Proyecto final de Estructuras de Datos Abstractas y algoritmos para Ingeniería

Tema: Quicksort

Estudiante: Steven Andrés Mora Barboza

Carné: B95109

Fecha de entrega: 14 de diciembre del 2020

Escuela de Ingeniería Eléctrica

II Semestre del año 2020

Contenido

Introducción

Quicksort, 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. Su creador es el científico Británico Charles Anthony Richard Hoare, mientras se encontraba en la Universidad Estatal de Moscú, en la Unión Soviética. “Hoare trabajó en un proyecto de traducción automática para el Laboratorio Nacional de Física (Reino Unido). Desarrolló el algoritmo para poder ordenar las palabras a ser traducidas, para volverlas más fácil de coincidir con un diccionario ya ordenado de ruso a inglés. El algoritmo original es recursivo, pero se utilizan versiones iterativas para mejorar su rendimiento [2] “

Discusión

Método Quicksort, ¿En qué consiste?

Consiste en ordenar listas pequeñas provenientes de una más grande a partir de un punto de referencia llamado pivote, con lo cual la lista original se descompone en dos partes y esas dos se descomponen recursivamente hasta que quede un solo número en cada lista creada.

Hay muchas versiones diferentes de Quicksort que seleccionan el pivote de diferentes maneras:

- Eligiendo el primer elemento de la lista como pivote.
- Eligiendo el último elemento de la lista cómo pivote (implementado a continuación).
- Eligiendo un elemento aleatorio de la lista cómo pivote.
- Eligiendo la mediana como pivote.

¿Cómo funciona?

1. Dada una lista, se elige uno de sus elementos, el cual se llama pivote. (En este caso, se tomará el último elemento de la lista siempre).

2. Se divide la lista principal en dos sublistas:

    - Una con los elementos mayores al valor del pivote seleccionado.
    - Otra con los elementos mayores al valor del pivote seleccionado.

3. De cada nueva sublista, se toma un valor cómo pivote, en este caso el último, y se hacen otras dos nuevas sublistas, las cuales tienen la misma forma que el punto 2

4. Recursivamente se aplican los mismos puntos (del 1 al 3), hasta obtener un único número
5. Armar una lista donde se encuentren primero los menores ordenados, y luego los mayores ordenados.

Y con este procedimiento en resumen, se obtiene la lista ordenada de menor a mayor.

¿Para qué se utiliza?

Este algoritmo es utilizado para el ordenamiento de valores numéricos, de menor a mayor, además, “Quicksort es actualmente el más eficiente y veloz de los métodos de ordenación interna.[4]”

En términos de complejidad, este algoritmo se puede comportar de diversas maneras. Por ejemplo; si en el mejor de los casos el pivote se encuentra en el centro de la lista, dividiéndola en dos listas de igual tamaño, el orden de complejidad del algoritmo es de 0(n*logn).
Si el algoritmo termina en un extremo de la lista, la cual es exactamente lo que se hace tanto en el código cómo ejemplo en la figura 1, su orden de complejidad es de $0(n^2)$. [3]

Aplicaciones del algoritmo Quicksort:

- Para ordenar una lista de números/nombres.
- Utilización antes de implementar una búsqueda binaria.
- Utilizado como el método de ordenamiento en tarjetas gráficas.

Ventajas y desventajas del algoritmo:

Ventajas:

- Necesita de pocos recursos en comparación a otros métodos de ordenamiento.
- En la mayoría de los casos, se requiere aproximadamente N log N operaciones.
- El ciclo interno es muy corto.
- No se requiere de espacio adicional durante la ejecución (in-place processing).

Desventajas:

- Se complica la implementación si la recursión no es posible.
- En el peor caso, su complejidad es N^2
- Un simple error en la implementación puede pasar sin detección, lo que provocaría un muy mal rendimiento.
- No es útil para aplicaciones de entrada dinámica, donde se requiere reordenar una lista de elementos con nuevos valores.
- Se pierde el orden relativo de elementos idénticos.

A continuación, un ejemplo gráfico de la implementación de Quicksort:

