Benemerita Universidad Autonoma de Puebla

Diseño y análisis de Algoritmos

Ordenamiento Burbuja

Integrantes:
Oscar Vazquez Orihuela
Berenice Juarez González
Kevin Gracia Guerra
Maria Fernanda Castro Salgado

Profesor
Pedro Tecuanhuehue Vera

Facultad de Computación

8 de febrero de 2023

Índice

1. Introducción

A lo largo del tiempo, la computación y el ámbito de la programación han evolucionado de tal forma que la cantidad de datos que procesa una computadora es cada vez más grande. Es por lo que se han implementado estrategias para su ordenamiento, puesto que el orden de datos en un programa es de vital importancia para que funcionen, de manera óptima y eficiente. Por lo tanto, este documento explicara el ordenamiento burbuja, el cual, está basado en comparar elementos adyacentes de la lista e intercambiar sus valores en caso de estar desordenados.
Esta documentación aclara cómo funciona dicho ordenamiento; mostrar la forma en la cual se programa en Python, su orden de complejidad, el cual será explicado a detalle para una mejor comprensión.
Al finalizar se mostrará una gráfica con dichos ordenamientos con una cantidad (n) de datos ejecutado de igual manera (n) veces.

2. Ordenamiento Burbuja

El ordenamiento burbuja (Bubble Sort) hace múltiples pasadas a lo largo de una lista. Compara los valores adyacentes e intercambia los que no están en orden. Cada pasada a lo largo de la lista ubica el siguiente valor más grande en su lugar apropiado; en esencia, cada valor "burbujea" hasta el lugar al que pertenece.

En la Figura 1 se puede apreciar como este ordenamiento inicia desde la posición 1 (0 desde un arreglo), lo compara con el valor adyacente (posicionado del lado derecha), en caso de ser mayor este se desplaza a la posición adyacente y el valor que es menor se desplaza a la posición donde estaba el valor más grande (izquierda). Con este medio el ordenamiento hace múltiples pasadas a todos los datos hasta que queden ordenados de manera correcta (de menor a mayor)

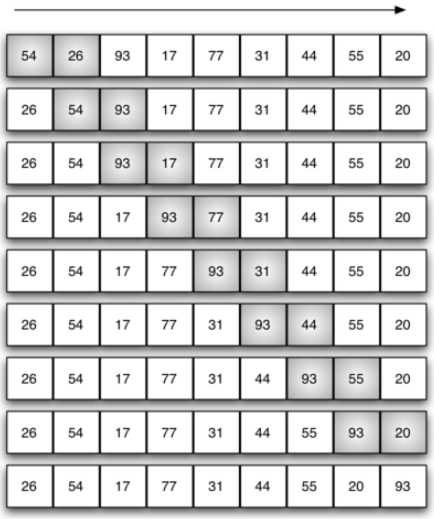


Figura 1: Comportamiento del ordenamiento Burbuja

2.1. ¿Cuándo es conveniente usar el ordenamiento burbuja?

El ordenamiento burbuja (Bubble Sort) se considera poco eficiente, debido a la cantidad de tiempo que toma ordenar la cantidad (n) de datos que hay en el arreglo.

Dado que, al hacer múltiples pasadas por todos los datos, esto produce que cuando hay una gran cantidad de datos, el tiempo sea muy prolongado para el ordenamiento. Por lo que se recomienda utilizar este ordenamiento cuando la cantidad de datos no es muy grande.

Como se puede observar en la Figura 2 el tiempo de ordenamiento para 500,000 datos es muy lenta.

