



INSTITUTO POLITÉCNICO NACIONAL
Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas.

MATERIA:
Análisis y Diseño de Algoritmos

DOCENTE:
Erika Sanchez Femat

PRACTICA:
○ QuickSort

ALUMNO:

- Fernando Antonio García Ruiz

INTRODUCCION

El algoritmo Quicksort, desarrollado por Tony Hoare en 1960, es uno de los algoritmos de ordenamiento más eficientes y ampliamente utilizados en la programación y la informática. Su popularidad radica en su capacidad para ordenar una lista o un arreglo de datos de manera muy eficiente. Quicksort se basa en el principio de "dividir y conquistar", lo que significa que divide un conjunto de datos en partes más pequeñas, las ordena y luego las combina en orden.

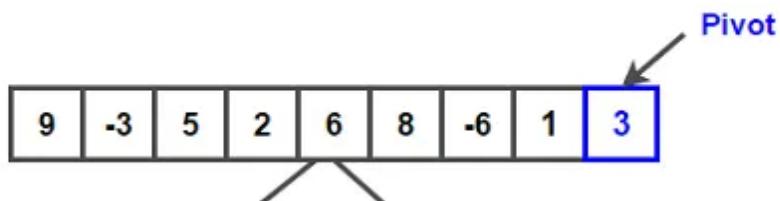
El corazón de Quicksort radica en la elección del "pivote", un elemento de la lista que se utiliza para dividir la lista en dos subconjuntos: uno que contiene elementos menores o iguales al pivote y otro con elementos mayores al pivote. Esta división se realiza de manera recursiva, lo que significa que se aplica repetidamente a las sublistas resultantes hasta que toda la lista esté ordenada.

La eficiencia de Quicksort se debe a su capacidad para ordenar en promedio en tiempo $O(n \log n)$, lo que lo hace ideal para ordenar grandes conjuntos de datos. Sin embargo, es importante señalar que el rendimiento puede variar según la elección del pivote y el estado inicial de los datos. En los casos menos favorables, la complejidad de Quicksort puede llegar a ser $O(n^2)$, pero estrategias de selección de pivote inteligentes, como la elección de la mediana de tres, pueden mitigar este problema.

¿COMO FUNCIONA?

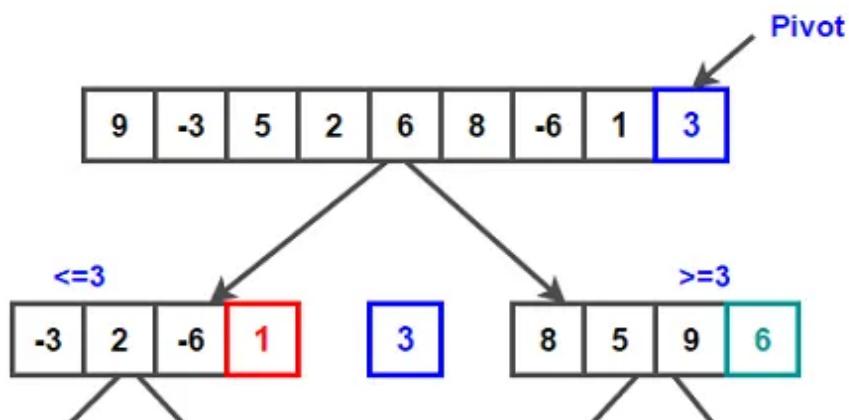
1.-Selección del pivote:

El primer paso implica elegir un elemento de la lista como pivote. El pivote actúa como una referencia para dividir la lista en dos subconjuntos: uno que contiene elementos menores o iguales al pivote y otro con elementos mayores al pivote.



