Devoir 2 IFT2015

Catherine Larivière 0955948

Dominique Vigeant 20129080

23 juin 2021

1 Partie pratique

Remis sous la forme Rope.java et Node.java.

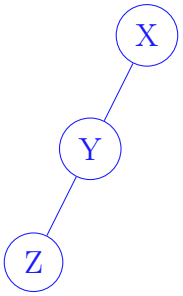
2 Partie théorique

1. (1 point) Dans la preuve du théorème sur la profondeur d'un arbre binaire moyen j'ai commencé par définir la valeur de I pour tous les arbres, à partir de $N = 1$ (arbre avec un seul noeud, $I = 0$), donc $N = 1, 2, 3....$ Après j'ai écrit que

$$I(\tau_N) = I(\tau_i) + I(\tau_{N-i-1}) + N - 1 \tag{1}$$

mais, un peu paresseux comme tout le monde, j'avais omis de mentionner une petite définition. Donnez un exemple d'un arbre avec, disons, $N \geq 3$, où des quantités dans l'équation (1) ne sont pas bien définies. Ensuite, fournissez la définition qui manque, et finalement, indiquez la conséquence pour la valeur de $D(0)$ (valeur utilisée d'ailleurs dans la preuve du théorème). (Je ne veux pas vous inquiéter : il n'y a pas d'erreur dans la preuve, il s'agit ici d'écrire les choses un peu plus au complet.)

Soit l'arbre binaire τ_N suivant où $N = 3$.



$$\begin{aligned} I(3) &= I(2) + I(0) + 3 - 1 \\ &= [I(1) + I(0) + 2 - 1] + I(0) + 3 - 1 \\ &= [0 + I(0) + 1] + I(0) + 2 \\ &= 2I(0) + 3 = ?? \end{aligned}$$

Pour l'équation (1), il nous manque la définition de $I(\tau_N)$ lorsque $N = 0$. Bien entendu, I(0) = 0. Cela fait en sorte que $D(0) = 0$.

2. (2 points) Nous avons remarqué dans l'un des Chat que le nombre de noeuds d'échec, dans un arbre de recherche avec N noeuds, est égal à $N+1$. La valeur de I est intéressante parce que le coût moyen de recherche (clef présente) pour un arbre particulier avec N noeuds est $C(N) = 1 + \frac{I(N)}{N}$. Si je fais la même chose pour la recherche de clefs non-présentes, je dois définir E, la longueur de chemin extérieur ("External Path Length"), qui est égale à la somme des profondeurs des noeuds d'échec. Le coût moyen de recherche (clef absente) pour un arbre particulier avec N noeuds est $C'(N) = \frac{E(N)}{(N+1)}$. Dans le Chat nous avons remarqué que

$$E(N) = I(N) + 2N \qquad N \geq 1 \tag{2}$$

et nous avons pensé faire une preuve par induction sur N.

(a) Démontrez que

$$C_N = \left(1 + \frac{1}{N}\right) C'_N - 1$$

$$\begin{aligned}
C_N &= \frac{I(N)}{N} + 1 \\
&= \frac{I(N)}{N} + 2 - 1 \\
&= \frac{I(N)}{N} + \frac{2N}{N} - 1 \\
&= \frac{I(N) + 2N}{N} - 1 \\
&= \frac{E(N)}{N} - 1 \\
&= \frac{E(N)(N+1)}{N(N+1)} - 1 \\
&= \frac{N\ E(N) + E(N)}{N(N+1)} - 1 \\
&= \frac{E(N)}{(N+1)} + \frac{E(N)}{N(N+1)} - 1 \\
&= C'_N + \frac{C'_N}{N} - 1 \\
&= \left(1 + \frac{1}{N}\right) C'_N - 1
\end{aligned}$$

(b) Il y a plus qu'une façon de faire une preuve de (2) par induction. Par exemple, nous pouvons regarder les sous-arbres, et procéder par l'induction forte ("strong induction", par exemple Johnsonbough, 7ième édition, p. 102). Prenons plutôt l'approche de regarder un noeud de profondeur maximale, et de procéder par induction simple en enlèvant ce noeud. Donnez la preuve de (2) en utilisant cette deuxième approche.

Preuve par induction sur N.

Cas de base : $N = 1$.
$E(N) = I(N) + 2N = I(1) + 2(1) = 0 + 2 = 2$, ce qui est vrai puisqu'un arbre binaire de 1 noeud a 2 noeuds d'échecs potentiels, chacun d'une profondeur de 1, donc $E(1) = 1 + 1 = 2$.

Pas d'induction :
Supposons $E(N) = I(N) + 2N$ vrai pour $N \geq 2$. On veut $E(N-1) = I(N-1) + 2(N-1)$.
Si on enlève 1 noeud de profondeur maximale, alors $I(N-1) = I(N) - d$, où d est la profondeur du noeud enlevé. Enlever ce noeud enlève aussi 2 noeuds d'échec de profondeur $d+1$, mais ajoute 1 noeud d'échec de profondeur d. Alors

$$\begin{aligned} E(N-1) &= E(N) - 2(d+1) + d \\ &= E(N) - d - 2 \\ &= I(N) + 2N - d - 2 \quad \text{Par hypothèse} \\ &= I(N-1) + d + 2N - d - 2 \\ &= I(N-1) + 2N - 1 \\ &= I(N-1) + 2(N-1) \end{aligned}$$

En continuant d'enlever un noeud de profondeur maximale, on se retrouve éventuellement au cas de base. Nous avons prouvé par induction que $E(N) = I(N) + 2N$ pour tout $N \geq 1$.

