ASOInforme

Fabian Cabrera 20242020020
Sebastian Charry 20231020095

September 2025

1 Introduccion

En este informe, se detallan los resultados obtenidos con el programa realizado para organizar a los candidatos a presidentes de la ASO.

2 Metodologia

El programa toma medidas del tiempo (en nanosegundos), las comparaciones y los intercambios que realiza cada algoritmo de ordenamiento. Las medidas se tomaron a partir de 5 candidatos con 100.000 eventos por cada atributo, cada uno. Para el orden "casi ordenado", se ordenaron los datos utilizando el algoritmo Merge y se introdujo un "desorden" de alrededor del 10% de los datos.

3 Resultados

3.1 Burbuja

3.1.1 Tiempo

Obtuvo, en promedio:
1) Para orden aleatorio: 15.360 ns
2) Para orden inverso: 13.420 ns
3) Para casi ordenado: 13.200 ns
Total promedio: 13.994 aproximadamente.

3.1.2 Comparaciones

Para todos los tipos de orden inicial, se obtuvieron 10 comparaciones

3.1.3 Intercambios

Obtuvo, en promedio:
1) Para orden aleatorio: 3.2 intercambios

2) Para orden inverso: 7.4 intercambios
3) Para casi ordenado: 4.2 intercambios
Total promedio: 5 aproximadamente

—

3.2 Seleccion

3.2.1 Tiempo

Obtuvo, en promedio:
1) Para orden aleatorio: 14.660 ns
2) Para orden inverso: 10.500 ns
3) Para casi ordenado: 14.860 ns
Total promedio: 13.340 aproximadamente.

3.2.2 Comparaciones

Para todos los tipos de orden inicial, se obtuvieron 10 comparaciones

3.2.3 Intercambios

Obtuvo, en promedio:
1) Para orden aleatorio: 4 intercambios
2) Para orden inverso: 4 intercambios
3) Para casi ordenado: 4 intercambios
Total promedio: 4 aproximadamente.

—

3.3 Insercion

3.3.1 Tiempo

Obtuvo, en promedio:
1) Para orden aleatorio: 9.940 ns
2) Para orden inverso: 8.320 ns
3) Para casi ordenado: 8.360 ns
Total promedio: 8.873 aproximadamente.

3.3.2 Comparaciones

Obtuvo, en promedio:
1) Para orden aleatorio: 7.4 comparaciones
2) Para orden inverso: 7.2 comparaciones
3) Para casi ordenado: 5.8 comparaciones
Total promedio: 6.8 aproximadamente.

3.3.3 Intercambios

Obtuvo, en promedio:
1) Para orden aleatorio: 4.6 intercambios
2) Para orden inverso: 4.6 intercambios
3) Para casi ordenado: 2.8 intercambios
Total promedio: 4 aproximadamente.

—

3.4 Quick

3.4.1 Tiempo

Obtuvo, en promedio:
1) Para orden aleatorio: 17.560 ns
2) Para orden inverso: 14.020 ns
3) Para casi ordenado: 17.120 ns
Total promedio: 16.233 aproximadamente.

3.4.2 Comparaciones

Obtuvo, en promedio:
1) Para orden aleatorio: 7.2 comparaciones
2) Para orden inverso: 8.6 comparaciones
3) Para casi ordenado: 8.2 comparaciones
Total promedio: 8.0 aproximadamente.

3.4.3 Intercambios

Obtuvo, en promedio:
1) Para orden aleatorio: 6 intercambios
2) Para orden inverso: 7 intercambios
3) Para casi ordenado: 9.4 intercambios
Total promedio: 7.467 aproximadamente.

—

3.5 Merge

3.5.1 Tiempo

Obtuvo, en promedio:
1) Para orden aleatorio: 18.060 ns
2) Para orden inverso: 15.120 ns
3) Para casi ordenado: 16.620 ns
Total promedio: 16.6 ns.

3.5.2 Comparaciones

Obtuvo, en promedio:
1) Para orden aleatorio: 7.6 comparaciones
2) Para orden inverso: 6.6 comparaciones
3) Para casi ordenado: 6.6 comparaciones
Total promedio: 6.933 aproximadamente.

3.5.3 Intercambios

Obtuvo, en promedio:
1) Para orden aleatorio: 2.8 intercambios
2) Para orden inverso: 3.6 intercambios
3) Para casi ordenado: 2.6 intercambios
Total promedio: 3.0 aproximadamente.

