Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas

Jared De Loera Espinosa

3CM2

Análisis y diseño de algoritmos

Practica02

1. Introducción

Dentro de la programación, es comun encontrarse con la necesidad de ordenar datos, y para esto hay diferentes tipos de algoritmos que resuelven esta problematica. Cada uno maneja un concepto y una ejecucion particularmente diferente. Y en esta practica se revisara el funcionamiento teórico, concepto y el funcionamiento práctico del **Ordemaniento Quicksort**

1.1. ¿Qué es?

El ordenamiento de Quicksort es un algoritmo de ordenación ampliamente utilizado que se basa en el principio de "dividir y conquistar". Fue desarrollado por Tony Hoare en 1960 y ha demostrado ser una de las técnicas de ordenación más eficientes en la práctica. Y este, aparte de que suele ser muy eficiente, tambien puede ser complicado de aplicar debido al uso de la iteracion y la forma en la que manipula los datos.

1.2. ¿Cómo funciona?

El algoritmo de Quicksort opera en una lista de elementos, dividiéndola en dos subconjuntos: uno que contiene elementos menores que un valor elegido llamado "pivotez otro que contiene elementos mayores. Luego, repite este proceso en los subconjuntos, dividiéndolos aún más, hasta que la lista completa esté ordenada.

El proceso de Quicksort se puede resumir en los siguientes pasos:

1. **Selección del pivote:**
   Se elige un elemento de la lista como pivote. La elección del pivote puede afectar el rendimiento del algoritmo, y existen diversas estrategias para su selección, como elegir el primer elemento, el último elemento o un elemento al azar.

2. **División:** Se reorganiza la lista de manera que todos los elementos menores que el pivote estén a su izquierda y todos los elementos mayores estén a su derecha. En este punto, el pivote ocupa su posición final en la lista.

3. **Recursión:** Se aplica el mismo proceso de Quicksort a los subconjuntos resultantes, es decir, a la parte izquierda y derecha del pivote.

4. **Combinación:** A medida que se resuelven los subconjuntos, se combinan para obtener la lista ordenada completa.

1.3. Complejidad

En general un algoritmo de ordenamiento Quicksort suele tener una complejidad de O(nlog n) en otras palabras tiene un rendimiento muy eficiente y es capaz de ordenar una lista de elementos en un tiempo proporcional a n log n, donde "n.^{es} el número de elementos a ordenar. Esta es una de las razones por las que Quicksort es uno de los algoritmos de ordenación más utilizados en la práctica.

1.4. Casos

Como en la mayoria de los algoritmos de ordenamiento se pueden diferenciar diferentes casos a los que el algoritmo puede verse enfrentado y aqui estan los 3 principales de este algoritmo:

1. *Mejor caso:*
   - El mejor caso ocurre cuando el pivote elegido divide el conjunto de datos en dos subconjuntos de aproximadamente el mismo tamaño. O en otras palabras, cuando se logra hacer que el pivote este colocado en la mitad del arreglo
   - En este caso, Quicksort tiene un rendimiento óptimo, con una complejidad de tiempo de O(n log n), donde "n.^{es} el número de elementos a ordenar.
2. *Caso promedio:*
   - El caso promedio se refiere al rendimiento esperado de Quicksort en una variedad aleatoria de datos de entrada.
   - En promedio, Quicksort tiene una complejidad de tiempo de O(n $logn$), lo que lo hace muy eficiente.
3. *Peor caso:*
   - El peor caso ocurre cuando el pivote elegido siempre es el elemento más pequeño o el más grande en cada iteración.
   - En este caso, Quicksort tiene un rendimiento pobre, con una complejidad de tiempo de O(n^2), donde "n.^{es} el número de elementos a ordenar.

2. Desarrollo

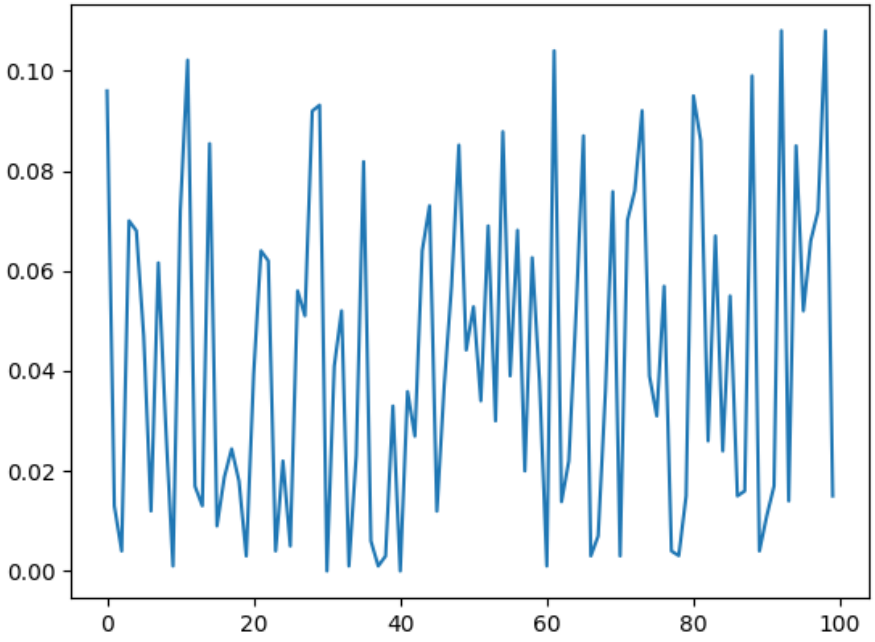
Ya dicho lo mas importante sobre el Quicksort, se va a implementar en un algoritmo.
La diferencia es que en este caso, para hacer un testeo relativamente masivo, se hará un algorimo que genere 100 arreglos, todos estos con un tamaño variable entre 10 y 10000 valores dentro de los mismos.
Al final se genera una grafica que contiene los arreglos y el tiempo que le tomó a la maquina resolverlos.

