Universidad de los Andes > FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Estructuras de Datos y Algoritmos (EDA)
Tarea 1: Análisis de Algoritmos y Tiempo de Procesamiento

Prof: José M. Saavedra Rondo
Ayudantes: Braulio Torres & Cristóbal Loyola & Francisco Jiménez

Agosto 2022

1. Objetivo

Comprender la importancia del diseño eficiente de algoritmos a través de una tarea experimental sobre análisis de tiempos de ejecución de algoritmos de ordenación.

2. Descripción

En esta tarea tendrán que implementar y evaluar cinco tipos algoritmos de ordenación y hacer un análisis sobre los tiempos de ejecución (milisegundos) generados. Para ello, dividiremos los experimentos en dos partes: 1) algoritmos genéricos y 2) algoritmos específicos.

2.1. Algoritmos de Ordenación Genéricos

En esta parte deberán evaluar los tiempos de ejecución reales de los siguientes algoritmos:

- Inserción $O(n^2)$
- Selección $O(n^2)$
- QuickSort $O(n \log n)$
- MergeSort $O(n \log n)$

Para lo anterior, tendrán que medir el tiempo de cada método para ordenar arreglos aleatorios de valores flotantes, para diferentes tamaños. Los tamaños de los arreglos serán divididos en dos grupos: 1) tamaños pequeños que varían entre 10.000 a 100.000 y tamaño grandes que varían entre 100.000 y 1.000.000.

- **Tamaños Pequeños**: aquí deben considerar los siguientes 10 tamaños: 10.000, 20.000, 30.000, 40.000, 50.000, 60.000, 70.000, 80.000, 90.000, 100.000.
- **Tamaños Grandes**: aquí deben considerar los siguientes 10 tamaños: 100.000, 200.000, 300.000, 400.000, 500.000, 600.000, 700.000, 800.000, 900.000, 1.000.000.

Para los experimentos de esta parte deben considerar arreglos de valores flotantes que varíen entre 0 y 1. Para ello, pueden utilizar el método *createRandomArray* de `https://github.com/jmsaavedrar/eda_cpp/`.

2.2. Algoritmos de Ordenación Específicos

En esta parte deben comparar un método genérico y uno específico para ordenar arreglos aleatorios de enteros, evaluando el tiempo de ordenación con el conjunto de tamaños grandes, especificado en la sección anterior. Los métodos a ordenar son:

- El mejor método que resulta del experimento anterior.
- RadixSort.

Para los experimentos de esta sección deben considerar arreglos de valores enteros que varíen entre 0 y 99999.

3. Código Base

Las implementaciones deben ser propias. Solamente pueden ocupar las implementaciones que vienen en el repositorio del curso `https://github.com/jmsaavedrar/eda_cpp/`.

4. Métricas y Evaluaciones

Para los experimentos anteriores deben presentar 3 tablas de resultados, dos para la evaluación de algoritmos genéricos y una para los específicos. Las dos tablas de los algoritmos genéricos corresponden a los dos grupo de tamaños involucrados. Cada tabla puede ser estructurada en forma similar a la tabla presentada en el Cuadro 1 de este documento.

Además, deben presentar 3 gráficos, dos para los algoritmos genéricos y uno para los específicos. Un ejemplo de un gráfico esperado para dos métodos se presenta en la Figura 1.

Tamaño (n)	Tiempo de Ejecución (ms)		
	Método 1	**Método 2**	...
10.000			
20.000			
30.000			
40.000			
⋮			⋱

Cuadro 1: Ejemplo de estructura para las tablas de resultados.

En la evaluación es importante relacionar los tiempos de ejecución real con los teóricos. Por ejemplo, es importante notar si un algoritmo de complejidad cuadrática o logarítmica sigue ese mismo comportamiento con respecto a los tiempos reales.

5. Informe

1. **Abstract o Resumen**: es el resumen del trabajo.
2. **Introducción**: aquí se describe cada uno de los algoritmos implementados y evaluados (10 %)
3. **Desarrollo**: aquí se describe el diseño e implementación de los programas necesarios para realizar sus experimentos. (40 %).

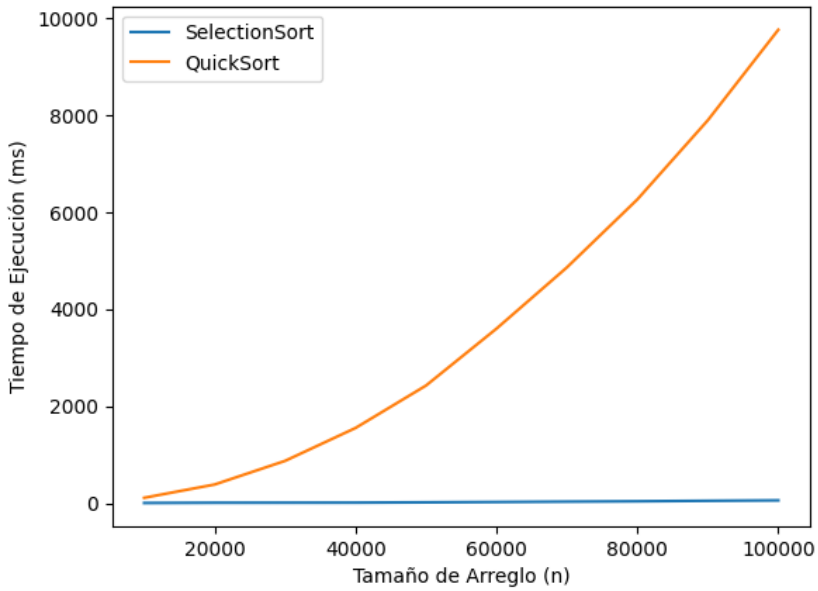

Figura 1: Ejemplo del comportamiento de dos métodos de ordenación, con respecto al tiempo de ejecución real para diversos tamaños de arreglos.

4. **Resultados Experimentales y Discusión**: aquí se presentan los resultados, pero lo más importante es analizarlos. Se debe observar y describir el comportamiento de los métodos. Es altamente importante relacionar los tiempos reales con los teóricos.

   Por favor, en esta sección deben incluir las características de la computadora sobre la que realizaron los experimentos. (40 %).

5. **Conclusiones**: ideas o hallazgos principales sobre el trabajo. (10 %)

6. Restricciones

1. Pueden trabajar en grupos de 2 estudiantes.
2. Todos los programas deben ser propios, permitiendo solamente utilizar el código disponible en el repositorio del curso `https://github.com/jmsaavedrar/eda_cpp/`.
3. El hallazgo de plagio será penalizado con nota 1.0, para todos los grupos involucrados.
4. Todos las implementaciones deben ser realizadas en C++.
5. **La entrega del informe es obligatorio**. Un trabajo sin informe no será calificado, asignando la nota mínima igual a 1.0.

7. Entrega

La entrega se debe realizar por canvas hasta el domingo 04 de septiembre, 2022, 23:50 hrs. La entrega debe incluir:

1. Código fuente (en C++), junto a un README con los pasos de compilación.
2. Informe