Devoir 3 IFT2015

Catherine Larivière 0955948

Dominique Vigeant 20129080

14 juillet 2021

1 Partie pratique

Remis sous la forme HashTable.java, Node.java, HashNode.java, Tombstone.java.

2 Partie théorique

1. (2 points) Buddy System

$b_n = 2^{2k-1}\alpha^2 + 2^{k-1}\alpha$:

On remarque que la première partie de la somme correspond à la somme des 2^k premiers entiers, qui est égale à $\frac{2^k(2^k+1)}{2}$. Cette somme à elle seule équivaut au cas où $\alpha = 1$. On traite ensuite le reste de la somme dans le cas où $\alpha > 1$. On comprend aussi que α sert à atteindre un nombre entre deux puissances de 2, ce qui nous aide a décomposer les termes de notre somme.

$$\begin{aligned}
b_{2^k\alpha} &= 1 + 2 + ... + 2^k + (2^k+1) + (2^k+2) + ... + (2^k + (2^k\alpha - 1 - 2^k)) + 2^k\alpha \\
&= \sum_{i=1}^{2^k} i + (2^k+1) + (2^k+2) + ... + (2^k + (2^k\alpha - 1 - 2^k)) + 2^k\alpha \\
&= \underbrace{\frac{2^k(2^k+1)}{2}}_{\text{première somme}} + \underbrace{(2^k+1) + (2^k+2) + ... + (2^k + (2^k\alpha - 1 - 2^k))}_{2^k\alpha - 1 - 2^k \text{ termes}} + 2^k\alpha \\
&= \underbrace{(2^{2k-1} + 2^{k-1})}_{\text{première somme}} + \underbrace{(2^k + ... + 2^k)}_{2^k\alpha - 1 - 2^k \text{ fois}} + \big(1 + 2 + ... + (2^k\alpha - 1 - 2^k)\big) + 2^k\alpha \\
&= (2^{2k-1} + 2^{k-1}) + 2^k(2^k\alpha - 1 - 2^k) + \sum_{i=1}^{2^k\alpha - 1 - 2^k} i \; + 2^k\alpha \\
&= (2^{2k-1} + 2^{k-1}) + (2^{2k}\alpha - 2^{2k} - 2^k) + \frac{(2^k\alpha - 1 - 2^k)(2^k\alpha - 1 - 2^k + 1)}{2} + 2^k\alpha \\
&= (2^{2k-1} + 2^{k-1}) + (2^{2k}\alpha - 2^{2k} - 2^k) + \frac{(2^k\alpha - 2^k - 1)(2^k\alpha - 2^k)}{2} + 2^k\alpha \\
&= (2^{2k-1} + 2^{k-1}) + (2^{2k}\alpha - 2^{2k} - 2^k) + \frac{2^{2k}\alpha^2 - 2^{2k}\alpha - 2^{2k}\alpha + 2^{2k} - 2^k\alpha + 2^k}{2} + 2^k\alpha \\
&= (2^{2k-1} + 2^{k-1}) + (2^{2k}\alpha - 2^{2k} - 2^k) + (2^{2k-1}\alpha^2 - 2^{2k}\alpha + 2^{2k-1} - 2^{k-1}\alpha + 2^{k-1}) + 2^k\alpha \\
&= 2^{2k-1}\alpha^2 + 2^{k-1}\alpha
\end{aligned}$$

$\underline{a_n = 2^{2k+1}(\alpha - \frac{2}{3}) + \frac{1}{3}}$:

On décompose la série un peu pour essayer de comprendre le fonctionnement. Posons c_{2^k}, la suite a_n avec $\alpha = 1$:

$$\begin{aligned}
c_{2^0} &= 1 \\
c_{2^1} &= 1+2 \\
c_{2^2} &= 1+2+4+4 \\
c_{2^3} &= 1+2+4+4+8+8+8+8 \\
&\vdots
\end{aligned}$$

Et donc, avec c_{2^k}, on trouve le polynôme caractéristique :

$$\begin{aligned}
4c_{2^k} - 1 &= c_{2^k+1} \\
4c_{2^k+1} - 1 &= c_{2^k+2} \\
c_{2^k+2} - c_{2^k+1} &= 4c_{2^k+1} - 4c_{2^k} \\
c_{2^k+2} &= 5c_{2^k+1} - 4c_{2^k}
\end{aligned}$$

$$\begin{aligned}
r^{2^k+2} &= 5r^{2^k+1} - 4r^{2^k} \\
\frac{r^{2^k+2}}{r^{2^k+2-2}} &= \frac{5r^{2^k+1}}{r^{2^k+1-2}} - \frac{4r^{2^k}}{r^{2^k-2}} \\
r^2 &= 5r^1 - 4r^0 \\
0 &= r^2 - 5r + 4 \\
r &= \frac{5 \pm 3}{2}
\end{aligned}$$

$$\begin{aligned}
c_{2^k} &= K_1\left(\frac{5+3}{2}\right)^k + K_2\left(\frac{5-3}{2}\right)^k \\
c_{2^0} &= 1 = K_1(4)^0 + K_2(1)^0 = K_1 + K_2 \\
c_{2^1} &= 3 = K_1(4)^1 + K_2(1)^1 = 4K_1 + K_2 \\
K_1 &= \frac{2}{3} \quad K_2 = \frac{1}{3} \\
c_{2^k} &= \frac{2}{3}4^k + \frac{1}{3}1^k \\
&= \frac{2^{2k+1}+1}{3}
\end{aligned}$$

