Título del reporte

Josué Torres Sibaja, C37853, josue.torressibaja@ucr.ac.cr

*Resumen*—En este trabajo se presenta un análisis del comportamiento de los algoritmos de ordenamiento por selección, inserción y mezcla. Para esto, se realizó una implementación de los algoritmos en C++, la cual se utilizó para generar gráficos de líneas basados en sus tiempos de ejecución. El resultado fue… Se concluye que…

Palabras clave—ordenamiento, selección, inserción, mezcla.

# Introducción

En este trabajo se presenta un análisis del comportamiento de los algoritmos de ordenamiento por selección, inserción y mezcla, para el que se utilizaron herramientas como su implementación en el lenguaje C++, cuadros comparativos de datos y gráficos de líneas.

# Metodología

Para lograr lo propuesto se realizó la implementación de los algoritmos en el lenguaje C++, modificando el archivo ‘Ordenador.hpp’ facilitado por el profesor, y se creó un archivo ‘main.c’ con el que se pudieran generar informes sobre los tiempos de ejecución de los algoritmos. El código se muestra en los apéndices, y está basado en el pseudocódigo del libro de Cormen y colaboradores [1]. El código para medir los tiempos de ejecución utiliza funciones para generar los arreglos aleatorios, medir el tiempo y poner a trabajar los algoritmos. Así mismo, se utilizan varios ciclos *for* anidados para llamar a estas funciones, además de arreglos que contienen los tamaños de los arreglos de números aleatorios y los nombres de los algoritmos, esto para imprimir de forma ordenada los resultados de tiempo de cada algoritmo. Es importante resaltar que el arreglo de números aleatorios es el mismo en cada ejecución de cada algoritmo, ya que se utiliza la misma semilla aleatoria para generarlo en los diferentes tamaños.

Para la ejecución del programa, se procuró cerrar todos los programas o ventanas extra para que la máquina no tuviera más procesos sucediendo al mismo tiempo y se garantizara la obtención de resultados competentes. Tras correr el programa y obtener el tiempo de cada ejecución, estos se registraron junto con su promedio en una tabla de Excel, facilitando trabajar con los datos posteriormente.

Finalmente, se utilizó la plataforma Canva para crear los gráficos de líneas para cada arreglo y el gráfico de comparación. Para esto se utilizó una plantilla predeterminada de la página, en la que se introdujeron los tamaños de los arreglos en el eje X, y los tiempos de ejecución en el eje Y, generando automáticamente la gráfica. Las imágenes también fueron editadas manualmente para agregar títulos y nombres a los ejes, así como crear una escala personalizada para la gráfica de comparación.

Cuadro I

Tiempo de ejecución de los algoritmos

|  | Tiempo (ms) | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Corrida | | | | | | |
| Algoritmo | Tam. (k) | 1 | 2 | 3 | 4 | 5 | Prom. |
| Selección | 50000 | 2528.65 | 2526.57 | 2527.16 |  |  | 2527.46 |
|  | 100000 | 10140.2 | 10122.5 | 10153.8 |  |  | 10138.9 |
|  | 150000 | 22890.5 | 22810.1 | 22838.7 |  |  | 22846.4 |
|  | 200000 | 40667.4 | 40472.1 | 40454.3 |  |  | 40531.3 |
| Inserción | 50000 | 1164.54 | 1164.68 | 1169.12 |  |  | 1166.11 |
|  | 100000 | 4715.16 | 4845.85 | 6998.94 |  |  | 5519.98 |
|  | 150000 | 14237.7 | 13579.7 | 13180.5 |  |  | 13666 |
|  | 200000 | 20792.3 | 20344.2 | 21477.6 |  |  | 20871.4 |
| Mezcla | 50000 | 19.7667 | 19.9541 | 20.4075 |  |  | 20.0428 |
|  | 100000 | 41.1392 | 40.7761 | 42.5718 |  |  | 41.4957 |
|  | 150000 | 62.6132 | 63.0288 | 62.327 |  |  | 62.6563 |
|  | 200000 | 84.6673 | 84.5185 | 85.1676 |  |  | 84.7845 |

Los tiempos de ejecución de las 3 corridas de cada algoritmo se muestran en el cuadro I.

# Resultados

Para el algoritmo de selección, los tiempos de ejecución se mantuvieron prácticamente iguales en las 3 corridas de cada tamaño, siendo la mayor diferencia 213.1 ms entre la primera y la última corrida con el arreglo de 200000 números.

Para el algoritmo de inserción, los tiempos tienden a variar un poco entre la primera y la última corrida con cada tamaño. Para el arreglo de 50000, 100000 y 200000 la primera corrida es más rápida que la última, siendo la diferencia más grande 2283.78 ms con el arreglo de 100000. Para el arreglo de 150000 la última corrida es más rápida que la primera. Así mismo, la segunda corrida suele tener un tiempo más cercano o similar al de la primera, a excepción de la de 150000.

