UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

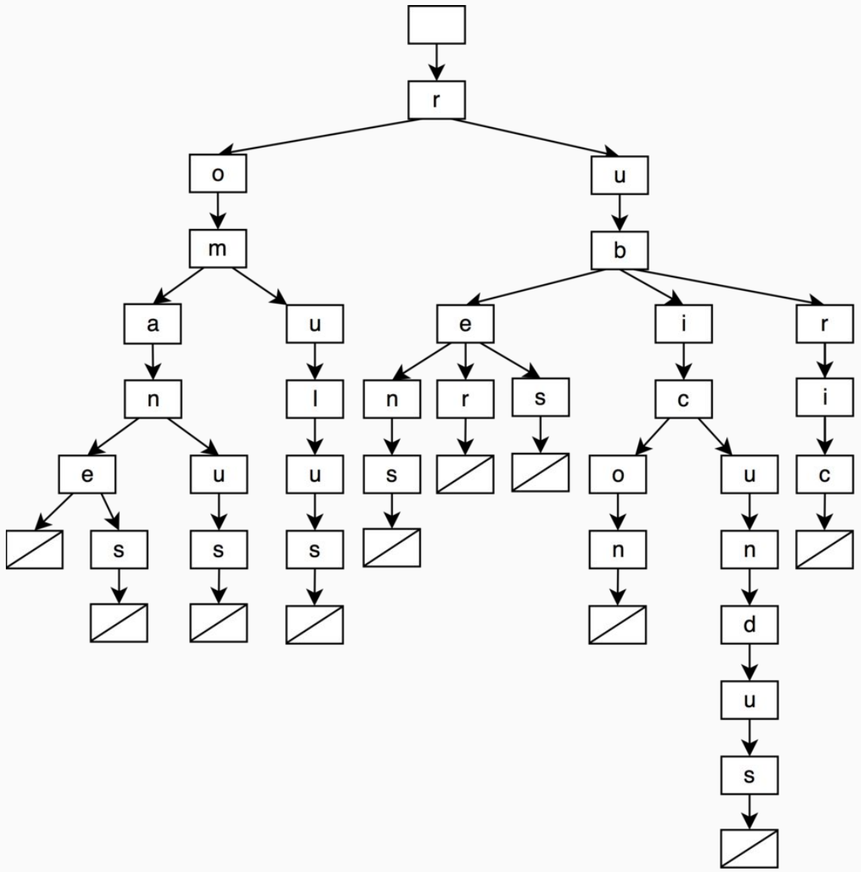
Welcome to Algorithms and Data Structures! - CS2100

Tries

Los árboles de prefijos *(efficient information reTRIEval data structure)*, es un tipo de árbol que es usualmente usado para almacenar caracteres. Normalmente, un tree representa un diccionario de palabras

A pesar de que cada nodo es utilizado para almacenar caracteres (también las aristas podrían representar los caracteres), los caminos entre estos representan palabras o partes de palabras. No olvidar el carácter de fin de cadena

Qué usos se les ocurren para las estructuras de datos que almacenan strings?
Autocompletar, editores de texto, procesadores de texto, etc



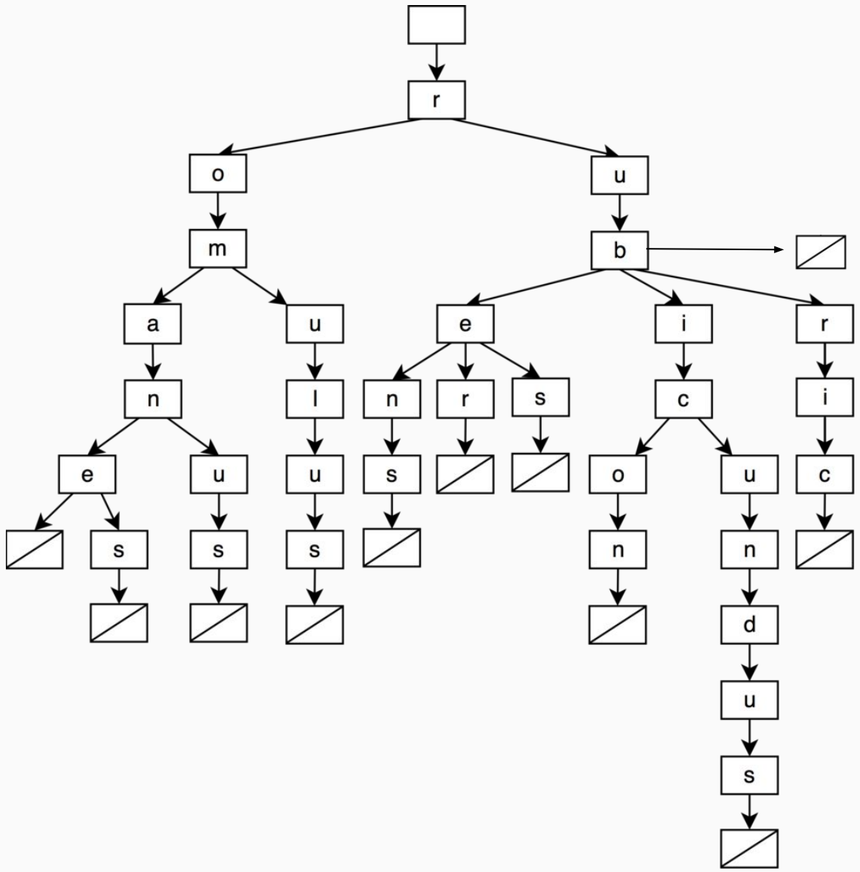
Tries (insertar)

