UT9_PD3

1)

Error en línea 3:

- permutar V[i]..V[j] de forma que, para alguna k tal que i+1 <= k < j, V[i].clave...V[k-1] .clave < Pivote y V[k].clave...V[j] .clave >= Pivote

En cada invocación recursiva del algoritmo quicksort(i,j), se elige un pivote y se reordenan los elementos del subarreglo V[i]..V[j] de forma que todos los elementos menores o iguales al pivote se sitúan a su izquierda, y todos los elementos mayores o iguales al pivote se posicionan a su derecha. Este proceso se repite para cada subarreglo recursivo.

2)

Selecciona el mayor de los 2 primeros elementos del conjunto.

44	**55**	**12**	**42**	**94**	**18**	**6**	**67**
6	55	12	42	94	18	44	67
6	18	12	42	94	55	44	67
6	18	12	42	44	55	94	67
6	18	12	42	44	55	94	67
6	12	18	42	44	55	67	94
6	12	18	42	44	55	67	94
6	12	18	42	44	55	67	94
6	12	18	42	44	55	67	94

Análisis de Llamadas y Profundidad Recursiva en Quicksort

¿Cuántas llamadas se realizan en total al método "Quicksort" (contando la inicial)?

En total, se realizan 7 llamadas al método principal Quicksort, incluyendo la llamada inicial.

¿Cuál es el máximo nivel de profundidad recursiva alcanzado?

El nivel máximo de profundidad recursiva alcanzado es 4. Esto se puede observar claramente en la tabla.

¿Cómo podrías medir este nivel en una implementación en JAVA?

Para medir el nivel de profundidad recursiva en una implementación en JAVA, una posible solución sería utilizar un contador dentro de la clase. Este contador incrementa cada vez que el algoritmo realiza una llamada recursiva. Esto permite rastrear el nivel de profundidad actual y compararlo con la profundidad máxima alcanzada.

3) Resumen de los conceptos más importantes presentes en el artículo:

Quicksort de Tres Vías

El quicksort de tres vías mejora la eficiencia al manejar datos con muchos elementos duplicados. Este algoritmo divide el array en tres partes:

- Elementos menores al pivote.
- Elementos iguales al pivote.
- Elementos mayores al pivote.

Esta estrategia es especialmente útil para datos con muchas repeticiones, ya que reduce significativamente el tiempo de ejecución al agrupar los elementos duplicados en una sola partición.

Tiempos de Ejecución

- Datos Aleatorios:
    - Quicksort Básico: 1222 ms
    - Quicksort de Tres Vías: 1295 ms
    - Quicksort de Doble Pivote: 1066 ms
- Datos Duplicados:
    - Quicksort Básico: 378 ms
    - Quicksort de Tres Vías: 15 ms
    - Quicksort de Doble Pivote: 6 ms

Los resultados muestran que el quicksort de tres vías maneja mejor los datos duplicados, aunque puede ser ligeramente más lento con datos aleatorios comparado con el quicksort básico de datos aleatorios grandes. La elección del algoritmo más adecuado depende de las características específicas de los datos a ordenar.

4) Análisis del Orden del Tiempo de Ejecución y Mejora para Conjuntos Pequeños

Análisis del Orden del Tiempo de Ejecución

Peor Caso: El peor caso en Quicksort ocurre cuando el pivote seleccionado es siempre el mayor o menor elemento del subarreglo. Esto provoca particiones extremadamente desequilibradas, resultando en una complejidad temporal de $O(n^2)$.

Probabilidad del Peor Caso: Si las claves están distribuidas uniformemente y cada clave tiene la misma probabilidad de aparecer en cualquier posición, la probabilidad del peor caso disminuye considerablemente. Utilizando un pivote aleatorio o el método de la Mediana de Tres, la probabilidad específica depende del método de selección del pivote.

2. Mejora para Conjuntos Pequeños

Peor Caso: El peor caso para conjuntos pequeños ocurre cuando el pivote seleccionado es siempre el elemento más pequeño o más grande del conjunto, resultando en particiones muy desequilibradas. En este caso, el tiempo de ejecución es $O(n^2)$.

Probabilidad del Peor Caso: Con una distribución uniforme de las posiciones de las claves, la probabilidad de seleccionar el peor pivote en cada paso es baja. La probabilidad exacta depende de la implementación del pivote, pero generalmente es baja debido a la selección aleatoria del pivote.

Modificaciones para Conjuntos Pequeños: Para optimizar el rendimiento de Quicksort en conjuntos pequeños, se puede combinar con un algoritmo de ordenación más sencillo y eficiente para pequeños volúmenes de datos, como el Insertion Sort.

Implementación: En una implementación típica, se cambia a Insertion Sort cuando el tamaño del subarreglo cae por debajo de un umbral predefinido (por ejemplo, 10 elementos). Esta combinación se conoce como "ordenación híbrida" y es común en implementaciones prácticas de Quicksort.