**INFORME DE INGENIERÍA**

**Paso 1: Identificación del problema**

**Definición del problema**

Encontrar los tres algoritmos de ordenamiento más eficientes para la implementación de un programa que tiene como base ordenar números enteros o reales.

**Justificación**

En busca de un programa más eficiente las empresas naturalmente usan coprocesadores cuya función es descargar trabajo del procesador principal ya que poseen una tarea especializada. Los coprocesadores pueden realizar operaciones nativas como lo es la función de ordenamiento. Por lo tanto, después de estudiar los costos de implementación de dicho algoritmo se ha decidido encontrar tres algoritmos de ordenamiento más eficientes dependiendo del caso de ordenamiento dado (real o entero, y el numero de elementos a ordenar).

**Requerimientos funcionales**

|  |  |
| --- | --- |
| **R1** | Ordenar datos ingresados |
| **Descripción** | Ordena datos (reales o enteros) ingresados por el usuario, y muestra el resultado. |
| **Entradas** | Datos por ordenar. |
| **Salidas** | Datos ordenados. |

|  |  |
| --- | --- |
| **R2** | Generar valores aleatorios ya ordenados |
| **Descripción** | Generar el número de valores que el usuario quiera y que estos estén ordenados |
| **Entradas** | Número de valores a generar |
| **Salidas** | Lista de valores ordenados |

|  |  |
| --- | --- |
| **R3** | Generar valores aleatorios de acuerdo a un % de desorden. |
| **Descripción** | Generar valores aleatorios con base en el tamaño de la secuencia y el % de desorden se obtiene un número k de cuantas posiciones deben estar desordenadas. |
| **Entradas** | Tamaño de la secuencia y % de desorden |
| **Salidas** | Lista de valores con k posiciones desordenadas |

|  |  |
| --- | --- |
| **R4** | Generar valores intercambiados |
| **Descripción** | Generar valores aleatorios ordenados y con base en el tamaño de la secuencia y el % de desorden se generan k/2 pares de posiciones diferentes y se intercambian los valores entre cada par de ellas. |
| **Entradas** | Tamaño de la secuencia y % de desorden |
| **Salidas** | Lista de valores con k/2 posiciones intercambiadas. |

|  |  |
| --- | --- |
| **R5** | Mostrar el tiempo de ordenamiento |
| **Descripción** | Mostrar el tiempo en que se ordenan los valores. |
| **Entradas** | Datos por ordenar |
| **Salidas** | Tiempo de ordenamiento. |

|  |  |
| --- | --- |
| **R6** | Restringir o permitir la ejecución de los algoritmos de ordenamiento. |
| **Descripción** | Dependiendo del número y tipo de datos a ordenar se debe restringir o permitir la ejecución de alguno de los tres algoritmos de manera que resulte ser el más eficiente. |
| **Entradas** | Datos por ordenar |
| **Salidas** | Opciones de algoritmo que pueda ser usado para ordenar esa secuencia. |

**Paso 2: Recopilación de la información**

**COMPLETAR (CAMILA)**

*Algoritmo*

Es una secuencia de pasos bien definidos que buscan resolver un problema computacional.

*Eficiencia*

Se dice que un algoritmo es eficiente cuando entrega la respuesta correcta para todos los casos y con la menor complejidad posible.

*Complejidad temporal*

Función que describe el comportamiento (en tiempo) de un algoritmo conforme se incrementa el tamaño de la entrada.

*Algoritmo de ordenamiento*

*Complejidad espacial*

*Notación asintótica*

Son aquellas notaciones utilizadas para describir el tiempo de ejecución asintótico de un algoritmo. Conocemos la notación O para el peor caso, Ω para el mejor caso y Ɵ para el caso promedio.

Según Kenneth H. Rosen en su libro *Matemática Discreta y sus aplicaciones*, los algoritmos más eficientes ordenados de manera descendente son aquellos que:

|  |  |
| --- | --- |
| Complejidad | Terminología |
| O(l) | Complejidad constante |
| O(log n) | Complejidad logarítmica |
| O(n) | Complejidad lineal |
| O(n log n) | Complejidad n log n |
| O( | Complejidad polinómica |
| O() | Complejidad exponencial |
| O(n!) | Complejidad factorial |

Por lo tanto, debemos tener como criterio de selección la complejidad temporal del algoritmo con base en esa tabla.

**Un análisis ya existente:**

Encontramos en un blog de *pereiratechtalks.com* perteneciente a Sergio Andrés Flórez un análisis hecho para encontrar el algoritmo de ordenamiento más eficiente:

Sergio toma siete algoritmos conocidos: Burbuja, inserción, selección, Heapsort (ordenamiento por montículos), conteo, merge sort y Quicksort, los implementa y los ejecuta en los maquinas con diferentes especificaciones para encontrar cuál es el algoritmo más eficiente para ordenar hasta 1.000.000.000 datos.

