TD n°4

Partie II

On note S un dictionnaire constitué d'un ensemble de n mots, appelés clés, de même longueur l, sur un alphabet v constitué des symboles 0, 1, 2,... $k-1$. Ces clés permettent d'accéder à une information. Les opérations sur le dictionnaire sont :

```
recherche : clé * dictionnaire -> booléen
insertion : clé * dictionnaire -> dictionnaire
suppression : clé * dictionnaire -> dictionnaire
```

On représente S par un arbre dont chaque nœud possède au maximum k fils. Chaque chemin entre la racine et un nœud de l'arbre est un préfixe d'une clé de S. Cette structure particulière d'arbre est appelée k-arbre. La figure 1 donne un exemple de cette représentation.

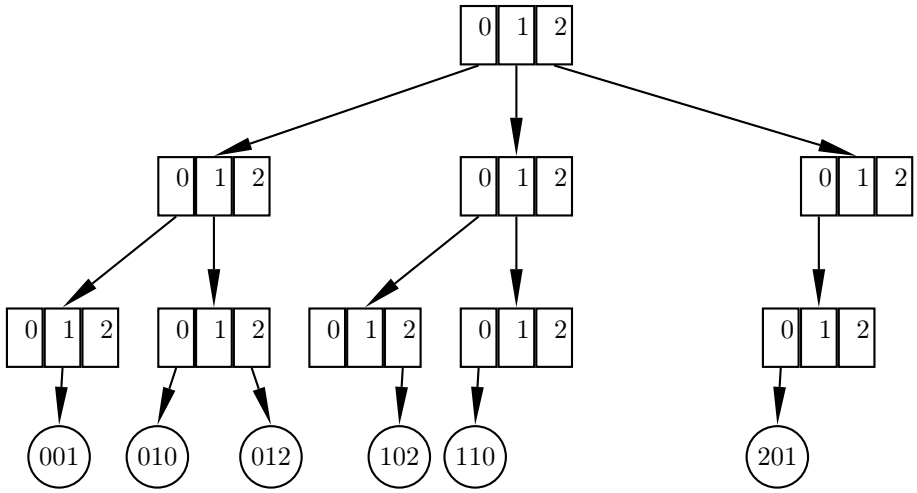

Fig. 1 – $S = \{001, 010, 012, 102, 110, 201\}$.

La structure de données utilisée pour représenter un k-arbre est la suivante en CAML :

```
type info = {cle : int vect ; valeur:int}
and noeudinterne = Nil | Feuille of info |
     Noeud of (noeudinterne vect)
and karbre = noeudinterne;;
```

II.A -

II.A.1) Déterminer la complexité en temps (nombre de nœuds parcourus) pour l'opération recherche.

II.A.2) Évaluer l'espace mémoire maximal utilisé pour représenter un dictionnaire S, de taille n, par un k-arbre. Commenter.

II.B - Une méthode simple pour réduire l'espace occupé par l'arbre consiste à supprimer les nœuds internes qui n'ont qu'un seul fils. Le passage d'un nœud père à un nœud fils nécessitera d'indiquer le nombre de lettres de la clé omises entre le père et le fils. Cette nouvelle structure s'appelle un k-arbre comprimé. La figure 2 présente le k-arbre de la figure 1 une fois comprimé.

II.B.1) Modifier la structure de données précédente pour représenter ce nouvel arbre. Donner un algorithme pour la fonction recherche dans un k-arbre comprimé. La démonstration de l'algorithme n'est pas demandée.

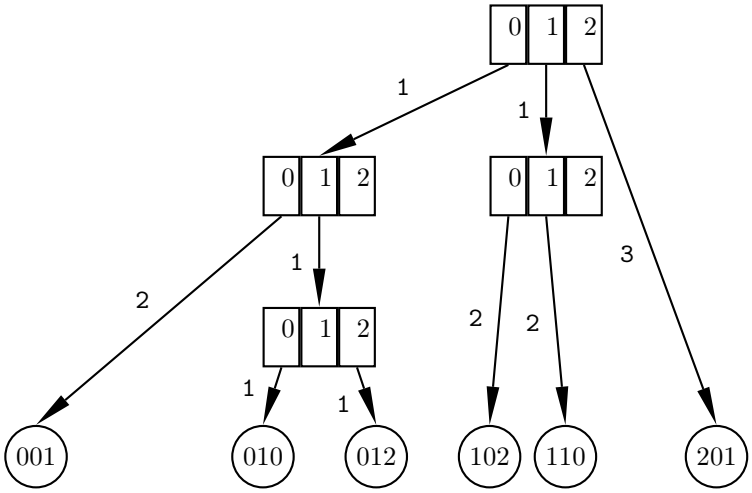

FIG. 2 – $S = \{001, 010, 012, 102, 110, 201\}$.

II.B.2) Construire à la main le k-arbre et le k-arbre comprimé associés au dictionnaire $S = \{001, 013, 310, 023, 230, 333, 121, 232, 321, 010\}$ sur l'alphabet $v = \{0, 1, 2, 3\}$.

II.B.3) Que devient l'espace mémoire requis pour représenter un dictionnaire S de taille n? Que devient la complexité en temps de la fonction recherche dans le pire cas ?

II.C - Soit à déterminer le temps moyen de recherche d'un élément d'un dictionnaire S quelconque, de taille n. On suppose que tous les dictionnaires de taille n sont équiprobables.

II.C.1) Calculer le nombre de k-arbres qui représentent des dictionnaires S de taille n. Montrer que parmi ces k-arbres, le nombre N_d de ceux dont le k-arbre comprimé à une profondeur strictement supérieure à d satisfait à l'inégalité :

$$N_d \leqslant \left(\begin{array}{c} k' \\ n \end{array} \right) \left[1 - \prod_{i=0}^{n-1} \left(1 - \frac{i}{k^{d-1}} \right) \right].$$

Soit $f_{d-1} : v' \to v^{d-1}$ la fonction qui à une clé associe son préfixe de longueur $d-1$. On pourra raisonner sur le nombre de sous-ensembles S de v' de taille n pour lesquels la restriction de la fonction f_{d-1} à S est injective.

II.C.2) En notant q_d le rapport

$$N_d / \left(\begin{array}{c} k' \\ n \end{array} \right)$$

montrer que la quantité $\overline{d_n}$, définie par

$$\overline{d_n} = \sum_{d \geqslant 1} d(q_{d-1} - q_d)$$

et correspondant à la profondeur moyenne du k-arbre comprimé pour un ensemble de taille n, satisfait à l'inégalité

$$\overline{d_n} \leqslant 2\lceil \log_k n \rceil + O(1)$$

La notation $\lceil x \rceil$ désigne le plus petit entier plus grand que x. L'inégalité suivante sera admise :

$$\prod_{i=0}^{n-1} \left(1 - \frac{i}{k^{d-1}} \right) \geqslant e^{\frac{n^2}{k^{d-1}}}$$

et on pourra montrer que

$$q_d \leqslant \min(1, \frac{n^2}{k^{d-1}}).$$