PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2233 - PROGRAMACIÓN AVANZADA

Ayudantía 06

2^o semestre 2018
Septiembre 2018

Estructuras de Datos: Árboles y listas ligadas

Basada en la AC 04 2017-2

Introducción

En esta ayudantía deberán implementar un **ContacTrie**, el cual es tipo especial de Trie. El Trie es un árbol utilizado para almacenar palabras de manera eficiente. Un ejemplo de un Trie es el que se visualiza continuación[1], el cual posee las palabras "dia", "dime", "mama" y "mia".

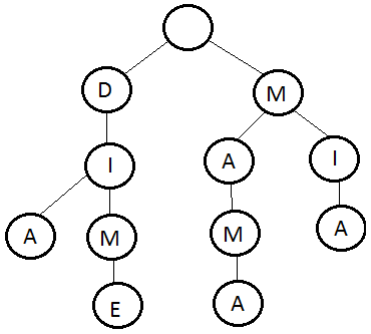

Figura 1: Trie con palabras "dia", "dime", "mama" y "mia"

Un Trie comienza con sólo la raíz, la cual no posee una letra propia. Cuando se ingresa una nueva palabra, se ve si algún nodo hijo del actual (en este caso la raíz) posee la primera letra de la palabra. Si ese nodo no existe, se crea. Luego, se pasa a ese nodo y se realiza lo mismo con el resto de la palabra.

Por ejemplo, a continuación veremos cómo se ingresa la palabra "dias" a un Trie vacío.

Figura 2: El Trie comienza con sólo la raíz

[1]Fuente: https://www.tecnohobby.net

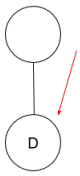

Figura 3: Se agrega un nodo hijo con la primera letra de “dias”

Como pueden ver se creó el nodo con la letra “D”. Ahora se debe continuar ingresando el resto de la palabra (quedan las letras “ias”).

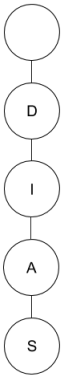

Figura 4: Se puede apreciar el Trie con la palabra “dias”

Si ahora se quisiera ingresar la palabra “mias”, se tendría que crear toda una nueva rama, ya que si bien comparten la mayoría de las letras, las primeras letras de las palabras son distintas.

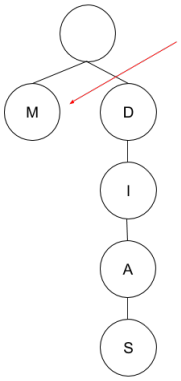

Figura 5: Como no hay un nodo con la letra “M” en la raíz, se crea uno.

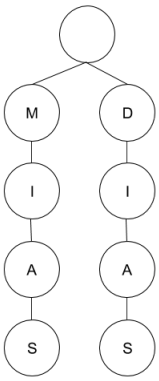

Figura 6: Trie con las palabras “mias” y “dias”

Por otro lado, si se quisiera agregar la palabra “diana”, sí se puede aprovechar de nodos ya existentes. Esto porque la raíz ya posee un nodo hijo con la letra “D”.

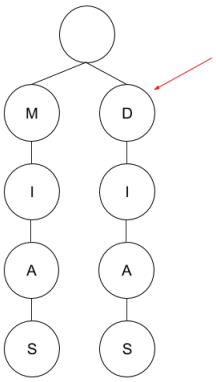

Figura 7: La raíz de este Trie ya posee un nodo con la letra “D”

De la misma forma puede avanzar hasta llegar al nodo con la letra “A”, el cual no posee un nodo hijo con la letra “N”. Por lo tanto, debe crear una nueva rama para poder ingresar el resto de la palabra.