Después de un análisis de las gráficas de los siete algoritmos, llegó a la conclusión de que en definitiva los perdedores eran burbuja, inserción y selección. Por lo tanto, se decide analizar más a de cerca los cuatro restantes, y se llega a la conclusión de que el algoritmo más eficiente fue el de conteo con una complejidad de O(n+k).

**Paso 3: Búsqueda de soluciones creativas**

Decidimos hacer una lluvia de ideas de posibles soluciones:

* Crear un árbol binario e ir insertado los valores ingresados y luego imprimirlos de acuerdo al recorrido inorden.
* Ordenamiento burbuja, selección o inserción.
* Bogosort: consiste en tomar el conjunto de valores ingresados y reorganizarlos de manera aleatoria en el conjunto, hasta que logre organizarlos de manera ascendente o descendente.
* Ordenamiento por panqueques: Ordena un arreglo de números enteros mediante rotaciones del arreglo en entre el valor más alto y el más bajo; este va reduciendo el tamaño del arreglo rotado mientras que se ordenan los valores en orden ascendente.
* Ordenamiento Radix: Es un método de ordenamiento que consiste en organizar valores de un arreglo en base a ciertas claves o categorías que el algoritmo aplica para separar los valores mediante la comparación de números en sus posiciones de cifra significativas.
* *Buscamos algoritmos ya conocidos: ordenamiento por casilleros, ordenamieno por mezcla, ordenamiento por montículos, ordenamiento rápido.*

**Paso 4: Diseños preliminares**

A primera vista, la opción de crear un árbol y luego imprimir su recorrido inorden queda descartada debido a su consumo de recursos de almacenamiento y de tiempo.

Los algoritmos inestables como el bogosort y el ordenamiento por panqueques también quedan descartadas porque no garantiza una respuesta segura y no tienen la capacidad de organizar arreglos grandes de manera eficiente.

Investigando más a fondo el algoritmo Radix queda descartado porque este algoritmo depende mucho de la eficiencia de las instrucciones es un interior y tiene dificultades con diferentes tipos de datos (enteros y decimales).

*ME PARECE QUE DEBERIA DESCARTARSE EL DE SELECCIÓN Y BURBUJA DE UNA VEZ.*

Por lo tanto, las ideas mas factiles son: algoritmo de selección, burbuja, inserción, por montículos, conteo, mezcla y rápido.

* Ordenamiento por selección
* Ordenamiento burbuja
* Ordenamiento de inserción
* Ordenamiento por montículos (Heapsort)
* Ordenamiento por conteo (Counting sort)
* Ordenamiento por mezcla (Merge sort)
* Ordenamiento rápido (Quicksort)

**Paso 5: Evaluación o selección de la mejor solución (Criterios y selección)**

Criterio A

Fácil implementación

Criterio B

Almacenamiento temporal adicional

Criterio C

Diferencia entre el mejor y el peor caso

Criterio D

Uso de recursividad

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Criterio A | Criterio B | Criterio C | Criterio D | Total |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Los tres algoritmos que elegir son:

**Paso 6: Preparación de informes**

**Diseño del diagrama de clases de la solución**

**Pseudocódigo de los tres algoritmos**

**Diseño de casos de pruebas unitarias**

*Diseña para cada uno de los 3 algoritmos relevantes al menos tres casos de prueba para verificar su correcto funcionamiento* ***(uno estándar, uno que pruebe casos límite y otro interesante)*** *cumpliendo con el formato donde se especifica la clase, el método a probar, el escenario, las entradas y la salida esperada.*

*Ordenados de mayor a menor y deba ordenarlos de menor a mayor*

*Todo ordenado*

*Normal*

*Que toda la secuencia sea de los mismos datos.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba 1: Algoritmo | | | | |
| Clase | Método | Escenario | Entrada | Resultado |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Análisis de complejidad temporal de los tres algoritmos**

**Análisis de complejidad espacial de los tres algoritmos**

Entradas, auxiliares y salidas

**Bibliografía**

Matemática Discreta y sus aplicaciones. Kenneth H. Rosen. Quinta edición.

<https://pereiratechtalks.com/analisis-de-algoritmos-de-ordenamiento/>

<https://techlandia.com/ventajas-desventajas-algoritmos-ordenamiento-info_181515/>

<https://jona83.wordpress.com/unidad-1/algoritmos-de-ordenamiento/>