Devoir Maison d'Algorithmique

Exercice 1. Arbre binaire optimal

On souhaite utiliser un arbre binaire de recherche (ABR) comme structure de donnée pour stocker un dictionnaire. Le coût de la recherche d'un mot dans ce dictionnaire est alors égale, en pire cas, à la profondeur de cet arbre. Le but cet exercice est de proposer une construction d'un tel ABR minimisant le temps de recherche moyen (en nombre de comparaisons). Un tel ABR est dit optimal.

Plus précisément, soit E un ensemble ordonné de n éléments $e_0 < e_1 < \cdots < e_{n-1}$ ayant chacun une probabilité p_i d'être recherché. Il s'agit de construire un ABR dont la profondeur moyenne est

$$\min_{A \in \{\text{ABR stockant } E\}} \sum_{i=0}^{n-1} p_i \times \Delta_A(e_i)$$

où $\Delta_A(e_i)$ vaut 1+ la profondeur de e_i dans A (par convention, la profondeur de la racine est 0). La figure 1 illustre les 5 configurations possibles pour un ABR optimal à trois éléments.

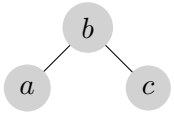
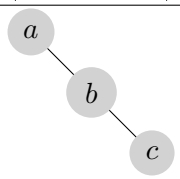
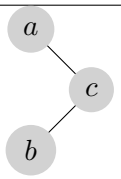
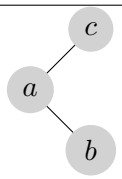
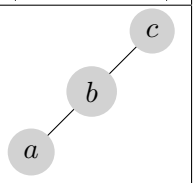
(p_a, p_b, p_c)	(0.2, 0.4, 0.4)	(0.5, 0.3, 0.2)	(0.6, 0.2, 0.2)	(0.3, 0.2, 0.5)	(0.1, 0.3, 0.6)
ABR optimal					
Nombre moyen de comparaisons	1.6	1.7	1.6	1.7	1.5

FIGURE 1 – Exemples d'ABR optimaux avec trois éléments $a < b < c$ de probabilités respectives p_a, p_b, p_c.

On note $C(i, j)$ le nombre moyen de comparaisons pour la recherche dans un sous-ABR optimal, stockant les éléments $e_i, \ldots, e_{j-1}$.

a. Justifier que tout sous-arbre d'un ABR optimal est un ABR optimal.

b. En déduire une caractérisation récursive de $C(i, j)$ en précisant les conditions aux bords.

c. Proposer un algorithme recursif avec mémoisation calculant la valeur le coût optimal $C(i, j)$.

d. Analyser son coût en temps et en mémoire.

e. Proposer un algorithme itératif calculant cette valeur et construisant l'ABR optimal correspondant.

f. Analyser son coût en temps et en mémoire.

g. Analyser son coût en nombre de défaut de cache, dans un cache de taille Z et de longueur de ligne $L \ll Z$. On supposera qu'un tableau bi-dimensionnel T de dimension $m \times n$ est stocké par un tableau unidimensionnel S de taille mn tel que : $T[i, j] = S[i*n+j]$. Si ce coût vous semble améliorable, proposer comment l'améliorer en modifiant l'algorithme itératif de la question e.