Cómo agregar una nueva palabra? O(WORD SIZE)

1. Buscar el camino del string
2. Si se encuentra un null pointer, entonces se crea un nuevo nodo
3. Se adiciona una secuencia de hijos como letras falten de la palabra, finalizando con el fin de cadena

En el diccionario actual tenemos palabras como:
ROMANE, ROMANES, ROMANUS, ROMULUS, RUBENS, RUBES, RUBER, RUBICON, RUBICUNDUS y RUBRIC

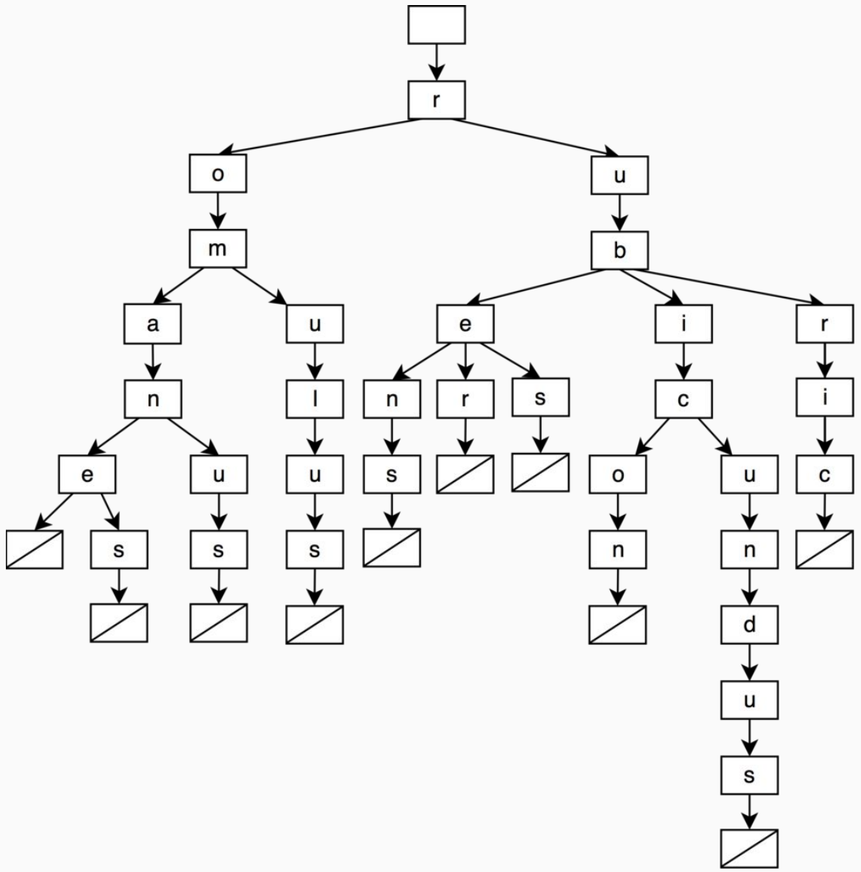
Cómo agregar RUB?



