Buscar en Dominios Discretos/Finitos

Note Title 9/10/2009

(en un arreglo ordenado)

- ~~Busqueda Binaria~~
- ~~Quick Search~~
- ~~Doubling Search~~

} no utilizan el dominio discreto/finito

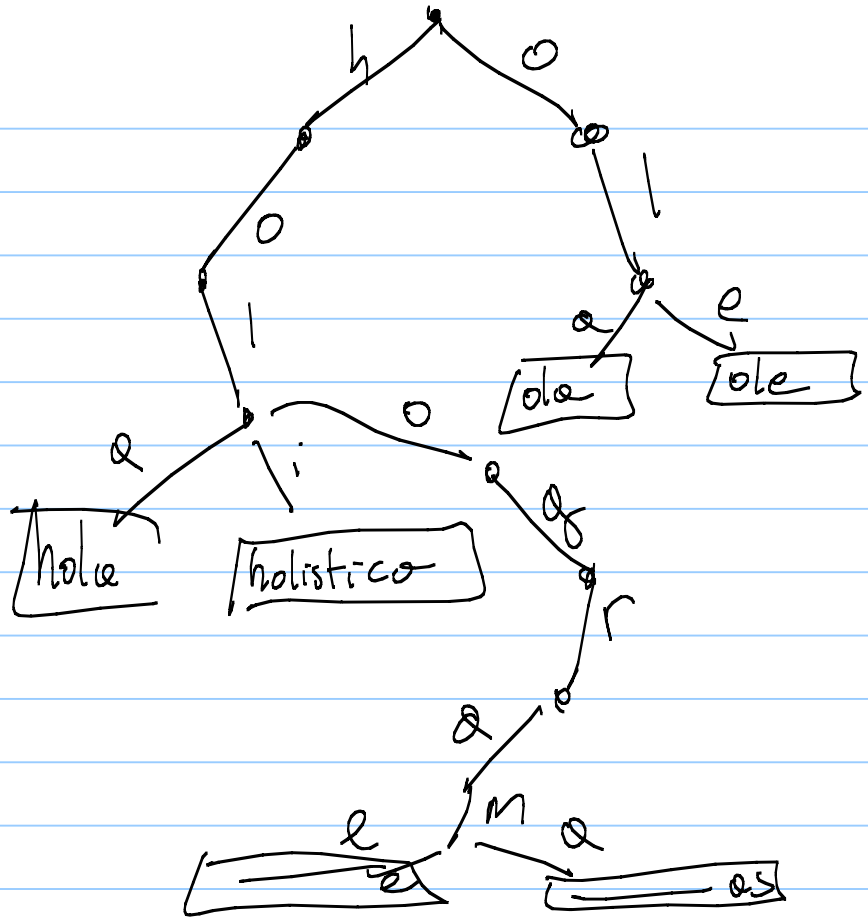
- Interpolation Search

(agenda telefonica)