3.6 Graficos

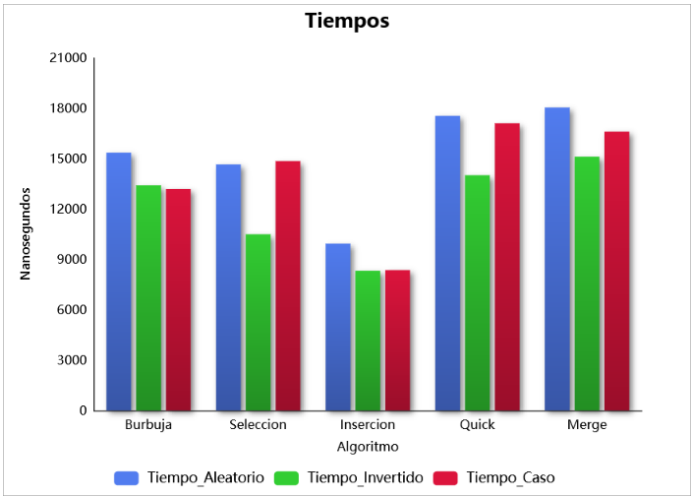

Figure 1: Tiempos

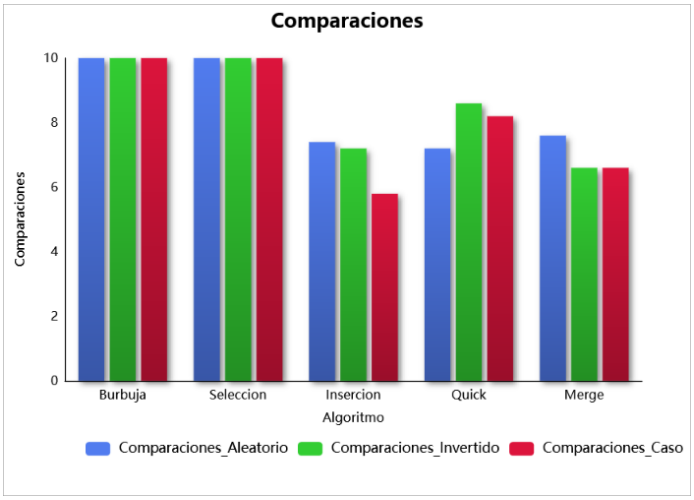

Figure 2: Comparaciones

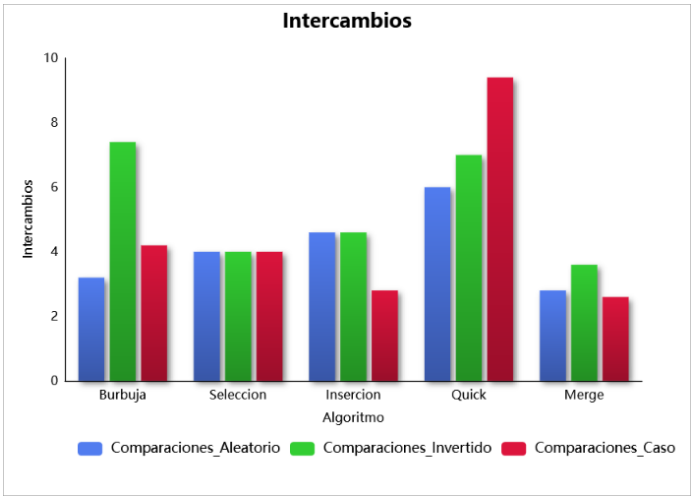

Figure 3: Intercambios

4 Conclusiones

4.1 Respecto al tiempo

Los algoritmos Merge y Quick son los algoritmos más "lentos", con 16.600 y 16.233 nanosegundos respéctivamente. Les siguen Burbuja, Seleccion y por ultimo Insercion, con 13.994, 13.340 y 8.873 réspectivamente.

4.2 Respecto a las comparaciones

Los algoritmos Burbuja y Seleccion comparten un primer puesto en cuanto a mayor cantidad de comparaciones, ambos con 10, les sigue Quick con 8, y Merge e Insercion empatan (aproximadamente) el ultimo puesto con 7 comparaciones.

4.3 Respecto a los intercambios

Quick realizó la mayor cantidad de intercambios con 7.5 en promedio, le sigue Burbuja con 5, Seleccion e Insercion empatan en la mitad con 4, y Merge ocupa el ultimo puesto con 4.

4.4 Respecto al orden inicial

A diferencia de lo esperado, la reduccion de tiempo más notable se da cuando el orden de los datos está invertido, no cuando la lista está casi ordenada (excepto con el algoritmo Burbuja, pero la reduccion es practicamente despreciable).

Los algoritmos de Burbuja y Seleccion presentan la particularidad de siempre realizar 10 comparaciones, que se corresponde al doble de la cantidad de candidatos comparados. Esto querria decir que estos algoritmos revisan la lista por completo en al menos dos ocasiones. Para el algoritmo de Insercion, la cantidad de comparaciones se redujo significativamente cuando los datos estaban casi ordenados, mientras que para Quick, aumentaron, especialmente cuando la lista se encontraba invertida. Para Merge, se redujo de igual forma cuando los datos estaban invertidos o casi ordenados.

El algoritmo de Seleccion presentó la particularidad de siempre hacer 4 intercambios, correspondiente a la cantidad de candidatos-1. La cantidad de intercambios para los algoritmos Burbuja y Quick aumentaron significativamente cuando el orden era invertido y casi ordenado, respectivamente. El algoritmo de Insercion realizó la misma cantidad de intercambios cuando el orden era invertido, más se redujo cuando el orden era casi ordenado. El algoritmo Merge realizó más intercambios cuando el orden era invertido y menos cambios cuando era ordenado, pero el cambio entre la cantidad de intercambios en orden aleatorio y casi ordenado es poco significativo.

4.5 Respecto al rendimiento

Viendo unicamente los valores obtenidos, el algoritmo de Insercion presentó el mejor rendimiento entre todos los demás algoritmos, mientras que Merge presentó el peor. Igualmente, el rendimiento del algoritmo Quick parece disminuir significativamente cuando los datos se encuentran casi ordenados, necesitando realizar muchos más intercambios de los que hace cuando el orden es aleatorio.

Asi, se puede concluir que debido a la naturaleza respectivamente pequeña de los datos, la utilización del algoritmo de Insercion resulta la más apropiada.