Comparativa Quicksort

*Nota: Los subtítulos no son capturados en Xplore y no deben usarse

Alexis Raciel Ibarra Garnica
Facultad de Informática
Universidad Autómona de Querétaro
Santiago de Querétaro, Qro, México
direccion.correo@email.com

Pablo Natera Bravo
Facultad de Informática
Universidad Autómona de Querétaro
Santiago de Querétaro, Qro, México
pablonatera16@gmail.com

***Abstract*—In this paper we implemented QuickSort in C sharp along with other in-place sorting algorithms such as BubbleSort, Flag BubbleSort, SelectionSort and InsertionSort. This with the intention of comparing performance while maintaining $O(1)$ spacial complexity. We concluded that of all the in-place sorting algorithms, QuickSort is the best option given that the pivot is chosen adequately.**

***Index Terms*—QuickSort, Divide and Conquer, Big O**

I. Introducción

A. Antecedentes Históricos

El algoritmo *QuickSort*, fue desarrollado por Tony Hoare en 1959 mientras era estudiante de ciencias de la computación en la Universidad Estatal de Moscú, surgió de la necesidad de ordenar listas de palabras para traducirlas del ruso al inglés. Su propuesta superó en eficiencia al método de *Insertion Sort*, haciendo uso del nuevo concepto de particiones e implementándose inicialmente en Mercury Autocode. Posteriormente, al regresar a Inglaterra, Hoare adaptó su idea para mejorar el algoritmo de *Shellsort*, lo que lo llevó a publicar QuickSort en 1961, y un año más tarde una versión mejorada. QuickSort está basado en el paradigma de *Divide and Conquer*, creando particiones recursivas en el arreglo hasta llegar a un caso base. [1]

B. Divide and Conquer

El QuickSort es un algoritmo que usa la técnica de *Divide and Conquer* [1], en donde

C. Algoritmo

Los pasos principales que sigue el algoritmo son:

1) Si la longitud del arreglo es menor a dos, se retorna el elemento directamente (caso base).
2) Si la longitud es dos o más, se selecciona un pivote (existen distintas técnicas para ello).
3) El arreglo se reordena colocando los elementos menores al pivote a la izquierda y los mayores a la derecha.
4) Los pasos anteriores se aplican recursivamente a cada subarreglo hasta alcanzar el caso base.

Identifique la agencia de financiación aplicable aquí. Si no hay, elimine esta línea.

D. complejidad y pivote

Escoger bien el pivote... En cuanto a la complejidad, el peor caso ocurre cuando las particiones dividen el arreglo de manera muy desigual (por ejemplo, $n-1$ y 0 elementos), resultando en una complejidad de $O(n^2)$. El mejor caso se logra cuando el pivote divide el arreglo en dos mitades balanceadas en cada paso, alcanzando una complejidad de $\Theta(n \log n)$. En la práctica, el caso promedio se acerca bastante al mejor caso debido a que la recurrencia converge a $n \log n$, por lo que QuickSort es considerado uno de los algoritmos de ordenamiento más eficientes.

E. Algoritmos de ordenamiento in-place

II. Metodología

Aquí debe describir los métodos, enfoques o procedimientos utilizados en su investigación.

III. Resultados

Presente los hallazgos y datos recopilados de su metodología. Incluya figuras y tablas como se muestra a continuación.

IV. Conclusiones

Resuma las conclusiones clave extraídas de los resultados y discuta las implicaciones de su trabajo.

El quicksort es el mejor de los algoritmos de in-place dado que se escoga bien el pivote.

Referencias

References

[1] L. Khreisat, "Quicksort a historical perspective and empirical study," 07 2008.