# DANIEL SANTOS - OBSERVACIONES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| % | Tamaño (Array\_list) | Insertion\_sort (ms) | Selection(ms) | Shell(ms) | Merge\_sort |
| 5 | 245 | 36.97 | 58.5 | 33.66 | 6.5 |
| 20 | 980 | 588.84 | 967.52 | 191.06 | 27.4 |
| 50 | 2451 | 3899.79 | 6167.97 | 560.81 | 79.35 |
| 100 | 4903 | 16203.72 | 25672 | 1436.84 | 174.92 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| % | Tamaño (Single\_linked) | Insertion\_sort (ms) | Selection(ms) | Shell(ms) | Merge\_sort |
| 5 | 245 | 634 | 591.62 | 90.35 | 20.26 |
| 20 | 980 | 41722.93 | 38218.62 | 2255.34 | 255.977 |
| 50 | 2451 | 687799 | NN | 15928 | 1530.21 |
| 100 | 4903 | NN | NN | 82718.1 | 6143.9 |

1. ¿El comportamiento de los algoritmos es acorde a lo enunciado teóricamente?

Para la primera máquina se observa que los datos parecen lineales, sin embargo, el mejor ajuste es un ajuste exponencial, este resultado se debe a que los datos son muy pequeños y prevalece el comportamiento lineal, sin embargo, cuando son grandes volúmenes de datos, el comportamiento logaritmo comienza a prevalecer. Corresponde al mejor ajuste posible el ajuste exponencial y se observa que se comporta de acuerdo con la teoría.

1. ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?

Sí, en las máquinas con procesadores más rápidos y con más RAM, el programa corre más rápido, sin embargo, se mantiene el comportamiento linearítmico.

1. De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?

Se deben a si se están ejecutando tareas en segundo plano que puedan consumir recursos, y debido a lo anteriormente mencionado, a los procesadores y a la RAM.

Comparando se obtuvo la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Algoritmo | Array\_list | Single\_linked |
| Merge | Mejor | Mejor |
| Quick | Peor | Peor |

a. ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?

Si solo se considera eso, lo mejor sería Merge sort, debido a que tiene menor tiempo de respuesta.

b. Para el escenario de ordenamiento de impuestos, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor - en relación con los tiempos de ejecución) para ordenar la mayor cantidad de impuestos.

|  |
| --- |
| Sort(menos tiempo a más) |
| Merge |
| Quick |
| Shell |
| Insertion |
| Selection |

# MARIA PAZ RAMIREZ - OBSERVACIONES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| % | Tamaño (Array\_list) | Insertion\_sort (ms) | Selection(ms) | Shell(ms) | Merge(ms) | Quick(ms) |
| 5 | 245 | 467.48 | 589.2 | 76.46 | 78.61 | 65.61 |
| 20 | 980 | 7698.93 | 9452.42 | 456.71 | 155.17 | 139.17 |
| 50 | 2451 | 44122.96 | 65276.66 | 1892.53 | 1062.86 | 1138.6 |
| 100 | 4903 | 185311.36 | 247904.58 | 3656.77 | 1983.61 | 2136.97 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| % | Tamaño (Single\_linked) | Insertion\_sort (ms) | Selection(ms) | Shell(ms) | Merge(ms) | Quick(ms) |
| 5 | 245 | 10459.16 | 9339.28 | 1436.59 | 309.79 | 1196.08 |
| 20 | 980 | 658386.97 | 537709.66 | 32040.38 | 1024.79 | 5977.27 |
| 50 | 2451 | 9310745.91 | 8055380.47 | 215971.39 | 4148.74 | 24937.88 |
| 100 | 4903 |  |  |  | 86352.97 | 752585.61 |

**a. ¿El comportamiento de los algoritmos es acorde a lo enunciado teóricamente?**

Si, puesto que los tipos de ordenamiento que realizan un procedimiento mas largo son más lentos al ejecutar el codigo, y los tipos de ordenamiento con mayor eficiencia en el codigo son más rapidos.

**b. ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?**

Si, ya que el programa corre mas o menos rapido segun los componentes del computador, en mi caso al no tener una memoria RAM y procesador mejores algunos ordenamientos se demoran demasiado tiempo en ejecutarse.

**c. De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?**

Al tipo de procesador y que tan veloz sea al realizar este tipo de procesos, además de la memoria RAM la cual al ser más grande permite llevar a cabo los procesos más eficazmente, y el tipo de OS instalado también permite mejorar o empeorar la velocidad.

