**REPORTE DE PRÁCTICA – ESTRUCTURA DE DATOS**

Nombre: ALEX ANTONIO, JUSTIN AVIK, IAN PADRON Fecha Entrega:13/10/202 . No Practica: . Fecha Recepción: .

**1. Definición del Problema: El objetivo de la práctica es implementar el algoritmo de ordenación Quicksort para ordenar una lista de números enteros de forma ascendente.**

**2. Definición de Variables y constantes:**

| **De entrada** | **Formulas** | **De Salida** |
| --- | --- | --- |
| * **array: El objetivo de la práctica es implementar el algoritmo de ordenación Quicksort para ordenar una lista de números enteros de forma ascendente.** * **inicio: indice de inicio de la sublista que se va a ordenar.** * **fin: indice de inicio de la sublista que se va a ordenar.** | **El algoritmo: Quicksort divide el arreglo en subarreglos basados en un pivote. La función de partición coloca los elementos menores o iguales al pivote a la izquierda y los mayores a la derecha. Después, aplica recursividad en ambos subarreglos.** | **El arreglo array se ordena de forma ascendente al finalizar la ejecución del algoritmo.** |

**3. Diagrama de flujo**

| **+---------------------------+**  **| INICIO |**  **+---------------------------+**  **|**  **v**  **+---------------------------+**  **| ¿inicio < fin? |**  **+---------------------------+**  **| Sí | No |**  **| v**  **| +---------------------+**  **| | FIN |**  **| +---------------------+**  **v**  **+-------------------------------+**  **| Pivote = Partición |**  **| (Arreglo, inicio, fin) |**  **+-------------------------------+**  **|**  **v**  **+-------------------------------------+**  **| Quicksort(Arreglo, inicio, |**  **| pivote - 1) |**  **+------------------------------------ +**  **|**  **v**  **+---------------------------------------+**  **| Quicksort(Arreglo, pivote +|**  **| 1, fin) |**  **+---------------------------------------+**  **|**  **v**  **+---------------------------+**  **| FIN |**  **+---------------------------+** |
| --- |

**4. Prueba de Escritorio**

| | **Accion** | **elementos** | **pivote** | **salida** | | --- | --- | --- | --- | | **Llamada a Quicksort(0, 6)** | **[(0, 6)]** |  |  | | **Llamar a Partición(0, 6), pivote = 70** | **[(0, 6)]** | **70** |  | | **Intercambiar 80 y 30** | **[(0, 6)]** | **70** | **{10, 30, 80, 90, 40, 50, 70}** | | **Intercambiar 80 y 40** | **[(0, 6)]** | **70** | **{10, 30, 80, 90, 40, 50, 70}** | | **Intercambiar 90 y 50** | **[(0, 6)]** | **70** | **{10, 30, 80, 90, 40, 50, 70}** | | **Intercambiar 80 y 70** | **[(0, 6)]** | **70** | **{10, 30, 80, 90, 40, 50, 70}** | | **Quicksort izquierda (0, 3)** | **[(0, 3)]** |  |  | | **Llamar a Partición(0, 3), pivote = 50** | **[(0, 3)]** | **50** |  | | **Intercambiar 30 y 30** | **[(0, 3)]** | **50** | **{10, 30, 40, 50, 70, 90, 80}** | | **Intercambiar 40 y 40** | **[(0, 3)]** | **50** | **{10, 30, 40, 50, 70, 90, 80}** | | **Colocar pivote 50** | **[(0, 3)]** | **50** | **{10, 30, 40, 50, 70, 90, 80}** | | **Quicksort izquierda (0, 2)** | **[(0, 2)]** |  |  | | **Llamar a Partición(0, 2), pivote = 40** | **[(0, 2)]** | **40** |  | | **Intercambiar 30 y 30** | **[(0, 2)]** | **40** | **{10, 30, 40, 50, 70, 90, 80}** | | **Colocar pivote 40** | **[(0, 2)]** | **40** | **{10, 30, 40, 50, 70, 90, 80}** | | **Quicksort izquierda (0, 1)** | **[(0, 1)]** |  |  | | **Llamar a Partición(0, 1), pivote = 30** | **[(0, 1)]** | **30** |  | | **Colocar pivote 30** | **[(0, 1)]** | **30** | **{10, 30, 40, 50, 70, 90, 80}** | | **Quicksort derecha (5, 6)** | **[(5, 6)]** |  |  | | **Llamar a Partición(5, 6), pivote = 80** | **[(5, 6)]** | **80** |  | | **Colocar pivote 80** | **[(5, 6)]** | **80** | **{10, 30, 40, 50, 70, 80, 90}** | | **Final** |  |  | **{10, 30, 40, 50, 70, 80, 90}** | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**5. Seudo-código (Español estructurado)**

