Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas Instituto Politécnico Nacional

UNIDAD 2

REPORTE

QUICKSORT

Nombre	Boleta
Enid Aimee Perez Robles	2023670020

Grupo:
3CM2

Docente:
M. en C. Erika Sánchez-Femat

Introducción

En el mundo de la informática y la ciencia de la computación, los algoritmos de ordenamiento desempeñan un papel fundamental. Estos algoritmos son herramientas esenciales para organizar datos de manera eficiente y efectiva. Uno de los algoritmos más destacados en este contexto es Quicksort.

El objetivo de esta práctica es que los estudiantes implementen el algoritmo Quicksort en Python, midan su rendimiento a través de la medición de tiempos de ejecución en 100 casos de prueba diferentes en arreglos aleatorios y visualicemos los resultados mediante gráficos. Este enfoque nos permite comprender el funcionamiento del algoritmo, adquirir habilidades de programación, aprender a analizar datos y promover la experimentación y el pensamiento crítico en el contexto de la ordenación de datos.

Desarrollo

Para comprender mejor el funcionamiento de este algoritmo, se tendrá que generar 100 casos de prueba, es decir, se tendrá que generar 100 arreglos de números aleatorios y de tamaño aleatorio. Con el fin de cubrir la mayor cantidad de casos en los que este algoritmo se puede aplicar. Para este punto se tendrá que generar números aleatorios, para esto lo tendremos que hacer con la librería ***random.***

Luego haremos una función para medir el tiempo de ejecución de cada prueba de QuickSort,este será muy importante para la representacion de la grafica y ver los 100 casos de prueba de ejecución en ella.Para esta parte se utilizó la libreria ***import time.***

Posteriormente, mediremos el tiempo de ejecución para cada caso de prueba y alamcenaremos los resultados. En la cual, se seleccionara los tres casos de prueba con el menor tiempo de ejecución y los tres casos que tardaron más tiempo para un análisis detenido.

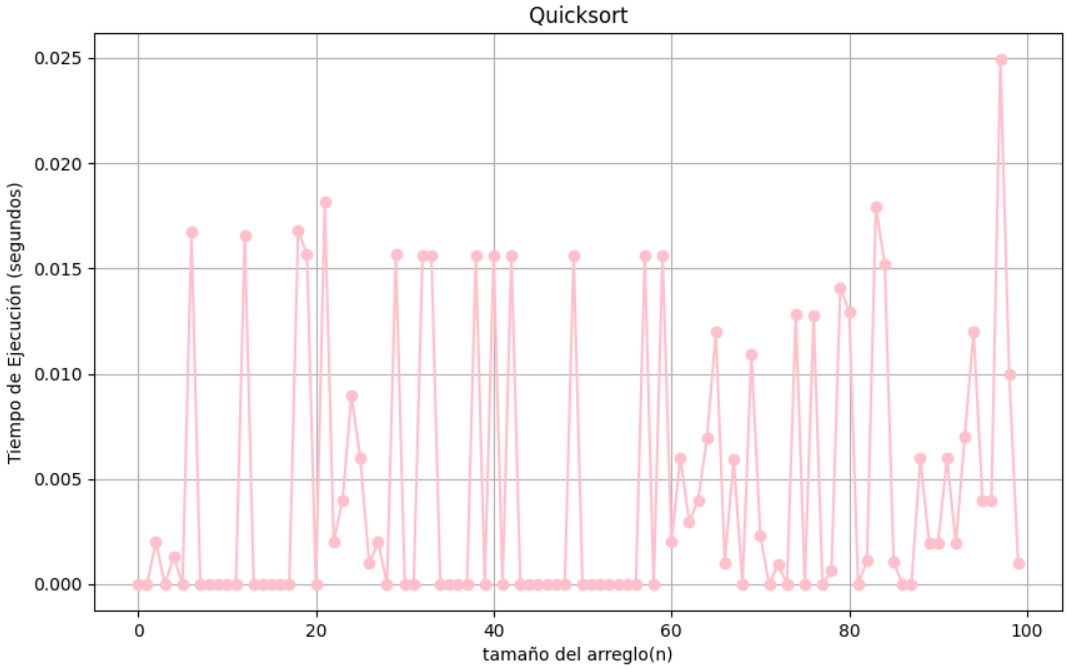
Procederemos a Ordenar los resultados por tiempo de ejecución, para ello utilizamos *lambda* lo cual nos ayuda a ordenar en orden ascendente según el tiempo de ejecución. Se mostrará los arreglos de los 3 casos de prueba con menor y mayor tiempo de ejecución en segundos(Por cada ejucución, será diferente su tiempo de ejecución de Quickort).

Por ultimo, utilizaremos la libreria ***import matplotlib.pyplot as plt*** para crear una gráfica donde nos muestre los resultados de ejecución y que nos guarde la gráfica como una imagen y que nos permita modificar el nombre de la grafica a guardar(*plt.savefig*).

A continuación mostraremos la gráfica generada muestra el tiempo de ejecución del algoritmo Quicksort para 100 casos de prueba diferentes, cada uno con un arreglo de números aleatorios de tamaño variable. La gráfica tiene el tiempo de ejecución en el eje vertical (Y) y el número de caso de prueba en el eje horizontal (X). Cada punto en la gráfica representa el tiempo que tardó Quicksort en ordenar un arreglo específico.

La altura del punto en la gráfica indica el tiempo de ejecución del algoritmo Quicksort para un caso de prueba específico. Puntos más bajos en la gráfica indican un menor tiempo de ejecución y, por lo tanto, una ejecución más rápida del algoritmo.

Figure 1: Ejemplo de rendimiento de QuickSort con 100 pruebas



Conclusión

En este estudio, se observó que el algoritmo Quicksort es altamente eficiente para ordenar arreglos de números aleatorios en una variedad de tamaños. Los tres casos de prueba que mostraron los tiempos de ejecución más bajos indican que el Quicksort puede manejar eficazmente arreglos de diversos tamaños y complejidades, mostrando su versatilidad y rapidez.

Por otro lado, los tres casos con mayor tiempo de ejecución sugieren que, aunque el Quicksort es generalmente rápido, puede haber situaciones particulares en las que su rendimiento se degrade, posiblemente debido a la estructura específica de los datos de entrada.

La gráfica permite visualizar cómo el algoritmo Quicksort se comporta en una variedad de situaciones. Si la mayoría de los puntos están en la parte inferior de la gráfica, indica que Quicksort es generalmente rápido para los casos de prueba proporcionados.

En resumen, el algoritmo Quicksort es una opción sólida y eficiente para el ordenamiento de arreglos en la mayoría de los casos. Sin embargo, al trabajar con conjuntos de datos muy específicos, puede ser necesario considerar otras alternativas o realizar ajustes en el algoritmo para garantizar un rendimiento óptimo en todas las situaciones.

Referencias

-http://mis-algoritmos.com/ordenamiento-rapido-quicksort
-https://www.youtube.com/watch?v=0tG8G-aRmSU
-https://www.youtube.com/watch?v=d0V6ibXxI5M
-http://aniei.org.mx/paginas/uam/CursoAA/curso$_a a_1 9.html$

-https://www.techiedelight.com/es/quicksort/