ITSRLL
INSTITUTO TECNOLÓGICO SUPERIOR DE LA REGIÓN DE LOS LLANOS

Ingeniería Mecatrónica

PROGRAMACIÓN AVANZADA

Enero – Junio 2025

M.C. Osbaldo Aragón Banderas

UNIDAD:

1	2	3	4	5

Nombre de actividad:

Evaluación de Métodos de Ordenamiento

Actividad realizada por:

Diego Alberto Alcaraz Gonzalez

Guadalupe Victoria, Durango

Fecha de entrega:

12	02	2025

Introducción

En este proyecto, se comparan dos algoritmos de ordenamiento clásicos: el **método de burbuja** y el **Quick Sort**. El objetivo es analizar su rendimiento en términos de tiempo de ejecución para listas de diferentes tamaños, esta comparación es relevante porque permite entender cómo la complejidad computacional de los algoritmos afecta su eficiencia en la práctica. El código implementa ambos algoritmos, genera listas de números aleatorios de diferentes tamaños y mide el tiempo que tarda cada algoritmo en ordenar estas listas. Los resultados se presentan en una tabla comparativa, lo que facilita la visualización de las diferencias en el rendimiento.

Entorno de programación Jupyter

Jupyter es un entorno interactivo de programación que permite escribir y ejecutar código en Python y otros lenguajes de forma estructurada y visual. Es ampliamente utilizado en ciencia de datos, inteligencia artificial, simulaciones y control de hardware, ya que facilita la integración con bibliotecas como NumPy, Pandas, Matplotlib, TensorFlow y OpenCV. Su interfaz basada en celdas permite organizar el código en fragmentos ejecutables, documentar con Markdown, visualizar gráficos y analizar datos de manera eficiente. Además, se ejecuta en el navegador, lo que facilita su uso remoto y la colaboración. Es especialmente útil en proyectos que requieren análisis de datos, comunicación con microcontroladores como ESP32 y Raspberry Pi, o experimentación con modelos de machine learning. Para instalarlo, basta con ejecutar pip install jupyter, y se inicia con jupyter notebook.

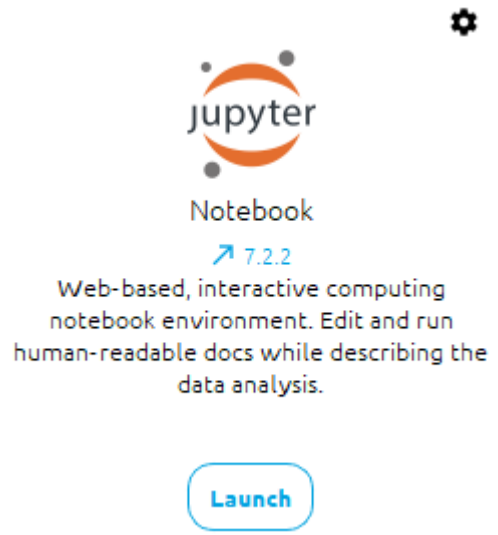

Ilustración 1: Logo Jupyter

Ordenamiento Burbuja (Bubble Sort)

El **algoritmo de ordenamiento burbuja** es un método sencillo para organizar los elementos de una lista en orden ascendente o descendente. Funciona comparando pares de elementos adyacentes y los intercambia si están en el orden incorrecto. Este proceso se repite varias veces hasta que la lista esté completamente ordenada. Su nombre proviene de la forma en que los valores más grandes "burbujearán" hacia el final de la lista en cada iteración.

A pesar de ser fácil de entender e implementar, **Bubble Sort no es eficiente para listas grandes**, ya que su complejidad en el peor caso y el caso promedio es de $O(n^2)$. Esto significa que el tiempo de ejecución crece rápidamente a medida que aumenta el número de elementos en la lista. Sin embargo, en el mejor caso (cuando la lista ya está ordenada), su complejidad se reduce a $\boldsymbol{O(n^2)}$. Debido a su ineficiencia, este algoritmo se usa más en entornos educativos para explicar conceptos básicos de ordenamiento que en aplicaciones reales.

Ordenamiento Rápido (Quick Sort)

El **algoritmo Quick Sort** es un método eficiente de ordenamiento basado en la técnica de **divide y vencerás**. Consiste en elegir un **pivote** dentro de la lista y dividir el conjunto en dos sublistas: una con los elementos menores al pivote y otra con los mayores. Luego, el algoritmo se aplica recursivamente a ambas partes hasta que toda la lista esté ordenada.

Una de las principales ventajas de Quick Sort es su eficiencia. En el **caso promedio y en el mejor caso**, tiene una complejidad de **O(n*log*n)**, lo que lo hace significativamente más rápido que Bubble Sort para listas grandes. Sin embargo, en el peor de los casos (cuando el pivote siempre es el menor o el mayor de la lista), su complejidad puede degradarse a $\boldsymbol{O(n^2)}$. Para evitar esto, se pueden emplear estrategias de selección de pivote, como elegir uno aleatorio o el valor mediano.

Gracias a su velocidad y flexibilidad, **Quick Sort es uno de los algoritmos de ordenamiento más utilizados en la informática**, siendo empleado en bases de datos, sistemas operativos y estructuras de datos avanzadas. A diferencia de

Bubble Sort, es ideal para manejar grandes volúmenes de datos y es ampliamente preferido en aplicaciones del mundo real.