Comme pour b_n, on calcule le reste de la somme pour $\alpha > 1$.

$$\begin{aligned}
a_n &= c_{2^k} + \underbrace{2^{k+1} + 2^{k+1} + \ldots + 2^{k+1}}_{2^k\alpha - 2^k \text{ fois}} \\
&= c_{2^k} + \left(2^k\alpha - 2^k\right)2^{k+1} \\
&= \frac{2^{2k+1}+1}{3} + 2^{2k+1}\alpha - 2^{2k+1} \\
&= 2^{2k+1}\left(\alpha - \frac{2}{3}\right) + \frac{1}{3}
\end{aligned}$$

$\frac{4}{3} \leq \rho(\alpha) \leq \frac{3}{2}$:

Borne inférieure : La fonction $\rho(\alpha)$ est minimale à ses extrémités, c'est-à-dire

$$\rho(1) = \frac{4(1-\frac{2}{3})}{1^2} = \frac{4}{3}$$
$$\lim_{\alpha \to 2^-} \rho(\alpha) = \frac{4(2-\frac{2}{3})}{2^2} = \frac{4}{3}$$

Le minimum de la fonction $\rho(\alpha)$ est $\frac{4}{3}$.

Borne supérieure : On trouve le maximum de $\rho(\alpha)$ avec la dérivée

$$\begin{aligned}
\rho'(\alpha) &= \frac{d}{d\alpha}\ \frac{4(\alpha-\frac{2}{3})}{\alpha^2} \\
&= \frac{d}{d\alpha}\left(\frac{4}{\alpha} - \frac{8}{3\alpha^2}\right) \\
&= \frac{-4}{\alpha^2} + \frac{16}{3\alpha^3} \\
0 &= \frac{-4}{\alpha^2} + \frac{16}{3\alpha^3} \\
\alpha &= \frac{4}{3} \\
\rho\left(\frac{4}{3}\right) &= \frac{4(\frac{4}{3}-\frac{2}{3})}{\frac{4}{3}^2} = \frac{3}{2}
\end{aligned}$$

Point critique : $\left(\frac{4}{3}, \frac{3}{2}\right)$

$$\begin{aligned}
\rho''(\alpha) &= \frac{d}{d\alpha}\left(\frac{-4}{\alpha^2} + \frac{16}{3\alpha^3}\right) \\
&= \frac{8}{\alpha^3} - \frac{16}{\alpha^4} \\
\rho''\left(\frac{4}{3}\right) &= \frac{-48}{19} < 0
\end{aligned}$$

$$\left(\frac{4}{3}, \frac{3}{2}\right) \text{ est bien un maximum.}$$

Le maximum de la fonction $\rho(\alpha)$ est $\frac{3}{2}$.

2. (1 point) Algorithme de Prim

a) Même difficulté pour un arbre sous-tendant minimal que maximal. L'algorithme de Prim se fait en étapes successives, où à chaque étape, on choisit un noeud comme racine et on ajoute une arête associée. Selon Weiss [4] : "The algorithm then finds, at each stage, a new vertex to add to the tree by choosing the edge (u, v) such that the cost of (u, v) is the smallest among all edges where u is in the tree and v is not." Pourtant, il est tout aussi possible de choisir la plus grande valeur d'arête dans les arêtes, pour obtenir l'arbre sous-tendant maximal, comme le montre cet algorithme-ci, de GeekforGeeks [2]. Donc, l'algorithme de Prim trouve aussi efficacement un arbre sous-tendant maximal que minimal.

b) L'algorithme fonctionne encore sans problème. On prend seulement l'arête avec le poids minimal dans toutes les arêtes. Dans l'algorithme de Prim, l'arête ajoutée à l'arbre sous-tendant est l'arête de poids minimal qui connecte à un noeud en dehors de l'arbre sous-tendant. Une comparaison de nombres négatifs ou positifs ne change rien dans la comparaison du poids, puiqu'on cherche le plus petit.

Une manière de le voir pourrait être d'ajouter le même gros nombre positif à toutes les arêtes de sorte à les rendre toutes positives, et voir que l'algorithme fonctionne puisque chacune des arêtes est positive [3].

3. (1 point) Hashage pour chaînes

Avec les propriétés du modulo [5], on prouve facilement [1] :

$$\begin{aligned}(((ax) \bmod M) + b) \bmod M &= [((ax \bmod M) \bmod M) + (b \bmod M)] \bmod M \\ &= [(ax \bmod M) + (b \bmod M)] \bmod M \\ &= (ax + b) \bmod M\end{aligned}$$

Références

[1] FLEABLOOD. *Solving a congruence/modular equation.* URL : https://math.stackexchange.com/questions/2855137/solving-a-congruence-modular-equation-ax-mod-m-b-mod-m-ax-b.

[2] Geeks for GEEKS. *Maximum Spanning Tree using Prim's Algorithm.* URL : https://www.geeksforgeeks.org/maximum-spanning-tree-using-prims-algorithm/.

[3] Jackson TALE. *Is Minimum Spanning Tree afraid of negative weights ?* URL : https://stackoverflow.com/questions/10414043/is-minimum-spanning-tree-afraid-of-negative-weights.

[4] Mark Allen WEISS. *Data Structures and Algorithm Analysis in Java.* Pearson Education Inc., 2012. ISBN : 978-0-13-257627-7.

[5] WIKIPEDIA. *Modulo operation.* URL : https://en.wikipedia.org/wiki/Modulo_operation.