

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Section d'Informatique et de Systèmes de Communication

Corrigé de la série 5

25 Octobre 2010

1. Algorithmes récursifs: Le diamètre d'un arbre binaire

a) Notons r la racine de T . Pour un chemin dans T il y a trois possibilités:

- (1) Le chemin se trouve entièrement dans T_1 ,
- (2) Le chemin se trouve entièrement dans T_2 ,
- (3) Le chemin passe par (ou se termine à) la racine r .

Pour pouvoir dire quelque chose sur le diamètre de T , nous considérons un chemin γ de longueur maximale. Si γ est du premier type, alors la longueur de γ est certainement égale à $\text{diam}(T_1)$, et si γ est du deuxième type, sa longueur est égale à $\text{diam}(T_2)$.

Le troisième cas nécessite une discussion un peu plus détaillée. Dénotons le dernier élément du chemin dans T_1 par x_1 et le dernier élément dans T_2 par x_2 . Le chemin est alors comme suit:

$$x_1 \rightarrow \cdots \rightarrow \text{racine de } T_1 \rightarrow \text{racine de } T \rightarrow \text{racine de } T_2 \rightarrow \cdots \rightarrow x_2.$$

La longueur de ce chemin est donc

$$\text{profondeur}_{T_1}(x_1) + 1 + 1 + \text{profondeur}_{T_2}(x_2),$$

où $\text{profondeur}_T(x)$ dénote la profondeur du sommet x dans l'arbre T (voir p.56 du cours pour voir la définition de profondeur).

La maximalité du chemin implique maintenant que les deux termes $\text{profondeur}_{T_1}(x_1)$ et $\text{profondeur}_{T_2}(x_2)$ doivent être maximaux, i.e. que les sommets en question se trouvent en tout en bas de l'arbre et donc que la profondeur des sommets en question est égale à la hauteur du sous-arbre correspondant. En résumé donc, la longueur du chemin est, dans ce cas, égale à

$$h_1 + 2 + h_2,$$

où h_i dénote la hauteur de l'arbre T_i .

Nous déduisons donc que la longueur du plus long chemin dans T (et alors le diamètre de T) est égale à

$$\max\{\text{diam}(T_1), \text{diam}(T_2), h_1 + h_2 + 2\}.$$

Soit h la hauteur de T . Il est alors assez clair que

$$h = \max(h_1, h_2) + 1.$$

b) Nous construisons une fonction récursive qui calcule à la fois la hauteur et le diamètre d'un arbre binaire étant donné sa racine:

Call: HAUTEURETDIAM

Input: r : Racine d'un arbre binaire.

Output: (h, d) : Hauteur et diamètre de l'arbre.

```

if  $r = 0$  then
  return  $(-1, -1)$ 
 $(h_1, d_1) \leftarrow$  HAUTEURETDIAM( $r[\text{left}]$ )
 $(h_2, d_2) \leftarrow$  HAUTEURETDIAM( $r[\text{right}]$ )
 $h \leftarrow \max\{h_1, h_2\} + 1$ 
 $d \leftarrow \max\{d_1, d_2, h_1 + h_2 + 2\}$ 
return  $(h, d)$ 

```

Remarquons que cet algorithme a bien un temps de parcours linéaire: Si nous ignorons pour un instant les appels récursifs, nous voyons que, puisqu'il n'y a pas de boucles, l'algorithme opère en temps constant. Mais l'appel de la fonction HAUTEURETDIAM se fait exactement une fois pour chaque sommet; il en résulte que l'algorithme est $O(|V|)$.

2. Algorithmes par induction: Le problème de la célébrité

On précise que dans nos graphes non-orientés on suppose qu'un sommet n'est jamais connecté à lui-même, c'est-à-dire que la diagonale de la matrice d'adjacence ne contient que des 0.

- Le sommet 2 est une célébrité puisqu'il est connu de tout le monde mais ne connaît personne (son in-degree vaut $n - 1$ alors que son out-degree vaut 0).
- Supposons que nous avons un graphe avec deux célébrités (que nous appelons les sommets i et j). i ne connaît personne, donc en particulier i ne connaît pas j . j ne peut donc pas être une célébrité et nous avons une contradiction. Un graphe ne peut donc pas avoir deux célébrités.
- On rappelle que pour tout graphe orienté, avec comme matrice d'adjacence A , A_{ij} vaut 1 s'il y a un arc du sommet i au sommet j , et 0 sinon. Ainsi la ligne k montre les arcs qui sortent du sommet k , et la colonne k ceux qui vont vers k .

Si le sommet k est une célébrité, la ligne k ne contient donc que des 0. De même, la colonne k ne contient que des 1, sauf sur la diagonale (l'entrée A_{kk}) qui vaut 0 (puisque nous supposons qu'il n'y a jamais d'arc d'un sommet à lui-même).

- L'algorithme naïf serait de prendre chaque sommet un par un et de regarder s'il s'agit d'une célébrité. Etant donné un sommet k , pour déterminer si c'est une célébrité nous devons:
 - (1) vérifier que tous les autres sommets connaissent k
 - (2) vérifier qu'aucun autre sommet n'est connu de k

Pour (1) nous devons vérifier que $A_{ik} = 1$ pour tout $i \in \{1, \dots, n\} \setminus \{k\}$ ($n - 1$ questions).
 Pour (2) nous devons vérifier que $A_{kj} = 0$ pour tout $j \in \{1, \dots, n\} \setminus \{k\}$ ($n - 1$ questions).
 Pour vérifier si k est une célébrité il nous faut donc $2 \cdot (n - 1)$ questions.