**a. ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?**

La estructura de datos Array list, merge sort es el mejor, ya que sus tiempos de ejecución son más rápidos

**b. Para el escenario de ordenamiento de impuestos, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor - en relación con los tiempos de ejecución) para ordenar la mayor cantidad de impuestos.**

1. Merge\_sort

2. Quick\_sort

3. Shell\_sort

4. Insertion\_sort

5. Selection\_sort

|  |  |  |
| --- | --- | --- |
|  | Merge (Array\_list) | Merge (Single\_linked) |
| 5% | 78,61 | 309,79 |
| 20% | 155,17 | 1024,79 |
| 50% | 1062,86 | 4148,74 |
| 100% | 1983,61 | 86352,97 |

|  |  |  |
| --- | --- | --- |
|  | Quick (Array\_list) | Quick (Single\_linked) |
| 5% | 65,61 | 1196,08 |
| 20% | 139,17 | 5977,27 |
| 50% | 1138,6 | 24937,88 |
| 100% | 2136,97 | 752585,61 |

# LUIS FELIPE SALES – OBSERVACIONES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **%** | **Tamaño (Array\_list)** | **Insertion (ms)** | **Selection (ms)** | **Shell(ms)** | **Merge (ms)** | **Quick (ms)** |
| 5 | 245 | 27.04 | 42.08 | 5.40 | 11.71 | 10.58 |
| 20 | 980 | 421.61 | 667.43 | 30.22 | 49.32 | 51.06 |
| 50 | 2451 | 2750.73 | 4183.65 | 89.95 | 114.11 | 134.08 |
| 100 | 4903 | 11459.35 | 16918.12 | 213.09 | 184.04 | 189.38 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **%** | **Tamaño (Single\_linked)** | **Insertion\_sort (ms)** | **Selection(ms)** | **Shell(ms)** | **Merge (ms)** | **Quick (ms)** |
| 5 | 245 | 523.30 | 506.44 | 80.54 | 40.63 ms | 134.39 |
| 20 | 980 | 35379.91 | 32724.64 | 1904.81 | 306.65 | 1862.44 |
| 50 | 2451 | 625419.08 | - | 16529.26 | 1680.54 | 13253.04 |
| 100 | 4903 | - | - | 74022.33 | 6553.06 | 57091.13 |

**A. ¿El comportamiento de los algoritmos es acorde a lo enunciado teóricamente?**

Efectivamente el comportamiento de los algoritmos es lo acorde a lo enunciado, sin embargo, por la diferencia tan abrumadora de tiempos respecto a los porcentajes de datos analizados, las graficas de algunos de ellos se disparan de manera exponencial dejando muy por debajo los tiempos de los algoritmos de mayor rapidez a quienes se les debió hacer un ajuste logarítmico puesto que sus tiempos tienen un bajo crecimiento en función del tiempo.

**b. ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?**

Así es, puede existir una pequeña o una gran diferencia dependiendo de varios factores, y este caso no fue la excepción.

**c. De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?**

Al existir diferencias en los sistemas operativos, la capacidad de almacenamiento y la memoria RAM u otros factores como el número de aplicaciones abiertas o la temperatura del equipo, los resultados obtenidos varían.

**d. ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?**

La estructura de datos Merge Sort es la mejor en cuanto a optimización del tiempo de ejecución; seguida muy de cerca por Quick Sort que cuenta con datos bastante aproximados, pero aún inferiores.

**e. Para el escenario de ordenamiento de impuestos, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor - en relación con los tiempos de ejecución) para ordenar la mayor cantidad de impuestos.**

1. Merge sort

2. Quick sort

3. Shell sort

4. Insertion sort

5. Selection sort

|  |  |  |
| --- | --- | --- |
| % | Merge (Array\_list) | Merge (Single\_linked) |
| 5 | 11,71 | 40,63 |
| 20 | 49,32 | 306,65 |
| 50 | 114,11 | 1680,54 |
| 100 | 184,04 | 6553,06 |

|  |  |  |
| --- | --- | --- |
| % | Quick (Array\_list) | Quick (Single\_linked) |
| 5 | 10,58 | 134,39 |
| 20 | 51,06 | 1862,44 |
| 50 | 134,08 | 13253,04 |
| 100 | 189,38 | 57091,13 |