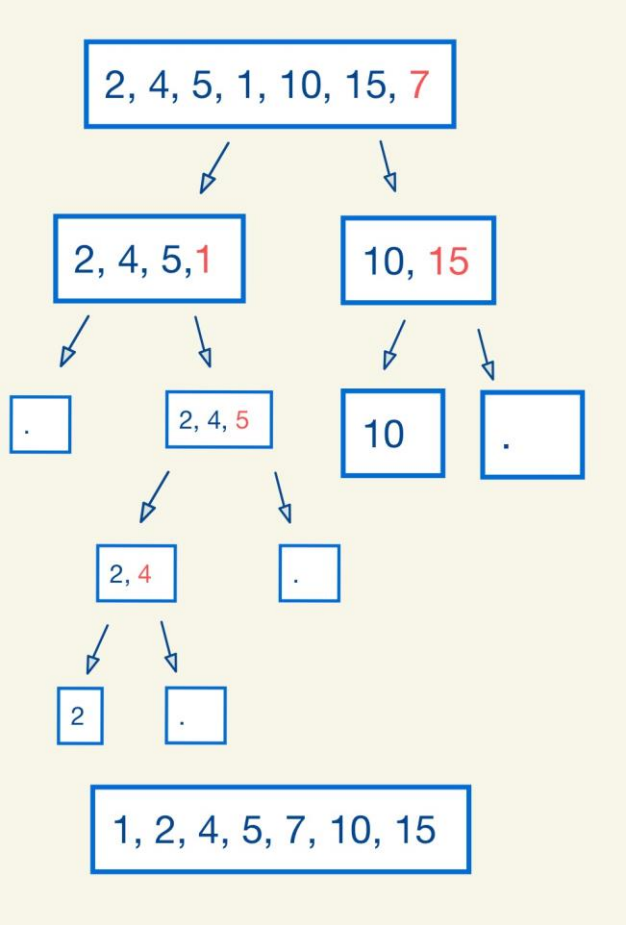

Figura 1. Ejemplo del ordenamiento de Quicksort.

En la figura 1 se muestra inicialmente una lista, la cual sus valores están ordenados al azar, luego, se toma cómo pivote el último elemento de la lista para ejecutar el algoritmo, y se divide la primera lista en dos sublistas; las listas de la izquierda siempre son las de los menores valores, los de la derecha son los mayores y el número en color rosado, es el pivote.

Finalmente, se juntan los valores marcados con “.” de “izquierda a derecha”, para así obtener la lista ordenada, hay que tomar en cuenta que justo en el centro de ambas primeras sublistas, se debe encontrar el número 7.

Código de Quicksort en Python por Mohit Kumra:

```
# La función toma el último elemento como el pivote

def quickSort(lista,pequeno,grande):
    if pequeno < grande:
        pi = partidor(lista,pequeno,grande)
        quickSort(lista, pequeno, pi-1)
        quickSort(lista, pi+1, grande)

def partidor(lista,pequeno,grande):
    i = ( pequeno-1 )         # índice del pequeño
    pivote = lista[grande]     # pivote

    for j in range(pequeno , grande):
        if   lista[j] < pivote:
            i = i+1
            lista[i],lista[j] = lista[j],lista[i]

    lista[i+1],lista[grande] = lista[grande],lista[i+1]
    return ( i+1 )

lista = [10, 7, 80, 8, 9, 1, 5, 90]
n = len(lista)
quickSort(lista,0,n-1)
print ("La lista ordenada seria:")
for i in range(n):
    print ("%d" %lista[i])
```

Código hecho por Mohit Kumra[1]

Dando como resultado la lista ordenada (mostrada individualmente) en la figura 2.

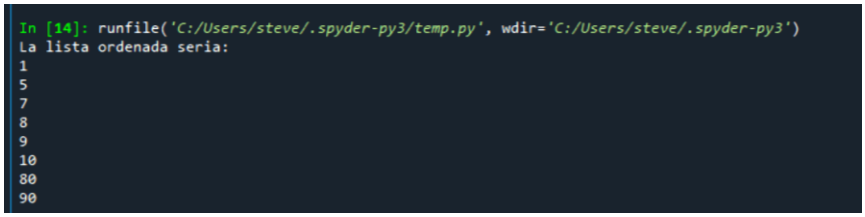

Figura 2. Ejecución del código en Python.

Explicación rápida del código:

Inicialmente se tiene una lista, compuesta de números al azar, luego se establece el tamaño de la lista con la variable n, que al ejecutar por primera vez la función Quicksort, se le resta 1, debido que la posición de cada valor vale n-1 porque comienza desde lista[0], y al querer usar cómo pivote más adelante al último valor, se utiliza n-1,la función Quicksort se encarga de llamar la función partidor, la cual ordena los valores menores y mayores para tenerlos en diferentes bloques, para finalmente, juntarlos y obtener los valores ordenados de menor a mayor.

Conclusiones

El algoritmo Quicksort es una forma muy rápida y eficaz para el ordenamiento de números y diversos usos cómo el ordenamiento en tarjetas gráficas.
Si se utiliza los valores de los extremos de la lista de valores, este podría ser más lento debido a su complejidad de $0(N^2)$.
Si se utiliza el ordenamiento más rápido, la complejidad sería de aproximadamente N log N, lo cual implica dividir la lista en dos sublistas de un tamaño igual.

Referencias

[1]M. Kumra, "QuickSort - GeeksforGeeks", *GeeksforGeeks*, 2020. [Online]. Disponible: https://www.geeksforgeeks.org/quick-sort/. [Accesado: 10- Dec- 2020].

[2]G. Sauceda and O. Ramirez, "Quicksort", *Gsauceda1*. [Online]. Disponible: https://gsauceda1.wixsite.com/quicksort/historia. [Accesado: 10- Dec- 2020].

[3]"QuickSort - EcuRed", *Ecured.cu*. [Online]. Disponible: https://www.ecured.cu/QuickSort. [Accesado: 10- Dec- 2020].

[4]"Método Quick Sort", *Cidecame.uaeh.edu.mx*. [Online]. Disponible: http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro9/mtodo_quick_sort.html#:~:text=Quicksort%20es%20un%20algoritmo%20basado,los%20m%C3%A9todos%20de%20ordenaci%C3%B3n%20interna. [Accesado: 10- Dec- 2020].