3. (2 points) La fonction H_N est approximativement égal à lnN, et dans le Tp5 du 2 juin vous avez vu démontrées des bornes logarithmiques inférieures et supérieures dans le cas spécial où N a la forme 2^{m+1}. Ce qui veut dire que H_N tend vers l'infinie, mais pas très vite. (Moins vite est meilleur pour nous.) On pourrait même dire que la série H_N frôle la finitude, i.e., elle échappe à peine de rester finie, parce que la série

$$H_N^{(r)} = \sum_{i=1}^{N} \frac{1}{i^r}$$

reste finie pour toute valeur de r strictement supérieure à 1.

Démontrez que $H^{(r)}_{2^m-1}$ est inférieur à $\sum_{k=0}^{m-1} \frac{2^k}{2^{kr}}$, ce qui est inférieur à $\frac{2^{r-1}}{2^{r-1}-1}$ (aussi à démontrer). De ce point de vue, H_N ne tend pas très vite vers l'infini.

Preuve par induction sur m.

Cas de base $m = 1$:

$$H_{(1)} = 1 \leq \sum_{k=0}^{0} \frac{2^0}{2^0} = 1$$

Hypothèse d'induction : On suppose vrai pour :

$$H^{(r)}_{2^m-1} \leq \sum_{k=0}^{m-1} \frac{2^k}{2^{kr}}$$

Pas d'induction :

$$\begin{aligned} H^{(r)}_{2^{m+1}-1} &= H^{(r)}_{2^m-1} + \frac{1}{(2^m)^r} + ... + \frac{1}{(2^{m+1}-1)^r} \\ &\leq \sum_{k=0}^{m-1} \frac{2^k}{(2^k)^r} + \frac{1}{(2^m)^r} + ... + \frac{1}{(2^m)^r} \\ &= \sum_{k=0}^{m-1} \frac{2^k}{(2^k)^r} + \frac{2^m}{2^{mr}} \\ &= \sum_{k=0}^{m} \frac{2^k}{(2^k)^r} \end{aligned}$$

■

Pour la deuxieme partie, on cherche $\sum_{k=0}^{m-1} \frac{2^k}{(2^k)^r} \leq \frac{2^{r-1}}{2^{r-1}-1}$. On remarque ici la forme de la somme géométrique.

$$\begin{aligned} \sum_{k=0}^{m-1} \frac{2^k}{(2^k)^r} &\leq \sum_{k=0}^{\infty} \frac{2^k}{(2^k)^r} \\ &= \frac{1}{1-\frac{1}{2^{r-1}}} \\ &= \frac{2^{r-1}}{2^{r-1}-1} \end{aligned}$$

■

4. (2 points)

(a) How many bits are required per node to store the height of a node in an N-node AVL tree ?

Comme la hauteur d'un arbre AVL de N noeuds est d'environ $\log N$, et que l'entier le plus grand représenté par k bits est $2^k - 1$. On a donc que $hauteur = \log N = 2^k - 1$. En isolant k, on se retrouve avec $k = \log(\log N + 1)$. On peut donc estimer le nombre de bits requis pour entreposer la hauteur d'un noeud dans un arbre AVL à N noeuds comme étant $O(\log(\log N))$ [1]

(b) What is the smallest AVL tree that overflows an 8-bit height counter ?

On prends la hauteur h du plus petit arbre qui utilise un compteur $k = 7$ bits (car le premier des 8 bits est gardé pour la représentation complément-à-deux). De la question (a), on sait que $h = \log N$ et $\log(\log N + 1) = k = 7$. On obtient $log(h + 1) = 7$, ce qui implique $h = 2^7 - 1$. La hauteur est donc au moins 127. [1]

(c) Quelle est la pertinence de la question ? Pourquoi Weiss l'a-t-il posé ?

On comprend que garder la hauteur de l'arbre comme attribut dans chaque noeud prend peu d'espace mémoire. Ainsi, si on garde l'information, il ne faut pas recalculer la hauteur à chaque fois que l'information doit être utilisée.

(d) Si la réponse à la partie 4b est h, donnez une borne inférieure grossière pour G_h, $G_h >> \phi^{h+3}$. Parce que $\phi > 1.618$, en faisant le carré cinq fois nous pouvons facilement calculer que $\phi^{32} >> 4 \cdot 10^6$ par exemple. Quelle est la pertinence de ma question ?

On remplace tout simplement h par la réponse trouvée en (b), soit 127. Ce qui donne $G_h >> \phi^{127+3}$, où $\phi^{127+3} = 1.47 \cdot 10^{27}$. Donc $G_h >> 1.47 \cdot 10^{27}$. Cette borne inférieure nous confirme encore la question (c), où l'on voit que la quantité de mémoire dans un arbre immense reste très faible.

Références

[1] Anonyme. *Problem DS-04-21*. URL : http://ms.ntub.edu.tw/~spade/teaching/x-DS2006-1/DS-04-21.pdf.