

Búsqueda digital, tries, etc.



Motivación

- Tiempo menos dependiente de la cantidad de claves
 - Rendimiento razonable en el peor caso
 - Rápidos en la práctica
 - Adecuados para claves de tamaño variable
 - Adecuados para otro tipo de aplicaciones (p. ej. pattern matching, indexación, en general “text retrieval”, compresión de textos)
-

Idea:

- No hacer comparaciones de claves completas, sino de **partes** de ellas
 - Requiere: poder hacer operaciones sobre esas partes.
 - Por ejemplo:
 - si las claves son strings, vamos a trabajar con caracteres
 - si las claves son enteros, vamos a trabajar con dígitos, o bits.
-

Desventajas

- Algunas implementaciones pueden requerir mucha memoria
 - Las operaciones sobre las componentes de las claves puede no ser fácil, o ser muy ineficiente en algunos lenguajes de alto nivel.
-

Árboles de búsqueda digital

- “Parecidos” a los ABB:
 - Para guardar una clave, vamos recorriendo el camino en el árbol que nos lleva a su posición
 - Pero la posición no está determinada por comparaciones por $>$ o $<$, sino por los bits (o dígitos, o caracteres) de la clave

■ Ejemplo:

E	00101
J	01010
M	01101
P	10000
L	01100
O	01111
D	00100
B	00010
U	10101
S	10011
Q	10001
A	00001

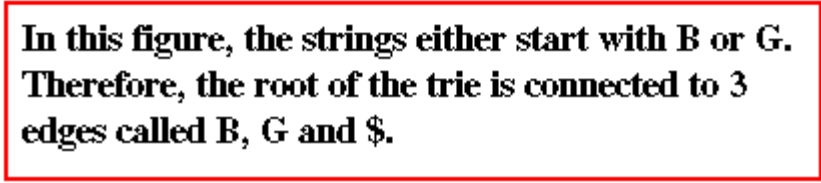
Propiedades

- El peor caso es mucho mejor que el de los ABBs si
 - El número de claves es grande y
 - Las claves no son largas
- Longitud del camino más largo = mayor número de bits sucesivos iguales de dos claves cualesquiera a partir del bit más a la izquierda (o sea, mayor prefijo común)
- Búsqueda o inserción en un árbol de n claves de b bits, necesita en promedio (suponiendo bla, bla, bla...) $\log n$ comparaciones de clave completa, y b en el peor caso.

Tries

- Debidos a Friedkin, años `60
 - Nombre proviene de “retrieval”
 - Árbol $(k+1)$ -ario para alfabetos de k elementos.
 - Los ejes del árbol toman interés: representan componentes de las claves
 - Por ejemplo:
 - si las claves son strings, los ejes representan caracteres
 - si las claves son enteros, los ejes representan dígitos, o bits.
 - Cada subárbol representa al conjunto de claves que comienza con las etiquetas de los ejes que llevan hasta él
 - Los nodos internos no contienen claves.
 - Las claves o los punteros a la info, se almacenan en las hojas (a veces ni eso es necesario).
-

Ejemplo



LEAVES ARE GREEN.
THE SYMBOL "\$" TERMINATES EACH
WORD.

