FACULTAD DE INGENIERÍA Y TECNOLOGÍAS

Algoritmos y Estructura de Datos I

TAREA UT3_TA1 2022 EJ 2

DOCENTES:

Profesor Nicolás Navarro

Profesora Agustín Paredes

FECHA:

Apr 3, 2022

INTEGRANTES:

Levi De Los Santos

Rafael Durán

Fabio Luzzatto

Pablo Saravia

Alejandra Sosa Dias

Clase TLista:

TLista.Insertar

Lenguaje Natural:
Recibe un nuevo nodo del Tipo TNodo luego agregamos el nodo a la lista.
Insertamos el nodo siempre al comienzo de la lista.
No estamos controlando repeticiones de etiqueta.

PRECONDICIONES
- El dato a insertar será una nueva instancia de TNodo.
- El Nodo no será Nulo.

POSTCONDICIONES

- El Nodo deberá estar insertado en la lista.
- La lista tendrá un elemento (contenido dentro de un nodo) más que antes.

PSEUDOCÓDIGO

```
insertar(nodo)       O(n)
COMIENZO
        aux <- primero   O(1)
        SI esVacia    O(1)
                primero <- nodo O(1)
        SINO
                MIENTRAS aux.getSiguiente <> nulo     O(n)
                        aux.getSiguiente O(1)
                aux.setSiguiente(nodo) O(1)
                FinMientras
        FinSi
FIN
```

TLista.Buscar

Lenguaje Natural
Recibe una etiqueta y recorre la lista comparando la etiqueta proporcionada con el valor de etiqueta de cada nodo, si ambas etiquetas coinciden entonces retorna el Nodo o sino retorna null.

PRECONDICIONES
- La etiqueta no será nula.
- No habrá etiquetas duplicadas.

POSTCONDICIONES
- Retorna un sólo elemento.
- Retorna null si no lo encuentra.

PSEUDOCÓDIGO

```
buscar(etiqueta)   O(n)
COMIENZO
        Actual <- primero O(1)
        Mientras Actual <> nulo O(n)
                Si Actual.Etiqueta = etiqueta O(1)
                        Devolver Actual O(1)
                FinSi
                Actual <- Actual.Siguiente O(1)
        FinMientras
        Devolver nulo
FIN
```

TLista.InsertarOrdenado

Lenguaje natural

Recibe un objeto, este se inserta en la lista manteniendo un criterio para que la lista esté ordenada. Como las etiquetas son comparables las utilizamos para saber en qué órden van.

PRECONDICIONES

- El dato a insertar será una nueva instancia de TNodo.
- El Nodo no será Nulo.
- La etiqueta no será nula.
- No habrá etiquetas duplicadas.
- Debe existir una lista.

POSTCONDICIONES

- La nueva lista contendrá los productos ordenados correctamente.
- Se inserta el nuevo nodo con el producto a la lista.
- Que estén todos los nodos que estaban antes y el último ingresado

PSEUDOCÓDIGO:

TLista.insertarOrdenado(TNodo nodo) O(n)

```
COMIENZO
        SI esVacia     O(1)
                insertar(nodo)   O(1)
                return true O(1)
        FIN SI
        actual <- primero   O(1)
        siguiente <- primero.getSiguiente   O(1)
        SI actual.getEtiqueta.compareTo(nodo.getEtiqueta) > 0   O(1)
                primero <- nodo   O(1)
                nodo.setSiguiente(actual)   O(1)
                return true O(1)
        FIN SI
        MIENTRAS siguiente <> nulo      O(n)
                SI actual.getEtiqueta.compareTo(nodo.getEtiqueta) < 0 AND
siguiente.getEtiqueta.compareTo(nodo.getEtiqueta > 0   O(1)
                        actual.setSiguiente(nodo)   O(1)
                        nodo.setSiguiente(siguiente)   O(1)
                        return true O(1)
                FINSI
                actual <- actual.getSiguiente   O(1)
                siguiente <- siguiente.getSiguiente   O(1)
        FIN MIENTRAS
        SI actual.getEtiqueta.compareTo(nodo.getEtiqueta) < 0   O(1)
                actual.setSiguiente(nodo) O(1)
                devolver true O(1)
        FIN SI
        devolver false
FIN
```

Clase Almacen:

Almacen.agregarProducto():

Lenguaje natural:

Método que inserta un producto a la lista TLista.

PRECONDICIONES:

- Debe existir una lista para almacenar los productos .

POSTCONDICIONES:

- Tras llamar al método imprimirLista debe mostrarse el contenido de la lista por consola.

PSEUDOCÓDIGO:

```
Almacen.agregarProducto(producto)   O(n)
        aux <- nuevo TNodo(producto.getEtiqueta, producto) O(1)
        this.listaProductos.insertarOrdenado(aux)    O(n)
```

Almacen.agregarStock()

Lenguaje natural:
Este método debe ser capaz de agregar una cantidad determinada de productos del mismo tipo (aumentar el contador de un tipo de producto).

PRECONDICIONES:

- Debe existir al menos un producto del tipo de stock que se quiera agregar en la lista de productos.

POSTCONDICIONES:

- Se le agrega una cantidad de stock determinada al stock previamente existente.

PSEUDOCÓDIGO:

```
COMIENZO
Almacen.agregarStock(Comparable etiqueta,  cantidad)   O(n)
        aux <- listaProductos.buscar(etiqueta) O(n)
        SI aux <> nulo   O(1)
                stockActual <- aux.getDato.getStock   O(1)
                aux.getDato.setStock(stockActual+cantidad)   O(1)
        FIN SI
FIN
```

Almacen.reducirStock()

Lenguaje natural:
Este método debe ser capaz de reducir una cantidad determinada de productos del mismo tipo (reducir el contador de un tipo de producto).

PRECONDICIONES:

- Debe existir al menos un producto del tipo de stock que se quiera agregar en la lista de productos.
- El valor de stock que se quiera reducir debe ser menor o igual al stock actual.

POSTCONDICIONES:

- Se le resta una cantidad de stock determinada al stock previamente existente.

PSEUDOCÓDIGO:

```
COMIENZO
Almacen.reducirStock(Comparable etiqueta, cantidad)       O(n)
        aux <- listaProductos.buscar(etiqueta)    O(n)
        SI aux <> nulo      O(1)
                stockActual <- aux.getDato.getStock     O(1)
                SI stockActual - cantidad < 0       O(1)
                        aux.getDato.setStock(0)      O(1)
                SINO
                        aux.getDato.setStock(stockActual - cantidad)      O(1)
                FIN SI
        FIN SI
FIN
```