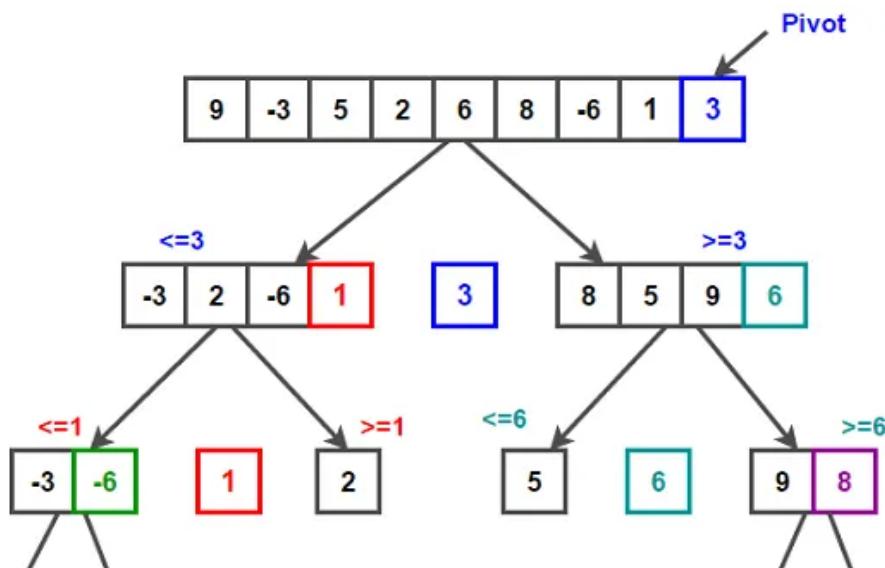
2.-Partición:

A continuación, el algoritmo recorre la lista y reorganiza los elementos de modo que aquellos menores o iguales al pivote se encuentren en un lado, mientras que los elementos mayores al pivote estén en el otro lado. Este proceso se denomina "partición".



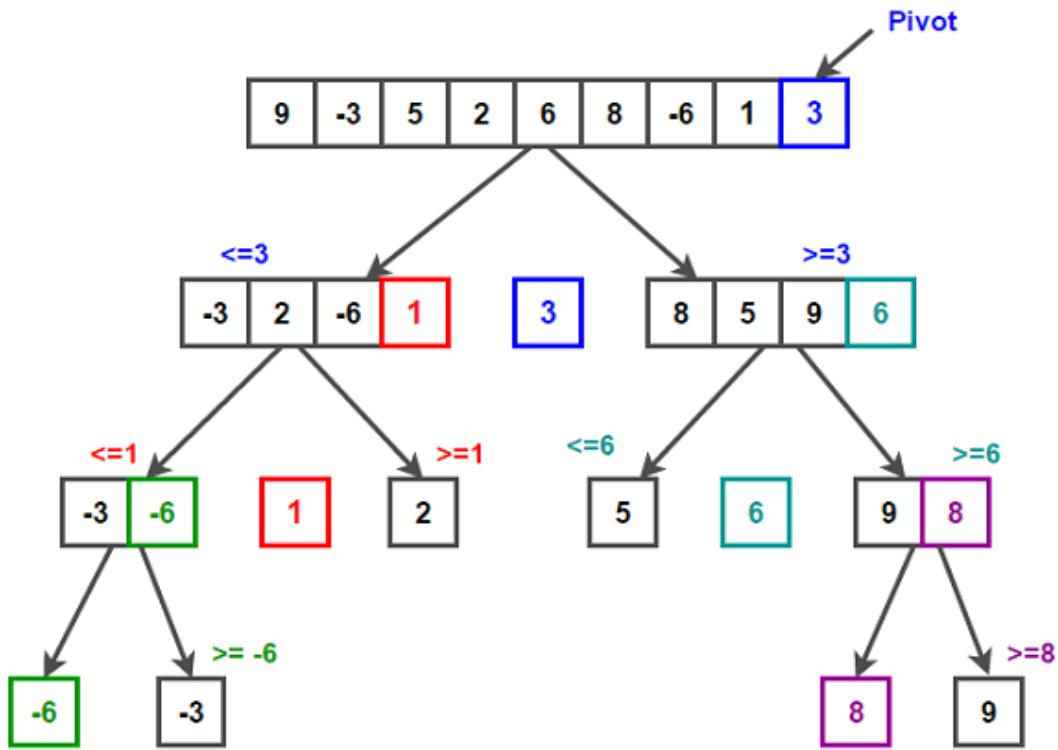
3.-Recursión:

