Unidad 3

Fundamentos de Programación Competitiva

Logro de sesión

Al finalizar la sesión, el estudiante comprenderá los conceptos de Trie

Semana 12

Trie

Contenido:

- Conceptos de Trie
- Inserción
- Búsqueda

Trie

- Trie es una estructura de datos que permite almacenar y realizar la búsquedas de datos.
- Trie permite realizar operaciones más eficientes que una tabla hash.
- Trie se usas para búsqueda basada en prefijos.

Trie - Ventajas

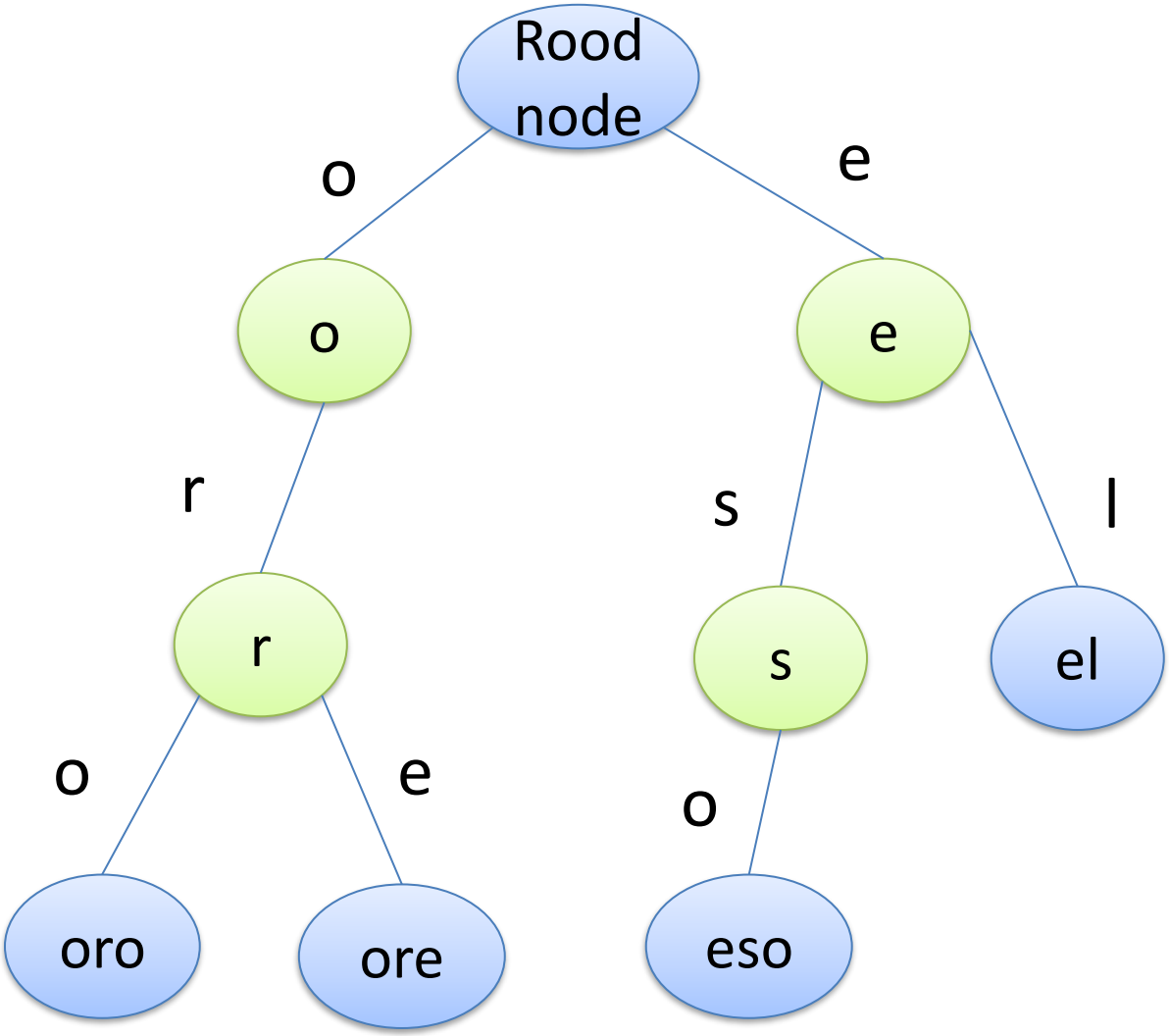
- Ventajas:
  - ✓ Realizar búsquedas de prefijos (o aucompletar).
  - ✓ Se imprimir con mayor facilidad las palabras en orden alfabético.
  - ✓ La búsqueda de una cadena en una colección de es mucho más eficiente, O(L), donde L es la longitud de la cadena.

Trie

- Estructura de datos Trie
  - ✓ Un nodo raíz en cada Trie.
  - ✓ Cada nodo de un Trie representa una cada y un borde representa un carácter.
  - ✓ Cada nodo consta de hashmaps o una matriz de punteros.
  - ✓ Cada índice representa un carácter y una bandera para indicar si alguna cadena termina en el nodo actual.
  - ✓ Trie puede contener los siguientes datos: alfabetos, números y caracteres especiales.

