Análisis de resultados:

David Samuel Rojas Sánchez - 202214621 (ds.rojass1@uniandes.edu.co)

Marco Alejandro Ramírez Camacho – 202210308 (ma.ramirez23@uniandes.edu.co)

Análisis de complejidad:

Requerimiento 1:

Complejidad en notación O: $O(2V + E)$

Justificación: La función primero ejecuta un DFS cuya complejidad es $O(V + E)$. Luego, si hay camino entre las estaciones ingresadas por parámetro, se itera la pila correspondiente a los vértices que se recorren entre la estación inicial y la de destino. En el peor caso, la pila contiene todos los vértices del grafo, permitiendo que la complejidad del requerimiento sea de $O(2V + E)$.

Requerimiento 2:

Complejidad en notación O: $O(2V + E)$

Justificación: En este requerimiento se empieza por ejecutar el algoritmo BFS cuya complejidad en el peor caso es $O(V + E)$. Posteriormente, de existir camino entre las estaciones ingresadas por parámetro, se itera la pila que contiene las estaciones de las cuales consta el camino entre los vértices inicial y destino. En el peor caso, el camino contiene todos los vértices del grafo, lo cual se acota por $O(2V + E)$

Requerimiento 3 (David Samuel Rojas Sánchez - 202214621):

Complejidad en notación O: $O(V + E + V^2)$ (Sin tener en cuenta ordenamiento)

Justificación: La función que satisface este requerimiento primero hace un llamado al algoritmo de Kosaraju para identificar los componentes conectados del grafo en un tiempo $O(V + E)$. Posteriormente, se procede a reorganizar la respuesta que devuelve la implementación de DISC.lib lo cual es, en términos generales, la creación de una estructura auxiliar y la iteración de una estructura que contiene todos los vértices del grafo (acción que se realiza en un tiempo $O(V^2)$ en el peor caso). En consecuencia, el procedimiento es acotado mejor por $O(V + E + V^2)$.

Requerimiento 4 (Marco Alejandro Ramírez Camacho – 202210308):

Complejidad en notación O: $O(V^2 \cdot E \cdot \log(V))$

Justificación: Primero se procede a encontrar las estaciones más cercanas a las dos ubicaciones geográficas ingresadas por parámetro en un tiempo $O(N)$ con N igual al total de estaciones con ayuda de un hash y la iteración de cada uno de los vértices del grafo. Luego, se procede a crear un árbol RBT ordenado según el peso de los caminos correspondientes a las posibles estaciones origen y destino con ayuda del algoritmo de Dijkstra. Teniendo en cuenta las iteraciones que se realizan sobre la totalidad de los vértices, la complejidad de este requerimiento debería ajustarse a $O(V^2)$ y a $O(E \cdot \log(V))$ por las invocaciones a Dijkstra.

Requerimiento 6:

Complejidad en notación O: $O(V \cdot E \log(V))$

Justificación: En este requerimiento se filtran todas las estaciones que pertenecen al barrio destino. Luego, se iteran ejecutando el algoritmo de Dijkstra $O(E \log(V))$ para cada una desde la estación origen. Luego se elige el camino más conveniente de acuerdo al peso de cada camino con ayuda de un mapa ordenado. Teniendo en cuenta que, en el peor de los casos, es decir, cuando todos los vértices hacen parte del mismo vecindario, el algoritmo de Dijkstra se ejecuta el mismo número de veces que la cantidad de vértices, todo el requerimiento se ve mejor acotado por $O(V \cdot E \log(V))$

Requerimiento 7:

Complejidad en notación O: $O(V^2)$

Justificación: En este requerimiento, lo primero que se hace es iterar los vértices del grafo en busca de adyacentes que se comuniquen con la estación inicial $O(V)$. Luego se iteran los adyacentes con el propósito de corroborar que se dirijan al origen. Entonces, se filtran aquellos vértices cuyo camino no cumpla los requisitos del instructivo, para luego, ejecutar DFS y asegurar la existencia de un camino circular desde la estación destino. Si bien DFS tiene una complejidad $O(E + V)$, en el peor de los casos, las instrucciones que tendrán más peso en la complejidad temporal serán aquellas que se ejecutan en la iteraciones del comienzo, ajustándose así la complejidad en términos generales del requerimiento a $O(V^2)$ en notación Big O.

