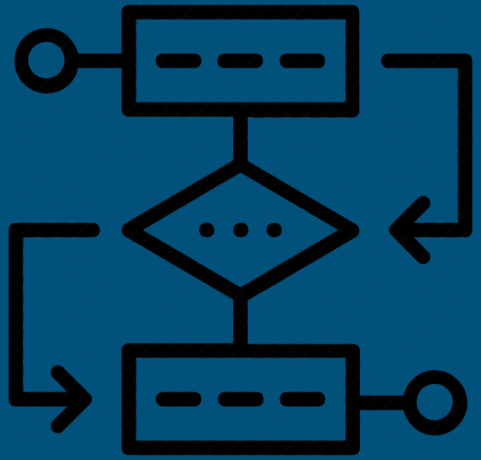
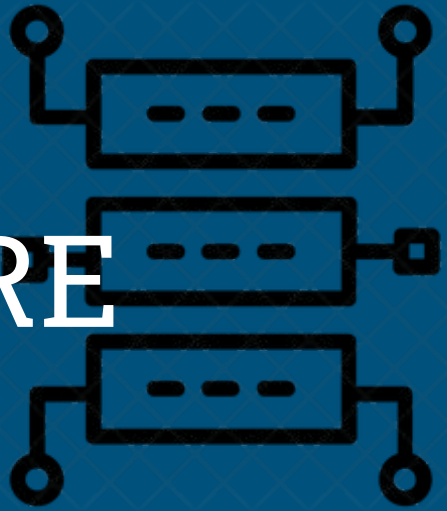
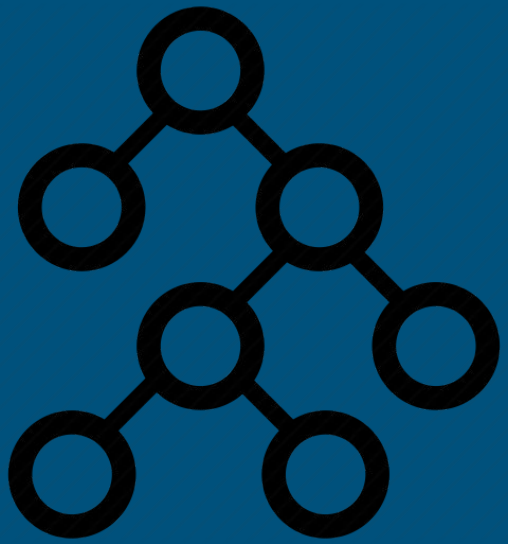
DATA STRUCTURE





