POLYTECH MARSEILLE Aix*Marseille Université **HUGO3**

Algorithmique

TD n°7 : les arbres et leurs parcours

Exercice 1 – Arbres binaires d'entiers

Q 1.1 Définissez le type `ab` représentant les arbres binaires dont les étiquettes des nœuds sont des entiers.

Q 1.2 Écrivez une fonction `enumeration` qui, pour un arbre donné, renvoie une liste composée par la concaténation de l'énumération du sous-arbre gauche, d'une liste contenant la racine, et l'énumération du sous-arbre droit.

Q 1.3 Écrivez une fonction `profondeur` qui, pour un arbre donné et un entier `k`, renvoie la liste des nœuds à la profondeur `k` s'il y a des nœuds à cette profondeur, et renvoie une liste vide sinon.

Exercice 2 – Manipulation d'arbres binaires d'entiers

Q 2.1 Étant donné deux arbres de type `ab`, écrivez une fonction `egalite` qui renvoie un booléen indiquant s'il y a égalité entre ces deux arbres.

Q 2.2 Écrivez une fonction `somme` qui, étant donné un arbre de type `ab`, renvoie la somme des éléments de cet arbre.

Q 2.3 Écrivez une fonction `miroir` qui, étant donné un arbre de type `ab`, renvoie l'image miroir de cet arbre. Le miroir d'un arbre consiste à inverser les fils gauches et les fils droits de l'arbre.

Q 2.4 Le Strahler d'un arbre binaire d'entiers est un nombre qui se définit de la façon suivante :

1. le Strahler de l'arbre vide vaut 0,
2. le Strahler d'un arbre non vide est égal :
   (a) à 1 + le Strahler d'un des sous-arbres si les Strahlers des sous-arbres gauche et droit sont égaux,
   (b) au max des deux Strahlers s'ils sont différents.

Écrivez une fonction `strahler` qui, étant donné un arbre de type `ab`, renvoie le Strahler de cet arbre.

Exercice 3 – Codage de Huffman

Comme nous l'avons vu dans le TD n°2, L'algorithme de Huffman consiste :

1. à dresser une liste des symboles de l'alphabet triée par ordre décroissant de fréquence d'apparition des symboles dans le texte (on en profite pour calculer ces fréquences). L'exemple ci-dessous vous rappellera cela.
2. puis à construire un arbre, dans lequel chaque symbole se trouve sur une feuille, de la manière suivante :
   (a) on part d'un arbre vide ;
   (b) on prend les deux symboles de la liste ayant les plus petites fréquences, appelons-les a_i et a_j, et on les enlève de la liste des symboles ;
   (c) on crée un symbole représentant la réunion des deux choisis, appelons-le a_{ij}, auquel on affecte la somme des fréquences de a_i et a_j, et on le rajoute à la liste ;
   (d) on rajoute alors à l'arbre un arbre binaire constitué d'une racine, a_{ij}, et de deux feuilles, a_i et a_j ;
   (e) on itère à partir de l'étape (b) jusqu'à ce qu'il ne reste plus qu'un seul symbole.
3. à parcourir l'arbre ainsi créé afin de déterminer le codage de chaque symbole de l'alphabet, comme le montre l'exemple ci-dessous.

Exemple : considérons cinq symboles, a_1, a_2, a_3, a_4 et a_5 apparaissant dans un texte avec les fréquences (probabilités) suivantes :

$$a_1 \equiv 40\% = 0,4 \qquad a_2 \equiv 0,2 \qquad a_3 \equiv 0,2 \qquad a_4 \equiv 0,1 \qquad a_5 \equiv 0,1.$$

L'algorithme de Huffman effectue alors les opérations suivantes :

1. a_4 est combiné avec a_5 car ce sont les caractères ayant les plus petites fréquences d'apparition. Leur combinaison forme alors un nouveau symbole que nous appellerons a_{45} et dont la fréquence d'apparition dans le texte est $0,1 + 0,1 = 0,2$. On créé un arbre binaire ayant pour racine a_{45} et comme feuilles a_4 et a_5 (cf. la figure 1.a).
2. On a maintenant 3 symboles avec les probabilités 0,2 : a_2, a_3 et a_{45}. On doit en combiner deux, peu importe lesquels. On va sélectionner arbitrairement les symboles a_3 et a_{45}. Ceux-ci en forment donc un nouveau, que nous appellerons a_{345} et dont la fréquence d'apparition est $0,2 + 0,2 = 0,4$. Cela nous donne l'arbre de la figure 1.b.

