Tarea Extraclase #3
Integrantes: Marlon Vega y Paola Villegas

Ordenes de Crecimiento

- Constante
    - Insertar en una linked list

```
void LinkedList::insertStart(int value) {
    Node *temp = new Node(value); //2T
    if (size > 0) { //2T
        temp->setNext(head); // T
    }

    head = temp; //T
    size++; //T

    //f(n) = 7T -> funcion constante
}
```

Algunas ayudas para determinar un orden de crecimiento O(1), sería el caso en que no haya ningún ciclo, y que con todas las entradas se hace lo mismo. Al hacer el conteo de instrucciones da una función constante, como se muestra en la imagen anterior.

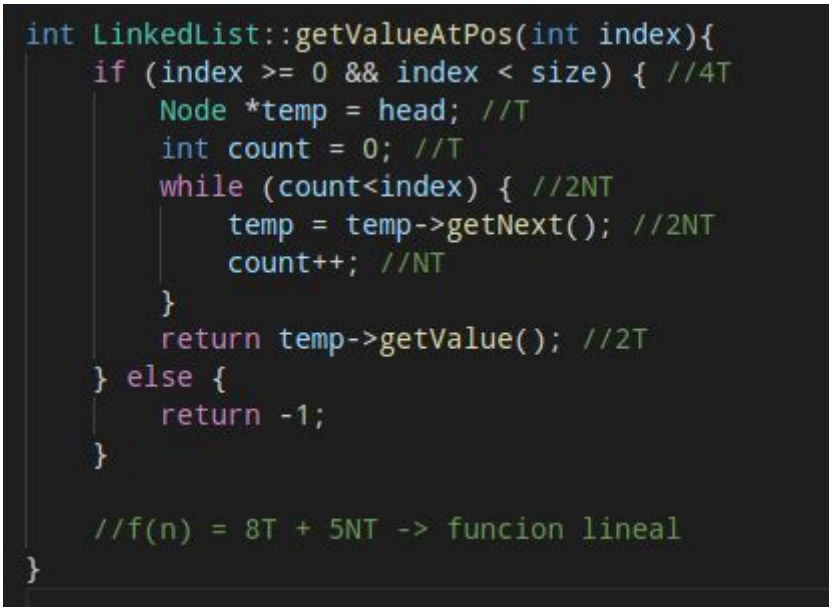
- Logaritmo Log(n)
    - Binary search

```
int binarySearch(int arr[], int l, int r, int x) {
    while (l <= r) {
        int m = l + (r - l) / 2;
        if (arr[m] == x)
            return m;
        if (arr[m] < x)
            l = m + 1;
        else
            r = m - 1;
    }
    return -1;
}
```

En el orden de crecimiento O(log n), sería el caso en que por ejemplo el tamaño de un array se va reduciendo a la mitad, como lo es la búsqueda binaria. Por ejemplo en un array de n elementos en el que se realicen 3 iteraciones (tomando i = 3), en cada iteración el array se divide a la mitad. En la iteración 1 el largo del array es n, es la segunda iteración el largo del array es n/2, en la tercera iteración el largo del array sería (n/2)/2, por lo tanto la iteración final sería (n/2)^i.

En la búsqueda binaria al final el largo del array es 1, por lo tanto (n/2)^i = 1, eso quiere decir que n = 2^i y si se aplica la función log a ambos lados y simplificando se obtiene que la complejidad es log(n).

- Lineal N
    - Buscar en linked list



```
int LinkedList::getValueAtPos(int index){
    if (index >= 0 && index < size) { //4T
        Node *temp = head; //T
        int count = 0; //T
        while (count<index) { //2NT
            temp = temp->getNext(); //2NT
            count++; //NT
        }
        return temp->getValue(); //2T
    } else {
        return -1;
    }

    //f(n) = 8T + 5NT -> funcion lineal
}
```

El orden de crecimiento lineal sería cuando hay solamente un for en un algoritmo o un while. Por ejemplo al hacer una búsqueda secuencial, como en una linkedlist, al hacer el conteo de instrucciones se obtiene una función lineal.

- Lineal logarítmica N log (n)
    - Merge sort

- Cuadrático N^2
    - Quicksort

```
void quick_sort( BidirectionalIterator first, BidirectionalIterator last, Compare cmp )
    if( first != last ) { //1T
        BidirectionalIterator left  = first; //1T
        BidirectionalIterator right = last; //1T
        BidirectionalIterator pivot = left++; //1T

        while( left != right ) { //NT
            if( cmp( *left, *pivot ) ) { //2NT
                ++left; //1NT
            } else {
                while( (left != right) && cmp( *pivot, *right ) ) //3TN^2
                    --right; //TN^2
                std::iter_swap( left, right ); //TN
            }
        }

        --left; //1T
        std::iter_swap( pivot, left ); //1T

        quick_sort( first, left, cmp ); //1T
        quick_sort( right, last, cmp ); //1T
    }
}

//f(n) = 4TN^2 + 5TN + 8T
```

En el caso del orden de crecimiento cuadrático, las formas de identificarlo sería que hayan dos while anidados o bien dos for anidados. Como se muestra en la imagen anterior del código de ordenamiento quicksort, al realizar el conteo de instrucciones se obtiene una función cuadrática.

- Cúbico N^3
    - Floyd Warshall

```
void floydWarshall (int graph[][V])
{
    int dist[V][V], i, j, k; //4T

    for (i = 0; i < V; i++)  //1T + 2NT
        for (j = 0; j < V; j++)  //1T + 2NT^2
            dist[i][j] = graph[i][j];  //2NT^2

    for (k = 0; k < V; k++) { //1T + 2NT
        for (i = 0; i < V; i++) {  //1T + 2NT^2
            for (j = 0; j < V; j++) { //1T + 2NT^3
                if (dist[i][k] + dist[k][j] < dist[i][j]) //5NT^3
                    dist[i][j] = dist[i][k] + dist[k][j];  //5NT^3
            }
        }
    }
}

//f(n) = 12NT^3 + 4NT^2 + 4NT + 9T
```

Para explicar el orden de crecimiento cúbico, se puede tomar como referencia el algoritmo Floyd Warshall, en la imagen anterior se observa como hay 3 for anidados, este sería una ayuda rápida para determinar el orden de crecimiento cúbico. Podemos ver que al hacer el conteo de instrucciones del algoritmo, se obtiene como resultado una función cúbica.