Université Bordeaux I
U.F.R. Mathématiques et Informatique

Licence Informatique semestre 6
Année 2004-2005

Lundi 21 mars 2005, (1 heure)
Sans document

Alain Griffault

SUJET + CORRIGE

Exercice 1 (4 points) *Quelles sont les valeurs possibles de n, pour que n personnes soient tel que :*

1. $\lceil \frac{n}{2} \rceil$ *personnes d'entre elles soient l'ami de 4 personnes exactement du groupe,*
2. $\lfloor \frac{n}{2} \rfloor$ *personnes d'entre elles soient l'ami de 3 personnes exactement du groupe.*

Justifiez votre réponse.

Réponse : La relation *ami* est représentable par un graphe non orienté. Pour calculer les valeurs possibles de n, on utilise la propriété qui énonce que la somme des degrés des sommets dans un graphe est pair.

n **pair :** Posons $n = 2 \times k$. Les conditions deviennent :

$$\begin{cases} n > 4 & \text{ami de 4 personnes} \\ n > 3 & \text{ami de 3 personnes} \\ \sum_{i=1}^{i=n} degre(s_i) \text{ pair} & \end{cases} \Rightarrow \begin{cases} n > 4 \\ (k \times 4 + k \times 3)\%2 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} n > 4 \\ k\%2 = 0 \end{cases}$$

$$\Rightarrow \{n \text{ tel que } n > 4 \text{ et } n\%4 = 0\}$$

n **impair :** Posons $n = 2 \times k + 1$. Les conditions deviennent :

$$\begin{cases} n > 4 & \text{ami de 4 personnes} \\ n > 3 & \text{ami de 3 personnes} \\ \sum_{i=1}^{i=n} degre(s_i) \text{ pair} & \end{cases} \Rightarrow \begin{cases} n > 4 \\ ((k+1) \times 4 + k \times 3)\%2 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} n > 4 \\ (7 \times k + 4)\%2 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} n > 4 \\ k\%2 = 0 \end{cases}$$

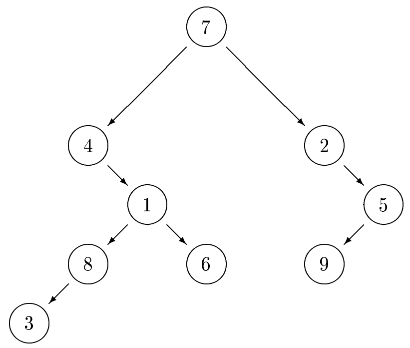
$$\Rightarrow \{n \text{ tel que } n > 4 \text{ et } n\%4 = 1\}$$

L'ensemble des valeurs possibles est donc $\{n \text{ tel que } n > 4 \text{ et } (n\%4 = 0 \text{ ou } n\%4 = 1)\}$

Exercice 2 (4 points) *Dessinez l'arbre binaire dont la liste des sommets est :*

- *dans l'ordre préfixe : 7, 4, 1, 8, 3, 6, 2, 5, 9*
- *dans l'ordre infixe : 4, 3, 8, 1, 6, 7, 2, 9, 5*

Réponse :



Exercice 3 *Soit le code suivant de l'algorithme de tri par permutations.*

```
T[1..N] un tableau d'entiers.
pour i de 1 a N-1 faire
  pour j de i+1 a N faire
    si (T[i] > T[j]) alors echanger(T[i], T[j]);
```

Question 3.1 (2 points) *Donnez et justifiez la complexité en terme du nombre de tests* `si (T[i] > T[j])` *?*

Réponse : *Le nombre de tests* `si (T[i] > T[j])` *est :*

$$\sum_{i=1}^{i=N-1}(N-(i+1)+1) \quad = \quad N \times (N-1) \div 2$$

La complexité en terme du nombre de tests `si (T[i] > T[j])` *est donc* $\Theta(N^2)$

Question 3.2 (2 points) *Donnez et justifiez la complexité en terme du nombre d'appels à la procédure* `echanger(x,y)` *?*

Réponse : *Du fait du test qui retourne soit* vrai *soit* faux, *le nombre d'appels à la procédure* `echanger(x,y)` *est compris entre :*

$$\begin{array}{lllll} & \sum_{i=1}^{i=N-1}(N-(i+1)+1) & = & N \times (N-1) \div 2 & \textit{toujours vrai} \\ \textit{et} & & & & \\ & \sum_{i=1}^{i=N-1} 0 & = & 0 & \textit{toujours faux} \end{array}$$

Le cas toujours faux *est possible si le tableau est déjà trié ; et le cas toujours* vrai *est possible si le tableau est trié dans l'ordre inverse.*
La complexité en terme du nombre d'appels à la procédure `echanger(x,y)` *est donc* $O(N^2)$ *et* $\Omega(1)$.