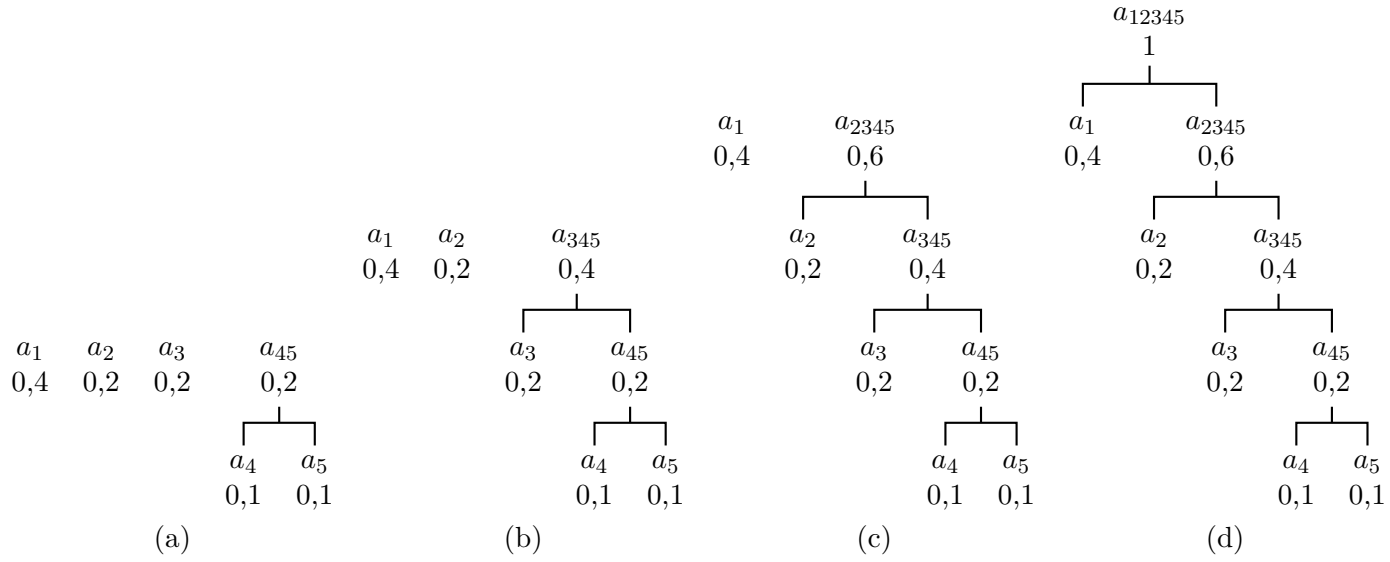

FIGURE 1 – Construction de l'arbre de Huffman.

3. On a maintenant deux symboles avec des fréquences de 0,4 : a_1 et a_{345}, plus un symbole, a_2, avec une fréquence de 0,2. On doit donc combiner a_2 avec, au choix, a_1 ou a_{345}. Choisissons de combiner a_2 et a_{345}. On obtient le symbole a_{2345} dont la fréquence est 0,6 (cf. figure 1.c).
4. Enfin, il ne reste plus que deux symboles, a_1 et a_{2345}. On les combine pour obtenir a_{12345} dont la fréquence d'apparition est 1. À cette étape, l'arbre déterminant les codes des symboles est construit : les symboles d'origine sont les feuilles et chaque « combinaison » représente un nœud de l'arbre (cf. figure 1.d).
5. À chaque nœud de l'arbre, on code l'une des deux arêtes sortantes avec un 1 et l'autre avec un 0 (cf. figure 2).

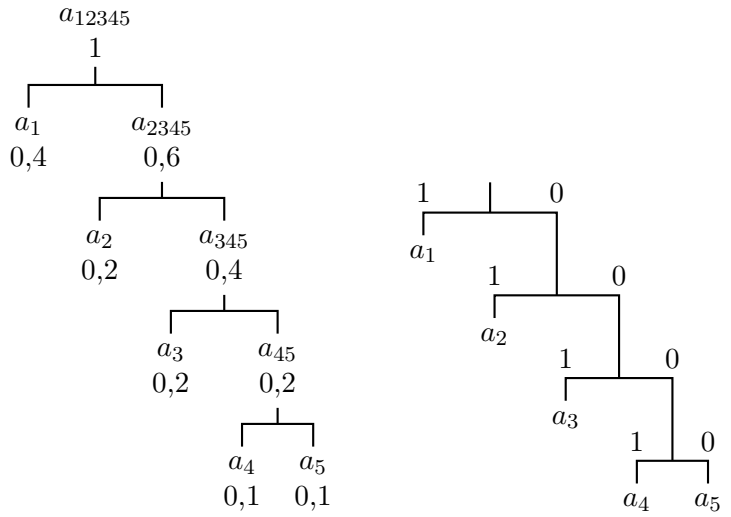

FIGURE 2 – Arbre de Huffman avec codes.

