Algoritmos y Estructuras de Datos

Facultad de Ingeniería UNER

Trabajo Práctico N° 1

Integrantes: Ríos Emiliano

Análisis del orden de complejidad del algoritmo de ordenamiento

En el ejercicio 1, se empleó el algoritmo de ordenamiento burbuja. En el caso de una lista de tamaño n, se requerirán n-1 pasadas para ordenarla en el peor escenario.

```python
def ordenar(self):
    """
    Ordena los elementos de la lista de menor a mayor.
    """
    if self.tamanio <= 1:
        return self

    nodo_actual = self.cabeza
    while nodo_actual is not None:
        nodo_siguiente = nodo_actual.siguiente
        while nodo_siguiente is not None:
            if nodo_actual.dato > nodo_siguiente.dato:
                nodo_actual.dato, nodo_siguiente.dato = nodo_siguiente.dato, nodo_actual.dato
            nodo_siguiente = nodo_siguiente.siguiente
        nodo_actual = nodo_actual.siguiente

    return self
```

La implementación del método implica dos bucles while anidados. El primero recorre cada nodo, mientras que el segundo compara los valores de los nodos. Si el valor del nodo actual es mayor que el del siguiente, se realiza un intercambio, siguiendo el ordenamiento de menor a mayor. En el mejor caso, cuando la lista ya está ordenada, no se realizan intercambios. Sin embargo, en el peor caso, cada comparación de elementos resultará en un intercambio.

Debido a los dos bucles anidados que se ejecutan n veces cada uno, el tiempo de ejecución es proporcional a n^2. Por lo tanto, podemos concluir que el orden de complejidad del algoritmo de ordenamiento implementado es **O(n2).**

Gráfica del orden de complejidad de la función de ordenamiento