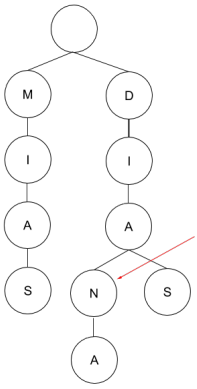

Figura 8: Trie con las palabras “mias”, “diana” y “dias”

Como pueden notar en cualquier nodo hijo (u hoja) se almacena una letra. Sin embargo, se puede almacenar más información en cada nodo. Para esta ayudantía deberán implementar un `ContacTrie` que almacene contactos, es decir, palabras asociadas a números. A continuación se explicarán los métodos que debe tener su ContacTrie.

Para este `ContacTrie` sólo los nodos con la letra final de cada palabra almacenen el número. Por ejemplo, supongamos que el siguiente Trie tiene los contactos "Hernan" y "Flor" con los números 9876 y 1234 respectivamente.

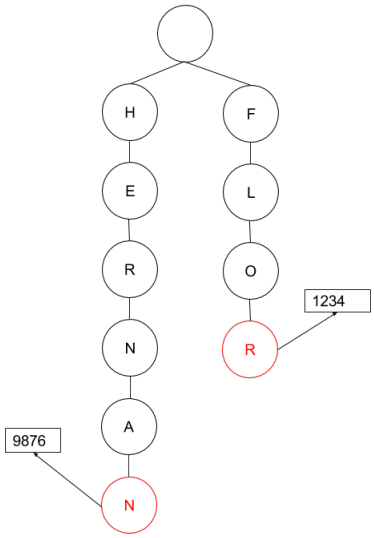

Figura 9: Las hojas de color rojo almacenan un número

De esta manera los nodos intermedios quedan "libres" para almacenar un número. Así, se puede agregar un contacto con nombre "flo" con el número 43567 sin tener que alterar la estructura del ContacTrie.

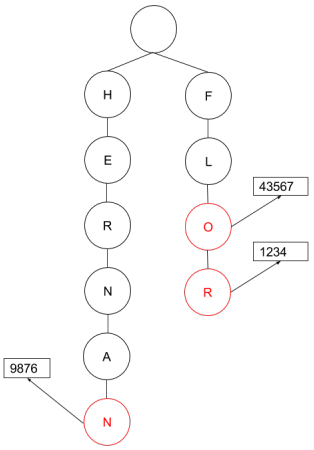

Figura 10: Se le agrega un número al nodo con la letra "O"

Ustedes deben crear la clase `ContactTrie` que tenga los siguientes métodos:

- `add_contact(name, number)`: Este método permite agregar contactos entregándole un string con el nombre y un entero con el número del contacto que se desea añadir. Primero se debe convertir el string a mayúsculas. Si el nombre del contacto ya existe en el ContacTrie, éste se debe sobreescribir, y por ende se modifica el número que ya existía por el nuevo número; de lo contrario, solo de debe añadir el nuevo número.

- `get_all_contacts( )`: Este método no recibe argumentos y debe retornar una lista con todos los contactos del `ContactTrie`. Cada contacto en la lista debe estar contenido en una namedtuple `Contacto` con los atributos `nombre` y `numero`. El orden en que se muestran los contactos es irrelevante.

- `ask_for_contact(name)`: Este método debe consultar por un número de un contacto. Para esto se le debe entregar un string con el nombre del contacto pedido. Este debe ser transformado a mayúsculas para poder iniciar la búsqueda. Si el contacto no existe se debe imprimir "Contacto Inexistente". Si el contacto existe se debe retornar el `Contacto` con el nombre y el número.

  Importante: Se debe tener en consideración que los nombres entregados a las funciones `add_contact` y `ask_for_contact` sean un *string*. De la misma manera, el número entregado a la función `add_contact` debe ser un *integer* . Si no lo son, se debe retornar `None` e imprimir `‘‘Error de tipos: Los tipos de los argumentos son incorrecto’’`. Si lo son, entonces la función hace lo deseado. Además, se debe verificar que el nombre entregado no sea vacío. Si es vacío se debe imprimir `‘‘Error de valor: Nombre no puede ser vacío’’`. **Hint:** recuerde la existencia de decoradores.

La idea es que construyan el árbol siguiendo estas reglas :).