TRIES



SEARCH STORE WORDS

- CAMALEÓN
- BICICLETA
- CORRECAMINO
- CILINDRO
- DIAMANTE
- RECUADRO

CAMALEÓN

BICILETA

RECUADRO

DIAMANTE

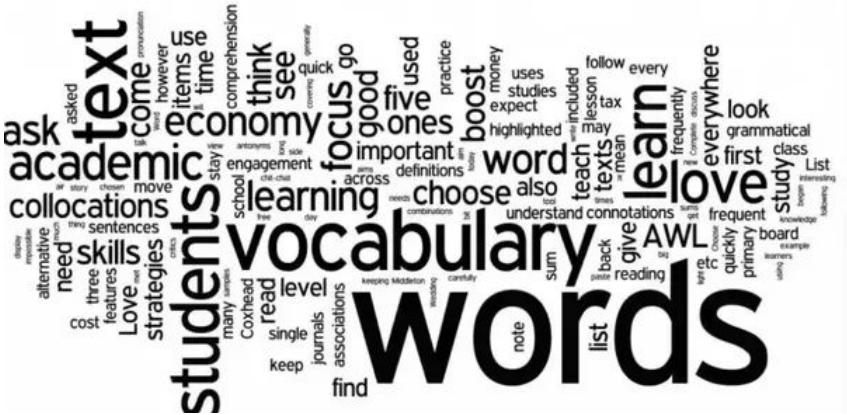
CORRECAMINO

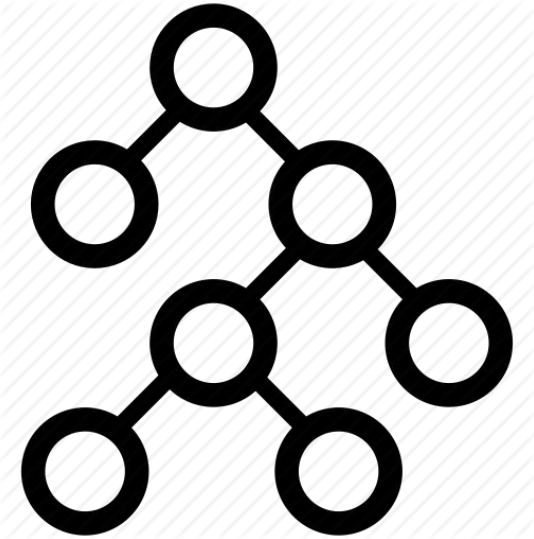
$O(nm)$

A TRIE is ...

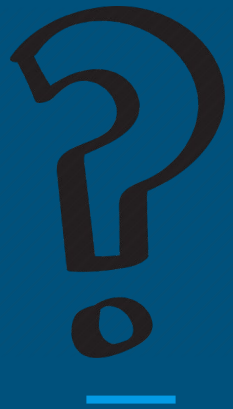
reTRIEval

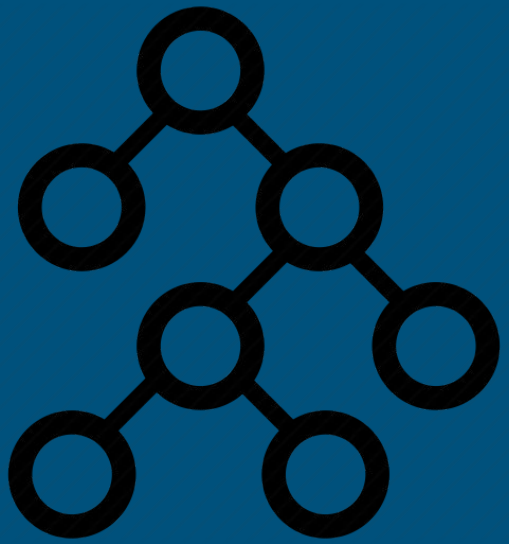
- Data Structure
- Tree
- Information Retrieval





HOW TRIE WORKS





Insertion Algorithm

```
INSERT ( CADENA S)
    Para cada letra de S
        if( nodoHijo == NULL )
            Creo nuevo Nodo
        NodoActual = NodoHijo
    NodoActual = IsPalabra
```