Pour obtenir le codage d'un symbole, il suffit de partir du nœud-symbole ayant une fréquence de 1 (autrement dit, la racine tout en haut de l'arbre), et de parcourir l'arbre jusqu'à ce qu'on atteigne le symbole désiré. Au fur et à mesure du cheminement, on note les 0/1 des arcs parcourus. Ceux-ci forment le code correspondant au symbole. Ainsi, le code de a_3 sera obtenu en partant de a_{12345}, en allant sur la branche vers a_{2345} (on obtient le premier bit du code : 0), puis en allant vers a_{345} (bit 0), puis en allant vers a_3 (bit 1). Le code de a_3 sera donc 001. Les arbres ci-dessus nous donnent donc les codes suivants :

$$a_1 \equiv 1 \qquad a_2 \equiv 01 \qquad a_3 \equiv 001 \qquad a_4 \equiv 0001 \qquad a_5 \equiv 0000.$$

Q 3.1 Écrivez une fonction `frequence`, prenant en paramètre une chaîne de caractères `str` ainsi qu'un tableau de 256 entiers et qui remplit ce dernier de telle sorte que chaque case contienne le nombre d'occurrences dans `str` du caractère dont le code ASCII est l'index de la case. Par exemple, après exécution de `frequence("aabd",tab)`, le tableau `tab` ne contient que des cases à 0, excepté `tab[97]` (caractère 'a') qui vaut 2, `tab[98]` et `tab[100]` qui valent 1.

Q 3.2 Proposez une structure `huffman_t` pour représenter un nœud d'un arbre de Huffman.

Q 3.3 Soit `abr_t` un type permettant de représenter des arbres binaires de recherche dont les clés sont des entiers et les valeurs sont des nœuds d'un arbre de Huffman. Écrivez une fonction `insert_abr` telle que `insert_abr (car,nb_occ,arbre)` rajoute à l'arbre `arbre` de type `abr_t` un couple dont la clé est `nb_occ` et la valeur est un nœud de type `huffman_t` sans enfant dont le caractère est `car`.

Q 3.4 Écrivez une fonction `cree_abr` prenant en argument un tableau `tab` généré par la fonction

`frequence` ainsi qu'un arbre vide de type `abr_t`. La fonction remplit l'arbre avec tous les éléments du tableau `tab` dont les fréquences sont non nulles.

Q 3.5 Écrivez une fonction `reunion` prenant en argument un arbre `arbre` de type `abr_t`. Cette fonction enlève de l'arbre les 2 nœuds de plus petites clés, appelons-les `a` et `b`. Elle rajoute alors à `arbre` un nouveau nœud correspondant à la réunion de `a` et `b` (étape 2 de l'algorithme de Huffman). Le caractère correspondant à une fusion sera `'\0'`.

Q 3.6 Écrivez une fonction `cree_huffman` qui prend en argument une chaîne de caractères et un arbre vide de type `abr_t`. La fonction remplit tout l'arbre de Huffman.

Q 3.7 Écrivez une fonction récursive `cree_code` qui prend en argument un nœud de type `abr_t` ainsi qu'une chaîne de carctères `str` et un tableau `tab` de 256 $\times$100 caractères. Si le nœud courant `n` de type `abr_t` est une feuille, la fonction copie `str` dans l'élément de `tab` correspondant au caractère du nœud `n`. Sinon, elle concatène à la fin de `str` un 0 avant d'appeler `cree_code` sur le fils gauche `n` ou un 1 avant d'appeler `cree_code` sur le fils droit (si ceux-ci existent).

Q 3.8 Écrivez une fonction `str2code` qui prend en argument une chaîne de caractères `s` ainsi qu'un tableau `tab` de 256 $\times$100 caractères. La fonction crée à partir de `s` l'arbre de Huffman puis remplit `tab` avec les codes de Huffman correspondant aux caractères de `s`.

Q 3.9 Écrivez une fonction `encode` qui prend en argument une chaîne de caractères `s` ainsi qu'un tableau `code` de 1000 caractères. La fonction place dans `code` la chaîne de caractère correspondant à `s` encodé par Huffman.