Université Lille1 Sciences et Technologies

Info 204 - ASD

2010–2011
Licence STS - Semestre 4

DS3 - rattrapage - documents de cours, TD, TP autorisés

Exercice 1 : Complexité (15 minutes)

Q 1.1 Démontrer le comportement asymptotique de algorithme $\mathcal{A}$ dont la fonction de complexité est :

$$\begin{cases} c(0) &= 1 \\ c(n) &= 3c\left(\frac{n}{2}\right) + n^2 \end{cases}$$

Q 1.2 On dispose d'un algorithme récursif réalisant un traitement sur un tableau dont on sait que le comportement asymptotique en temps est en $\Theta(n^3)$. Cet algorithme utilise le principe diviser pour régner en traitant des demi-tableaux. L'équation de récurrence de l'algorithme est

$$c(n) = ac\left(\frac{n}{b}\right) + f(n)$$

Donner des valeurs possibles pour a, b et $f(n)$. Démontrer.

Exercice 2 : Tri par sélection (15 minutes)

On rappelle le principe du tri par sélection du minimum :

```
soit E une suite d'elements a trier
soit F une suite vide
tant qu'il reste des elements dans E
  soit x l'element minimum de E
  ranger x dans F (de maniere ordonnee)
fin tant que
retourner F
```

On souhaite utiliser cet algorithme pour trier les éléments d'une pile. La pile est la seule structure de données autorisée dans cet exercice (on n'aura droit ni aux tableaux, ni aux files, ni aux listes, ni aux arbres). Ainsi votre algorithme pourra utiliser d'autres piles et bien sûr des variables.

Q 2.1 Décrire un algorithme en français réalisant le tri des éléments d'une pile utilisant le principe du tri par sélection du minimum.

Q 2.2 Quelle est la complexité asymptotique en temps ? Justifiez.

Exercice 3 : Valeur la plus fréquente (25 minutes)

Ci-dessous est donné un algorithme qui permet de calculer la valeur qui apparaît le plus grand nombre de fois dans un tableau t comportant n cases.

```
soit ht une table de hachage dont les clés sont les valeurs du tableau et les valeurs
    associées sont des compteurs
pour chaque valeur v de t
  s'il y a un compteur c associée à cette valeur v dans ht alors
    incrémenter c
  sinon
    ajouter dans ht un nouveau compteur initialisé à 1, associé à la clé v
  fin si
  retenons la plus grande valeur cmax de tous les compteurs
      et la clé vmax qui permet d'accéder à ce compteur
fin pour
retourner vmax
```

Q 3.1 En utilisant les primitives offertes par le module `Hashtbl` (ou la classe `Hashtable` en Java) écrire en OCaml (ou en Java) cet algorithme.

Q 3.2 Donner le comportement asymptotique en temps de cet algorithme (préciser les opérations comptées).

Q 3.3 Donner le comportement asymptotique en espace.

Exercice 4 : A nouveau ce tri étrange (30 minutes)

Nous re-présentons ci-dessous un algorithme de tri :

```
let rec tri t i j =
  if j > i then
    if i + 1 = j then begin
      if t.(i) > t.(j) then
        (* echange dans t les valeurs aux indices i et j *)
        echanger t i j;
    end else begin
      tri t i (j - 1);
      tri t (i + 1) j;
      tri t i (j - 1);
    end
```

Nous avons déjà vu au DS précédent que le dernier appel récursif `tri t i (j - 1)` était nécessaire pour être certain de replacer la valeur minimale à la bonne place.

Q 4.1 Proposer une variation de cet algorithme en remplaçant ce dernier appel par une suite d'instructions tirant parti du fait que la tranche $i+1..j$ est déjà triée après les deux premiers appels récursifs.

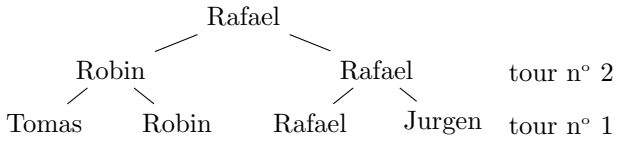
Q 4.2 Identifier un pire des cas dans le cas où c'est le nombre d'échanges qui sont comptés.

Q 4.3 Exprimer sour forme d'une équation de récurrence fonction de la longueur de la tranche de tableau traitée (c'est-à-dire $n = j - i + 1$) le nombre d'échanges dans le pire des cas.

Q 4.4 Donner le comportement asymptotique dans le pire des cas.

Exercice 5 : Tournois sportifs (35 minutes)

Dans un tournoi, n joueurs s'affrontent deux par deux dans des matchs, le vainqueur de chaque match passe au tour suivant. Le processus est itéréré jusqu'à ce qu'il ne reste qu'un joueur, le vainqueur. Le premier tour est appelé tour numéro 1, le suivant numéro 2 et ainsi de suite. Un tounoi est souvent représenté sous forme d'un arbre, par exemple ici un tournoi à 2 tours avec 4 joueurs :



Pour simplifier, les n joueurs seront n entiers et le vainqueur est l'entier le plus grand. Les n entiers de départ sont contenus dans un tableau t. *On supposera que n est une puissance de 2.*

Le principe de l'algorithme de création du tournoi est le suivant. Mettre dans les feuilles les valeurs initiales du tableau (tour numéro 1), puis pour chaque nœud père, mettre le maximum des deux fils, recommencer jusqu'à avoir rempli tout l'arbre.

Q 5.1 Dessiner l'arbre correspondant au tournoi avec les valeurs aux feuilles 8, 1, 2, 6, 7, 5, 3, 4.

Q 5.2 Donner, en fonction de n, le nombre m de tours du tournoi.

Q 5.3 Etant donné un tournoi sous forme d'arbre avec les déclarations suivantes :

```
type tournoi = Vide | Cons of noeud
and noeud = { valeur : int; mutable droit : tournoi; mutable gauche : tournoi }
```

proposer le code d'une fonction `joueurs_du_tour : tournoi -> int -> int List` qui étant donné un tournoi sous forme d'arbre et un numéro de tour retournera la liste des joueurs du tour.

Q 5.4 Quel est le comportement asymptotique de cette fonction dans le pire des cas (vous indiquerez quel est le pire des cas) ?