- DADO
- CARPA
- DONA
- CA
- CIRCO
- CASA
- CI

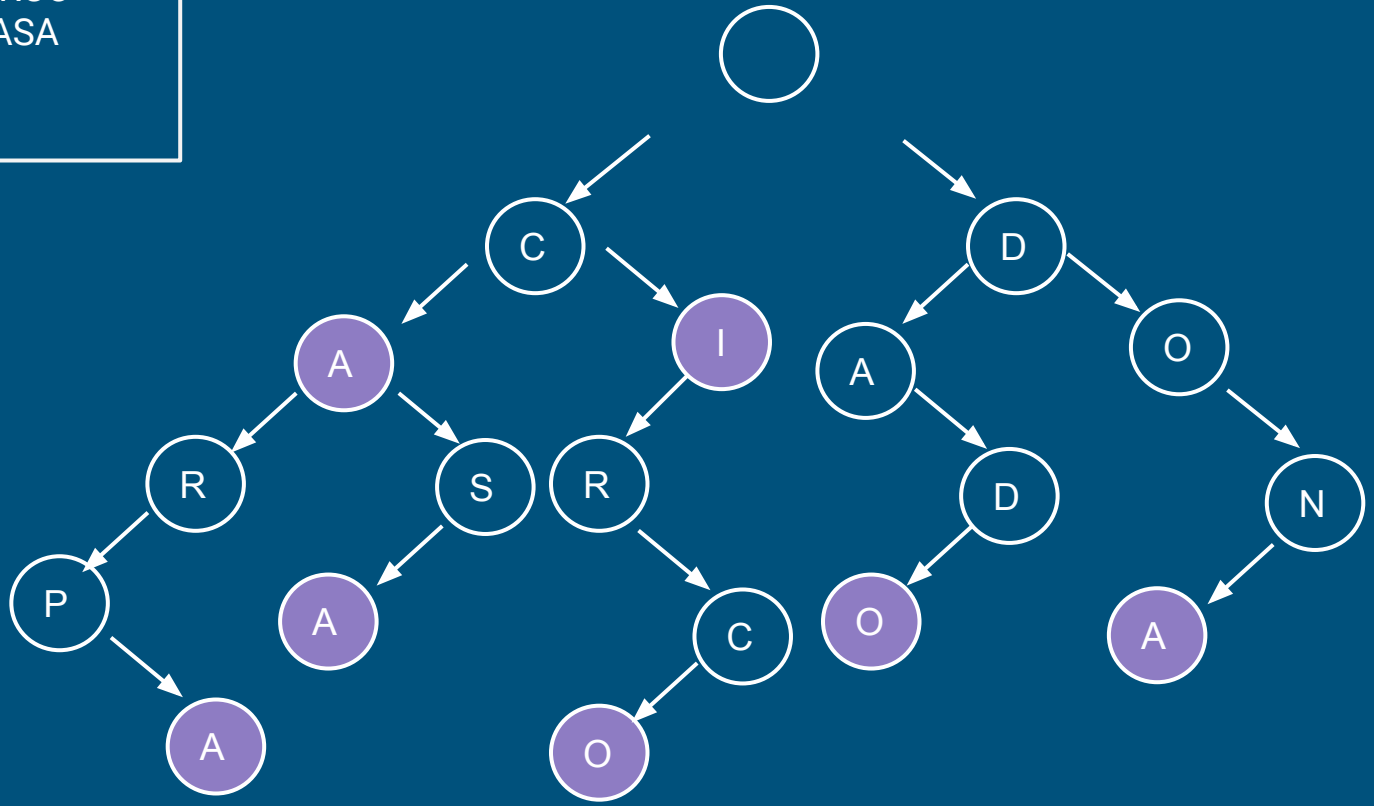
CARPA

CIRCO

DADO

DONA

CASA



Search Algorithm

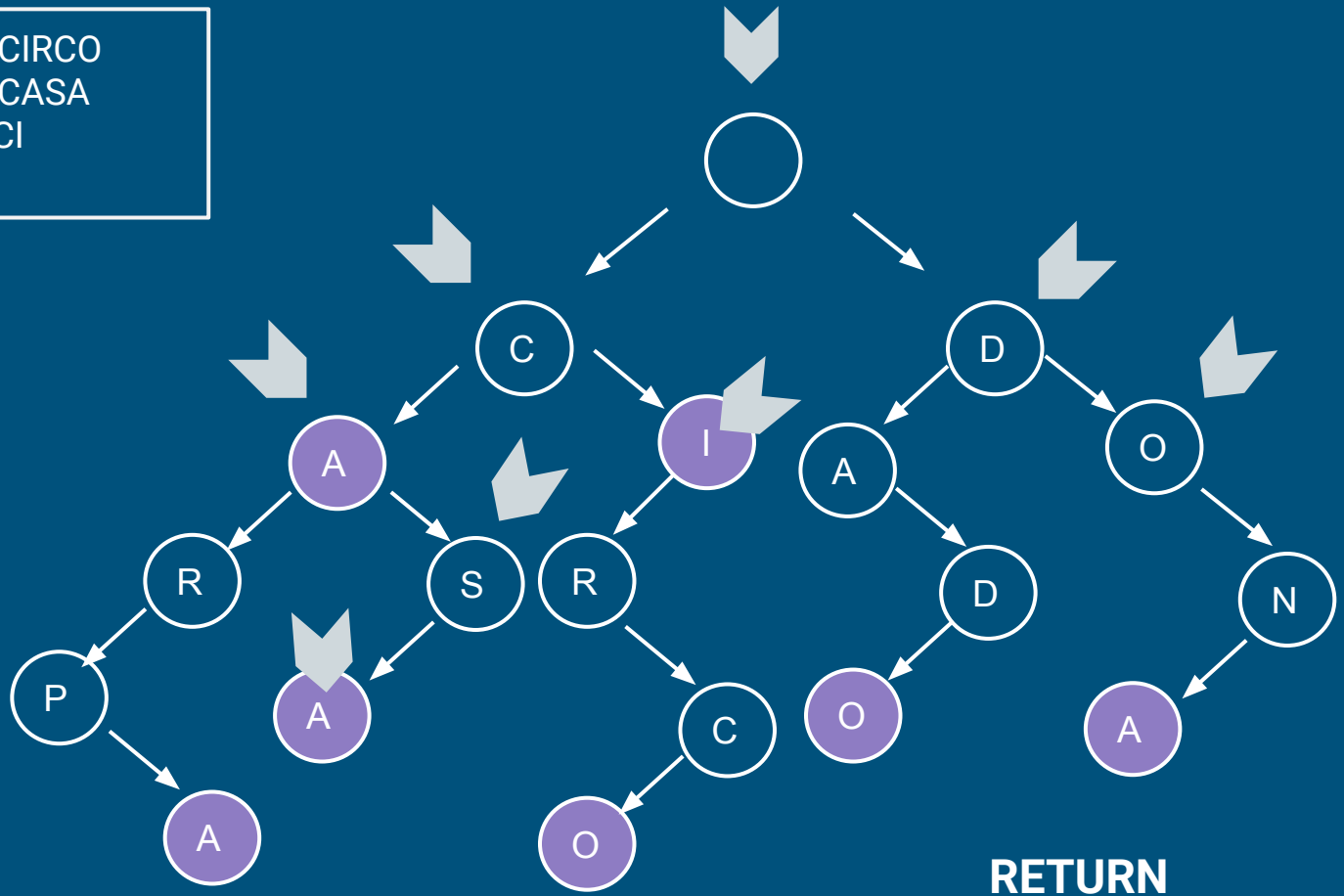
```
SEARCH ( CADENA S)
    Para cada letra de S
        if( nodoHijo == NULL )
            return False
        NodoActual = NodoHijo
    return NodoActual-Leaf
```