Exercice 4 *Soit l'algorithme suivant*

```
transforme(A : Arbre) {
  parcours(racine(A));
}

parcours(a){
  Fils[1..n]; // un tableau de pointeurs sur les fils du sommet ``a''
  f1, f2; // deux pointeurs à nil
  si (n > 0) alors {
    f1 = Fils[1];
    pour i de 2 à n faire
      Fils[i-1].f2 = Fils[i];
    pour i de 1 à n faire {
      parcours(Fils[i]);
      Fils[i] = nil;
    }
  }
}
```

Question 4.1 (3 points) *Appliquez l'algorithme sur l'arbre suivant :*

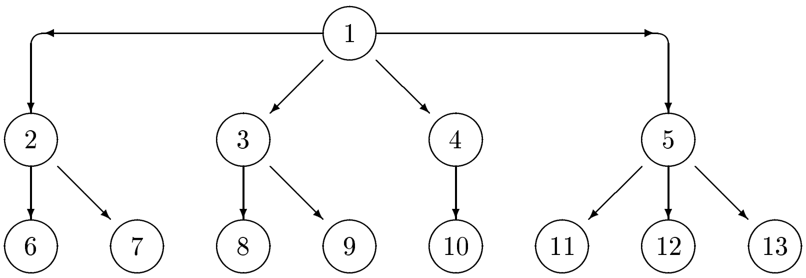
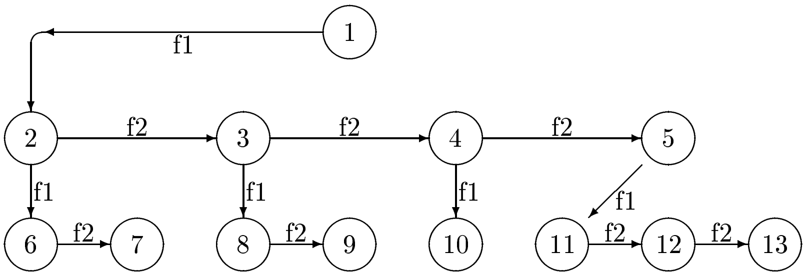


FIG. *1 – Un arbre de racine* 1

Réponse :



Question 4.2 (1 point) *Montrez que l'algorithme* `transforme(A : Arbre)` *conserve le nombre de pointeurs différents de* `nil`*.*

Réponse : *La structure de l'algorithme* `transforme(A : Arbre)` *est celle d'un parcours en profondeur d'un arbre. Dans un tel parcours, chaque sommet est visité une et une seule fois.*
Pour chaque sommet `a`*, si son nombre de fils n est strictement positif :*
Pendant le traitement préfixe *:*
– *son pointeur* `f1` *passe de* `nil` *à* `Fils[1]` *qui n'est pas* `nil`*.*
– *$n-1$ pointeurs* `Fils[i-1].f2` *passent de* `nil` *à* `Fils[i]` *qui ne sont pas* `nil`*.*
et pendant le traitement suffixe *:*
– *n pointeurs* `Fils[i]` *passent à* `nil`*.*
Soit CP le nombre de pointeurs différents de `nil` *dans l'arbre original. Après chaque visite d'un sommet, ce nombre devient $CP + 1 + (n-1) - n = CP$.*
L'algorithme `transforme(A : Arbre)` *conserve donc le nombre de pointeurs différents de* `nil`*.*

Question 4.3 (2 points) *Montrez que dans le résultat de l'algorithme* `transforme(A : Arbre)`*, tous les sommets restent accessibles depuis la racine.*

Réponse : *Par récurrence sur la profondeur h du sommet (distance du sommet à la racine).*

$h = 0$ **:** *la racine est inchangée donc encore accessible.*

$h = 1$ **:** *Soit* `Fils[i]` *le* `ième` *fils de la racine. Dans la nouvelle structure, le chemin $racine(A).f1.(f2)^{i-1}$ pointe sur le* `ième` *fils de la racine.*
Tous les sommets de profondeur 1 sont donc accessibles.

vrai jusqu'à h **:** *Soit s un sommet de profondeur $h+1$ dans l'arbre original. Dans l'arbre original, ce sommet est le* `ième` *fils de son père, et ce père est un sommet ayant une profondeur h. D'après l'hypothèse de recurrence, ce père est accessible dans la nouvelle structure par un chemin C. Le sommet s est donc accessible par le chemin $C.f1.(f2)^{i-1}$.*
Tous les sommets de profondeur $h+1$ sont donc accessibles.

Dans le résultat de l'algorithme `transforme(A : Arbre)`*, tous les sommets restent donc accessibles depuis la racine.*

Question 4.4 (1 point) *Montrez que le résultat de l'algorithme* `transforme(A : Arbre)` *est un arbre de même racine.*

Réponse : *La structure obtenue possèdent les propriétés suivantes :*

1. *une racine identique à* `racine(A)`*,*
2. *tous ses sommets sont accessibles depuis la racine,*
3. *le nombre d'arcs est égal au nombre d'arcs de l'arbre initial, à savoir le nombre de sommets moins un.*

D'après les théorèmes et propriétés vus en cours, la structure obtenue est un arbre.

Question 4.5 (1 point) *Montrez que dans l'arbre résultat, chaque sommet possède au plus deux pointeurs différents de* `nil`*.*

Réponse : *Dans l'arbre résultat, pour chaque sommet, tous les pointeurs* `Fils[i]` *sont à* `nil`*. Les seuls pointeurs qui peuvent être différents de* `nil` *sont* `f1` *et* `f2`*.*
Chaque sommet possède donc au plus deux pointeurs différents de `nil`*.*

Question 4.6 (1 point) *Que fait l'algorithme* `transforme(A : Arbre)`*.*

Réponse : *L'algorithme* `transforme(A : Arbre)` *produit à partir d'un arbre un arbre binaire ayant :*
– *même racine,*
– *même nombre de sommets,*
– *pour chaque sommet :*
  – *un pointeur* `f1` *qui pointe vers le* fils ainé
  – *un pointeur* `f2` *qui pointe vers le* frère suivant