
Instituto Politécnico Nacional

Escuela Superior de Computo

(ESCOM)


Materia:

Análisis y Diseño de Algoritmos.

Tema:

Cuadro sinóptico de algoritmos de ordenamiento.

Alumno:

Casiano Granados Brandon Antonio

Carrera:

Ingeniería en Sistemas Computacionales.

Grupo:

3CV14

Profesora:

Moreno Galván Elizabeth

## Ordenamiento por inserción.

- La lista desordenada la dividirá en dos sublistas una donde ya se encuentren los datos ordenados y la otra los datos a ordenar.
- Su programación se basa en ingresar un dato desordenado a la lista ordenada y compararlo con los elementos de esta hasta colocarlo en su respectivo lugar.
- Su complejidad en el peor de los casos será: $O(n^2)$.
- Su complejidad en el mediano de los casos será: O (n log n).
- Su complejidad en el mejor de los casos será: O(n).

## Ordenamiento por selección.

- En la lista buscaría al nodo más pequeño, y lo cambiaría por el nodo de la primera posición, después buscaría al segundo más pequeño y lo cambiaría por el de la posición 2, y seguirá esta sucesión hasta ordenar la lista.
- Su complejidad en el peor, mediano y mejor caso es: $O(n^2)$.

## Ordenamiento de Burbuja.

- Este algoritmo se basa en comparar un nodo con el siguiente, en caso de que se cumpla la condición de ordenamiento los intercambia. Esto se repite en varias ocasiones hasta que la lista se encuentre ordenada en su totalidad.
- Su complejidad en el peor, mediano y mejor caso es: $O(n^2)$.

## Ordenación del peine.

- El algoritmo tiene el mismo principio que el de burbuja, la diferencia es que compara un nodo actual, con otro de cualquier posición y no necesariamente la siguiente. Lo cual permite que sea más eficiente.
- Su complejidad en el peor, mediano y mejor caso es: O (n log n).

## Ordenación de Shell.

- El algoritmo divide al vector en subvectores (la división en subvectores depende de la formula matemática que se utilice), comparar la posición 0 del subvector 0 con las posiciones 0 de los demás subvectores, esto sucedería con las demás posiciones de todos los subvectores. Cada iteración de comparación hace que los subvectores disminuyan de tamaño hasta que no exista la necesidad de dividir cuando la lista se ordenó.
- Su complejidad en el peor de los casos será: $O(n^2)$.
- Su complejidad en el mediano y mejor de los casos será: O (n log n).

## Ordenamiento rápido.

- El algoritmo escoge un pivote colocando todos los elementos menores al pivote en un subvector izquierdo y todos los elementos mayores en un subvector derecho, repetiría todo lo anterior hasta que los subvetores creados en las iteraciones sean de dos elementos.
- Su complejidad en el peor de los casos será: $O(n^2)$.
- Su complejidad en el mediano y mejor de los casos será: O (n log n).

## Ordenamiento de burbuja bidireccional.

- Es casi el mismo algoritmo que el de burbuja ordinario, la diferencia es que recorre el vector de forma ascendente y cuando llega al final empieza del final al principio. Este proceso de manera vulgar seria recorrer el vector de cabeza a pies y viceversa.
- Su complejidad en el peor, mediano y mejor caso es: $O(n^2)$.

## Ordenación Gnome.

- El algoritmo crea un subvector el cual incrementa un elemento a la vez por cada iteración, dicho elemento se ordenada en el subvector.
- Su complejidad en el peor de los casos será: $O(n^2)$.
- Su complejidad en el peor, mediano y mejor caso es: $O(n^2)$.

## Ordenación Merge.

- El algoritmo escoge un pivote y divide el vector en dos subvectores en donde va a poner los elementos mayores al pivote den el subvector derecho y los menores el subvector derecho, después los 2 subvectores se unen. Se repite lo anterior varias veces.
- Su complejidad en el peor de los casos será: $O(n^2)$.
- Su complejidad en el mediano y mejor de los casos será: $O (n \log n)$.