UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Ordenamiento

Programación I

Versión 24

Introducción

Los algoritmos de **ordenamiento** son técnicas utilizadas para organizar elementos en una secuencia específica. En informática, el ordenamiento es una tarea fundamental y se emplea en una amplia gama de aplicaciones, desde bases de datos hasta algoritmos de búsqueda y análisis de datos.

Introducción

Nosotros vamos a utilizar estos algoritmos para ordenar nuestros arrays, ya sean unidimensionales como multidimensionales.

Con esto podemos lograr cosas como:

- Ordenar alfabeticamente un array
- Ordenar un array de menor a mayor
- Ordenar un array de mayor o menor

Métodos de Ordenamiento

Existen varios métodos de ordenamiento como:

- Burbujeo (Bubble Sort)
- Selección (Selection Sort)
- Inserción (Insertion Sort)
- Merge Sort
- Quick Sort
- Heap Sort
- Entre otros

Clasificación

- La cantidad de **intercambios**: esta es la cantidad de veces que el algoritmo intercambia elementos.
- El número de **comparaciones**: este es el número de veces que el algoritmo compara elementos.
- La cantidad de **espacio** adicional requerido: algunos algoritmos pueden ordenar una lista sin la necesidad de crear listas nuevas.
- Utilizan **recursividad** o no: algunos algoritmos usan técnicas recursivas y otros no.

Método Burbujeo

En nuestro caso vamos a ver el **método burbujeo** ya que gracias a su simpleza es una buena introducción para ordenar arrays.

Método Burbujeo

¿Qué es el Método de Burbujeo?

- Es un algoritmo de ordenamiento simple y fácil de entender.
- Funciona comparando pares de elementos adyacentes y realizando intercambios si es necesario hasta que la lista esté ordenada.

Ejemplo

Vamos a abstraernos un poco de la programación primero para explicar este método.

Imaginemos que tenemos está lista de números

[5,3,1,7,9]

Y queremos ordenarla de esta manera

[9,7,5,3,1]

Ejemplo

Para ello debemos comparar cada uno de sus elementos entre si de la siguiente manera:

1)	5	3	1	9	7
2)	5	3	1	9	7
3)	5	3	1	9	7
4)	5	3	1	9	7
5)	9	3	1	5	7
6)	9	3	1	5	7
7)	9	3	1	5	7
8)	9	3	1	5	7
9)	9	5	1	3	7
10)	9	7	1	3	5
11)	9	7	1	3	5
12)	9	7	3	1	5
13)	9	7	5	1	3
14)	9	7	5	1	3
15)	9	7	5	3	1
16)	9	7	5	3	1

Ejemplo

1) Se toma de referencia el primer elemento de mi lista (5) y se lo va a comparar con todos los elementos que tiene a la derecha (3,1,9,7)

2) Se compara el primer elemento (5) con el primero que tiene a la derecha (3) En este caso como queremos ordenar de mayor a menor tenemos que verificar si el elemento de la derecha es mayor al de la izquierda o sea (5 < 3) como está comparación no es verdadera no se produce ningún movimiento en mi lista.

3) Ahora se compara el primer elemento (5) con el segundo que tiene la derecha (1) por ende (5 < 1) como está comparación no es verdadera no se produce ningún intercambio en mi lista.

Ejemplo

4) Ahora el primer elemento (5) se compara con el tercer elemento a su derecha (9) por ende (5 < 9) en este caso el 9 es mayor a 5 por ende se produce un intercambio en mi lista (swap), por ende al final de este paso el (9) pasa a ser el primer elemento y el (5) ahora se convierte en el cuarto elemento de mi lista, esto es conocido como swap (las listas intercambian indices)

5) Ahora el primer elemento que es el (9) se compara con el cuarto elemento a su derecha (7) por ende (9 < 7) como no se cumple no se produce un intercambio

6) En este paso ya el primer elemento de mi lista está ordenado por ende este mismo ya no tiene protagonismo.

Ejemplo

7) Se vuelve a repetir el procedimiento pero ahora el segundo elemento de mi lista se va a comparar con todos los que tiene a su derecha (3) se compara con (1) (3 < 1) no hay intercambio

8) (3 < 5) hay intercambio ahora el segundo elemento pasa a ser el (5)

9) (5 < 7) hay intercambio ahora el segundo elemento pasa a ser el (7)

10) En este paso ya el segundo elemento de mi lista está ordenado por ende este mismo ya no tiene protagonismo.

11) (3 < 1) no hay intercambio

Ejemplo

12) (3 < 5) hay intercambio ahora el tercer elemento pasa a ser el (5)

13) En este paso ya el tercer elemento de mi lista está ordenado por ende este mismo ya no tiene protagonismo.

14) (1 < 3) hay intercambio ahora el cuarto elemento pasa a ser el (3)

14) En este paso ya el cuarto elemento de mi lista está ordenado por ende este mismo ya no tiene protagonismo.

15) En este paso ya no quedan elementos a la izquierda por ende no se compara con nada

16)Lista ordenada

Ejemplo

Si quisiera ordenar de menor a mayor **[1,3,5,7,9]** tengo que aplicar el mismo procedimiento pero con una comparación diferente (a > b) en vez de (a < b)

1)	5	3	1	9	7
2)	5	3	1	9	7
3)	3	5	1	9	7
4)	1	5	3	9	7
5)	1	5	3	9	7
6)	1	5	3	9	7
7)	1	5	3	9	7
8)	1	3	5	9	7
9)	1	3	5	9	7
10)	1	3	5	9	7
11)	1	3	5	9	7
12)	1	3	5	9	7
13)	1	3	5	9	7
14)	1	3	5	9	7
15)	1	3	5	7	9
16)	1	3	5	7	9

Ejemplo en código

Ahora con este ejemplo planteado, necesitamos generar un algoritmo que logre ordenarme para ello vamos a necesitar recorrer dos for

Un for principal que va a recorrer uno por uno los elementos de mi array, y otro **for secundario** que va a estar dentro del principal que me va a recorrer todos los elementos que esten al lado al elemento que estoy recorriendo.

Ejemplo en código

```
numeros = [5,3,1,7,9]

for i in range(len(numeros)): -> For principal

    for j in range(i+1,len(numeros),1): -> For secundario

        if(comparación):

            aux = numeros[i] -> intercambio

            numeros[i] = numeros[j] -> intercambio

            numeros[j] = aux -> intercambio
```

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Ejemplo en código #1

```
numeros = [5,3,1,7,9]


#ORDENO DE MAYOR A MENOR
for i in range(len(numeros)):
    for j in range(i+1,len(numeros),1):
        if(numeros[i] < numeros[j]):
            aux = numeros[i]
            numeros[i] = numeros[j]
            numeros[j] = aux


print(numeros)
```

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Ejemplo en código #2

```
numeros = [5,3,1,7,9]


#ORDENO DE MENOR A MAYOR
for i in range(len(numeros)):
    for j in range(i+1,len(numeros),1):
        if(numeros[i] > numeros[j]):
            aux = numeros[i]
            numeros[i] = numeros[j]
            numeros[j] = aux


print(numeros)
```