'est pas borne
*/
val h : string -> int;
```

On utilise une résolution des collisions par sondage linéaire et la même fonction d'insertion que celle du cours. La table ne peut contenir qu'une seule valeur associée à une clé donnée.

On souhaite disposer d'une fonction de suppression d'un couple (clé,valeur) de la table. La suppression aura pour effet de réaliser un décalage des couples ayant créé une collision. Par exemple, sur une table contenant 6 alvéoles :

(Clé,Valeur)	Adresse primaire
(ab,0)	1
(bc,1)	5
(ae,2)	1
(cb,3)	2
(ax,4)	1

AVANT

0 :	
1 :	(ab,0)
2 :	(ae,2)
3 :	(cb,3)
4 :	(ax,4)
5 :	(bc,1)

APRES suppression de la clé "ae"

0 :	
1 :	(ab,0)
2 :	(cb,3)
3 :	(ax,4)
4 :	
5 :	(bc,1)

Q 2.1 Si la table ne contient que des clés ayant même adresse primaire que les indices de la table, décrire en français le travail qu'il y a à faire pour supprimer l'une des clés.

Q 2.2 Dans le cas contraire (il y a au moins deux clés avec même adresse primaire, on souhaite en supprimer une), décrire en français ce qu'il y a à faire lorsqu'on rencontre une clé d'adresse primaire différente de celle de la clé à supprimer.

Q 2.3 Ecrire le code CAML d'une fonction `supprimer [k]` qui supprime l'association de la clé `[k]` dans la table `[t]` si elle existe. Sinon elle ne fait rien.

Q 2.4 Quelle est la complexité asymptotique en temps de la fonction de suppression ? Justifier, indiquer l'opération prise en référence. (on suppose avoir inséré n clés dans la table, et que sa taille est M).

Exercice 3 : Comparaison de listes

Etant donné deux listes d'**entiers compris entre 0 et 99**, on souhaiter écrire un prédicat qui teste si les deux listes de longueur respective n et m sont *équivalentes*. On dira que deux listes sont équivalentes si elles contiennent les mêmes éléments, peu importe l'ordre et le nombre d'occurrences. Par exemple :

`[ 5; 6; 1; 9; 5]` et `[1; 9; 6; 1; 1; 5]` sont équivalentes,

`[ 5; 6; 1; 9; 5]` et `[ 6; 1; 9; 1]` ne sont pas équivalentes (la valeur 5 n'apparaît pas).

On suppose que les listes utilisées correspondent à celles qui ont été vues en TP, on ne peut donc les manipuler qu'avec des itérateurs. Il est strictement interdit d'utiliser une fonction de tri pour résoudre le problème.

Q 3.1 Décrire en français la stratégie que vous allez utiliser et la(les) structure(s) de données annexes.

Q 3.2 Ecrire en CAML la fonction booléenne `equivalents` prenant en entrée deux listes et retournant vrai si les deux listes sont équivalentes, faux sinon.

Q 3.3 Quelle est la complexité asymptotique en temps de votre algorithme ? Justifier.

Q 3.4 Quelle est la complexité asymptotique en espace de votre algorithme ? Justifier.

Exercice 4 : Construction d'un ABR optimal par programmation dynamique

Etant donné un ensemble de valeurs entières distinctes croissantes $\{v_1, \ldots, v_n\}$ et des fréquences de recherche de ces valeurs $\{f_1, \ldots, f_n\}$ on souhaite construire un ABR $\mathcal{T}$ contenant ces valeurs et qui minimise la fonction de coût suivante afin d'optimiser l'arbre par rapport aux recherche effectuées :

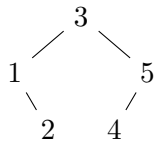
$$\text{coût}(\mathcal{T}) = \sum_{i=1}^{n} f_i \times p_{\mathcal{T}}(v_i)$$

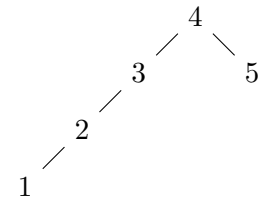
où $p_{\mathcal{T}}(v_i)$ est la profondeur du nœud contenant la valeur v_i dans $\mathcal{T}$. On considère ici que la profondeur de la racine vaut 1, celles de ses fils 2, etc.

Supposons par exemple qu'on construise un arbre avec les valeurs et les fréquences de recherche suivantes :

v_i	1	2	3	4	5
f_i	0.1	0.1	0.5	0.2	0.1

et si on considère les deux ABR suivants :





l'arbre de gauche aura pour coût $(0.5 \times 1 + 0.1 \times 2 + 0.1 \times 2 + 0.1 \times 3 + 0.2 \times 3) = 1.8$ alors que l'arbre de droite aura pour coût $(0.2 \times 1 + 0.5 \times 2 + 0.1 \times 2 + 0.1 \times 3 + 0.1 \times 4) = 2.1$

Q 4.1 Proposer un arbre de meilleur coût (dessinner l'arbre et détailler le calcul du coût comme ci-dessus).

On peut voir le problème récursivement : quelle est la meilleure valeur v_k à mettre à la racine qui minimise le coût des sous-arbres gauche contenant les valeurs v_1 à v_{k-1} et droit contenant les valeurs v_{k+1} à v_n ?

Si on note $\mathcal{T}_{i,j}$ l'arbre optimal pour les valeurs v_{i+1} à v_j, $w_{i,j} = \sum_{k=i+1}^{j} f_k$, et $c_{i,j}$ le coût de l'arbre $\mathcal{T}_{i,j}$ ($c_{i,i} = 0$ pour l'arbre vide $\mathcal{T}_{i,i}$), alors on peut prouver que [1] :

$$c_{i,j} = w_{i,j} + \min_{k \in \{i+1..j\}}(c_{i,k-1} + c_{k,j})$$

Il suffit donc de calculer $c_{0,n}$ pour obtenir le coût minimum.

Cette expression récursive permet de mettre en œuvre une résolution par programmation dynamique. On supposera disposer d'une fonction `w` à deux paramètres permettant de calculer $w_{i,j}$.

Q 4.2 Quelle est la dimension de la table de programmation dynamique ?

Q 4.3 Quelle est la taille de la table de programmation dynamique ?

Q 4.4 Ecrire le code permettant d'initialiser la table sachant que les fréquences sont contenues dans un tableau `f`.

Q 4.5 Grâce à un dessin, indiquer toutes les cases qu'il faut calculer avant de calculer $c_{i,j}$.

Q 4.6 Ecrire le code permettant de remplir la table.

Q 4.7 Sur le même dessin, indiquer la case où se trouve le résultat final.

Q 4.8 Quelle est la complexité en temps du calcul de $c_{(0,n)}$? Justifier, indiquer l'opération prise en référence.

1. ne cherchez pas à prouver cette formule !