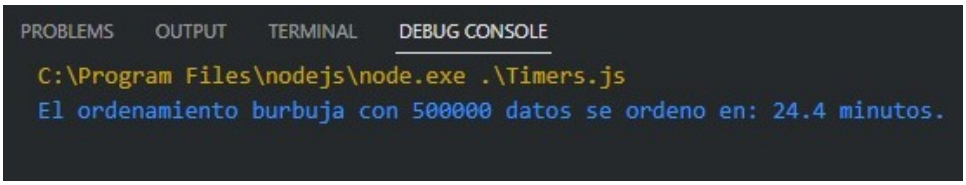


Figura 2: Tiempo de ordenamiento burbuja a 500,000 datos

3. Pseudocódigo

Algoritmo

```
Inicio Burbuja (X[k]) //Inicio del ordenamiento

    para i = 1 a n-1 hacer //Ciclo for de i a n-1 que recorre la posicion del arreglo
        para j = 1 a n-1 hacer /*Ciclo for de j a n-1 que recorre los valores en
        cada posicion del arreglo*/
            Si (X[j] > X[j+1]) entonces /*Si el valor de j en la posicion i es
            mayor al valor de la posicion i + 1 entonces*/

                auxiliar = X[j] //El valor se pasa a la variable auxiliar
                X[j] = X[j+1] /*El valor de i + 1 pasa a la posicion de i y se
                asigna su valor*/
                X[j+1] = auxiliar /*El valor almacenado en la variable auxiliar
                pasa al valor de la posicion i + 1*/

            Fin Si //Fin de la estructura de control condicion If
        Fin para //Fin de la estructura de control repetitiva anidada
    Fin para //Fin de la estructura de control repetitiva
Fin Burbuja //Fin del ordenamiento
```

4. Codigo del programa

```
import random #biblioteca de un numero aleatorio
import numpy as np #importar libreria numpy para leer los datos
#de los archivos txt
import matplotlib.pyplot as plt #libreria para poder crear graficos en
#dos dimensiones

arr = [] #crear un arreglo llamado arr

#crear numero aleaotrios
for i in range(100): #se inicializa la variable i en 100
    arr.append(random.randint(100, 200)) #se generan los 100 numeros entre
    #100 y 200
print(arr) #imprime el arreglo

contador = 0 #se crea un variable llamada contador
#Establecemos la variable analisis en False para entrar en el bucle al menos
#una vez
analisis = False

while analisis:
    analisis = True #se cambia el valor de la variable a True
```

```
    for i in range(len(arr) - 1): #un ciclo para recorrer todo el arreglo
        if arr[i] > arr[i + 1]: #comparar el numero de la posicion 0 con el
        #de la siguiente posicion
            contador += 3 #el contador sumara de 3 en 3 por el numero de
                          #asignaciones que se tienen
            #sucede los intercambios de los datos
            aux = arr[i]
            arr[i] = arr[i + 1]
            arr[i + 1] = aux
           # cambiamos a False la variable ya que ha habido un intercambio
            analisis = False

#imprime la cantidad de ordenamientos al ejecutarse
print("La cantidad de ordenamientos fue de: " + str(contador))

#usar la variable x,y para crear la grafica, luego con el np cargaremos el archivo
#txt para los valores de x,y
#El skiprows es para que empieze a leer los datos desde la linea 2
#(da un salto de linea de donde se encuentra x,y)
#usecols es para leer los datos tomando como los datos de x=0 y los de y=1
#el unpack nos sirve para poder leer los daots del archivo txt
x1, y1 = np.loadtxt('primera.txt', skiprows=1, usecols=[0, 1], unpack=True)
x2,y2=np.loadtxt('segunda.txt', skiprows=1, usecols=[0,1], unpack=True)
x3,y3=np.loadtxt('tercera.txt', skiprows=1, usecols=[0,1], unpack=True)
x4,y4=np.loadtxt('cuarta.txt', skiprows=1, usecols=[0,1], unpack=True)
x5,y5=np.loadtxt('quinta.txt', skiprows=1, usecols=[0,1], unpack=True)
x6,y6=np.loadtxt('sexta.txt', skiprows=1, usecols=[0,1], unpack=True)
x7,y7=np.loadtxt('septima.txt', skiprows=1, usecols=[0,1], unpack=True)
x8,y8=np.loadtxt('octava.txt', skiprows=1, usecols=[0,1], unpack=True)
x9,y9=np.loadtxt('novena.txt', skiprows=1, usecols=[0,1], unpack=True)
x10,y10=np.loadtxt('decima.txt', skiprows=1, usecols=[0,1], unpack=True)
#imprimir los datos de todos los archivos txt
print(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8,x9,y9,x10,y10)
#para poder cargar los datos en la grafica
plt.plot(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8,x9,y9,x10,y10)
#crear una cuadricula en la grafica
plt.grid(True)
#para agregar los titulos dentro de la grafica
plt.title("GRAFICA METODO BURBUJA")
plt.xlabel("numero de datos")
plt.ylabel("numero de operaciones")
#mostrar el grafico
```