Una vez que se ha realizado la partición, Quicksort se aplica de manera recursiva a las dos sublistas resultantes: la lista de elementos menores o iguales al pivote y la lista de elementos mayores al pivote. Esto significa que se repiten los pasos 1 y 2 para estas sublistas hasta que toda la lista esté ordenada.



4.-Combinación:

A medida que las llamadas recursivas se desenrollan, las sublistas se combinan de manera que todos los elementos menores estén a la izquierda del pivote y todos los elementos mayores a la derecha. El pivote ocupa su posición final en la lista.



resultado final es que la lista original se encuentra completamente ordenada, con los elementos en la posición correcta según el criterio de orden especificado. En el caso del Quicksort, este algoritmo tiene una eficiencia promedio de $O(n \log n)$, lo que lo convierte en uno de los métodos de ordenamiento más rápidos y eficientes disponibles.

COMPLEJIDAD

Puede ser complejidad $O(n \log n)$ ú $O(n^2)$

ANALISIS DE CASOS

Mejor caso:

El mejor caso ocurre cuando el pivote elegido en cada partición divide la lista en dos subconjuntos de tamaño aproximadamente igual. En este escenario ideal, el Quicksort tiene un rendimiento óptimo, y su complejidad es de $O(n \log n)$, donde "n" es el número de elementos a ordenar. Esta eficiencia se debe a la naturaleza de "dividir y conquistar" del algoritmo, que divide la lista en partes equitativas y luego las combina eficientemente.

Caso promedio: El caso promedio se refiere al rendimiento típico de Quicksort en una variedad de conjuntos de datos aleatorios. En promedio, Quicksort tiene una complejidad de $O(n \log n)$. La elección aleatoria de pivotes y otras estrategias, como la elección de la mediana de tres, ayudan a mitigar el riesgo de caer en el peor caso y permiten que el algoritmo mantenga su eficiencia en una amplia gama de situaciones.

Peor caso: El peor caso se da cuando el pivote elegido en cada partición es extremadamente inadecuado y divide la lista en un subconjunto grande y otro pequeño. En este escenario, la complejidad del Quicksort puede ser de $O(n^2)$, lo que lo convierte en un algoritmo ineficiente para grandes conjuntos de datos. El peor caso suele ocurrir cuando la lista ya está ordenada o está ordenada en orden inverso, y el pivote se elige como el primer o el último elemento en cada iteración.

CONCLUSION

En conclusión, el método de Quicksort es un algoritmo de ordenamiento altamente eficiente que se basa en el principio de "dividir y conquistar" para organizar una lista o un arreglo de datos de manera rápida y efectiva. Se destaca por su capacidad para dividir la lista en subconjuntos más pequeños, ordenarlos y luego combinarlos en orden, lo que lleva a una complejidad promedio de $O(n \log n)$, haciéndolo ideal para ordenar grandes conjuntos de datos.

Sin embargo, es importante tener en cuenta que la eficiencia de Quicksort puede verse afectada por la elección del pivote y el estado inicial de los datos. En el peor caso, la complejidad puede ser de $O(n^2)$, pero estrategias inteligentes de selección de pivote, como la elección de la mediana de tres, pueden mitigar este problema.

REFERENCIAS

<https://www.genbeta.com/desarrollo/implementando-el-algoritmo-de-quicksort-en-javascript>

<https://latteandcode.medium.com/algoritmos-de-ordenaci%C3%B3n-quicksort-en-javascript-f064db39e6ad>

http://aniei.org.mx/paginas/uam/CursoAA/curso_aa_19.html

<https://numerentur.org/quicksort/>