2.1. Resultados

Al final de la practica los resultados de los tiempos de ejecucion fueron los siguientes:


```
[0.0959775447845459, 0.013003349304199219, 0.004000663757324219, 0.07001566886901855, 0.0680243968963623, 0.04705739021301269 5, 0.012006044387817383, 0.06159663200378418, 0.029215335845947266, 0.0010001659393310547, 0.07207417488098145, 0.10213088989257812, 0.01697087287902832, 0.013022899627685547, 0.0854341983795166, 0.00900554656982421 9, 0.01873946189880371, 0.024427890777587 89, 0.0180101394653320 3, 0.0029995441436767 58, 0.0392668247222900 4, 0.0640144348144531 2, 0.0620141029357910 16, 0.0040009021759033 2, 0.0220048427581787 1, 0.0050013065338134766, 0.05600118637084961, 0.05102801322937012, 0.09196162223815918, 0.0931186676025390 6, 0.0, 0.040973663330078125, 0.05202531814575195, 0.0010001659393310547, 0.023147583007812 5, 0.0818462371826171 9, 0.0060048103332519 53, 0.0009999275207519531, 0.0030009746555151367, 0.0329887866973876 95, 0.0, 0.0358896255493164 06, 0.0269520282745361 33, 0.0640144348144531 2, 0.0730190277099609 4, 0.0119636058807373 05, 0.0370082855224609 4, 0.0570130348205566 4, 0.0851452350616455 1, 0.0441658496856689 45, 0.0528364181518554 7, 0.0340163707733154 3, 0.0689885616302490 2, 0.0300123691558837 9, 0.0878260135650634 8, 0.0390088558197021 5, 0.0681207180023193 4, 0.0200076103210449 22, 0.0626404285430908 2, 0.0380156040191650 4, 0.0010006427764892 578, 0.1040005683898925 8, 0.0138559341430664 06, 0.0220041275024414 06, 0.0521385669708251 95, 0.0870003700256347 7, 0.0030076503753662 11, 0.0069961547851562 5, 0.0359988212585449 2, 0.0758368968963623, 0.0030105113983154297, 0.07014036178588867, 0.07600259780883789, 0.09201431274414062, 0.0390071868896484 4, 0.0309929847717285 16, 0.0569281578063964 84, 0.0040011405944824 22, 0.0030727386474609375, 0.01500654220581054 7, 0.0949096870422363, 0.0860095024108886 7, 0.0259945392608642 58, 0.0669875144958496 1, 0.0239975452423095 7, 0.0549124526977539, 0.0150101184844970 7, 0.0160079002380371 1, 0.0989906787872314 5, 0.0040004253387451 17, 0.0110065937042236 33, 0.0169987678527832 03, 0.1080119609832763 7, 0.0140066146850585 94, 0.0850480651855469, 0.051998138427734375, 0.06601834297180176, 0.07195377349853516, 0.10800480842590332, 0.015007734298706055]
```


Y el resultado del contedo de datos graficado por el programa fueron estos:



Es necesario recalcar que la razon por la que los resultados de los tiempos de ejecucion se realizan todo dentro de un solo arreglo para facilitar la graficacion de los datos, ya que funcionan con la libreria matplotlib.

3. Conlcusión

El algoritmo de Quicksort destaca por su eficiencia en el caso promedio, con una complejidad de tiempo de O(n $logn$), lo que lo convierte en una elección sólida para la ordenación de datos a gran escala. Sin embargo, debe manejarse con cuidado en el peor caso, que puede alcanzar una complejidad de tiempo de O(n^2), pero con implementaciones cuidadosas y estrategias de selección de pivotes apropiadas, este escenario se puede evitar en la mayoría de las situaciones.
La capacidad de Quicksort para dividir eficazmente la lista y resolver subconjuntos hace que sea un algoritmo poderoso para el procesamiento de datos en la programación y la informática en general.

4. Referencias

- *PEÑA M., Ricardo. Diseño de programas. Formalismo y abstracción. Prentice-Hall, 1998*
- *WIRTH, Niklaus. Algoritmos y Estructuras de Datos. Pentice Hall, 1987.*
- *MacKay, David (December 2005). "Heapsort, Quicksort, and Entropy". Archived from the original on 1 April 2009.*