① Tries

@Ejemplo

hola n_1
holistico n_2
holograma n_3
hologramas n_4
ola n_5
ole n_6

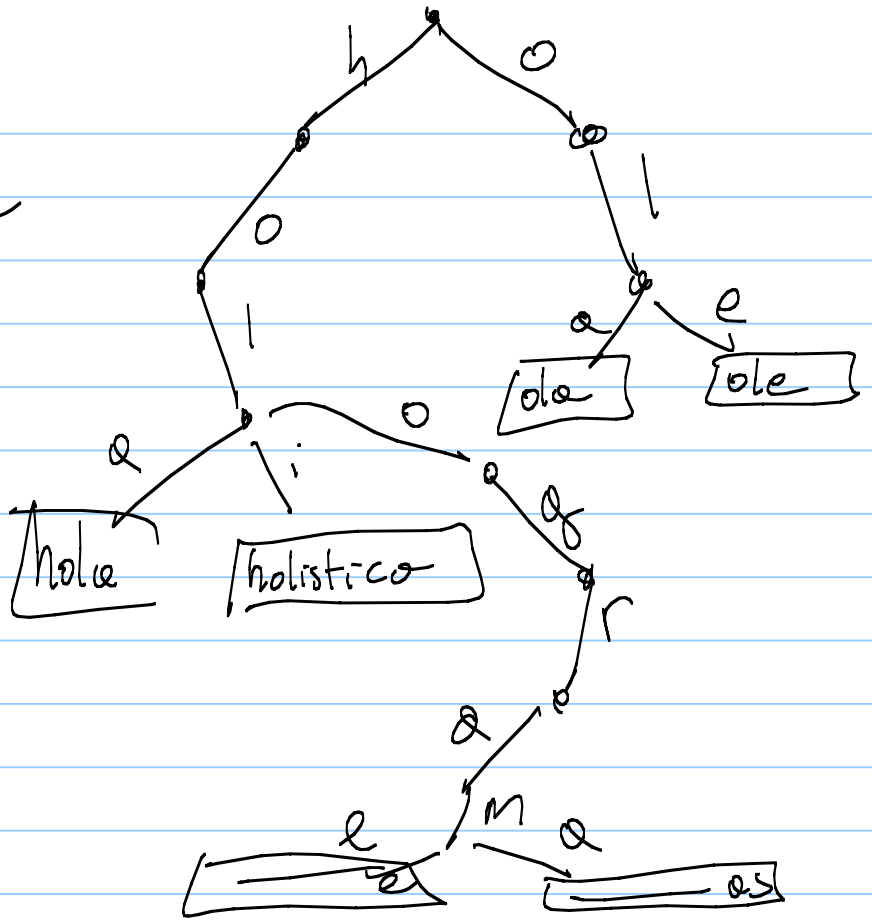


Notaciones

$n = \sum n_i$, suma de los tamaños de los strings

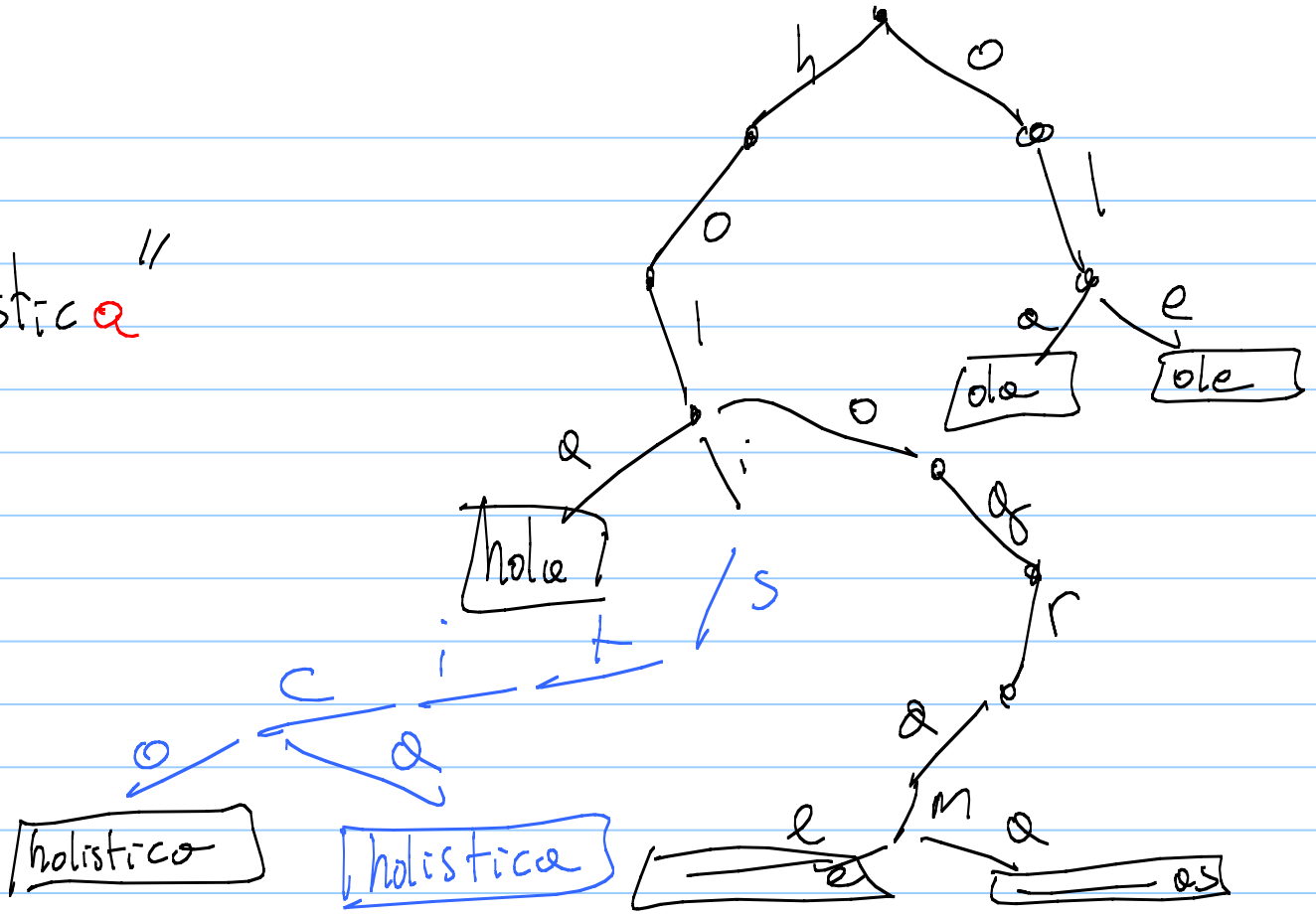
⑥ Implementaciones

- con arreglos de tamaño σ
  - $O(1)$ tiempo } cada nodo
  - $O(\sigma)$ espacio }
- con arreglo de tamaño variable
  - $O(\lg \sigma)$ tiempo cada nodo
  - $O(n)$ en total
- con hashing
  - $O(1)$ tiempo/nodo en promedio
  - $O(n)$ espacio en total



