**INFORME: POGRAMACION II**

Luraschi, Roberto Gino

**Trabajo Practico 2:**

**Agregar**

Método que agrega nodos con valores dentro del Trie:

**public** **void** agregar(String clave, V valor) {

**int** i = 0;

Nodo<V> nodo = raiz;

**char**[] aClave = clave.toCharArray();

**for** (i = 0; i < clave.length(); i++) {

**if** (nodo.hijo(alf.indice(aClave[i])) == **null**)

nodo.setHijo(alf.indice(aClave[i]), **new** Nodo<V>(alf.tam()));

nodo = nodo.hijo(alf.indice(aClave[i]));

}

nodo.val = valor;

}

La complejidad de este metodo es de **O(|c|)**, siendo c el tamaño de la clave. Lo que hace el *for* es recorrer los nodos navegando por la posicion del arreglo *hijo* hasta llegar a cumplir la clave. Cuando encuentra la clave se inserta el valor en el nodo creado.

**Busqueda**

El método busqueda, busca el elemento en la clave y en el sub arbol que le sigue. Recorriendo cada nodo(Excluyendo los que son nulos, debido a que al ingresar en ellos no los tiene en cuenta). Este ejercicio cuenta con 2 metodos:

**List<V>** lista = **new** ArrayList<V>();

**public** List<V> busqueda(String prefijo) {

**int** i = 0;

Nodo<V> nodo = raiz;

**char**[] aClave = prefijo.toCharArray();

**for** (i = 0; i < prefijo.length(); i++) {

**if** (nodo.hijo(alf.indice(aClave[i])) == **null**)

**return** lista;

nodo = nodo.hijo(alf.indice(aClave[i]));

}

busqueda(nodo);

**return** lista;

}

Lista es el arreglo donde se guardan los elementos, este metodo tiene complejidad **O(|c|)** Siendo c el tamano de la clave. Recorre la clave hasta cumplirla quedando con el Nodo que inicia el sub arbol a recorrer a continuacion.

**private** **void** busqueda(Nodo<V> nodo) {

**if** (nodo == **null**)

**return**;

**if** (nodo.val != **null**)

lista.add(nodo.val);

**for** (**int** x = 0; x < alf.tam(); x++)

busqueda(nodo.hijo(x));

}

La complejidad de este método que sobrecarga a Busqueda(String) es de O(s) que son la cantidad de elementos que tiene dentro del sub arbol(Mientras no sean nulos).

La suma de la complejidad de estos métodos(Se suman debido a que dentro de la misma ejecucion se ejecutan uno despues del otro) da como resultado: ***O(|c|+s)***.