Université d'Ottawa
Faculté de génie

École d'ingénierie et de
technologie de l'information

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

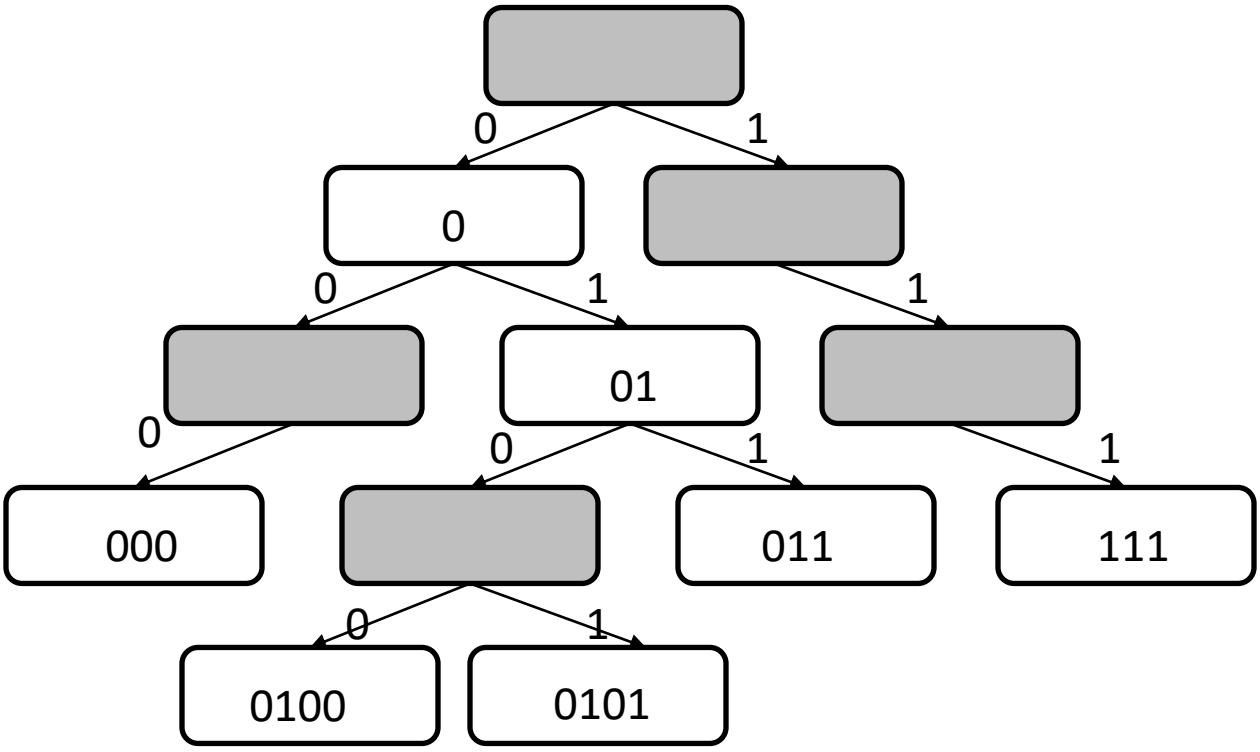
School of Information
Technology and Engineering

Devoir 2 (7.5%)

CSI2510

A remettre sur le Campus Virtuel avant 23h55 le 5 novembre 2017

Dans ce devoir, on vous demande de réaliser un arbre binaire contenant des chaines binaires telles que '01001' ou '011'. La séquence de chiffres contenue dans la chaine est utilisée comme clé. Ainsi un '0' dans la séquence créera une branche gauche alors que un '1' créera une branche droite. Voici une illustration d'un tel arbre contenant six chaines binaires.