Ejemplo

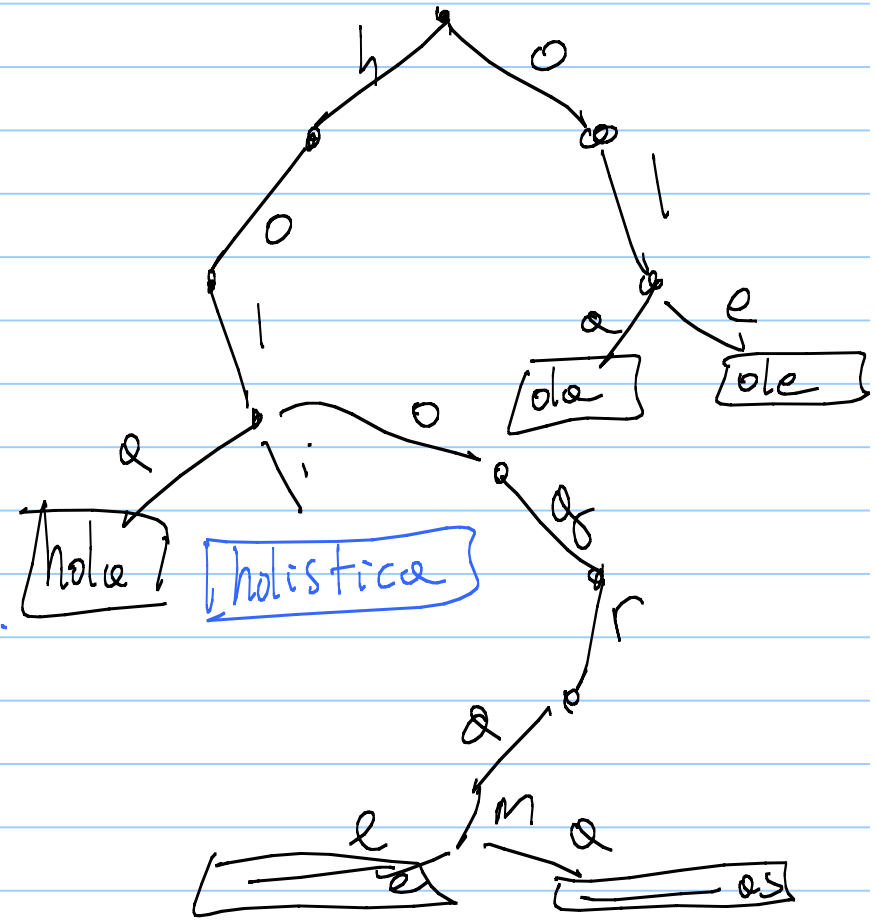
Insertar "holistica"



buscar holistico

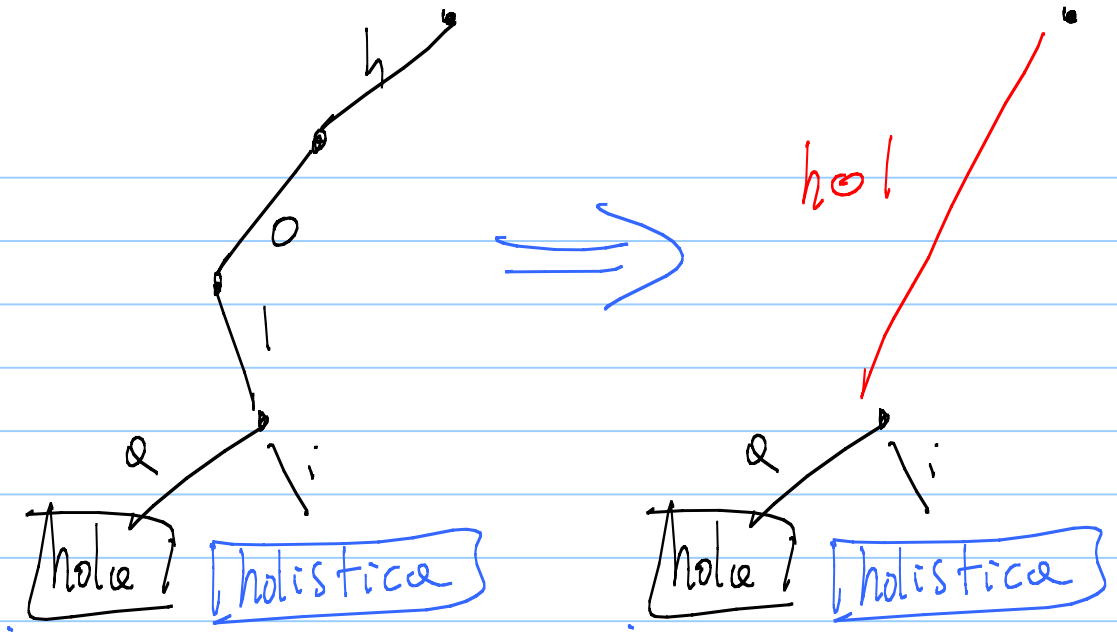
Complejidad?