Puisque nous devons répéter cette procédure pour tous les $k \in \{1, \dots, n\}$ on obtient un total de $2 \cdot n \cdot (n - 1)$ questions dans le pire des cas, donc un algorithme $O(n^2)$.

e) Rappelons qu'on sait qu'il existe une célébrité dans le graphe qui nous est donné.

- (i) Si on a deux sommets a et b on peut poser la question " a connaît-il b ?". Si la réponse est "oui" on sait que a ne peut pas être une célébrité, et puisque l'un des deux doit être une célébrité, il s'agit forcément de b . De même, si la réponse est "non", on sait que b ne peut pas être une célébrité, et donc la célébrité est a .
- (ii) Nous supposons qu'étant donné un groupe de n personnes contenant une célébrité, nous pouvons la retrouver en posant $T(n)$ questions.

Etant donné un groupe de $n + 1$ personnes nous procédons comme suit:

- nous choisissons deux personnes a et b parmi les $n + 1$.
- nous posons la question " a connaît-il b ?"
- Si la réponse est "oui" nous savons que a n'est pas la célébrité. Celle-ci se trouve donc parmi les n autres personnes (dont b). Nous pouvons donc la retrouver en posant $T(n)$ questions. Nous avons donc dû poser $T(n) + 1$ questions au total.
- Si la réponse est "non" nous savons que b n'est pas la célébrité. Nous pouvons donc de la même façon la retrouver parmi les n personnes restantes en posant $T(n)$ questions. Nous avons donc dû poser $T(n) + 1$ questions au total.
- (iii) L'algorithme récursif a été décrit dans la partie (ii). En pseudo-code on obtient:

Call: FINDCELEBRITY

Input: $S = \{s_1, \dots, s_n\}$: Un ensemble de n personnes, dont une célébrité

Output: c : la célébrité

```

if  $n = 2$  then
  if  $s_1$  connaît  $s_2$  then
    return  $s_2$ 
  else
    return  $s_1$ 
if  $s_1$  connaît  $s_2$  then
  return FINDCELEBRITY( $S \setminus \{s_1\}$ )
else
  return FINDCELEBRITY( $S \setminus \{s_2\}$ )

```

Si on appelle $T(n)$ le nombre de questions dont il a besoin pour trouver la célébrité parmi n personnes on a donc $T(2) = 1$ et $T(n) = T(n - 1) + 1$ pour $n \geq 2$. Il est assez clair que cela nous donne

$$T(n) = n - 1$$

En effet par récurrence on voit que l'égalité est vraie pour $n = 2$ (base), et si elle est vraie pour n alors

$$T(n + 1) = T(n) + 1 = (n - 1) + 1 = n$$

Ainsi c'est un algorithme $O(n)$.

3. Multiplication de plusieurs matrices

Rappelons que $m_{i,j}$ est le nombre minimal de multiplications scalaires pour le calcul de $A_{i\dots j}$ et on a :

$$m_{ij} = \begin{cases} 0 & \text{si } i = j = 0, \\ \min_{i \leq k < j} \{m_{ik} + m_{k+1,j} + p_{i-1} \cdot p_k \cdot p_j\} & \text{si } 1 \leq i \leq j \leq n. \end{cases}$$

Rappelons aussi que s_{ij} est la valeur de k pour laquelle le minimum de la formule est atteint. Considerons que $M[i, j] = m_{ij}$ et $S[i, j] = s_{i,j}$.

$$\begin{aligned} m_{1,2} &= 5 \cdot 10 \cdot 3 = 150 \\ m_{2,3} &= 10 \cdot 3 \cdot 12 = 360 \\ m_{3,4} &= 3 \cdot 12 \cdot 5 = 180 \\ m_{1,3} &= \min \begin{cases} m_{1,2} + 5 \cdot 3 \cdot 12 = 330 \\ m_{2,3} + 5 \cdot 10 \cdot 12 = 960 \end{cases} \\ m_{2,4} &= \min \begin{cases} m_{2,3} + 10 \cdot 12 \cdot 5 = 960 \\ m_{3,4} + 10 \cdot 3 \cdot 5 = 330 \end{cases} \\ m_{1,4} &= \min \begin{cases} m_{2,4} + 5 \cdot 10 \cdot 5 = 580 \\ m_{1,2} + m_{3,4} + 5 \cdot 3 \cdot 5 = 405 \\ m_{1,3} + 5 \cdot 12 \cdot 5 = 630 \end{cases} \end{aligned}$$

Les matrices M et S sont alors :

$$M = \begin{pmatrix} 0 & 150 & 330 & 405 \\ & 0 & 360 & 330 \\ & & 0 & 180 \\ & & & 0 \end{pmatrix} \quad S = \begin{pmatrix} 0 & 1 & 2 & 2 \\ & 0 & 2 & 2 \\ & & 0 & 3 \\ & & & 0 \end{pmatrix}$$

$S[1, 4] = 2$ signifie que $A_1 \cdot A_2 \cdot A_3 \cdot A_4$ est construit de manière optimale à partir de $A_1 \cdot A_2$ et $A_3 \cdot A_4$. Le nombre de multiplications nécessaires est donc 405.