OBSERVACIONES DEL RETO 3

Estudiante 1 (Nathalia Quiroga) Cod 202013212
Estudiante 2 (David Valderrama) Cod 201910987

El presente documento hace un análisis de la complejidad y eficiencia, entendida como consumo de datos y tiempo de ejecución, del reto 4.

Análisis de complejidad

- El requerimiento 1 tiene una complejidad estimada en O(V+E) para el algoritmo inicial de Kosaraju (*scc*) y luego de O(1.5) aproximadamente, esto porque se busca en la estructura que crea Kosaraju (*map*) si los dos vértices dados se encuentran en el mismo *cluster*.

- El requerimiento 2 tiene una complejidad estimada en O(V^2), ya que el teorema dice que realiza esta cantidad de operaciones (sumas y comparaciones) para determinar la longitud del camino mas corto de dos vértices en un grafo. Por otro lado, al hacer uso de *pathTo()* se le suma una complejidad aproximada de O(1.5) ya que se busca en la estructura creada (*map*) el camino que se requiere.

- El requerimiento 3 tiene una complejidad estimada de O(ElogV) + O(V^2) que resulta de la suma del uso del algoritmo *Prim(Eager)* y Dijsktra respectivamente.

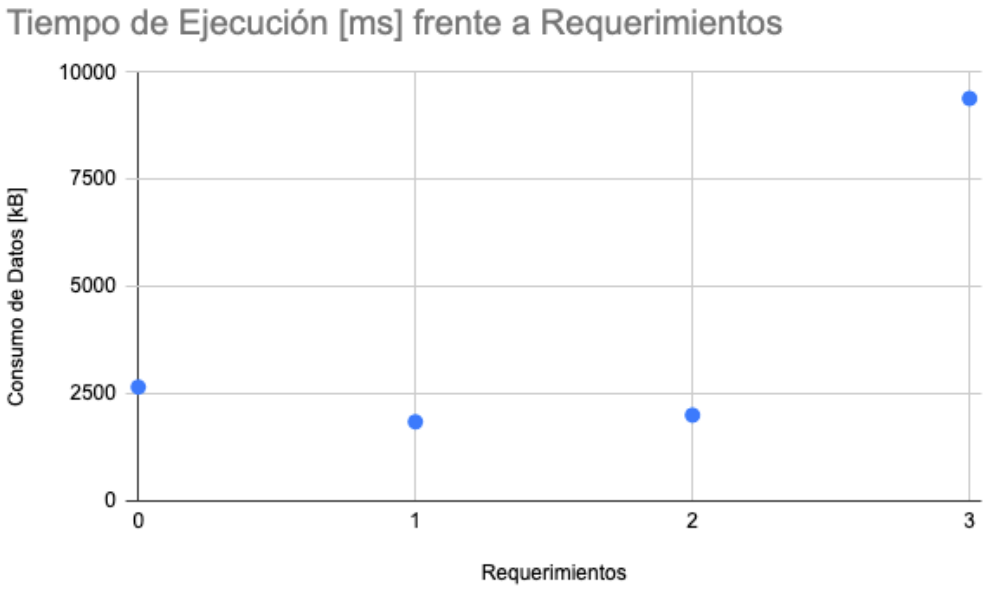
Registro de pruebas

Requerimientos	Consumo de Datos [kB]	Tiempo de Ejecución [ms]
Carga de datos	10608,752	2657,050
1	3425,216	1846,016
2	2372,841	2002,064
3	2945,803	9390,718

*Los datos presentados en las tablas son el promedio de 3 ejecuciones en una sola máquina.
**El análisis que se presentará a continuación solo contempla las funciones principales de cada requerimiento

Gráficas

*El 0 es equivalente a la carga de datos



*El 0 es equivalente a la carga de datos

Conclusiones sobre la eficiencia en pruebas

- En cuanto al espacio usado la carga de los datos naturalmente ocupó un mayor espacio con el fin de procurar tiempos más cortos de consulta y ejecución de los algoritmos sobre el grafo en los requerimientos. Los estuvieron bastante a la par, no obstante, el primer requerimiento ocupó más espacio en memoria que los demás, esto se debe al uso del algoritmo de Kosaraju, pues este necesita encontrar el grafo invertido y buscar sobre este almacenando la información en estructuras auxiliares.

- Naturalmente el requerimiento 3 tuvo la mayor complejidad temporal porque traspasa la estructura que obtiene de ejecutar el algoritmo Prim(Eager) al analizador y allí se vale de comparaciones por medio del algoritmo para establecer la ruta para llegar al último vértice de la rama más larga de ese grafo.