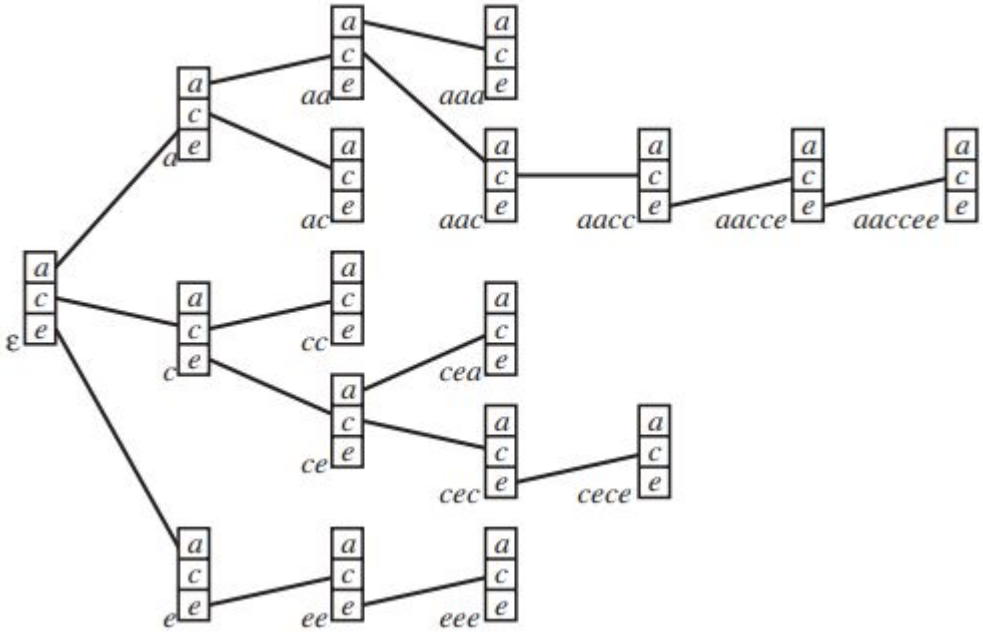
Data structure for text

String

- Hasta ahora asumimos que los datos de entrada son del mismo tamaño
  - Numeros…
- Los fragmentos de texto tiene un largo
  - no son objetos elementales que el procesador puede procesar en un solo paso
- Necesitamos estructuras diferentes
  - Balanced binary search trees
    - No eficiente para Strings
    - El ordenamiento por orden lexicográfico tiene poco sentido para Strings.
- Ejemplo de aplicación : DNA/RNA Alfabeto de 4 / 20

