



# Tecnológico de Monterrey

## **Actividad 2.1**

**Programacion de estructuras de datos y algoritmos fundamentales**

**TC1031 Grupo 4**

**Prof. Jesús Guillermo Falcón Cardona**

## **Integrantes:**

**Jorge Leonardo Garcia Reynoso**

**A01734836**

**30/09/2021**

## Análisis de complejidad

### Create

```
template <class T>
void LinkedList<T>::create(T data,int pos){
    if(pos>size){
        cout<<"La lista es muy pequena para anadir el valor"<<endl;
        return;
    }
    if(pos==0){
        addFirst(data);
        return;
    }
    if(pos==size){
        addLast(data);
        return;
    }
    Node<T> *current=head;
    Node<T> *next;
    for(int i=0;i<pos-1;i++){
        current=current->getNext();
    }
    next=current->getNext();
    current->setNext(new Node<T>(data,next));
    size++;
}
```

Handwritten complexity annotations for the 'Create' function:

- $\Theta(1)$  for the first `if` block.
- $\Theta(1)$  for the second `if` block.
- $\Theta(1)$  for the third `if` block.
- $\Theta(1)$  for `*current=head;`
- $\Theta(1)$  for `*next;`
- $\Theta(n)$  for the `for` loop.
- $\Theta(1)$  for `next=current->getNext();`
- $\Theta(1)$  for `current->setNext(...);`
- $\Theta(1)$  for `size++;`

### Read

```
template <class T>
T LinkedList<T>::read(int pos){
    if(pos>=size){
        cout<<"Ese valor no existe"<<endl;
        return -1;
    }
    Node<T> *current=head;
    for(int i=0;i<pos;i++){
        current=current->getNext();
    }
    return current->getData();
}
```

Handwritten complexity annotations for the 'Read' function:

- $\Theta(1)$  for the `if` block.
- $\Theta(1)$  for `*current=head;`
- $\Theta(n)$  for the `for` loop.
- $\Theta(1)$  for `return current->getData();`

### Update

```

template <class T>
void LinkedList<T>::update(T data,int pos){
    if(pos>=size){
        cout<<"Este valor no existe"<<endl;
        return;
    }
    Node<T> *current=head;
    for(int i=0;i<pos;i++){
        current=current->getNext();
    }
    current->setData(data);
}

```

Handwritten complexity annotations for the `update` function:

- $\Theta(1)$  for the `if` block.
- $\Theta(1)$  for the `Node<T> *current=head;` line.
- $\Theta(n)$  for the `for` loop.
- $\Theta(1)$  for the `current->setData(data);` line.

## Delete

```

template <class T>
void LinkedList<T>::del(int pos){
    Node<T> *current=head;
    if(pos==0){
        head=head->getNext();
        delete current;
        size--;
        return;
    }
    Node<T> *last, *next;
    for(int i=0;i<pos;i++){
        last=current;
        current=current->getNext();
    }
    next=current->getNext();
    delete current;
    size--;
    last->setNext(next);
}

```

Handwritten complexity annotations for the `del` function:

- $\Theta(1)$  for the `Node<T> *current=head;` line.
- $\Theta(1)$  for the `if` block.
- $\Theta(1)$  for the `Node<T> *last, *next;` line.
- $\Theta(n)$  for the `for` loop.
- $\Theta(1)$  for the `next=current->getNext();` line.
- $\Theta(1)$  for the `delete current;` line.
- $\Theta(1)$  for the `size--;` line.
- $\Theta(1)$  for the `last->setNext(next);` line.

Debido a que cada una de estas funciones requieren del uso de la función `for` (la cual tiene un costo de  $\Theta(n)$ ), todas las funciones tienen una complejidad de  $O(n)$ .