Otro ejemplo

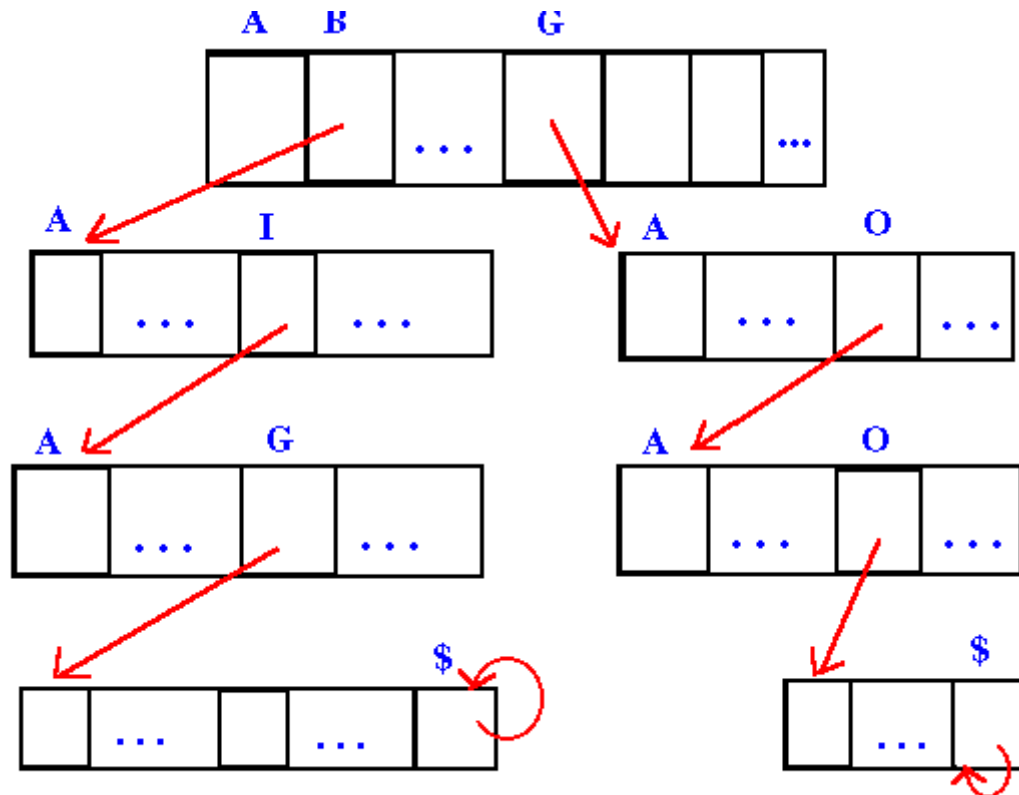
E	00101
J	01010
M	01101
P	10000
L	01100
O	01111
D	00100
B	00010
U	10101
S	10011
Q	10001
A	00001

Propiedades

- A lo sumo una comparación de clave completa (cuando llego a una hoja)
 - La estructura del trie es la misma independientemente del orden en el que se insertan las claves (o sea...hay un único trie para un conjunto de claves)
 - Búsqueda/inserción en un trie construido a partir de n claves de b bits, necesita $\sim \log n$ comparaciones **de bits** en promedio (suponiendo bla bla bla....) y b comparaciones en el peor caso.
-

Implementación de tries

- Primera posibilidad: en cada nodo interno, un arreglo de punteros

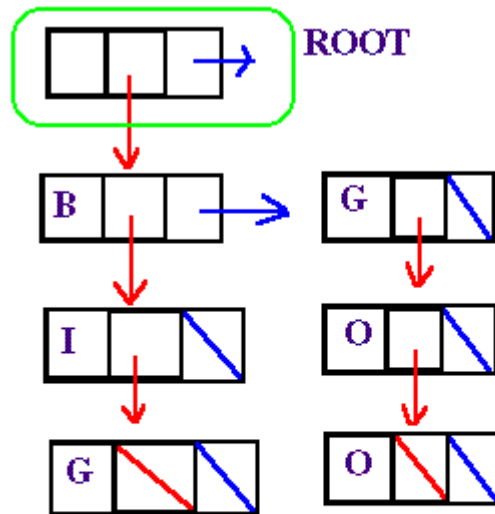


- Muy eficiente en términos de tiempo
- Algoritmo simple
- ¡Puede ser extremadamente ineficiente en términos de memoria!
- Especialmente cuando:
 - Alfabeto grande
 - Diccionario chico
- Mitigaciones posibles:
 - Por ejemplo, agrupar caracteres poco frecuentes
 - (ver ejemplo con bits)

Trie representation for words BIG and GOO using array of pointers

Implementación de tries

- Segunda posibilidad: hijos de un nodo representados a través de una lista encadenada.



