INSTITUTO POLITÉCNICO NACIONAL

Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas.

MATERIA:

- Ánalisis y Diseño de Algoritmos

Reporte 01:

- QuickSort

DOCENTE:

- Erika Sánchez Femat

Alumno:

- Gael García Torres

Introducción

En este Reporte, exploraremos los conceptos fundamentales de Quicksort, su definicion , cómo funciona y su rendimiento en diferentes situaciones. Además, analizaremos los casos de mejor, peor y promedio para comprender la complejidad temporal del algoritmo.

Método de QuickSor:

QuickSort, método de ordenamiento rápido. El método de ordenamiento QuickSort es actualmente el más eficiente y veloz de los métodos de ordenación interna. Este método es una mejora sustancial del método de intercambio directo y recibe el nombre de QuickSort por la velocidad con que ordena los elementos del arreglo. 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". Fue desarrollado por el científico de la computación británico Tony Hoare en 1960 y se ha convertido en uno de los algoritmos de ordenación más eficientes en la práctica.

Como funciona:

Éste algoritmo se basa en el principio de divide y conquistarás. Resulta más fácil ordenar listas pequeñas que una grande, con lo cual irá descomponiendo la lista en dos partes y ordenando esas partes.

Para ésto utiliza la recursividad.

• Dada una lista, elegir uno de sus elementos, que llamamos pivote

• Dividir la lista en dos sublistas:

• una con los elementos "menores"

• otra con los elementos "mayores"

• Ordenar recursívamente ambas sublistas

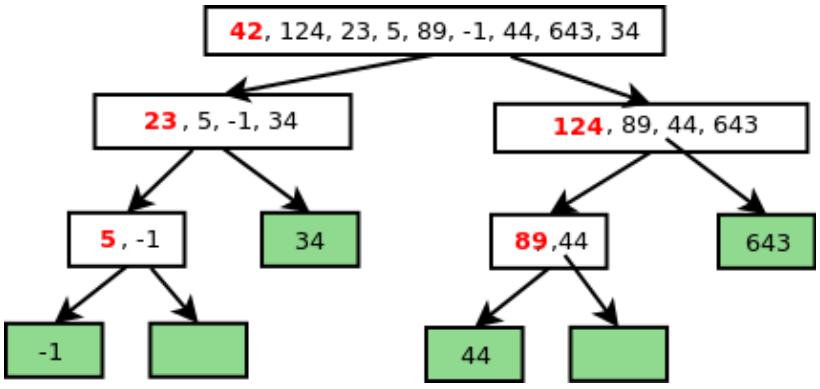
• Armar la lista resultado como: menoresOrdenados + pivote + mayoresOrdenados

La mejora es que, como al final de cada iteración el elemento mayor queda situado en su posición, ya no es necesario volverlo a comparar con ningún otro número, reduciendo así el número de comparaciones por iteración.

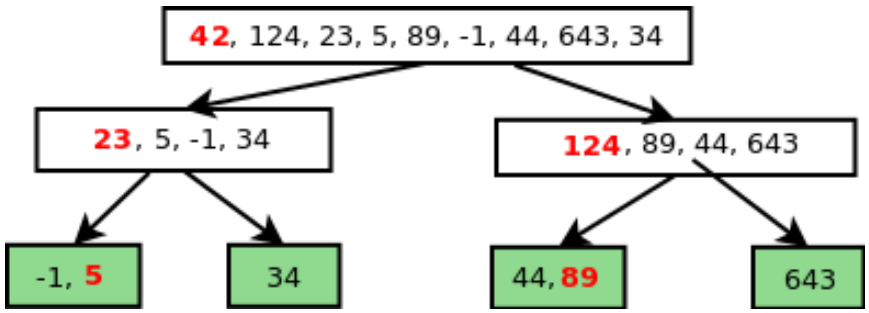
Veamos un ejemplo para ordenar la siguiente lista:
[42, 124, 23, 5, 89, -1, 44, 643, 34]

En nuestro ejemplo vamos a seleccionar el primer elemento de la lista como el pivote.

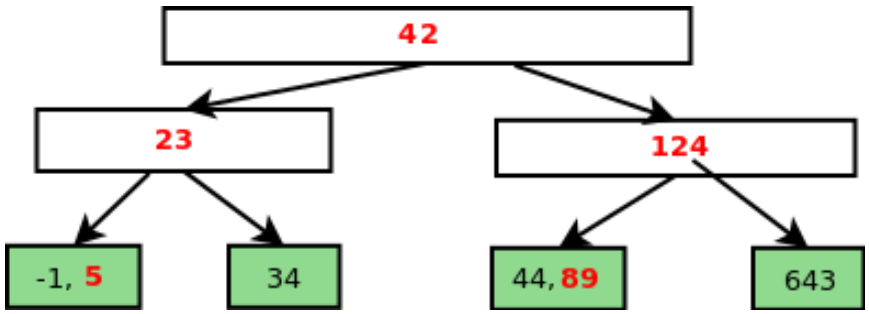
Veamos entonces, paso a paso se va a ir formando este árbol de llamados recursivos. En rojo mostramos el pivot. A izquierda las sublistas menores, y a la derecha la de los mayores.