Para el algoritmo por mezcla, para los arreglos de 100000 y 150000, la corrida más diferente es la segunda, siendo más rápida que las otras dos para el primer tamaño, y más lenta que las otras dos para el segundo. En el caso del arreglo de 50000 la primera corrida es la más rápida y la última la más lenta, y para el arreglo de 200000 la segunda corrida es la más rápida y la tercera la más lenta. En este algoritmo no se encontraron patrones tan claros en el comportamiento de los tiempos como en los algoritmos anteriores.

La forma de las curvas fue la esperada, ya que las curvas de los algoritmos Θ(n²) tuvieron una forma aproximadamente parabólica y las curvas de los algoritmos Θ(n) y Θ(n log n) tuvieron una forma aproximadamente lineal.

Los tiempos promedio se muestran gráficamente en la figura 1.

Gráfico, Gráfico de líneas, Gráfico de dispersión

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Figura 1. Tiempos promedio de ejecución de los algoritmos de ordenamiento por selección, inserción y mezcla.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamenteLas curvas se muestran de forma conjunta en la figura 2.

Figura 2. Gráfico comparativo de los tiempos promedio de ejecución de algoritmos de ordenamiento.

En la gráfica se aprecian enormes diferencias entre los tiempos de ejecución, ya que el algoritmo de mezcla tuvo promedios sumamente bajos en comparación de los otros dos algoritmos, manteniéndose por debajo de los 100 milisegundos, cuando los otros rebasaban los 1000 milisegundos en sus ejecuciones más rápidas.

# Discusión

Como primer punto, con base en los resultados presentados, se puede observar que el algoritmo de mezcla es muchísimo más rápido que los otros algoritmos. Esto se debe a la complejidad temporal de cada uno, ya que los algoritmos de selección e inserción tienen una complejidad de Θ(n²), lo que los hace preferible para arreglos pequeños (Computer Science Stack Exchange, n.d.). Por otro lado, el algoritmo de mezcla tiene una complejidad de Θ(n log n), lo que hace que sea superior al trabajar con tamaños de arreglos grandes (Algorithms, 2023). En este caso, como todos los arreglos son de miles de números, el algoritmo de mezcla es el más eficiente, pero si se usaran arreglos mucho más pequeños probablemente los otros dos algoritmos destacarían por encima de este (templatetypedef, 2020).

Adicionalmente, los experimentos realizados revelan que, aunque tanto el algoritmo de selección como el de inserción comparten la misma complejidad de Θ(n²), sus tiempos de ejecución no son idénticos. En la mayoría de los casos, el algoritmo de selección fue más lento que el de inserción, sin embargo, el algoritmo de inserción tiene la ventaja de ser más eficiente en arreglos que ya están parcialmente ordenados (GeeksforGeeks, 2023).

Por otra parte, con los datos recolectados en la tabla …. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt.

# Conclusiones

A partir de los resultados se puede concluir que... Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

##### Referencias

1. GeeksforGeeks. (29 de enero de 2024). Merge Sort vs. Insertion Sort. <https://www.geeksforgeeks.org/merge-sort-vs-insertion-sort/J>.
2. Computer Science Stack Exchange. (n.d.). How does size of list in merge-sort, quick-sort, insertion-sort, matter? <https://cs.stackexchange.com/questions/138201/how-does-size-of-list-in-merge-sort-quick-sort-insertion-sort-matter>
3. Algorithms. (4 de marzo de 2023). *What are the benefits and drawbacks of using merge sort over insertion sort?*. LinkedIn. <https://www.linkedin.com/advice/3/what-benefits-drawbacks-using-merge-sort-over-insertion#:~:text=One%20of%20the%20main%20benefits,as%20the%20input%20size%20increases>.
4. *templatetypedef. (23 de abril de 2020). When would you use Selection sort versus Merge sort?*. Stack Overflow. <https://stackoverflow.com/questions/61393831/when-would-you-use-selection-sort-versus-merge-sort>
5. GeeksforGeeks. (2023, March 30). *Difference between Insertion sort and Selection sort*. GeeksforGeeks. <https://www.geeksforgeeks.org/difference-between-insertion-sort-and-selection-sort/>

|  |  |
| --- | --- |
| Male profile outline | **Your name** Some words about you, e.g. as a student, as a person in general, and what your interests are. |

# Apéndice A

Código de los Algoritmos

Esta sección no debe ir en en reporte escrito. Los siguientes bloques son solo demostrativos en caso de que considere necesario incluir código fuente o pseudocódigo en su reporte para guiar o complementar la discusión del reporte. Al igual que una figura o tabla, si agrega código fuente al reporte, debe referirse a este en el texto del documento. El código se muestra en los algoritmos 1 y 2. El primero es un ejemplo de código que se puede colocar en una columna. Si el código no cabe en una columna, en LaTeX, use la versión que se muestra en el segundo ejemplo. Para ello solo debe cambiar en formato de la sección para que sea de una columna en vez de dos columnas.

|  |
| --- |
| **Algoritmo 1** Algoritmo que despliega en la salida estándar una cadena de texto. |
| int main () {  cout << “Pura vida !\n”;  return 0;  } |

|  |
| --- |
| **Algoritmo 2** Primer código que normalmente se utiliza en un ejercicio de programación. |
| printf( “Hello World!”) |