Trie

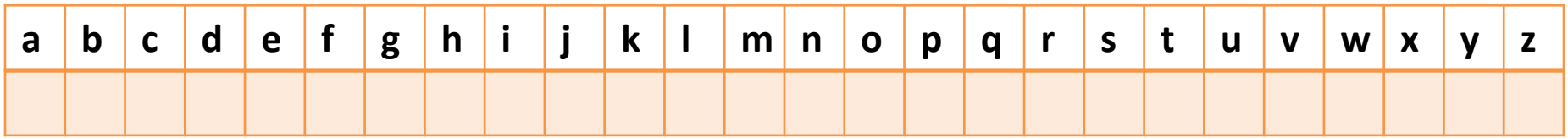
- oro
- ore
- eso
- el



Trie - Ejemplo

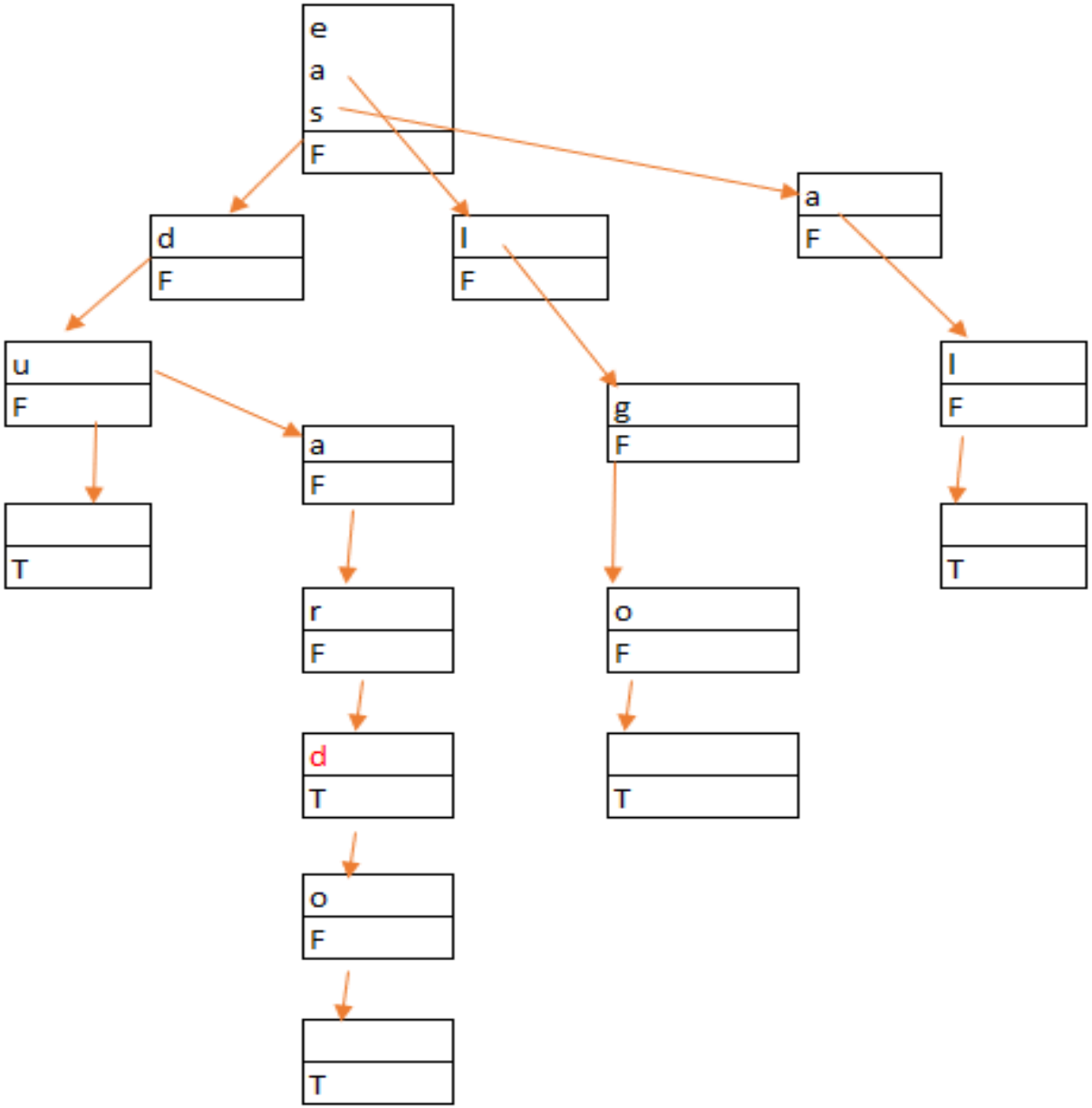
- Estructura de datos de Trie
  - Utilizaremos cadenas de caracateres desde la “a” hasta la “z”, se recibirán 26 punteros para cada nodo.
  - El índice 0 representa al carácter “a” y así sucesivamente.
  - Cualquier palabra en minúscula puede comenzar con cualquier letra del rango indicado.

Trie - Ejemplo



Una matriz de punteros dentro de cada nodo Trie

Trie - Ejemplo



Insertar

edu
eduar
algo
eduardo
sal

Trie - Creación

Creación:

```
11  struct TrieNode
12  {
13      struct TrieNode* children[26];
14      // isEndOfWord is true if the node represents
15      // end of a word
16      bool isEndOfWord;
17  };
```

Trie - Insertar

Insertar:

```
32  void insert(struct TrieNode* root, string key)
33  {
34      //Agrega datos en Key[i]
35      struct TrieNode* var = root;
36      for (int i = 0; i < key.length(); i++)
37      {
38          int index = key[i];
39          if (!var->children[index]) {
40              var->children[index] = getNode();
41          }
42          var = var->children[index];
43      }
44      var->isEndOfWord = true;
45  }
```

Trie - Buscar

Buscar:

```
50  bool search(struct TrieNode* root, string key)
51  {
52      struct TrieNode* var = root;
53      for (int i = 0; i < key.length(); i++)
54      {
55          int index = key[i];
56          if (!var->children[index])
57              return false;
58          var = var->children[index];
59      }
60      return (var->isEndOfWord);
61  }
```

Muchas Gracias!!!