5. Experimentos

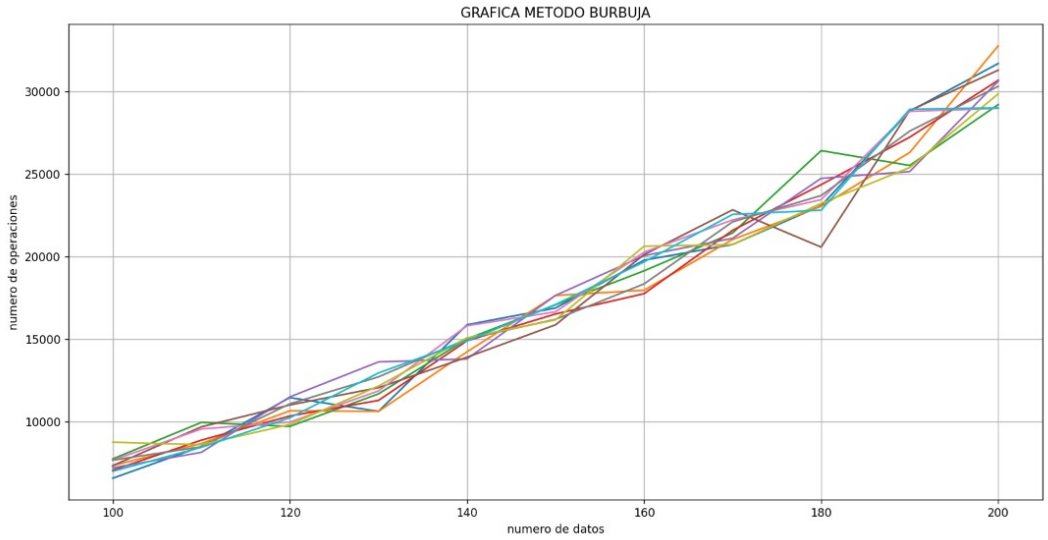
Se ejecutó el programa 10 veces con cada uno de los distintos parámetros establecidos por el profesor (100, 110, 120, 130, 140...200) y estos fueron los resultados

Datos	Primera	Segunda	Tercera	Cuarta	Quinta	Sexta	Septima	Octava	Novena	Decima
100	6564	7332	7734	7047	7194	7326	7656	7656	8742	6981
110	8481	8661	9942	8868	8130	9675	9543	8469	8592	8436
120	11445	10644	9702	10332	11487	11001	9963	11070	9819	10224
130	10611	10611	11670	11274	13605	12045	11847	12699	12141	12936
140	15855	14214	14994	14883	13788	13890	15795	14997	15066	14883
150	16869	17628	17064	16509	17631	15852	16641	16185	16152	17082
160	19767	17949	19122	17739	20040	20103	20250	18330	20610	19662
170	20718	21021	21411	21573	21093	22815	22209	22089	20715	22539
180	23070	23028	26403	24342	24723	20559	23433	23682	23208	22794
190	28782	26289	25503	27216	25134	28827	28779	27582	25377	28908
200	31680	32751	29193	30663	30612	31275	28962	30297	29841	28992

