Structures de Données — Epreuve de mai 2011
Durée deux heures — Aucun document autorisé

Ne recopiez pas l'énoncé sur votre copie. Indiquez seulement le numéro des questions. Des réponses courtes et claires seront appréciées.

1 Questions de cours (12 points)

1.1 Listes chaînées

Question 1 [1 pt]. On souhaite réaliser une file au moyen d'une liste chaînée. Parmi les différentes implantations possibles des listes chaînées étudiées en cours, laquelle vous semble la mieux adaptée ? Pourquoi ?

1.2 Mesures de performances

On considère la fonction `Richardson` de la Figure 2. On note $f(n)$ le nombre d'opérations arithmétiques sur les doubles exécutées lors d'un appel à cette fonction, y compris celles exécutées par v. On suppose que chaque appel à v exécute 30 opérations arithmétiques sur les doubles.

Question 2 [1 pt]. Modifier le code de la fonction pour qu'elle retourne $f(n)$.

Note : vous pouvez répondre directement sur l'énoncé, pour éviter d'avoir à recopier le code de la fonction.

Question 3 [2 pts]. On souhaite produire un fichier de mesures `stats` à partir de différentes exécutions de la fonction. Ce fichier comporte deux colonnes : une pour n, une pour $f(n)$. Ecrire un programme principal qui fabrique le fichier `stats` en faisant varier n de 1 à 20.

1.3 Notations asymptotiques

Question 4 [1 pt]. L'algorithme A a une complexité en temps en $O(n)$. L'algorithme B a une complexité en temps en $O(n^2)$. Peut-on affirmer que A est plus rapide que B, quand n tend vers l'infini ? Justifier.

Question 5 [1 pt]. L'algorithme A a une complexité en temps en $O(n)$. L'algorithme B a une complexité en temps en $\Omega(n^2)$. Peut-on affirmer que A est plus rapide que B, quand n tend vers l'infini ? Justifier.

Question 6 [1 pt]. L'algorithme A a une complexité en temps en $\Theta(n)$. L'algorithme B a une complexité en temps en $\Theta(n^2)$. Peut-on affirmer que A est toujours plus rapide que B. Justifier.

1.4 ABR

Question 7 [1 pt]. Insérer les symboles suivants dans un ABR initialement vide, dans l'ordre (on ne distingue pas les majuscules des minuscules). Dessiner uniquement l'ABR obtenu au final.

`Cette question porte sur les ABR`

Question 8 [1 pt]. A partir de la déclaration suivante, écrire en C une fonction, paramétrée par un `struct ABR*` a, qui retourne `true` si a est une feuille, `false` sinon.

```
struct ABR
{   struct ABR* gauche;   /* symboles inférieurs lexicographiquement à value */
    struct ABR* droit;    /* symboles supérieurs lexicographiquement à value */
    struct symbole value; /* le symbole associé au noeud */
};
#define NIL (struct ABR*)0
```

1.5 Tables de hachage

On considère l'insertion des clés 17, 39, 29, 21 dans une table de hachage de $N = 11$ alvéoles. La table est gérée en adressage ouvert, avec la technique du double hachage. La fonction de hachage est $h(s) = (h_1(s), h_2(s)) = (s \bmod N, 1 + s \bmod (N - 1))$.

Question 9 [1 pt]. Il y a 4 clés à insérer dans une table de 11 alvéoles. Est-on certain, pour autant, qu'on trouvera un alvéole libre pour chaque clé ? Justifier en quelques mots.

Question 10 [2 pts]. Donner le contenu de la table après insertion des clés. Indiquer les collisions qui se sont produites, s'il y en a eu.

