Tries

Fernando Schapachnik[1,2]

[1]En realidad... push('Fernando Schapachnik', push('Esteban Feuerstein', autores))
[2]Departamento de Computación, FCEyN, Universidad de Buenos Aires, Buenos Aires, Argentina

Algoritmos y Estructuras de Datos II, segundo cuatrimestre de 2018

(2) Cómo harían el coso de las búsquedas?

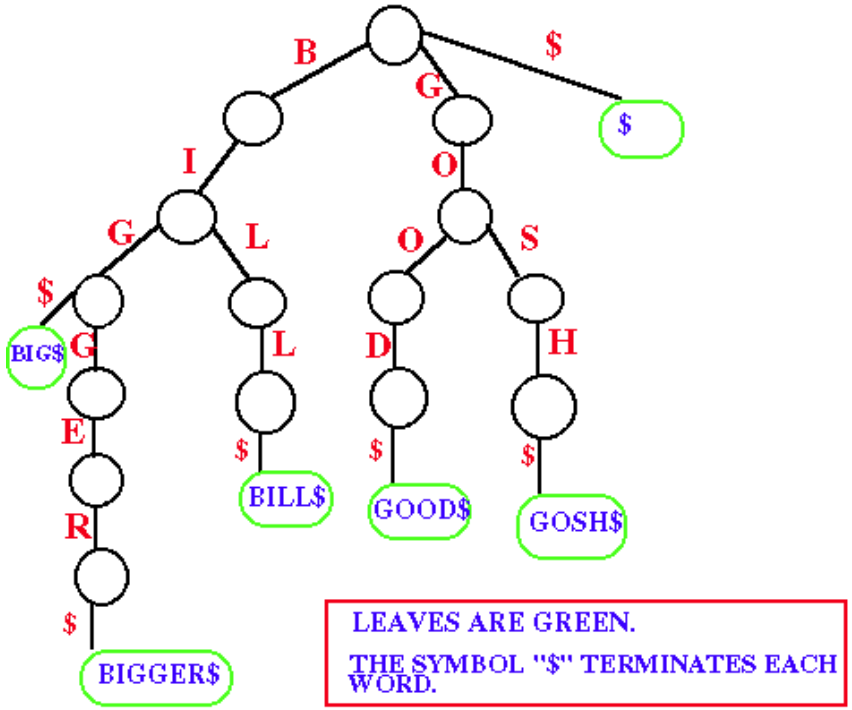
- El coso ése en el que uno tipeando y aparecen las palabras que coinciden...
- Con un ABB? Con un AVL? Con una tabla de hash?
- ¿Qué complejidad tienen esas búsquedas?
- Hoy vamos a ver una estructura que sirve para esos escenarios, además de permitir otras varias aplicaiones.

(3) Suposiciones

- Esta estructura de datos requiere concentrarse en las claves.
- Vamos a suponer que las claves x pertenecen a un alfabeto Σ^*.
- Tenemos que pensar que no tienen todas $O(1)$, sino que su tienen una longitud $|x|$.
- Esa longitud puede ser:
  - La longitud propiamente dicha, en caso de strings.
  - O $\log(x)$ si x es un entero.
- Además, definimos a $k = |\Sigma|$, como la cantidad de símbolos distintos del alfabeto Σ.
- Pensemos en árboles k-arios.

(4) Tries

Representemos {BIG, BIGGER, BILL, GOOD, GOSH}.

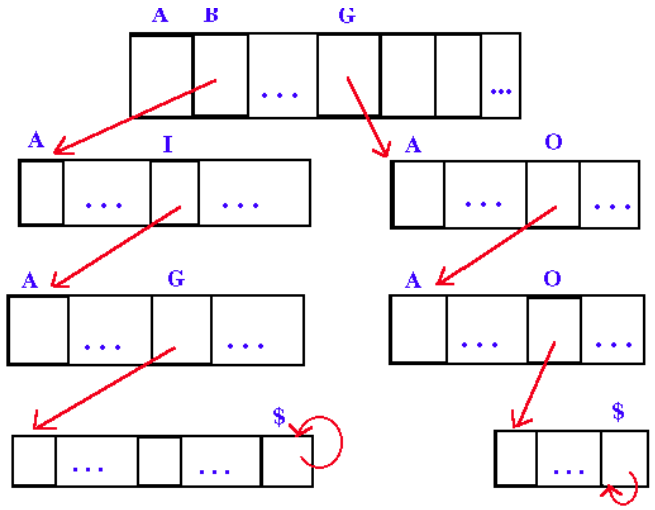


Debidos a Edward Fredkin, años ‘60. El nombre proviene de “retrieval”.

(5) Análisis

- Complejidad de insertar clave x? $O(|x|)$
- Y de buscar? $O(|x|)$.
- Notar: es independiente de la cantidad de elementos almacenados.
- Algoritmos: muy simples.
- Almacenamiento: punto flojo.