Ici, les nœuds en gris sont des nœuds intermédiaires ne contenant pas de chaines binaires. Nous montrons ici, les chaines stockés dans chaque nœud mais il faut noter qu'en fait, ce n'est pas nécessaire d'inclure ces chaines puisque la position du nœud dans l'arbre nous donne la chaine correspondante. Il suffit donc de simplement associer à chaque nœud une variable booléenne indiquant si le nœud contient ou non une chaine (i.e. les nœuds gris ont une valeur *fausse*).

Ce genre d'arbres est appelé Trie. Ces Tries sont binaires dans le cas de chaines de '0' et de '1'. On peut aussi insérer des chaines de caractères (des mots) ; dans ce cas chaque nœud de l'arbre pourra avoir jusqu'à 26 enfants. Les Tries sont utilisés, entre autres, pour compresser des fichiers, ou pour créer des dictionnaires. Par exemple, lorsque des mots sont entrés dans un Trie, il est facile de les afficher en ordre alphabétique, que nous appelons ordre lexicographique dans le cas générale. L'ordre lexicographique se définit formellement ainsi :

Soit deux chaines $a_0\, a_1 \ldots a_p$ et $b_1\, b_2 \ldots b_q$, la chaine a est lexicographiquement inférieure (ou égale) à b si :
1) il existe un entier j, $0 <= j <= \min(p,q)$ tel que $a_i = b_i$ pour $i=0,\ldots,j-1$ et $a_j < b_j$ ou
2) $(p<q)$ et $a_i = b_i$ for all $i=0,1,\ldots p$,
Les mots dans un dictionnaire sont listés dans l'ordre lexicographique.

On vous donne trois classes :
`TreeNode` : représentant les nœuds dans l'arbre ;
`MyTrie` : représentant l'arbre Trie ;
`TestTrie` : permettant de tester le Trie.

Vous devez donc :

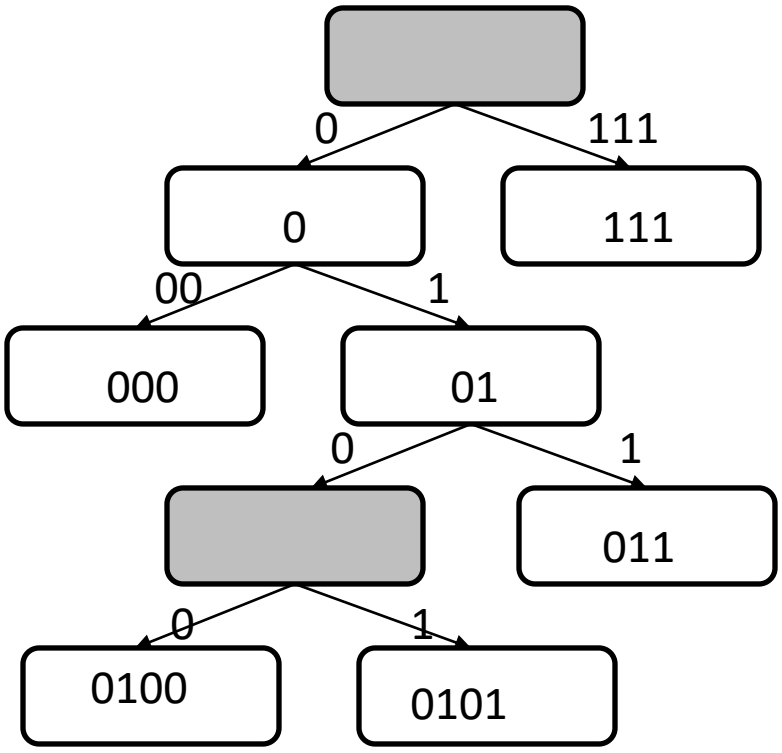
1) [60%] Ajouter les méthodes suivantes à la classe MyTrie :

   - `public boolean insert (String s)` : Cette méthode insère une nouvelle chaine binaire dans le Trie en créant les nœuds requis. Le nœud représentant cette chaine doit donc voir son attribut `isUsed` assigné à `true`. Si la chaine est déjà présente dans le Trie, la méthode retourne simplement false ;
   - `public boolean search(String s)` : Cette méthode retourne `true` si la chaine spécifiée se trouve dans le Trie.
   - `public void printStringsInLexicoOrder()` : Cette méthode affiche à la console toutes les chaines se trouvant dans le Trie dans l'ordre lexicographique. Vous devez donc choisir le bon type de parcours de l'arbre.

