**Experimentación de los algoritmos de ordenamiento Insertion-sort y Cocktail sort**

**Integrantes:**

**Emmanuel Zuluaga G1**

**Juan david Lopez G3**

**Julian Mabesoy G3**

**Ingeniería de sistemas**

**Universidad Icesi**

**Cali, Colombia**

**Análisis de Complejidad**

**Insertion Sort:** (En el peor de los casos)

**Análisis de complejidad temporal:**

|  |  |  |
| --- | --- | --- |
| **Insertion-Sort (A)** | **Costo** | **Tiempo** |
| **for** j= 2 to length[A] |  | n |
| **do** keyA[j] |  | n-1 |
| // insert A[j] to sorted sequence A[1...j-1] | 0 | n-1 |
| ij-1 |  | n-1 |
| **while** i>0 and A[i]>key |  |  |
| **do** A[i+1]A[i] |  |  |
| ii-1 |  |  |
| A[i+1]key |  | n-1 |

* **Peor caso:** los números están en orden inverso
* = =n(n+1)/2-1, y == n(n-1)/2
* T(n)= n+n-1)+n-1)+n(n+1)/2-1)+n(n-1)/2-1)+(n(n-1)/2)+n-1
* =((++)/2)+(+++/2-/2-/2+)n-(+++)
* =

**Análisis de complejidad espacial:**

Ya que es un algoritmo in-place(que no necesita una estructura de datos auxiliar, el algoritmo realiza los intercambiamos en el mismo arreglo), la complejidad espacial del algoritmo insertion es O(1).

**Cocktail Sort:**

**Análisis de complejidad temporal:**

|  |  |  |
| --- | --- | --- |
| **Cocktail-Sort (A)** | **Costo** | **Tiempo** |
| while i = 0 to length[A] | C1 | n+1 |
| do key ←- A[i] | C2 | n |
| add object | C3 | n |
| for j=i and j < A.length-1 | C4 |  |
| do A[j+1] ← A[i] | C5 |  |
| add objects | C6 |  |
| for j=A.length-1 and j > i | C7 |  |
| do A[j+1] ← A[i] | C8 |  |
| add objects | C9 |  |
|  |  |  |

* **Peor de los casos**: El orden del arreglo de encuentra desorganizado, entonces su complejidad temporal será de O(n^2).
* **mejor de los casos**: El orden del arreglo se encuentra ordenado, entonces su complejidad temporal será de O(n).

**PLANEACION Y REALIZACION**

**1. OBJETO DE ESTUDIO:**

Se estudiará los algoritmos de ordenamiento (**Insertion Sort-Cocktail Sort**), mediante arreglos de valores, teniendo en cuenta su estado inicial (tamaño, tipo de orden). Con base en esta prueba, se evaluará diferentes criterios y factores que influyen en el proceso de Entrada-Salida del arreglo, asimismo se identificará las ventajas y desventajas del uso de estos algoritmos en diversas aplicaciones.

**2. VARIABLES DE RESPUESTA:**

* Tiempo de ejecución del Algoritmo (Complejidad temporal y espacial).

Con esta variable, se espera llevar a cabo un análisis donde se pueda recopilar información, para encontrar cual es el algoritmo más eficiente y óptimo para el problema de ordenamiento en arreglos con variables numéricas.

**3. DETERMINACIÓN DE FACTORES, QUE INFLUYEN EN EL RESULTADO:**

* **Tamaño del arreglo:**

Este factor es sumamente importante, ya que con esto influye la rapidez y el tiempo de desarrollo de la solución del algoritmo, es decir que entre más sea el tamaño de del arreglo así va a variar el resultado final entre los algoritmos escogidos.

* **Estado de los valores en el arreglo:**

El estado del arreglo es un factor que da a conocer la complejidad, y el proceso que se llevaría a cabo en el proceso de ejecución, con este factor se puede evaluar si el algoritmo es óptimo o no, para el objetivo que se está evaluando.

* **RAM del computador donde se ejecuta el algoritmo:**

Este aspecto, es fundamental a la hora de ejecución de la solución del problema, ya que todos los computadores, no cuentan con los mismos recursos, sabiendo que por medio de esta experimentó el objetivo es analizar, si la implementación o la estructura del algoritmo es eficiente o adecuada.

* **Procesador del computador donde se ejecuta el algoritmo:**

Es importante saber las especificaciones del procesador ya que este es el cerebro y corazón de las computadoras, por eso sus características son esenciales, ya que consideran la velocidad, capacidad de memoria y seguridad. Toda acción realizada por la computadora en algún momento pasa por esta.

