UNICAEN

Année 2019-2020

Portail Mathématiques, Informatique
Licence deuxième année (L2 Info, L2 maths, L2 MIASHS)

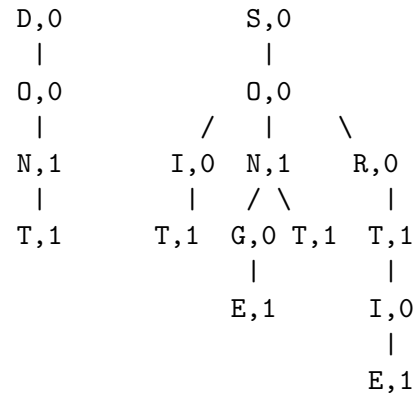
Algorithmique et structures de données
TD 10 – Trie ou arbre de préfixes

La structure de Trie (contraction de tree retrieval) a été inventé par Fredkin en 1959. Elle est également appelée arbre de préfixes (prefix tree) et arbre digital (digital tree).

Cette structure permet de représenter un ensemble de mots (un dictionnaire) par une forêt. On procède de la manière suivante : chaque nœud de la forêt est étiqueté par une lettre de l'alphabet et une valeur booléenne. Les enfants d'un nœud sont rangés par ordre alphabétique (les racines des arbres de la forêt sont également rangés par ordre alphabétique). Un mot est formé par concaténation des lettres des nœuds allant de la racine jusqu'à un nœud N ayant la valeur booléenne Vrai. Un préfixe de ce mot est obtenu par concaténation des lettres des nœuds allant de la racine jusqu'à un ancêtre de N. Certains de ces préfixes peuvent également être des mots du trie. C'est uniquement la valeur booléenne des nœuds qui permet de déterminer quels sont les mots du trie.

Soit T_0 le trie formé avec le dictionnaire $D_0 = \{$ DON, DONT, SOIT, SON, SONGE, SONT, SORT, SORTIE$\}$. Dans nos dessins des tries, nous écrirons 1 pour Vrai et 0 pour Faux.

Nous obtenons le trie suivant



Question 1. Dessinez le trie après insertion des mots DOIT et DO. Est-ce que la construction du trie dépend de l'ordre d'insertion de mots ?

Question 2. On implémente le trie avec un arbre binaire (relation premierEnfant-prochainFratrie). On considère la structure de nœud suivante

```
structure noeudT {
  valeur : caractère
  present : booléen
  enfant : pointeur sur noeudT
  fratrie : pointeur sur noeudT
}
type trie = pointeur sur noeudT
```

Dessinez T_0 avec cette représentation premierEnfant-prochainFratrie.

Question 3. Écrivez une procédure *longueurMax(T :trie) :entier* qui retourne la longueur d'un plus long mot du dictionnaire. A quoi correspond cette longueur ?

Question 4. Écrivez une procédure *nombreMotsTrie(T :trie) :entier* qui retourne le nombre de mots du trie.

Question 5. Expliquez ce que fait la procédure *mystère(M :chaîne de caractères) :trie* suivante :

```
mystère(M:chaîne de caractères):trie
   res: trie ; Nouveau(res) ; res->valeur = M[0] ; res->fratrie = None
   si longueur(M) = 1 alors
      res->present = Vrai ; res->enfant = None
   sinon
        res->present = Faux ; res->enfant = mystère(M[1:])
   retourner res
```

Question 6. Expliquez ce que fait la procédure *secret(T :trie, M :chaîne de caractères) :trie* suivante. Vous détaillerez à quoi correspondent les différents cas.

```
secret(T:trie, M:chaîne de caractères):trie
   si T = None alors retourner mystère(M)
   si T->valeur = M[0] alors
      si longueur(M) = 1 alors
         T->present = Vrai
      sinon
          T->enfant = secret(T->enfant, M[1:])
   sinon
          si T->valeur < M[0] alors
             T->fratrie = secret(T->fratrie, M)
          sinon
              tmp : trie ; tmp=mystère(M) ; tmp->fratrie = T
              T = tmp
   retourner T
```

Question 7. Écrivez une fonction *adresseMot(T :trie, M : chaîne de caractères) :trie* qui retourne un pointeur sur le nœud correspondant à M (le nœud contenant la dernière lettre de M) si ce mot est un mot du trie T et None sinon. Attention : M peut-être un préfixe sans être un mot du trie.

Question 8. Écrivez une procédure *afficheMotsPrefixe(T :trie, prefixe : chaîne de caractères)* qui affiche tous les mots de T commençant par le préfixe *prefixe*. En déduire une fonction *afficheMots(T :trie)* qui affiche tous les mots de T.

Question 9. * On définit la distance entre deux mots de même longueur comme le nombre de lettres différentes aux mêmes positions (cette distance est appelée distance de Hamming). Par exemple, dans le trie T_0, les mots DONT, SOIT et SORT sont à distances 1 de SONT.

Écrivez une procédure *rechercheProche(T :trie, M : chaîne de caractères) : booléen* qui renvoie VRAI si le trie T contient le mot M ou un mot à distance 1 de celui-ci.
Indication : utilisez la procédure *adresseMot*.

Question 10. * Modifiez cette procédure pour déterminer si le trie T contient un mot à distance k de M.

Question 11. Écrivez une procédure *nombrePrefixes(T :trie, M : chaîne de caractères) :entier* qui retourne le nombre de mots du trie T qui sont préfixes de M. Vous compterez 1 pour le mot lui-même lorsqu'il appartient à T.