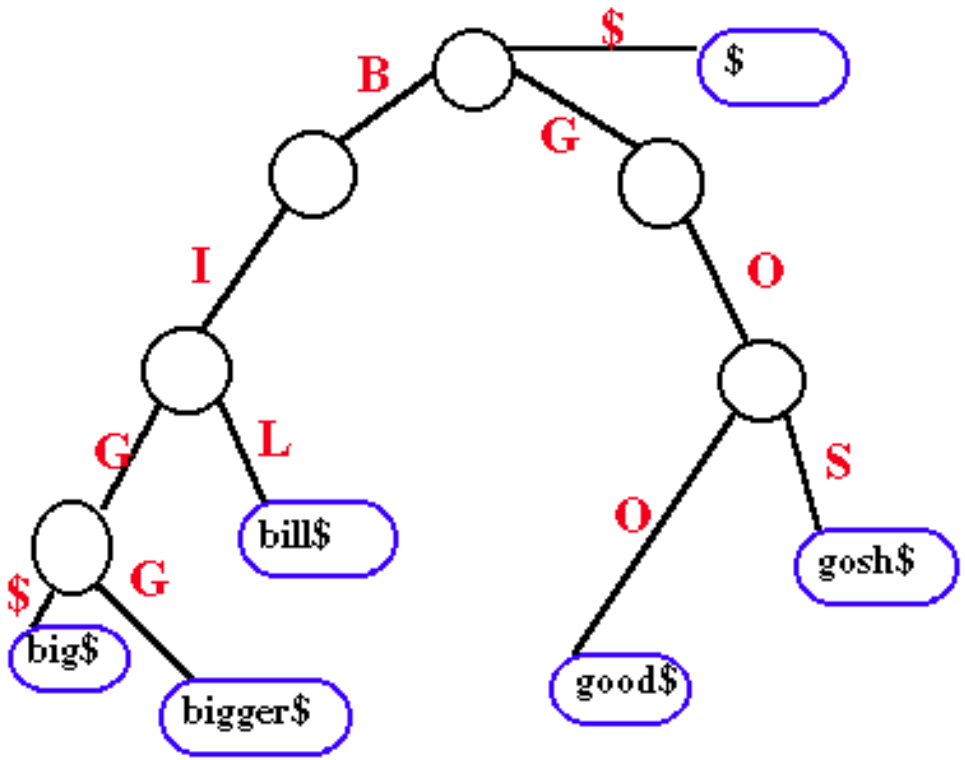
(6) Representación con arreglos y punteros



- Notar: puede ser muy ineficiente en términos de espacio.
- En especial cuando $|\Sigma|$ es grande y n es chico.
- ¿Podemos "comprimir" el arreglo? ¿Y qué pasa con la complejidad?
- En el peor caso se multiplica por k.

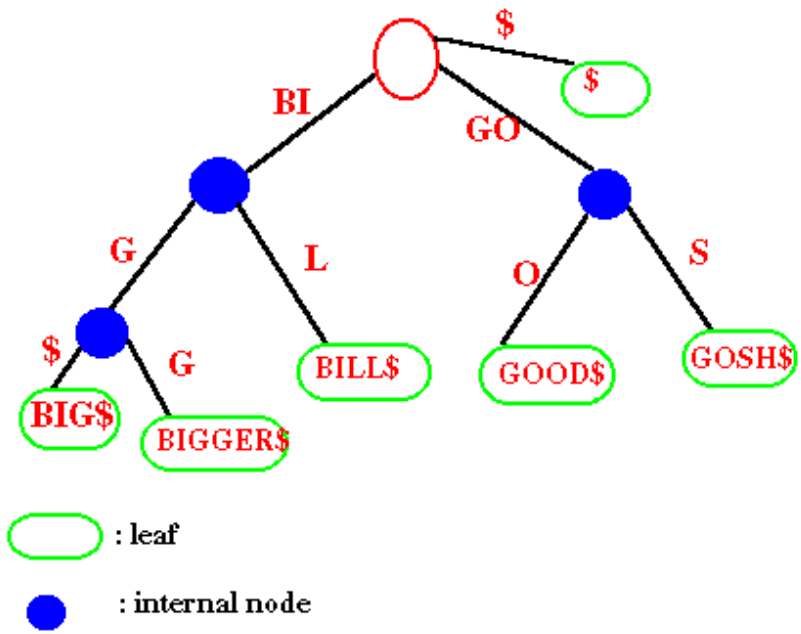
(7) Compactos

¿Qué pasa si compactamos las cadenas que sólo llevan a hojas?



(8) 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.



(9) El que sigue

- ¿Cuál es la complejidad de encontrar al siguiente?
- ¿Y el anterior?
- ¿Y en un ABB/AVL?
- Para resolver ese problema en 1975 Patrick Van Emde Boas inventó los árboles que llevan su nombre (también conocidos como vEB).

(10) vEB

- Operaciones:
  - Insertar
  - Borrar
  - Buscar
  - Próximo
  - Anterior
  - Mínimo/Máximo
- Mínimo y máximo en $O(1)$ (más adelante).
- El resto al mismo precio: $O(\log \log N)$ si las claves son $\{0 \dots N\}$.
- Notemos que si $N = 2^k$, entonces la complejidad es $O(\log k)$.
- Es decir que para palabras de k bits encontramos elemento, anterior y próximo en $O(\log k)$.
- Pero ocupan mucho espacio ($O(N)$) y son complejos de programar.

(11) vEB (cont.)

Intuición de vEB para el conjunto $\{2, 3, 4, 5, 7, 14, 15\}$. El resumen indica qué subárboles no están vacíos y cada subárbol tiene una parte de la información.