Universidad Central de Venezuela

Facultad de Ingeniería

Ciclo Básico de Ingeniería

Departamento de

Programación (0790)

Gabriela Jiménez

C.I. 28017436

Ensayo

Los métodos de ordenamiento son algoritmos fundamentales en ciencias de la computación que permiten organizar eficientemente los elementos de una colección de datos en un orden secuencial de acuerdo a un criterio de ordenamiento específico. Estos algoritmos son ampliamente utilizados en diversas aplicaciones, desde bases de datos y sistemas de búsqueda hasta la implementación de algoritmos más complejos. El objetivo principal de los métodos de ordenamiento es reorganizar los registros de una tabla o arreglo en un orden lógico, basado en el valor de algún campo o atributo. Esto facilita la búsqueda y recuperación de datos, mejora la eficiencia de los algoritmos que trabajan con esos datos y brinda una estructura más legible y comprensible para los usuarios.

Existen numerosos métodos de ordenamiento, cada uno con sus propias características y eficiencia en términos de tiempo y espacio. Uno de ellos es el método de la burbuja, este es un algoritmo sencillo pero efectivo para ordenar elementos en una colección de datos. Aunque puede ser considerado como uno de los métodos más simples, su simplicidad es una de sus fortalezas y ha demostrado ser útil en muchas situaciones. La belleza del método de la burbuja radica en su enfoque intuitivo y fácil comprensión. Este algoritmo funciona comparando pares de elementos adyacentes y reorganizándolos si están en el orden incorrecto. De esta manera, los elementos "burbujean" gradualmente hacia sus posiciones correctas a medida que se realizan múltiples pasadas sobre el conjunto de datos.

Aunque el método de la burbuja puede ser considerado como uno de los más simples, eso no significa que sea ineficiente o menos útil. En realidad, este algoritmo tiene varias ventajas que lo hacen valioso en ciertos contextos. Una de sus principales ventajas es su fácil implementación, lo que lo convierte en una opción atractiva para aplicaciones de menor escala o cuando la simplicidad es prioritaria.

```python
from random import sample
# Importamos un Método de la biblioteca random para generar listas aleatorias

lista = list(range(100)) # Creamos la lista base con números del 1 al 100

# Creamos una lista aleatoria con sample
#(8 elementos aleatorios de la lista base)
vectorbs = sample(lista,8)


def bubblesort(vectorbs):
    """Esta función ordenara el vector que le pases como argumento con el Método de Bubble Sort"""

    # Imprimimos la lista obtenida al principio (Desordenada)
    print("El vector a ordenar es:",vectorbs)
    n = 0 # Establecemos un contador del largo del vector

    for _ in vectorbs:
        n += 1 #Contamos la cantidad de caracteres dentro del vector

    for i in range(n-1):
    # Le damos un rango n para que complete el proceso.
        for j in range(0, n-i-1):
            # Revisa la matriz de 0 hasta n-i-1
            if vectorbs[j] > vectorbs[j+1] :
                vectorbs[j], vectorbs[j+1] = vectorbs[j+1], vectorbs[j]
            # Se intercambian si el elemento encontrado es mayor
            # Luego pasa al siguiente
    print ("El vector ordenado es: ",vectorbs)

bubblesort(vectorbs)
```

Explicación del código:

1. En la primera línea de código, se encuentra la función **sample** de la biblioteca **random**. Esta función nos permite generar una lista aleatoria de elementos tomados de una lista base.
2. Luego, se encuentra una lista base llamada **lista** que contiene números del 1 al 100 utilizando la función **range** y convirtiéndola en una lista.
3. Se utiliza la función **sample** para crear una lista aleatoria llamada **vectorbs** que contiene 8 elementos seleccionados al azar de la lista base **lista**.
4. Seguidamente se define la función **bubblesort** que toma como argumento un vector, aquel que se desea ordenar a través del método de ordenamiento de la burbuja).
5. La función imprime el vector inicial desordenado utilizando **print**.
6. Luego, se declara una variable **n** inicializada en 0 para contar el largo del vector.
7. Se inicia un bucle **for** que itera sobre cada elemento en **vectorbs** con el objetivo de contar la cantidad de elementos presentes y almacenarla en la variable **n**.

8. Después se inica otro bucle **for** que va de 0 a **n-1**. Este bucle representa las pasadas que se realizarán para ordenar el vector. Dentro de este segundo bucle **for**, se encuentra otro bucle **for** anidado que va de 0 a **n-i-1**, donde **i** representa la pasada actual. Este bucle itera sobre la matriz y compara los elementos adyacentes.
9. Si el elemento actual es mayor que el siguiente, se realiza un intercambio de posiciones entre ellos utilizando la técnica de "burbujeo". Esto significa que el elemento mayor "burbujea" hacia la derecha, moviéndose hacia su posición correcta.
10. Después de completar todos los bucles, la función imprime el vector ordenado utilizando **print**.