Tries

Entonces creemos el siguiente diccionario:

A
AN
ANCESTOR
AND
IN
IS
TAVERN
THERE
THE
TOWN

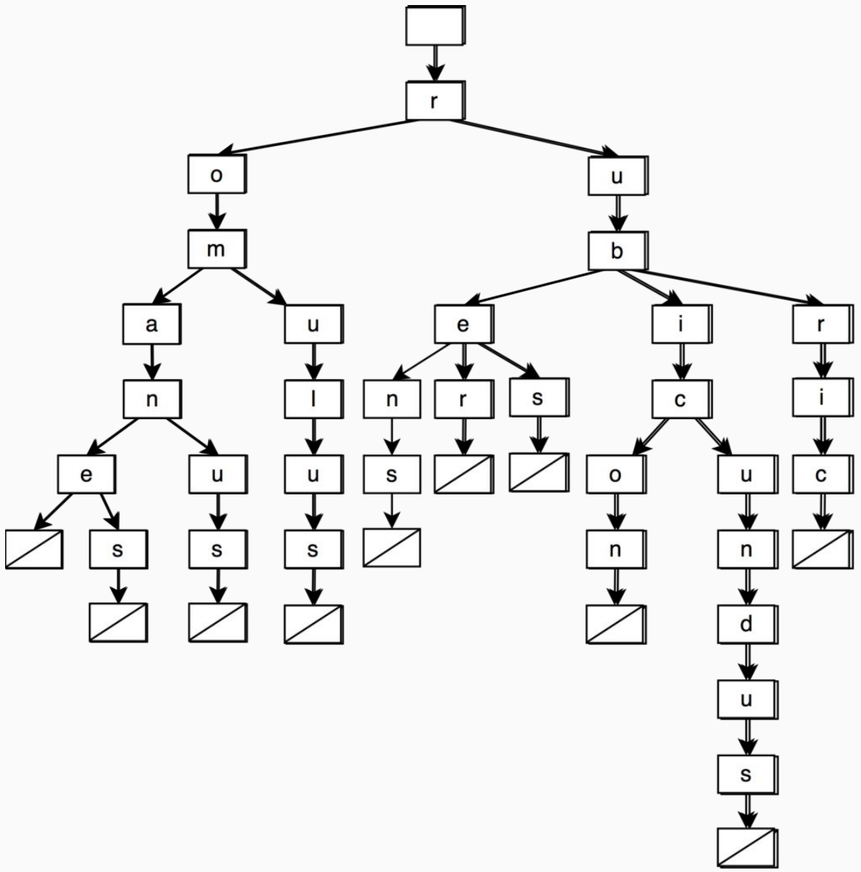


Tries (eliminar)

Cómo eliminar una palabra? O(WORD SIZE)

1. Se busca la hoja *“s”* que represente nuestra palabra
2. Empezamos a eliminar los nodos desde *“s”* hacia el root hasta encontrar un nodo que tenga más de un hijo

Cómo eliminar RUBENS?

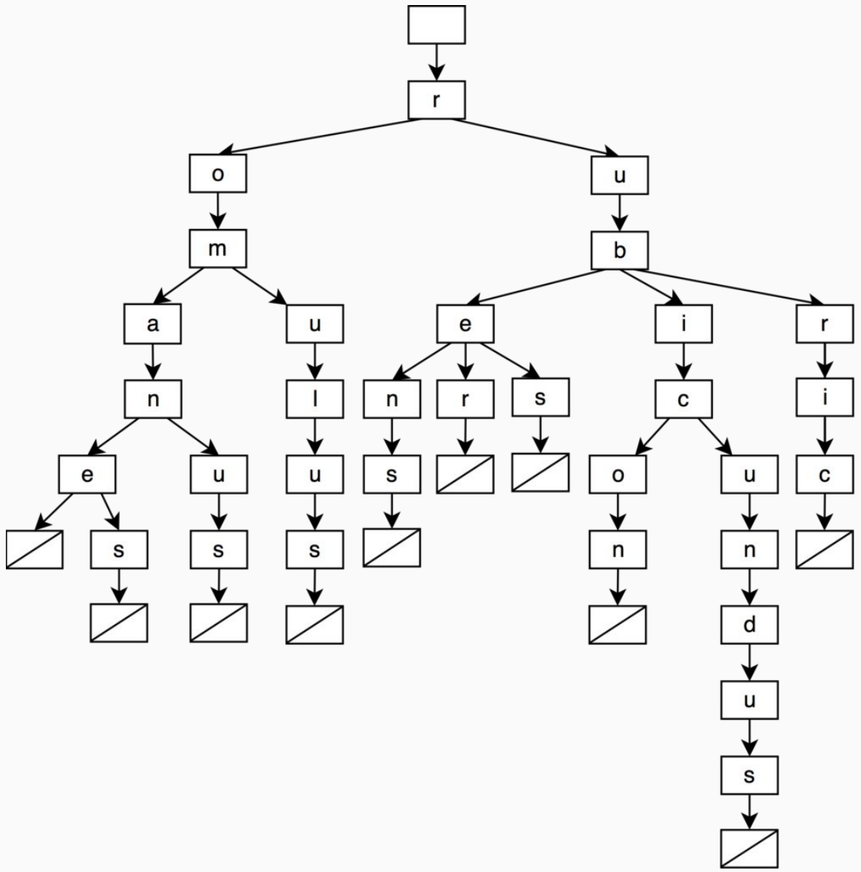


Tries (buscar)

Cómo buscar una palabra? O(WORD SIZE)

1. Se va buscando cada carácter de la palabra en el árbol
2. Si en algún punto nos encontramos con un null pointer, entonces la palabra no existe en el trie
3. De otra manera, si llegamos a una hoja que represente nuestra palabra, retornamos verdadero

Buscar ROMANE?