2 Problème (8 points)

On s'intéresse à la gestion d'un calendrier de réservation d'une certaine ressource. Ce calendrier est représenté par un arbre binaire. Les feuilles de l'arbre sont des périodes élémentaires de réservation (des intervalles bornés par deux dates, pas nécessairement tous de même durée). Une date est représentée par un entier. Les nœuds internes représentent les périodes couvertes par tous leurs descendants. Même pour les nœuds internes, on suppose qu'une période est un intervalle borné par deux dates. Un exemple de calendrier, couvrant une période de 24 heures, est donné figure 1. Les opérations de calendrier qui nous intéressent sont les suivantes :

1. la réservation de la ressource à une date d donnée ; cette opération a pour effet de mettre la période élémentaire de réservation contenant d dans l'état « réservé » (sur notre exemple, si on réserve la ressource à la date 10, la période élémentaire $7 - 18$ devient réservée) ;
2. la détermination de l'état de la ressource (soit « libre », soit « réservé ») à une date d donnée ; cet état est égal à l'état de la période élémentaire qui contient d.
3. l'annulation d'une réservation de la ressource à une date d donnée.

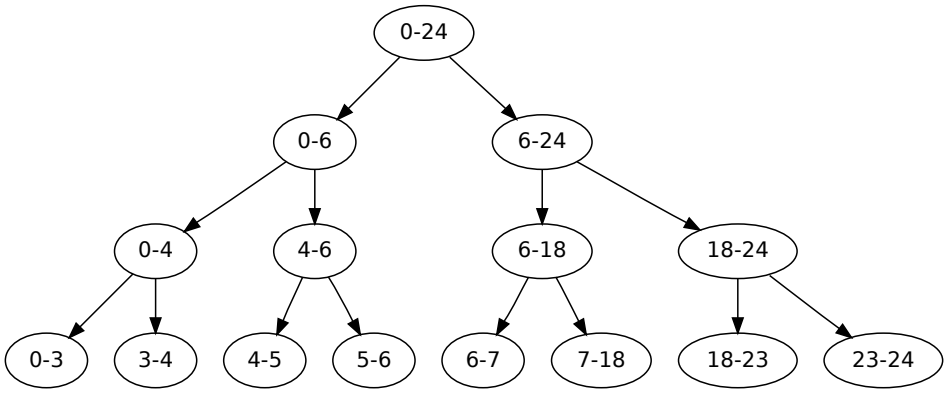

FIGURE 1 – Un exemple de calendrier

Question 11 [3 pts]. Proposer une structure de données, en C, permettant d'implanter un calendrier. Donner toutes les précisions nécessaires à la compréhension de la structure. Définir éventuellement des types intermédiaires.

Question 12 [2 pts]. Donner les prototypes des fonctions d'interface. Penser aussi aux fonctions de construction de calendrier. Spécifier ces fonctions.

Question 13 [2 pts]. Ecrire la fonction qui réserve la ressource à une date donnée. Il se peut que la date n'appartienne à aucune période élémentaire. Dans ce cas, ne rien réserver.

Question 14 [1 pt]. Estimer la complexité, en temps, dans le pire des cas, de la fonction de réservation. Mesurer pour cela le nombre de comparaisons de dates en fonction du nombre n de périodes élémentaires du calendrier. Pour simplifier, on suppose que tout nœud qui n'est pas une feuille, a exactement deux fils.

Numéro de place :

```
extern double v (double);

void Richardson (double* resultat, int n)
{   double A[n][n];  /* matrice intermédiaire */

    double powr[n];  /* tableau de puissances de r */

    double h, h0, r;

    int p, m;



    h0 = 1.0;

    r = 0.5;

    powr[0] = 1.0;                /* powr[0] = r^0 */

    A[0][0] = v(h0);

    for (p = 1; p < n; p++)

    {   powr[p] = powr[p-1] * r;  /* powr[p] = r^p */

        h = powr[p] * h0;

        A[p][0] = v(h);

        for (m = 1; m <= p; m++)

            A[p][m] = (A[p][m-1] - powr[m]*A[p-1][m-1])/(1.0 - powr[m]);

    }

    *resultat = A[n-1][n-1];

}
```

FIGURE 2 – Une implantation de la méthode de Richardson.