no más que un factor constante de la complejidad de la busqueda



Optimizaciones

* PAT arboles



* Estructura de Datos distinta para cada nodo o nivel.

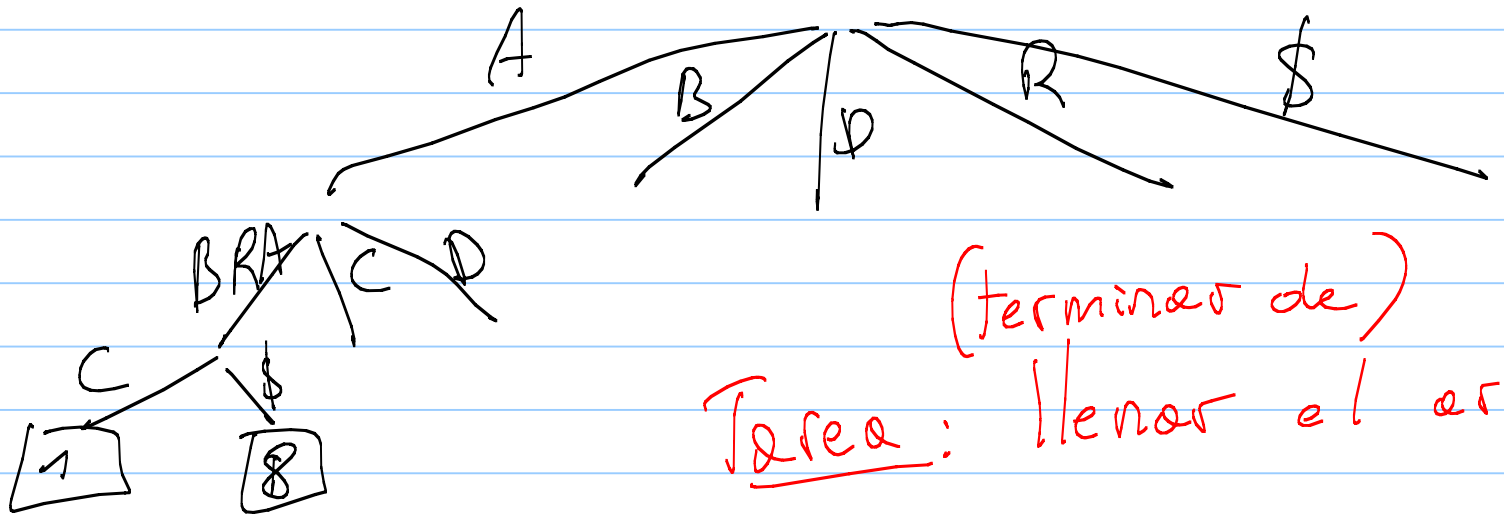
(2) Arbol de Sufijos

- Trie con todos los sufijos de un texto T
- Las hojas no tienen el sufijo, pero un puntador a su primera letra en T
- un subarbol codifica para una lista de matches de un patron

Ejemplo

1 2 3 4 5 6 7 8 9 10 11 12

T = Â B R A C A D A B R A $



(terminar de)

Tarea: llenar el arbol

Complejidad

* Construir el arbol $\Theta(n \lg n)$

* Espacio para el arbol $O(n)$ punteros
  $O(n \lg n)$ bits

* Tiempo de busqueda de un patron P de tamaño m
  (arreglo de tamaño variable) $O(m \lg \sigma)$

③ Arreglo de Sufijos

* Lista de sufijos, ordenados en orden lexicográfico.

* búsqueda de patrones
  = dos búsquedas binarias sobre n elementos
  cada comparación costa $\leq m$
  $\Rightarrow$ en total $O(m \lg n)$