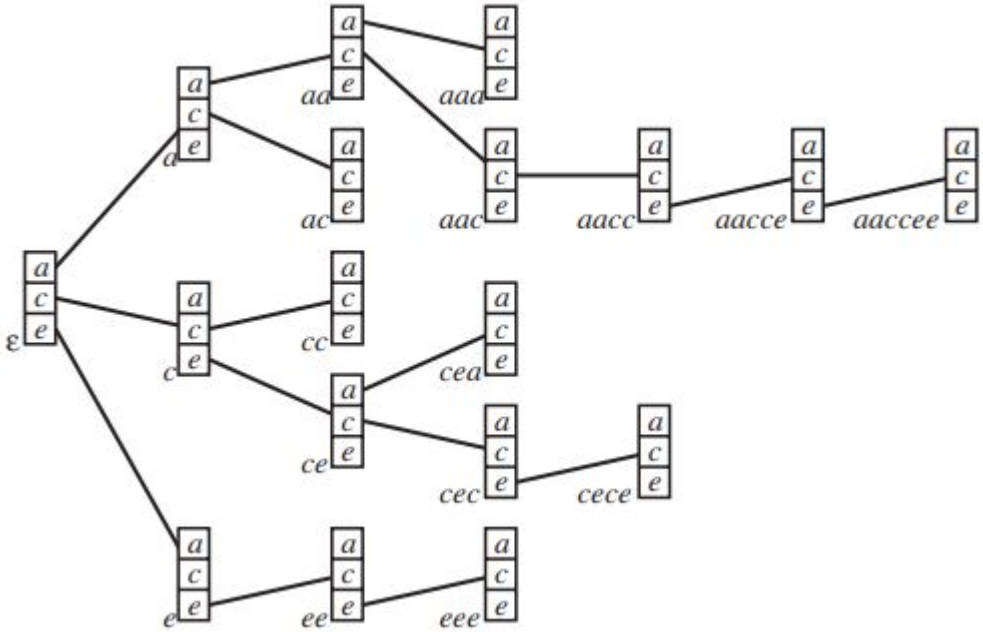
Trie

- Similar a un Balanced Binary Search Tree
- Cada nodo contiene potencialmente una arista saliente por cada caracter del alfabeto completo
- Cada nodo representa un prefijo, si un prefijo aparece varias veces entonces solo hay un no que lo representa
- La raíz es un prefijo vacío



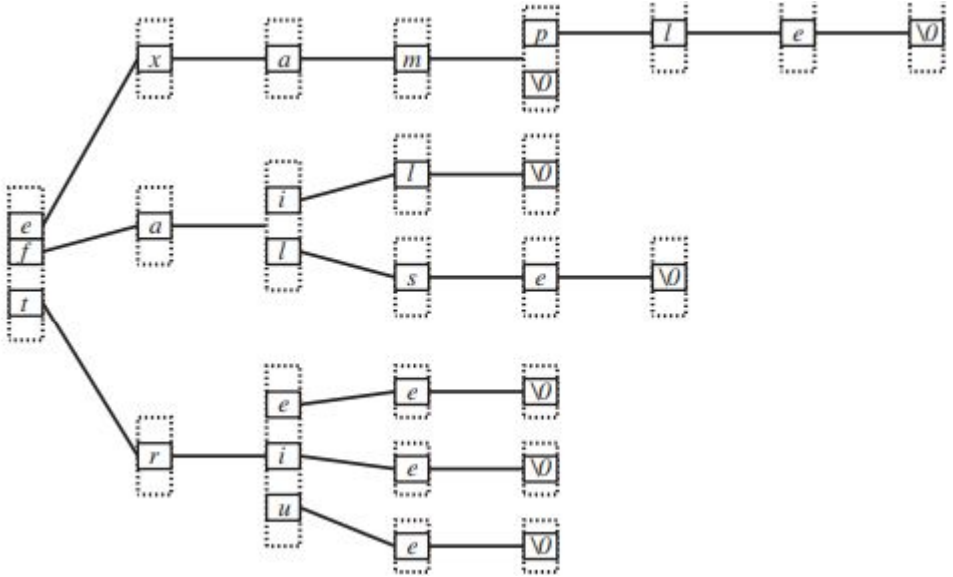
Trie

- Buscar
- Leemos el String de búsqueda siguiendo los caracteres
- Si llegamos al fin del texto sobre una hoja hemos encontrado el texto
- Usar el caracter ‘/0’



Trie

- Cada nodo tiene 256 puntero (char) entonces de 4-8 byte así que cada nodo pesa al menos 1kB
- Si los string que almacenamos tienen poco overlap, la estructura puede terminar siendo muy voluminosa