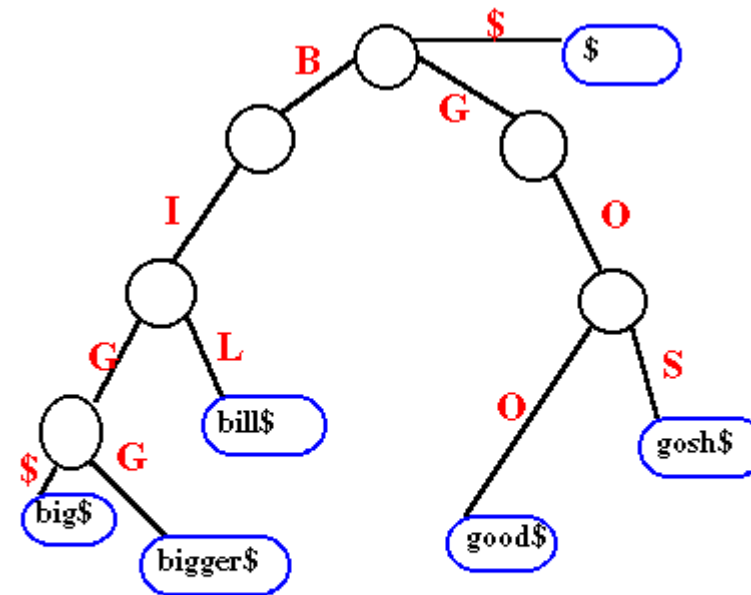
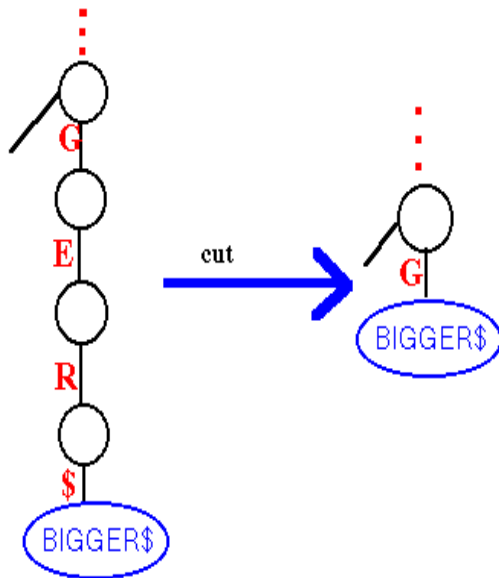
- Eficiente en términos de tiempo sólo si hay pocas claves
- Requiere algoritmos sobre listas
- Mucho más eficiente en términos de memoria

Trie representation for words BIG and GOO using linked lists

→ pointer to child → pointer to sibling

Tries Compactos

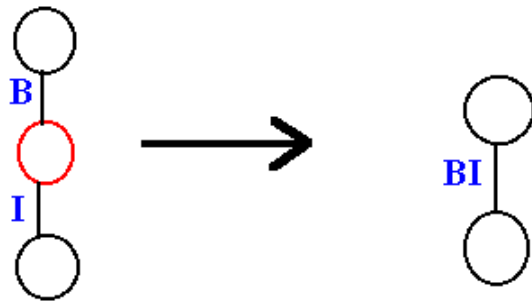
- Parecidos, pero.....colapsamos las cadenas que llevan hacia hojas



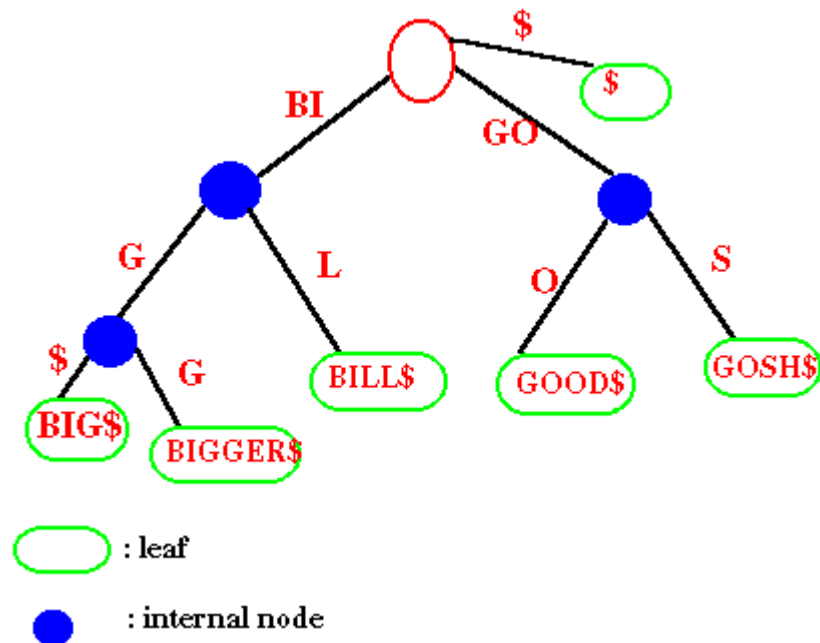
Trie for strings big, bigger, bill, good, gosh
This trie is more compact than the trie
in figure 2.

Tries más compactos: Patricia

- Patricia = Practical Algorithm To Retrieve Information Coded In Alphanumeric
- Debidos a D.R. Morrison
- Ahora colapsamos todas las cadenas
- Un eje puede representar más de un caracter.



Before, one edge is used to represent a character.
Now, the red node (unary node) is collapsed and
one edge can hold more than one character.



Para terminar

- La altura de un árbol Patricia binario está acotada por n (el número de claves).
 - Eso no sucede con los modelos anteriores.
 - En realidad, Patricia es un poco más elaborado, y hay numerosas variantes de los métodos vistos hoy.
-