- DADO
- CARPA
- DONA
- CA
- CIRCO
- CASA
- CI

CASA

DOS

CI



RETURN FALSE

Delete Algorithm

DELETE (CADENA S)

Ir al final de s para ver si existe

Si existe verificar si tiene hijos

Delete Algorithm

Si el último nodo tiene
tiene hijos

actual->fin_palabra = 0

Delete Algorithm

Si el último nodo no tiene tiene hijos

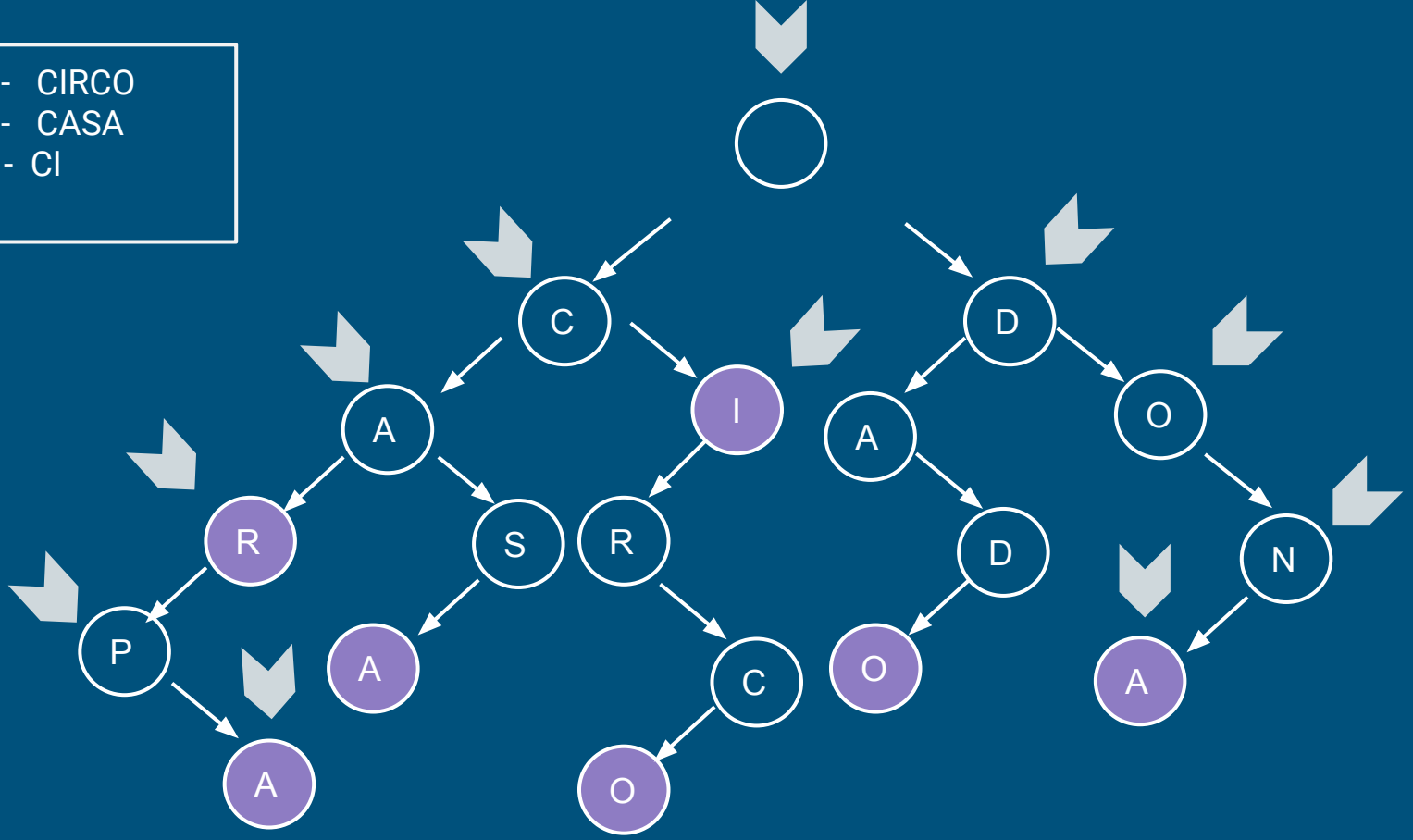
```
do{
    bool b1 = !actual->fin_palabra
    bool b2 = actual->n_hijos > 0
    if(!b1 || !b2){
        actual = actual->padre
        -- actual->padre->n_hijos
    }
}
while (b1 and b2 and actual != H);
```

