Analsis y diseño de algoritmos

Unidad II
Analisis de Complejidad Computacional
Practica 02: Analisis de Casos Quicksort
Fernando Ruiz Correa 3M2

Docente:
M. en C. Erika Sanchez-Fermat

Unidad Profesional Interdedisciplinaria de Ingenieria Campus Zacatecas
Instituto Politecnico Nacional
Octubre 2023

1. 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.

2. Desarrollo de la ´Práctica

2.1. Implementacón del Algoritmo de 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.

- Para implementar el algoritmo de Quicksort, es necesario identificar el caso base, que es cuando la longitud del arreglo es igual o menor que 1. Es decir que la cantidad de elementos que se encuentran dentro del arreglo es de 1.
- posteriormente se define el pivote, en este caso, el pivote se obtiene con el concepto de media Aritmetica. La media aritmetica, se obtiene sumando todos los elementos dela arreglo y dividrlos entre la cantidad de elementos del arreglo. El pivote será igual al valor mas cercano a la media.
- Una vez seleccionado el pivote se divide el arreglo en dos partes, la parte menor al pivote y la parte mayor al pivote.
- Finalmente se realiza este proceso de manera recursiva hasta obtener un caso base.

2.2. Generación de Casos Prueba

- Para generar los casos prueba, será necesaria una funcion que se encargue de realizar 100 arreglos con una cantidad aleatoria de numeros, que vayan del 1 al 100. Para esto se utiliza la función generar arreglos aleatorios.

2.3. Medición de tiempo

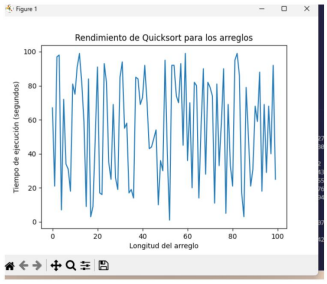
- Para medir los casos de prueba se utiliza la libreria timeit, que servira para medir el tiempo que tarda la función quicksort en ordenar cada arreglo que se genera.

,

2.4. Visualización de Datos

- Una vez que se realizó la ejecución de los 100 casos de prueba, se debe crear un gráfico que muestre el tiempo que tarda el algoritmo en ejecutar y ordenar cada uno de los casos prueba..

- Como se puede apreciar en la gráfica los resultados son muy variados, sin embargo los arreglos que mas tardaron en ordenarse son los que tenian a proximadamente una longitud de 20 elementos en su arrgelo, y los que tenian una longitud de 60 y 80 elementos. Por lo que el tiempo no depende de la cantidad de elementos que contenga el arreglo, si no de como se encuentran ordenados inicialmente

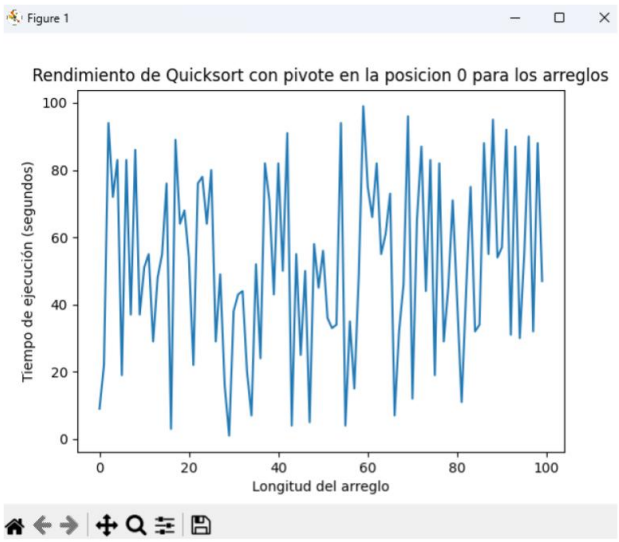


2.5. Medición de pivote

- Ahora, en lugar de utilizar el concepto de media aritmetica como pivote, se utiliza como pivote la posición 0 del arreglo, es decir la primera posición, esto con el objetivo de facilitar la división del arreglo principal.

- Una vez dividido el arreglo principal en dos arreglos se repite el proceso de selección de pivote hasta llegar al caso base antes mencionado.

- A continuación se muestra la gráfica obtenida por un nuevo algorimto con una selección de pivote diferentee, en está ocasión los resultados que muestra la gráfica son que tardó mas tiempo en ordenar los datos con 5, 53 y 67 elementos dentro de su



arreglo

3. Referencias

- QuickSort Algorithm Overview — Quick Sort (Article) — Khan Academy. (s. f.). Khan Academy. https://www.khanacademy.org/computing/computer-science/algorithms/quick-sort/a/overview-of-quicksort
- QuickSort Algorithm Overview — Quick Sort (Article) — Khan Academy. (s. f.). Khan Academy. https://www.khanacademy.org/computing/computer-science/algorithms/quick-sort/a/overview-of-quicksort