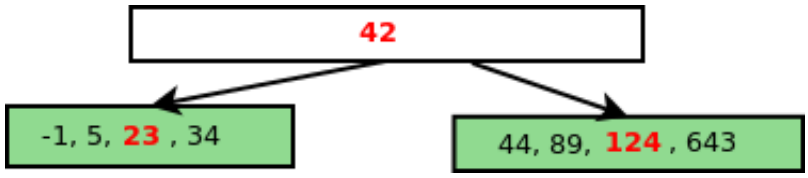
En verde están los nodos "caso base", es decir lista de un único elemento o bien lista vacía.
Entonces estos nodos se van a resolver fácil y se van a concatenar con los pivots de arriba. Así empezamos a volver en los llamados recursivos.



Si sacamos las listas originales de cada nodo y solo dejamos los pivots, queda más claro:



Ya tenemos todo el tercer nivel resuelto así que cada par va a retornar para concatenarse entre sí con el pivot en el medio.



Ya casi estamos. Tenemos dos listas ordenadas, la de los menores y la de los mayores, y el primer pivot. Concatenamos y eso nos da el resultado final:

[-1, 5, 23, 34, 42, 44, 89, 124, 643]

Complejidad:

Su tiempo de ejecución promedio es O(n log (n)), siendo en el peor de los casos O(n2), caso altamente improbable. El hecho de que sea más rápido que otros algoritmos de ordenación con tiempo promedio de O(n log (n)) (como SmoothSort o HeapSort) viene dado por que QuickSort realiza menos operaciones ya que el método utilizado es el de partición.

Mejor caso:

El mejor caso de quick sort ocurre cuando X divide la lista justo en el centro. Es decir, X produce dos sublistas que contienen el mismo número de elementos. En este caso, la primera ronda requiere n pasos para dividir la lista original en dos listas. Para la ronda siguiente, para cada sublista, de nuevo se necesitan n/2 pasos (ignorando el elemento usado para la división). En consecuencia, para la segunda ronda nuevamente se requieren $2(n/2) = n$ pasos. Si se supone que $n = 2^P$, entonces en total se requieren $p * n$ pasos. Sin embargo, $p = log2$ n. Así, para el mejor caso, la complejidad temporal del quick sort es O(n log n).

La recurrencia para el mejor caso se expresa como:
T(n) = 2 * T(n/2) + O(n)

Peor caso:

El peor caso del quick sort ocurre cuando los datos de entrada están ya ordenados o inversamente ordenados. En estos casos, todo el tiempo simplemente se está seleccionando el extremo (ya sea el mayor o el menor). Por lo tanto, el número total de pasos que se requiere en el quick sort para el peor caso es:

La recurrencia en el peor caso se expresa como: T(n) = T(n-1) + O(n)

Promedio Caso:

El caso promedio de Quicksort se encuentra entre el mejor y el peor caso. Para analizarlo, se utilizan conceptos de probabilidad y se asume que la elección del pivote es aleatoria y que todas las permutaciones de los elementos son igualmente probables. La recurrencia promedio se expresa de manera similar a la del mejor caso, lo que lleva a una complejidad promedio de O(n log n).

Conclusión:

Quicksort es un eficiente algoritmo de ordenación utilizado en informática que opera dividiendo una lista de elementos en subgrupos más pequeños, ordenándolos y luego combinándolos.

En resumen, Quicksort es un algoritmo de ordenación eficiente que se basa en la estrategia de dividir y conquistar. Su complejidad promedio de O(n log n) lo hace una opción preferida para ordenar listas en la mayoría de las situaciones. Sin embargo, es importante considerar la elección del pivote y la implementación para lograr un rendimiento óptimo en la práctica.

Referencias:

http://aniei.org.mx/paginas/uam/CursoAA/curso_aa_19.html#:~:text=El%20algoritmo%20quick%20sort%20se,Los%20resultados%20se%20unen%20despu%C3%A9s.

https://www.genbeta.com/desarrollo/implementando-el-algori

https://issuu.com/ernestoalonsopestanajimenez/docs/metodo_burbuja_optimizado.pptx#:~:text=Conclusiones%20El%20m%C3%A9todo%20Burbuja%20Optimizado,utilizando%20menos%20recursos%20del%20computador.

https://sites.google.com/site/programacioniiuno/temario/unidad-6---anlisis-de-algoritmos/algoritmo-quicksort