Tabla 1: Cantidad de asignaciones en cada ejecución del programa

Como se observa en la Tabla 1 el crecimiento de asignaciones es mayor cuando los datos son mayores, debido a que, al ser más datos la cantidad de repeticiones que hace el algoritmo es mayor. Mientras que en las ejecuciones de las mismas cantidades de datos a ordenar no hay un cambio muy notorio, ya que al ser datos aleatorios esto puede beneficiar o afectar al ordenamiento de dichos datos, pero el rango de diferencia entre una ejecución y otra no es muy notoria.

Como se observa en la gráfica 1 el crecimiento es notorio conforme la cantidad de datos a ordenar aumento, dandonos una idea de cuántas asignaciones genera (del mínimo al máximo) facilitando nuestro análisis.



6. Orden de complejidad

Se desarrolla el anáslisis para obtener la complejidad del algoritmo del metodo burbuja,en el algoritmo tenemos dos ciclos for donde una ocupara la variable i y el otro la variable j y el condicional if estan las asignaciones(operaciones basicas),

Representado de la siguiente manera:

$$\sum_{i=1}^{n-1}\sum_{j=1}^{n-i} 3 \tag{1}$$

Por propiedades de las sumatorias : $\sum_{i=p}^{n} a_i = n - p + 1$, la sumatoria que es del ciclo for para j la sustituimos de la siguiente manera:

$$\sum_{i=1}^{n-1}(n-i)(3) \tag{2}$$

Propiedad distributiva de la sumatoria:

$$3\sum_{i=1}^{n-1}(n-i) \tag{3}$$

Propiedad asociativa de la sumatoria del ciclo for para j se generan dos sumatorias:

$$3\left[\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i\right] \tag{4}$$

Separamos las dos sumatorias que se encuentran en el inciso (4)

$$\sum_{i=1}^{n-1} n \tag{5}$$

$$\sum_{i=1}^{n-1} i \tag{6}$$

Resolvemos la ecuación del inciso (5)

$$\sum_{i=1}^{n-1} n \to n\sum_{i=1}^{n-1} \to n\,(n-1) \tag{7}$$

Donde se obtiene

$$n^2 - n \tag{8}$$

Resolvemos la ecuación del inciso(6); sabiendo que cumple una propiedad de la sumatoria la cual es la siguiente:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{9}$$

Sustituimos la ecuación del inciso (6) en la del inciso (9)

$$\sum_{i=1}^{n-1} i = \frac{(n-1)((n-1)+1)}{2} = \frac{(n-1)n}{2} \tag{10}$$

Realizando el producto obtenemos lo siguiente:

$$\sum_{i=1}^{n-1} i \to \frac{(n-1)n}{2} \to \frac{n^2-n}{2} \tag{11}$$

Al sustituir los valores de ambas ecuaciones en la ecuación del inciso (4)

$$3\left[(n^2-n)-\left(\frac{n^2-n}{2}\right)\right] \rightarrow 3\left[\frac{2n^2-2n-n^2+n}{2}\right] \rightarrow \frac{3}{2}\left[2n^2-2n-n^2+n\right] \rightarrow \frac{3}{2}\left[n^2-n\right] \tag{12}$$

Finalmente el resultado del inciso (12) nos da la complejidad del ordenamiento de burbuja

$$O\left(n^2\right) \tag{13}$$

7. Conclusión

La evidencia que presentamos anteriormente demuestra que el Ordenamiento Burbuja es bastante funcional, en casos donde la cantidad de datos a ordenar no es muy grande, así como que su orden de complejidad es correcto y con base al análisis planteado se llegó a una respuesta acertiva para el ordenamiento de datos. Finalmente logramos concluir que hacer este análisis nos hizo comprender mejor como se analiza y desarrolla un algoritmo y nos da la posibilidad de adquirir un conocimiento mayor para algoritmos futuros.