TD 3 : Tarbres et listes à raccourcis

Exercice 1. *Opérations sur les tarbres*

1. Effectuer l'INSERTION des couples suivants dans l'ordre dans lequel ils sont donnés dans un tarbre initialement vide, en utilisant l'ordre alphabétique pour classer les lettres : $(C, 0.4)$, $(G, 0.3)$, $(F, 0.7)$, $(B, 0.8)$, $(E, 0.5)$, $(D, 0.9)$, $(A, 0.2)$, $(H, 0.1)$. *Représenter les tarbres obtenus à chaque étape.*
2. Supprimer l'élément D du tarbre de la question précédente.

Exercice 2. *Tarbre et insertion aléatoire*

1. Montrer qu'un tarbre pour un ensemble de valeur V est exactement l'arbre binaire de recherche qu'on obtiendrait en insérant les valeurs de V dans un arbre initialement vide, par ordre de priorité croissant, en utilisant l'algorithme d'INSERTION classique.
2. En déduire que l'espérance de la hauteur d'un arbre binaire de recherche pour un ensemble V de taille n est $O(\log n)$ lorsqu'on insère ses éléments dans un ordre aléatoire.

Exercice 3. *Simulation d'un réel aléatoire*

L'implantation des tarbres nécessite de savoir tirer un nombre réel aléatoirement, ce qui est bien sûr impossible en pratique. En pratique, on ne sait que produire des bits aléatoires [1]. Pour simuler l'obtention d'un réel aléatoire entre 0 et 1, on écrit chaque priorité p en base 2, sous la forme

$$p = \sum_{j=1}^{+\infty} b_j 2^{-j}$$

où les b_j sont des bits. L'idée est de ne tirer les bits de p que lorsque l'on en a besoin pour comparer p à une autre priorité q.
Plus précisément, chaque priorité est un tableau de bits de longueur variable, initialement de longueur nulle. Pour comparer deux priorités, on teste si les bits existants suffisent à les départager et sinon on tire aléatoirement des bits pour augmenter la taille des tableaux, tant qu'on ne peut pas départager les deux priorités.

1. Écrire l'algorithme de comparaison de deux priorités.
2. Calculer l'espérance du temps de calcul de l'algorithme proposé.

On veut faire le contraire pour les liste à raccourcis. Supposons qu'au lieu de tirer des bits aléatoires dans la construction d'une liste à raccourcis, on associe à chaque élément un réel tiré aléatoirement dans $[0, 1]$ et qu'on utilise ce réel pour déterminer le nombre de copies de l'élément.

3. Décrire une règle permettant de déterminer le nombre de copies d'un élément à partir du réel qui lui est associé, de manière à avoir le même comportement que les listes à raccourcis décrites en cours.

Exercice 4. *Taille des listes à raccourcis*

✎ Montrer que l'espérance du nombre total de nœud d'une liste à raccourcis de n éléments est $O(n)$.

1. Et encore !