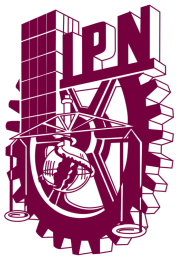


Instituto Politécnico Nacional

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA CAMPUS ZACATECAS

REPORTE PRÁCTICA 02: IMPLEMENTACIÓN Y EVALUACIÓN DEL ALGORITMO QUICKSORT

Docente: M. en C. Erika Sánchez-Femat

Lucia Iveth De La Vega Hernández
3CM2

22 Octubre 2023

Introducción

En esta práctica se comprenderá el comportamiento del método QuickSort, el cual es un algoritmo basado en la técnica de divide y vencerás, que permite, en promedio, ordenar n elementos en un tiempo proporcional a n log n. Se realizará la implementación de dicho método en Python.

Objetivo de la Práctica

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 arreglos aleatorios y visualicen los resultados mediante gráficos. Este enfoque les 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 de la Práctica

Implementación del Algoritmo QuickSort

Desarrollar el algoritmo de ordenamiento Quicksort en Python, teniendo en cuenta que la selección del pivote se debe hacer seleccionando la media del arreglo. Recuerda que este algoritmo es recursivo, por lo que tendrás que definir el caso base y el caso general.

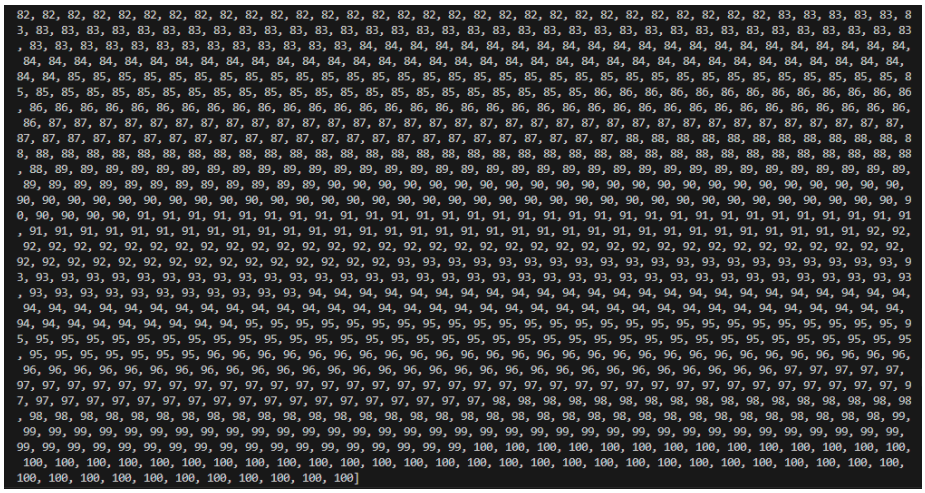
Nuestro caso base será cuando un arreglo contenga un solo elemento dentro de él, ya que de ese modo será el único y no tendrá más elementos para compararse. Esto sucede al definir el pivote o al final cuando termina de separar todos los elementos quedando arreglos individuales de un elemento.

El caso general será la comparación de los elementos restantes del arreglo con el pivote los cuales se irán agregando al arreglo de la izquierda o derecha según cumplan la condición de ser mayores o menores al pivote.

Generación de Casos de Prueba

Para comprender mejor el funcionamiento de este algoritmo, tendrás que generar 100 casos de prueba, es decir, tendrás que generar 100 arreglos de números aleatorios y de tamaño aleatorio. Con el fin de curbir la mayor cantidad de casos en los que este algoritmo se puede aplicar. Para este punto, tendrás que generar números aleatorios, esto lo puedes hacer con la librería random que ya viene incluida en la instalación de python. La puedes implementar usando: *import random*

A continuación se muestra una parte de los arreglos generados con números al azar de forma automática:



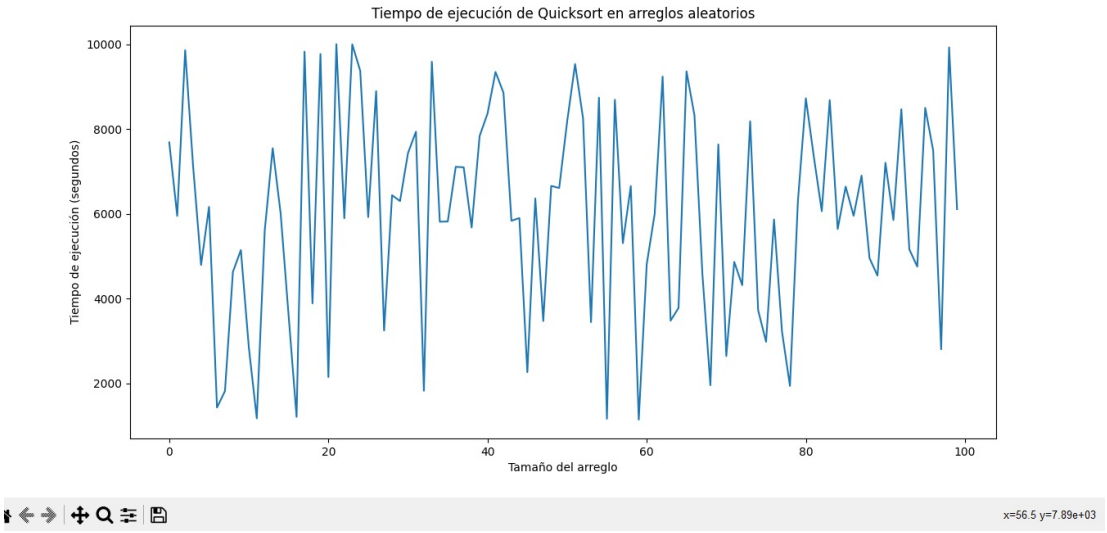
Medición del Tiempo

Deberás escribir una función que mida el tiempo de ejecución para cada caso de prueba.

Esto se realizó mediante la librería time y en un arreglo ir incluyendo los valores del tiempo que tardó en ejecutar la generación de cada uno de los cien arreglos.

Visualización de Datos

A continuación se muestra la gráfica de los 100 casos de prueba donde en el eje x tenemos el tamaño del arreglo y en el eje y el tiempo de ejecución de cada arreglo:



Entrega de Resultados

Deberás desarrollar un reporte en Overleaf con Portada, Introducción, Desarrollo, Conclusiones y Referencias. En la sección de desarrollo tendrás que explicar el procedimiento de implementación del algoritmo, cómo seleccionaste el pivote (si desarrollaste el punto 2.5) y agregar la gráfica con su explicación.

El código en python y el archivo PDF del reporte se deberán subir al repositorio que cada uno de los alumnos tiene en Github.

Conclusión

En conclusión el método QuickSort es una buena forma de ordenar los elementos de un arreglo, al poder utilizar la librería random y hacer un ciclo con 100 casos de prueba se pudo obtener una gráfica para visualizar los resultados según el tamaño del arreglo y el tiempo de ejecución. De esto se concluye que los arreglos que tienen menor tamaño son los que menor tiempo de ejecución toman, considerando qué tan desordenado genere los números.