* **Lenguaje de Programación:**

En el entorno de desarrollo y arquitectura de software, es importante saber cual es más eficiente, en la ejecución y la lectura de los algoritmos, con este factor se evaluará el consumo de los recursos del computador, el tiempo de ordenamiento y la rapidez de la lectura de cada línea de código.

**4. NIVELES DE LOS FACTORES TOMADOS EN CUENTA:**

**Algoritmos de entrada:**

Nivel 1: Insertion Sort

Nivel 2: Cocktail Sort

**Tamaño del arreglo a ordenar**

Nivel 1: 10.000

Nivel 2: 100.000

Nivel 3: 1’000.000

Nivel 4: 10’000.000

Nivel 5: 100’000.000

**Variables de ensayo:**

Nivel 1: Double.

Nivel 2: Int

Nivel 3: Float

**Entrada o estado inicial de los Arreglos:**

Nivel 1: Orden descendente.

Nivel 2: Orden aleatorio.

Nivel 3: Orden ascendente.

**Computadores para el uso del experimento:**

1. Portatil Lenovo ideapad 330s Ryzen 3 2200U RAM 8 GB

2. Portatil Asus Intel i7 5500U RAM 12 GB

3. Portatil Toshiba Intel i5 5200U RAM 12 GB

**Lenguajes de programación:**

Java-(Ejecutado en Eclipse IDE).

C# -(Ejecutado en Microsoft Visual Studio)

**5. PLANEACIÓN DEL EXPERIMENTO:**

Los algoritmos de ordenamiento se ejecutan individualmente en cada computador de los integrantes del grupo, se modifica la estructura del algoritmo, para que arroje los valores de tiempo, y con base a ellos se hace un tabulado, con sus respectivos análisis. La modificación del algoritmo es únicamente para el muestreo de datos en consola, sin perder, ni modificar la complejidad de estos. Esta toma de datos se evalúa y con base, a los resultados de la evaluación, se deduce que algoritmo es mejor para la organización de arreglos con valores numéricos.

**6. RESULTADO:**

**Computador:** Portatil Toshiba Intel i5 5200U RAM 12 GB.

**Lenguaje:** C#

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Int | | Double | | Long | |
| insertion | cocktail | insertion | cocktail | insertion | cocktail |
| 10 | 7 ticks | 1 tick | 3 ticks | 1 tick | 2 ticks | 1 tick |
| 100 | 93 ticks | 4 ticks | 63 ticks | 3 ticks | 70 ticks | 3 ticks |
| 1000 | 5779 ticks | 17 ticks | 5219 ticks | 22 ticks | 6777 ticks | 18 ticks |

**Lenguaje:** Java

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Int | | Double | | Long | |
| insertion | cocktail | insertion | cocktail | insertion | cocktail |
| 10 | 32 ticks | 23 ticks | 37 ticks | 23 ticks | 37 ticks | 27 ticks |
| 100 | 1292 ticks | 65 ticks | 1292 ticks | 51 ticks | 1357 ticks | 51 ticks |
| 1000 | 114972 ticks | 545 ticks | 133208 ticks | 541 ticks | 154002 ticks | 541 ticks |

**Computador:** Portatil Lenovo ideapad 330s Ryzen 3 2200U RAM 8 GB

**Lenguaje:** C#

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Int | | Double | | Long | |
| insertion | cocktail | insertion | cocktail | insertion | cocktail |
| 10 | 17 ticks | 3 ticks | 5 ticks | 3 ticks | 4 ticks | 2 ticks |
| 100 | 40 ticks | 2 ticks | 74 ticks | 5 ticks | 81 ticks | 3 ticks |
| 1000 | 5491 ticks | 462 ticks | 7192 ticks | 527 ticks | 7098 ticks | 31 ticks |

**Lenguaje:** Java

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Int | | Double | | Long | |
| insertion | cocktail | insertion | cocktail | insertion | cocktail |
| 10 | 41 ticks | 20 ticks | 32 ticks | 20 ticks | 36 ticks | 24 ticks |
| 100 | 2088 ticks | 94 ticks | 2101 ticks | 86 ticks | 976 ticks | 45 ticks |
| 1000 | 64329 ticks | 439 ticks | 61678 ticks | 874 ticks | 54287 ticks | 480 ticks |

**Análisis**

Con base en los resultados obtenidos por el experimento, no se puede afirmar, ni conjeturar cuál de los algoritmos evaluados son mejores para el uso global, cuales tiene mejor características, sin embargo, se puede exponer, cual es mas optimo para el uso académico, añadiendo que sea eficiente con los recursos de los equipos de los estudiantes.

**Interpretación**

**Control y conclusiones finales**