



Instituto Politecnico Nacional

Unidad Profesional Interdisciplinaria de
Ingenieria Campus Zacatecas

October 23, 2023

Ingenieria en Sistemas Computacionales

Practica 2:Implementación y Evaluación del
Algoritmo Quicksort

Maestro:

M. en C. Erika Sanchez Femat

Materia:

Analisis y Diseño de Algoritmos

Alumno:

Alejandro Ulloa Reyes

Grupo:

3CM2

1 Introduccion

Durante el inicio del semestre hemos analizado algunos tipos de algoritmos de ordenamiento como lo fue el ordenamiento burbuja, que si bien cumplia con su proposito de acomodar un arreglo no era el mas eficiente a lo que respecta, a continuacion, desglosaremos el como funciona otro algoritmo que cumple con la misma funcion pero de manera mas eficiente y eficaz **Quicksort**.

2 Quicksort

2.1 Logica y funcionamiento

Anteriormente se ah investigado sobre el algoritmo y sabemos que se trata del mas rapido hasta ahora en el mundo de la programacion pero ¿como es que tan eficiente?, sencillo, en primera instancia tendremos un arreglo o lista que tengamos en desorden con la cantidad de elementos que sean, despues de esto escogeremos un elemento pivote que en base a este se dividira el arreglo en dos secciones, en la parte izquierda estaran todos los elementos que sean menores que el pivote y del lado derecho todos aquellos que sean mayores, aunque no se ah mencionado todavia como seleccionar el pivote, bueno uno puede escoger cualquier elemento del arreglo como pivote pero eso no garantiza que sea la forma mas efectiva para el funcionamiento del quicksort ya que si bien conocemos este algoritmo implementado en su mejor caso nos dara un tiempo de $O(n\log n)$ y en su peor caso un tirmopo de forma $O(n^2)$, asi que para que funcione de manera adecuada el pivote sera escogido apartir de la media del arreglo.

Ahora si con nuestro pivote dividiremos el arreglo como se menciono anteriormente cabe destacar que el pivote y las otras dos secciones son arreglos que salieron del arreglo original, como el algoritmo trabaja de forma recursiva empezara a repetir el proceso con los arreglos que vaya dividiendo hasta completar el ordenamiento deseado. Finalmente quedaran n cantidad de arreglos cada uno con un elemento perteneciente del arreglo original, y asi todos esos arreglos individuales se juntan para formar solo un arreglo con la cantidad n original de elementos que habia en un inicio.

2.2 Implementacion

Para esta practica se requiere generar 100 arreglos aleatorios con una cantidad de elementos aleatoria ya sea entre 10 a 1000 para despues ser ordenados con el algoritno de **Quicksort** y contar el tiempo que se tardo en cada arreglo para asi hacer una grafica en relacion a los arreglos.

Como primera instancia colocaremos las librerias necesarias como random (para generar elementos aleatorios), time (para calcular los tiempos de cada arreglo en ser ordenados)y matplotlib (para graficar los arreglos en relacion al tiempo que tardaron en ser ordenados), despues iniciaremos con una funcion que defina cada arreglo de manera aleatoria con las caracteristicas deseadas y continuar con otra funcion solo para que imprima dichos arreglos junto con una etiqueta que indique que este sea el arreglo desordenado.

Ahora empieza la funcion para ordenar los arreglos con el **Quicksort** en cual para definir el caso base este no podra ser menor o igual a 1, calcula el pivote de los arreglos como se habia mencionado (calculando la media) para despues dividir el arreglo en tres partes, la izquierda para aquellos que sean menores que el pivote la derecha para los que sean mayores que el pivote y un ultimo para cuando sean igual que el pivote finalmente para la esta funcion es donde se aplicara recursivamente el alogoritmo de ordenamiento con las partes izquierda y derecha combiandolos para dejar un solo arreglo ordenado.

Otra funcion que utilizaremos y que es bastante escencial para este codigo es con la que mediremos los tiempos que se tardo en ordenar el algoritmo cada arreglo y para almacenar estos datos trabajaremos con dos listas una para guardar el tamaño de los arreglos y otra para el tiempo que tardo en ordenarlos.Como ultimo paso escribiremos los comandos necesarios para generar la grafica de tiempos de los arreglos ordenados.

3 Conclusiones y Resultados

Veremos pues que este ejercicio es muy completo ya que permite dominar temas como la recursividad multiples veces para casos alaetorios, asi como la medicion de tiempos para los algoritmos en casos particulares y lo mas destacable es el trabajar con nuevas librerias que descubren nuevas funciones que en lo personal desconocia que podia hacer el lenguaje como lo fue matplotlib dejando una perspectiva mas amplia de lo que puede llegar a hacerse con los lenguajes de programacion dando rienda suelta a los programadores que se van forjando y a los que se dedican profesionalmente a la creacion de algoritmos y codigos. Como resultados este codigo arroja 3 arreglos que fueron los mas lentos y 3 que fueron los mas rapidos en ser ordenados, pasa que con los tres primeros arreglos en entrar a ser ordenados fueron los mas rapidos ordenarse y los arreglos 70, 74 y 67 fueron los que mas tardaron aunque ninguno paso mas de 1 segundo en ser ordenado, lo que me demuestra lo efectivo que es el algoritmo **Quicksort**. A continuacion, se presentaran los 3 arreglos mas rapidos en ser ordenados y los 3 mas lentos.