Trie

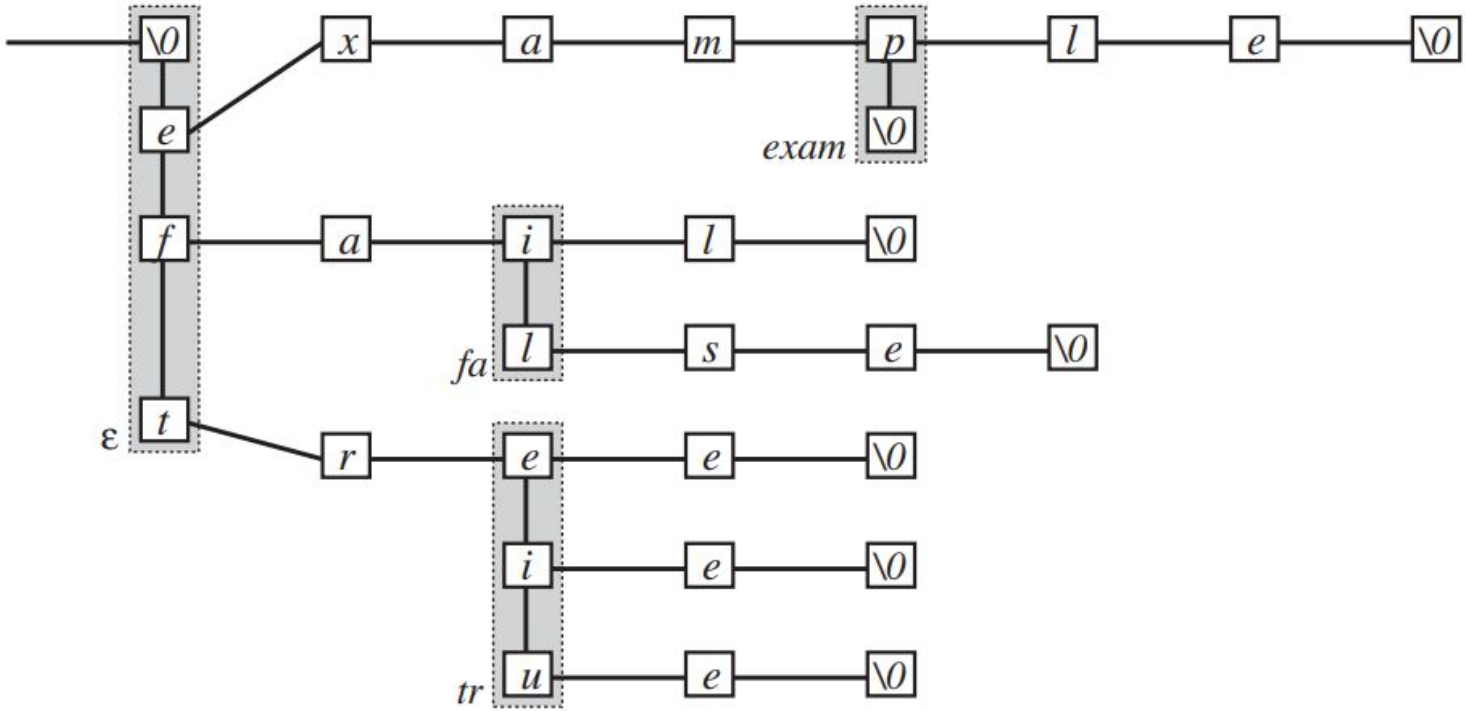
```
typedef struct trie_n_t {

    struct trie_n_t *next[256]; /* possibly additional
information*/

    } trie_node_t;
```

Trie

```
object_t *find(trie_node_t *trie, char *query_string) {
        trie_node_t *tmp_node;
        char *query_next;
        tmp_node = trie;
        query_next = query_string;
        while(*query_next != '\0') {
                if( tmp_node->next[(int)(*query_next)] == NULL)
                                return( NULL ); /* query string not found */
|               else {
                        tmp_node = tmp_node->next[(int) (*query_next)];
                        query_next += 1; /* move to next character of query */ }
                 }
        return((object_t *)
        tmp_node->next[(int)'\0']);
}
```

Trie

- Caracteristicas
  - t el texto
  - Find O(length(t))
  - Insert and Delete O(lenght(t)*length(alphabet))
  - Espacio para almacenar n Strings wo,w1,...,wn es length(alphabet)*Suma(length(wi).


\0
x
a
m
p
l
e
\0
e
exam
\0
f
a
i
l
\0
fa
l
s
e
\0
ε
t
r
e
e
\0
i
e
\0
tr
u
e
\0

```
typedef struct trie_n_t {
        char this_char;
        struct trie_n_t *next;
        struct trie_n_t *list; /
} trie_node_t;
```