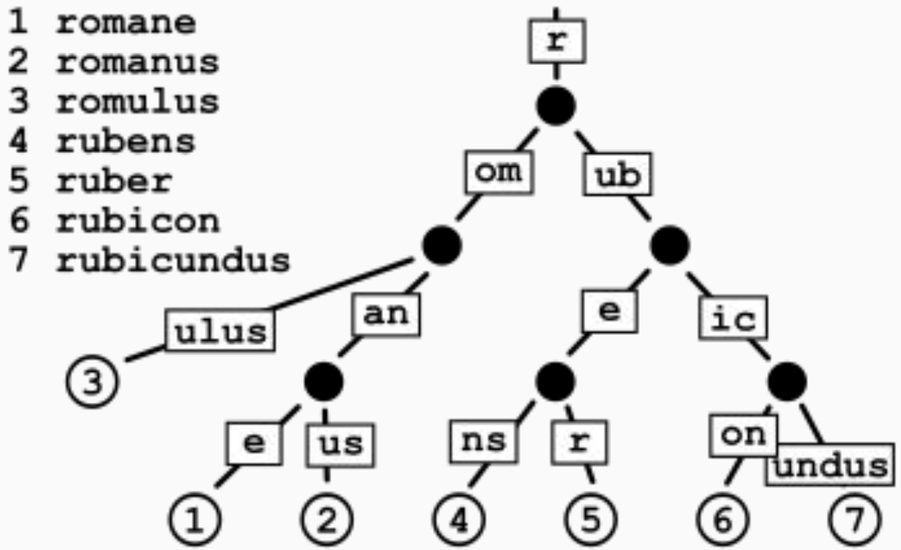
Patricia Tries

Es un tipo de trie que optimiza el espacio utilizado para representar el diccionario

El problema con los tries es que su conjunto de keys tiende a ser muy disperso. En muchos de los casos un nodo interno termina teniendo solo un descendiente

El problema anterior causa que los tries tengan una complejidad alta en términos de espacio

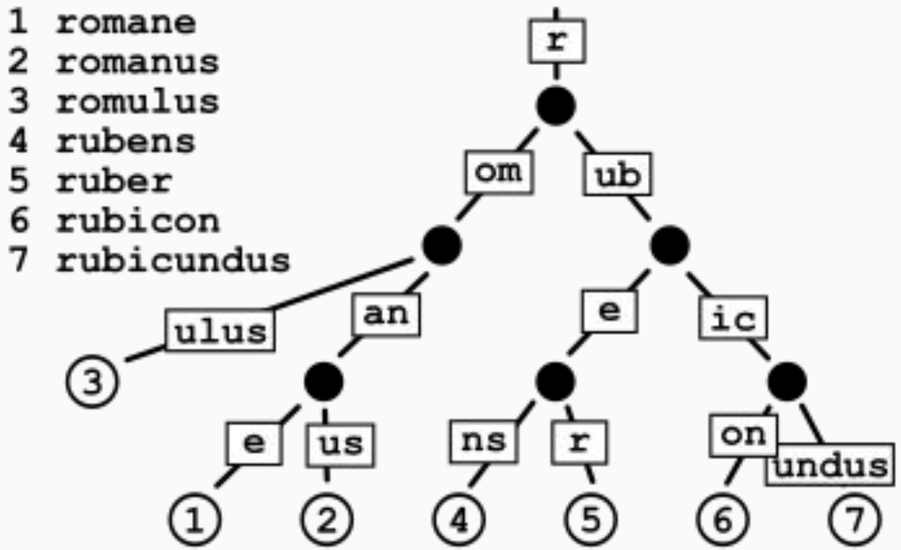
En vez de almacenar un solo carácter en un nodo, se almacena la mayor cantidad de caracteres posibles



Patricia Tries

Entonces creemos el siguiente diccionario:

TEST
TOASTER
TOASTED
TASTING
SLOW
SLOWLY
TESTIMONY
TESTAMENT
SLOWEST



UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

Welcome to Algorithms and Data Structures! - CS2100