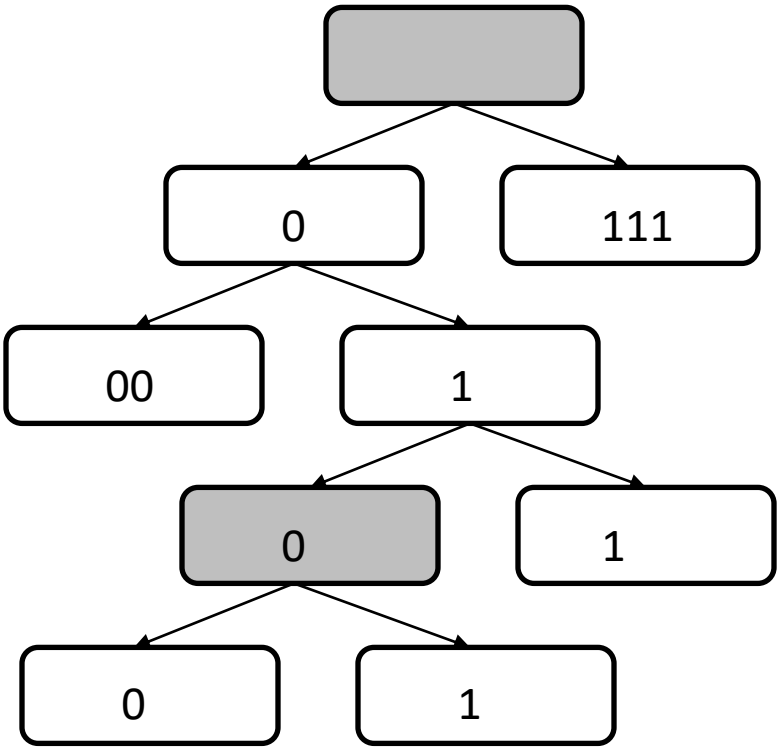
Important : Les classe `TreeNode` et `TestTrie` ne doivent pas être changés.

Créer aussi une classe `TestTrie2` testant chacune des méthodes demandées.

Les Tries peuvent être optimisés en compressant les nœuds intermédiaires ayant un seul enfant (les nœuds à deux enfants doivent demeurer dans le Trie). Cette stratégie est utile si plusieurs chaines partagent le même préfixe. Sur l'arbre de l'exemple précédent, la compression produira le résultat suivant :



On voit que dans ce cas, certaines branches ont été fusionnées ce qui signifie qu'une branche ne représente plus nécessairement un 1 ou un 0. Il faut donc inclure, dans chaque nœud, les bits correspondant au chemin compressé complet (on ne peut plus se contenter d'un simple booléen). Ce qui donne le résultat suivant :



2) [40%] La classe `TreeNodeWithData` peut être utilisée afin de représenter des nœuds contenant des chaines (cette classe ne doit pas être modifiée). La nouvelle classe Trie (avec compression) s'appelle `MyCompressedTree` et les méthodes suivantes doivent y être ajoutées :

   - `public MyCompressedTrie(MyTrie trie)` : un constructeur pour cette classe prenant en paramètre un Trie régulier qui doit être compressé.
   - `public void printStringsInLexicoOrder()` : Cette méthode affiche à la console toutes les chaines se trouvant dans le Trie dans l'ordre lexicographique. Vous devez donc choisir le bon type de parcours de l'arbre.

   Créer aussi une classe `TestCompressedTrie2` testant chacune des méthodes demandées.

Soumettre toutes vos classes sur le campus virtuel dans un fichier zip nommé d'après votre numéro d'étudiant, par exemple e12345.zip