3.1 Top 3 Arreglos mas rapidos en ser ordenados

1.-Arreglo desordenado: [101, 508, 622, 544, 598, 732, 493, 935, 828, 1, 975]

Arreglo ordenado: [1, 101, 493, 508, 544, 598, 622, 732, 828, 935, 975]

2.-Arreglo desordenado: [742, 121, 288, 854, 75, 238, 695, 630, 474, 862, 567, 122]

Arreglo ordenado: [75, 121, 122, 238, 288, 474, 567, 630, 695, 742, 854, 862]

3.-Arreglo desordenado: [405, 755, 85, 446, 620, 344, 260, 521, 839, 368]

Arreglo ordenado: [85, 260, 344, 368, 405, 446, 521, 620, 755, 839]

3.2 Top 3 Arreglos mas lentos en ser ordenados

1.-Arreglo desordenado: [579, 945, 117, 553, 771, 671, 966, 477, 109, 704, 319, 625, 890, 459, 496, 952, 466, 470, 234, 853, 909, 257, 526, 751, 766, 368, 563, 611, 921, 665, 731, 44, 275, 719, 458, 536, 489, 622, 978, 690, 918, 473, 588, 264, 599, 112, 268, 801, 421, 742, 995, 712, 513, 872, 177, 536, 567, 405, 572, 476, 800, 548, 205, 691, 281, 688, 707, 582, 698, 165, 452, 168, 37, 525, 39, 842, 388, 650, 649]

Arreglo ordenado: [37, 39, 44, 109, 112, 117, 165, 168, 177, 205, 234, 257, 264, 268, 275, 281, 319, 368, 388, 405, 421, 452, 458, 459, 466, 470, 473, 476, 477, 489, 496, 513, 525, 526, 536, 536, 548, 553, 563, 567, 572, 579, 582, 588, 599, 611, 622, 625, 649, 650, 665, 671, 688, 690, 691, 698, 704, 707, 712, 719, 731, 742, 751, 766, 771, 800, 801, 842, 853, 872, 890, 909, 918, 921, 945, 952, 966, 978, 995]

2.-Arreglo desordenado: [238, 198, 934, 631, 460, 879, 42, 327, 921, 68, 377, 272, 879, 747, 38, 555, 542, 480, 311, 131, 456, 857, 803, 683, 780, 593, 247, 491, 708, 347, 252, 699, 457, 942, 696, 483, 660, 967, 840, 110, 659, 433, 586, 575, 951, 113, 239, 562, 883, 402, 281, 217, 680, 123, 583, 127, 772, 334, 787, 182, 535, 403, 525, 858, 143, 897, 100, 640, 401, 776, 144, 952, 217, 805, 869, 207]

Arreglo ordenado: [38, 42, 68, 100, 110, 113, 123, 127, 131, 143, 144, 182, 198, 207, 217, 217, 238, 239, 247, 252, 272, 281, 311, 327, 334, 347, 377, 401, 402, 403, 433, 456, 457, 460, 480, 483, 491, 525, 535, 542, 555, 562, 575, 583, 586, 593, 631, 640, 659, 660, 680, 683, 696, 699, 708, 747, 772, 776, 780, 787, 803, 805, 840, 857, 858, 869, 879, 879, 883, 897, 921, 934, 942, 951, 952, 967]

3.-Arreglo desordenado: [627, 889, 327, 875, 221, 308, 808, 637, 967, 559, 868, 279, 237, 182, 462, 812, 926, 102, 782, 986, 842, 735, 982, 843, 410, 634, 523, 693, 825, 454, 435, 208, 266, 763, 430, 52, 900, 677, 668, 525, 583, 141, 938, 27, 740, 479, 367, 928, 695, 55, 410, 228, 16, 890, 702, 530, 276, 552, 805, 652, 889, 141, 166, 747, 793, 428, 400, 164, 578, 817, 138, 819, 704, 758, 342, 858, 703, 750, 112, 849, 60, 625, 582]

Arreglo ordenado: [16, 27, 52, 55, 60, 102, 112, 138, 141, 141, 164, 166, 182, 208, 221, 228, 237, 266, 276, 279, 308, 327, 342, 367, 400, 410, 410, 428, 430, 435, 454, 462, 479, 523, 525, 530, 552, 559, 578, 582, 583, 625, 627, 634, 637, 652, 668, 677, 693, 695, 702, 703, 704, 735, 740, 747, 750, 758, 763, 782, 793, 805, 808, 812, 817, 819, 825, 842, 843, 849, 858, 868, 875, 889, 889, 890, 900, 926, 928, 938, 967, 982, 986]

3.3 Grafica de tiempos

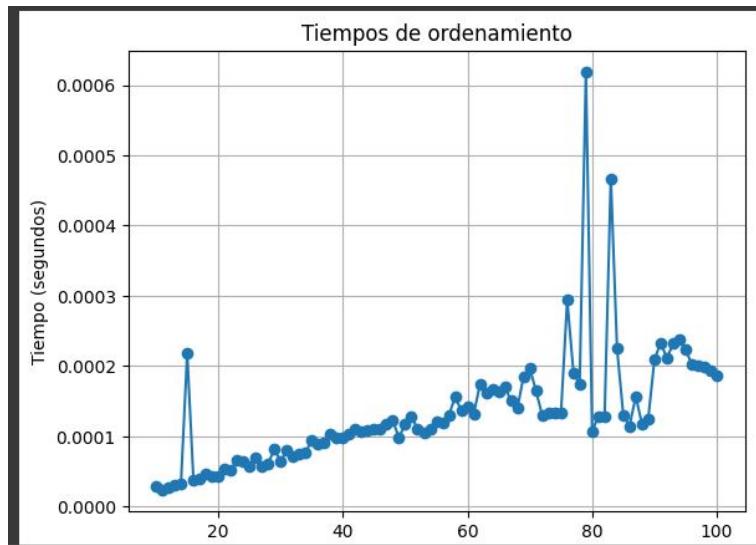


Figure 1: Grafica de tiempos de los arreglos