| **FUNCION Quicksort(arreglo[], inicio, fin)**  **SI inicio < fin ENTONCES**  **pivoteIndex = Particion(arreglo, inicio, fin)**  **Quicksort(arreglo, inicio, pivoteIndex - 1)**  **Quicksort(arreglo, pivoteIndex + 1, fin)**  **FIN SI**  **FIN FUNCION**  **FUNCION Particion(arreglo[], inicio, fin)**  **pivote = arreglo[fin]**  **i = inicio - 1**  **PARA j DESDE inicio HASTA fin - 1 HACER**  **SI arreglo[j] <= pivote ENTONCES**  **i = i + 1**  **intercambiar(arreglo[i], arreglo[j])**  **FIN SI**  **FIN PARA**  **intercambiar(arreglo[i + 1], arreglo[fin])**  **RETORNAR i + 1**  **FIN FUNCION**  **FUNCION intercambiar(a, b)**  **temp = a**  **a = b**  **b = temp**  **FIN FUNCION** |
| --- |

**6. Desarrollo de la practica: Planteamiento del algoritmo:**

* **El algoritmo Quicksort fue seleccionado por su eficiencia para ordenar listas grandes, ya que utiliza el paradigma de "divide y vencerás". El proceso se divide en dos fases: la partición del arreglo en base a un pivote y la recursividad para ordenar las sublistas.**

**Definición del código:**

* **Se implementó la función quicksort que toma como parámetros el arreglo a ordenar, el índice de inicio y el índice final.**
* **Se implementó la función particion, que es responsable de dividir el arreglo en dos partes según el pivote y luego devolver la nueva posición del pivote.**
* **La función intercambiar se utilizó para realizar los cambios de posición de los elementos del arreglo cuando sea necesario.**

**Ejecución del algoritmo:**

* **Se comenzó con un arreglo desordenado de números {10, 80, 30, 90, 40, 50, 70}.**
* **El pivote inicial fue seleccionado como el último elemento del arreglo. En cada paso, el arreglo fue dividido en subarreglos de elementos menores o mayores al pivote, y se realizaron intercambios de elementos según el algoritmo.**
* **El proceso continuó recursivamente para las sublistas hasta que todos los elementos quedaron ordenados.**

**Prueba de escritorio:**

* **Se realizó una prueba de escritorio paso a paso para verificar el correcto funcionamiento del algoritmo, con una tabla detallada de las llamadas a quicksort y el manejo de la pila durante el proceso.**

**Resultados:**

* **Al finalizar la ejecución del algoritmo, el arreglo fue ordenado exitosamente como {10, 30, 40, 50, 70, 80, 90}, cumpliendo con el objetivo de la práctica.**

**7. Conclusiones: El algoritmo Quicksort es eficiente para ordenar grandes cantidades de datos y tiene una complejidad promedio de O(n log n). Es un algoritmo de "divide y vencerás" que utiliza la recursión para ordenar sublistas.**

**8. Estado de la practica:** A) Terminada B) Inconclusa

**9. Aprovechamiento:** A) Excelente B) Bueno C) Regular D) Deficiente **10. Observaciones y/o sugerencias**