Código y explicación

```
import time
import random
```

Ilustración 2: Importación de módulos

El código comienza importando dos módulos esenciales: time y random. El módulo time se utiliza para medir el tiempo de ejecución de los algoritmos, lo que permite comparar su eficiencia. El módulo random se emplea para generar listas de números aleatorios, que son necesarias para probar los algoritmos en diferentes escenarios.

```
def metodo_burbuja(lista):
    for i in range(len(lista)):
        for j in range(0, len(lista) - i - 1):
            if lista[j] > lista[j + 1]:
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
    return lista
```

Ilustración 3: Algoritmo de Ordenamiento Burbuja

Se define la variable método burbuja, usando la condicional for para realizar un numero de pasadas según el numero de elementos en la lista, posteriormente usando la condicional if se compara si el elemento elegido es mayor al elemento siguiente, si el elemento elegido es mayor al siguiente elemento se cumple la condicional y cambian de lugar, pasando a la derecha el elemento elegido, repitiendo el proceso según el numero de elementos de la lista.

```python
def quick_sort(lista):
    if len(lista) < 2:
        return lista
    else:
        pivote = lista[0]
        menores = [x for x in lista[1:] if x < pivote]
        mayores = [x for x in lista[1:] if x >= pivote]
        return quick_sort(menores) + [pivote] + quick_sort(mayores)
```

Ilustración 4: Algoritmo de Quick Sort

En esta parte se define la función quick_sort para la lista, se establece con la condicional if que si el numero de elementos de lista solo son dos la regresara, si no se establece el primer elemento de la lista para hacer de pivote, dividiendo en dos la lista a partir de la comparación del pivote, creando una lista llamada menores y mayores almacenando elementos según si son mayores o menores al pivote, por ultimo regresa como resultado la suma de las listas en el orden: menores + pivote + mayores.

```python
def medir_tiempo(algoritmo, lista):
    inicio = time.time()
    algoritmo(lista.copy())
    fin = time.time()
    return fin - inicio
```

Ilustración 5: algoritmo para medir tiempo

Para comparar el rendimiento de ambos algoritmos, se implementa una función llamada medir_tiempo, esta función toma como argumentos un algoritmo de ordenamiento y una lista de números. Primero, registra el tiempo actual usando time.time(), luego ejecuta el algoritmo sobre una copia de la lista para no modificar la lista original. Finalmente, registra el tiempo nuevamente y calcula la diferencia entre ambos valores, que corresponde al tiempo que tardó el algoritmo en ordenar la lista. Este tiempo se devuelve como resultado.

```python
tamanos = [100, 2000, 5000, 10000]

# Comparación de tiempos para cada tamaño de lista
for tam in tamanos:
    lista_random = [random.randint(1, tam * 10) for _ in range(tam)]  # Lis

    print(f"\nOrdenando lista de {tam} elementos:")

    tiempo_burbuja = medir_tiempo(metodo_burbuja, lista_random)
    print(f"Tiempo del método de burbuja: {tiempo_burbuja:.6f} segundos")

    tiempo_quick_sort = medir_tiempo(quick_sort, lista_random)
    print(f"Tiempo del Quick Sort: {tiempo_quick_sort:.6f} segundos")
```

Ilustración 6: código de tamaño de listas y resultados

Ya por último el código se encarga de generar listas de diferentes tamaños y medir el tiempo que tarda cada algoritmo en ordenarlas, se definen cuatro tamaños de lista: 100, 2000, 5000 y 10000 elementos. Para cada tamaño se genera una lista de números aleatorios utilizando una comprensión de listas y la función random.randint (esta función es para que en la lista se puedan repetir los números generados de manera aleatoria). Luego se llama a la función medir_tiempo para medir el tiempo que tarda el método de burbuja y el Quick Sort en ordenar la lista, por ultimo los resultados se imprimen en la consola, mostrando el tiempo de ejecución de cada algoritmo para cada tamaño de lista.

Resultados

```
Ordenando lista de 100 elementos:
Tiempo del método de burbuja: 0.002000 segundos
Tiempo del Quick Sort: 0.000999 segundos

Ordenando lista de 2000 elementos:
Tiempo del método de burbuja: 0.652026 segundos
Tiempo del Quick Sort: 0.004530 segundos

Ordenando lista de 5000 elementos:
Tiempo del método de burbuja: 2.696266 segundos
Tiempo del Quick Sort: 0.014091 segundos

Ordenando lista de 10000 elementos:
Tiempo del método de burbuja: 11.233988 segundos
Tiempo del Quick Sort: 0.027651 segundos
```

A partir de los resultados obtenidos, se observa que el algoritmo de ordenamiento burbuja (Bubble Sort) es significativamente más lento que Quick Sort, especialmente a medida que el tamaño de la lista aumenta, para una lista de 100 elementos, la diferencia de tiempo entre ambos algoritmos es mínima, con Quick Sort siendo aproximadamente el doble de rápido. Sin embargo, conforme la cantidad de elementos crece, la diferencia en rendimiento se vuelve más evidente. Por ejemplo, en la ordenación de 2000 elementos, Bubble Sort toma 0.652 segundos, mientras que Quick Sort solo 0.0045 segundos, lo que representa una mejora de más de 140 veces en velocidad. Para 5000 y 10,000 elementos, la diferencia se hace aún más extrema: mientras que Bubble Sort tarda 2.69 y 11.23 segundos, Quick Sort resuelve el mismo problema en apenas 0.014 y 0.027 segundos, mostrando una mejora de más de 400 veces en el caso más grande.

Bibliografía

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2013). Introducción a los algoritmos (3.ª ed.). Alfaomega.

Knuth, D. E. (2000). El arte de la programación de computadores, Volumen 3: Ordenamiento y búsqueda (2.ª ed.). Addison-Wesley.