- DADO
- CARPA
- DONA
- CAR
- CIRCO
- CASA
- CI

CI

DONA

CARPA

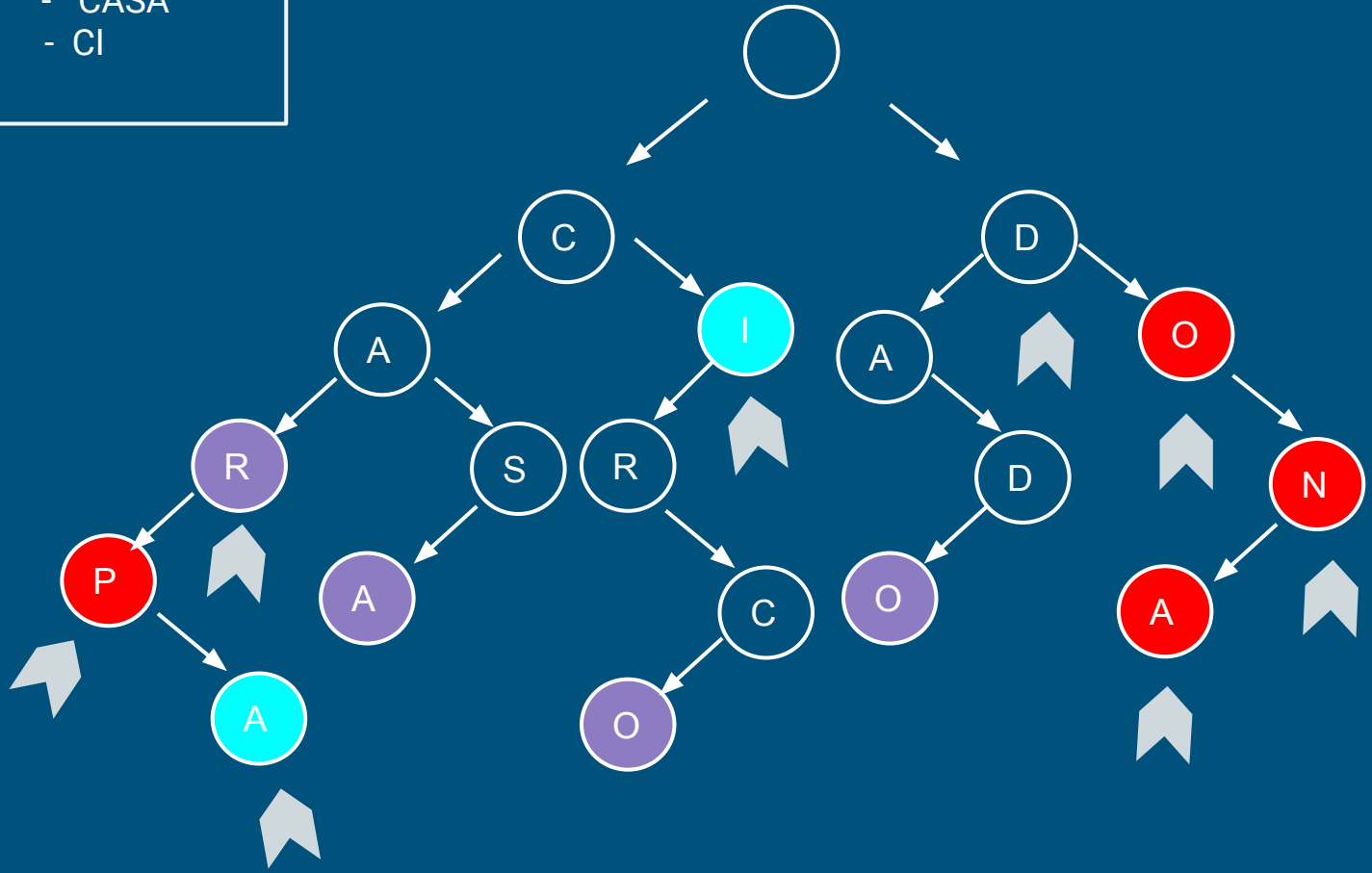


- DADO
- CARPA
- DONA
- CAR
- CIRCO
- CASA
- CI

CI

DONA

CARPA

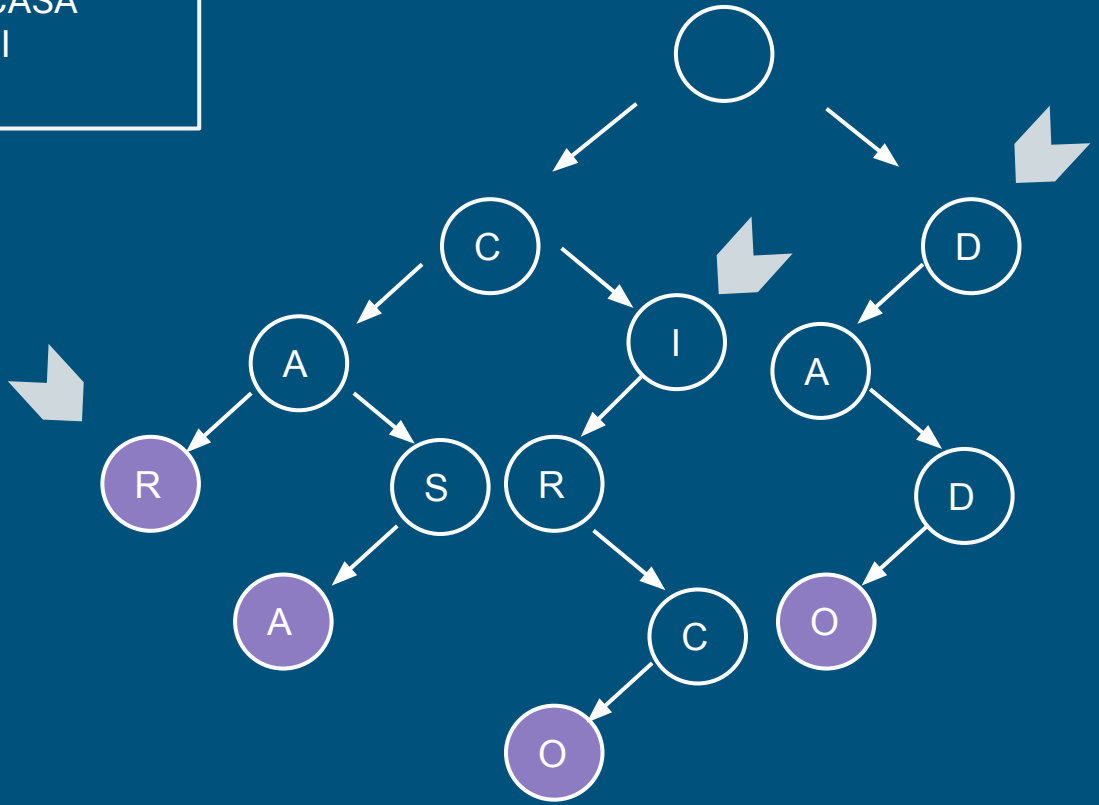


- DADO
- CARPA
- DONA
- CAR
- CIRCO
- CASA
- CI

CI

DONA

CARPA



```
#define CHAR_SIZE 128

class Trie
{
public:
  bool isLeaf;
  Trie* character[CHAR_SIZE];
  Trie(){
    isLeaf = false;
    for (int i = 0; i < CHAR_SIZE; i++)
      this->character[i] = NULL;
  }
  void insert(std::string);
  bool deletion(Trie*&, std::string);
  bool search(std::string);
  bool haveChildren(Trie const*);
};

void Trie::insert(std::string key){
  Trie* curr = this;
  for (int i = 0; i < key.length(); i++){
    if (curr->character[key[i]] == NULL)
      curr->character[key[i]] = new Trie();
    curr = curr->character[key[i]];
  }
  curr->isLeaf = true;
}
```

```
bool Trie::search(std::string key){
  Trie* curr = this;
  for (int i = 0; i < key.length(); i++){
    curr = curr->character[key[i]];
    if (curr == NULL)
      return false;
  }
  return curr->isLeaf;
}

bool Trie::haveChildren(Trie const* curr){
  for (int i = 0; i < CHAR_SIZE; i++)
    if (curr->character[i])
      return true;
  return false;
}
```

C++ Implementation

```
bool Trie::deletion(Trie*& curr, std::string key){
  if (curr == NULL) return false;
  if (key.length()){
    if (curr != NULL && curr->character[key[0]] != NULL &&
      deletion(curr->character[key[0]], key.substr(1)) &&
      curr->isLeaf == false){
      if (!haveChildren(curr)){
        delete curr;
        curr = NULL;
        return true;
      }
      else {return false;}
    }
  }

  if (key.length() == 0 && curr->isLeaf){
    if (!haveChildren(curr)){
      delete curr;
      curr = NULL;
      return true;
    }
    else{
      curr->isLeaf = false;
      return false;
    }
  }
  return false;
}
```

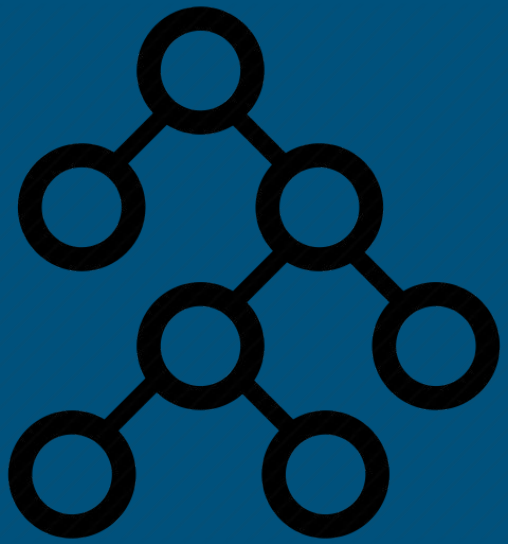
C++ Implementation

CHARACTERISTICS

- Store Words
- Replace Tablas Hash
- Trie don't have collisions
- Fast Retrieval

Tiempo
O(m)

Espacio
*O(Z*n*m)*



REFERENCES

- Trie Data Structure : https://www.geeksforgeeks.org/trie-insert-and-search/
- Trie Implementation : https://www.techiedelight.com/cpp-implementation-trie-data-structure/
- Trie Video : https://www.youtube.com/watch?v=AXjmTQ8LEoI
- Trie Visualization : https://www.cs.usfca.edu/~galles/visualization/Trie.html
- Trie Complexity : https://softwareengineering.stackexchange.com/questions/348444/what-is-the-space-complexity-for-inserting-a-list-of-words-into-a-trie-data-stru
- Algorithm AhoCorasick : https://en.wikipedia.org/wiki/Aho%E2%80%93Corasick_algorithm
- AhoCorasick : https://github.com/cjgdev/aho_corasick/blob/master/src/aho_corasick/aho_corasick.hpp
- AhoCorasick Video : https://www.youtube.com/watch?v=IcXimoT_YXA&list=TLPQMjYxMTIwMTkSJ3EP8PjRyA&index=3