Pruebas de tiempos de ejecución y memoria utilizada:

Requerimiento 1:

Tabla tiempos de ejecución:

Porcentaje de la muestra	REQ 1 [ms]
0.5%	0.1
5%	0.31
10%	10.73
20%	1.21
30%	1.66
50%	2.4
80%	92.09
100%	147.79

Tiempo [Milisegundos] vs porcentaje muestra

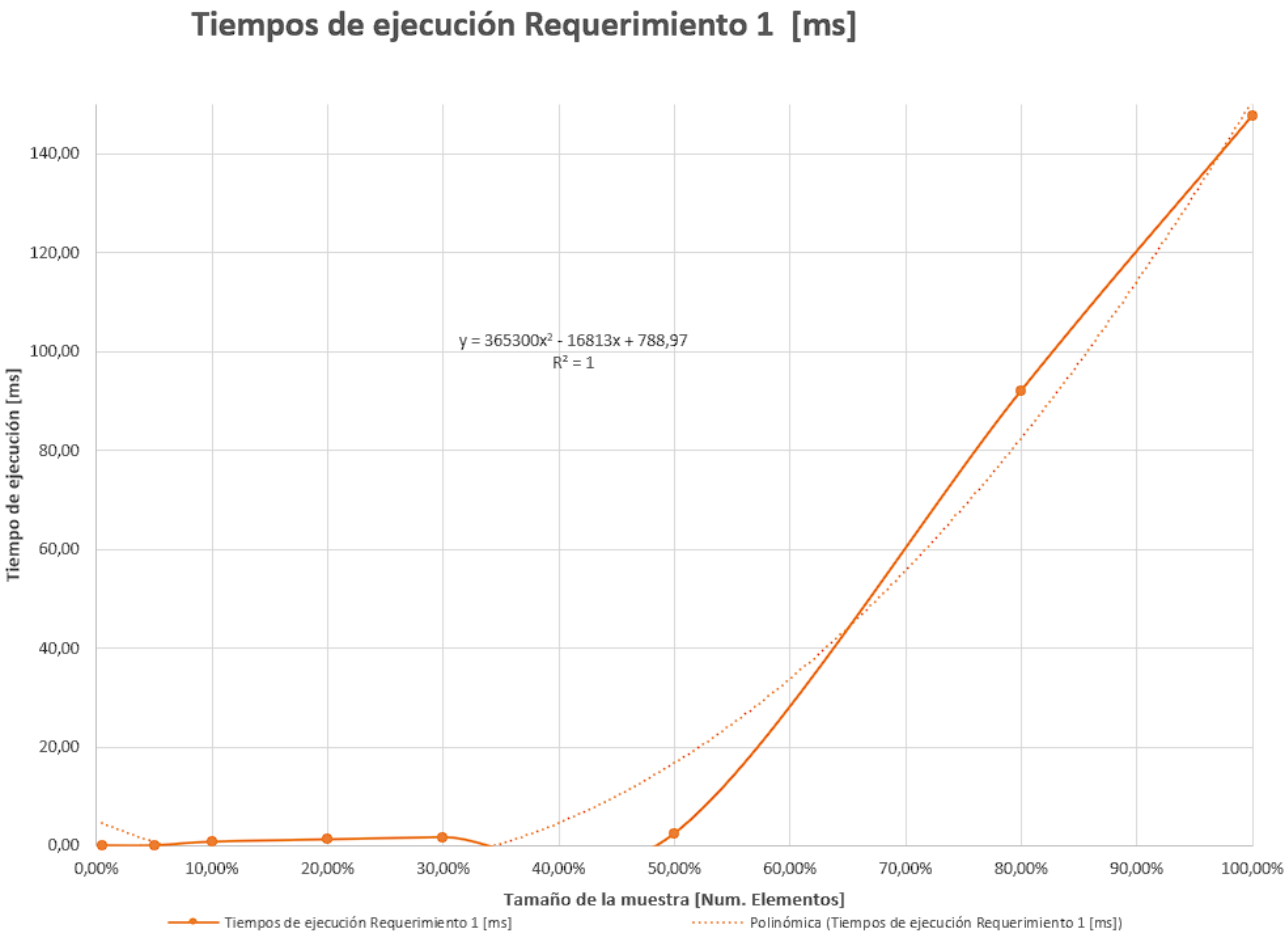
Tabla memoria utilizada:

Porcentaje de la muestra	REQ 1 [kB]
0.5%	1.2
5%	2.12
10%	3.12
20%	14.84
30%	21.23
50%	35.23

80%	57.82
100%	65.31

Memoria consumida [kB] vs porcentaje muestra

Gráfica:



Comparación con estimación: Se evidencia que la gráfica se comporta de manera lineal, como se predecía, solamente cuando se cuenta con más de la mitad del volumen total de datos. Esto puede suceder porque no se cuenta con el peor caso en las pruebas. Esto es, cuando el camino a la parada destino tiene una longitud cercana al número total de vértices. Se aproximan mejor a las condiciones del peor caso cuando el volumen de datos crece. Recordemos que el algoritmo DFS se ejecuta de manera distinta cada vez que se invoca.

Requerimiento 2:

Tabla tiempos de ejecución:

Porcentaje de la muestra	REQ 2 [ms]
0.5%	0,31
5%	0,45
10%	12,02
20%	10,20
30%	13
50%	50,04
80%	92,09
100%	152,15

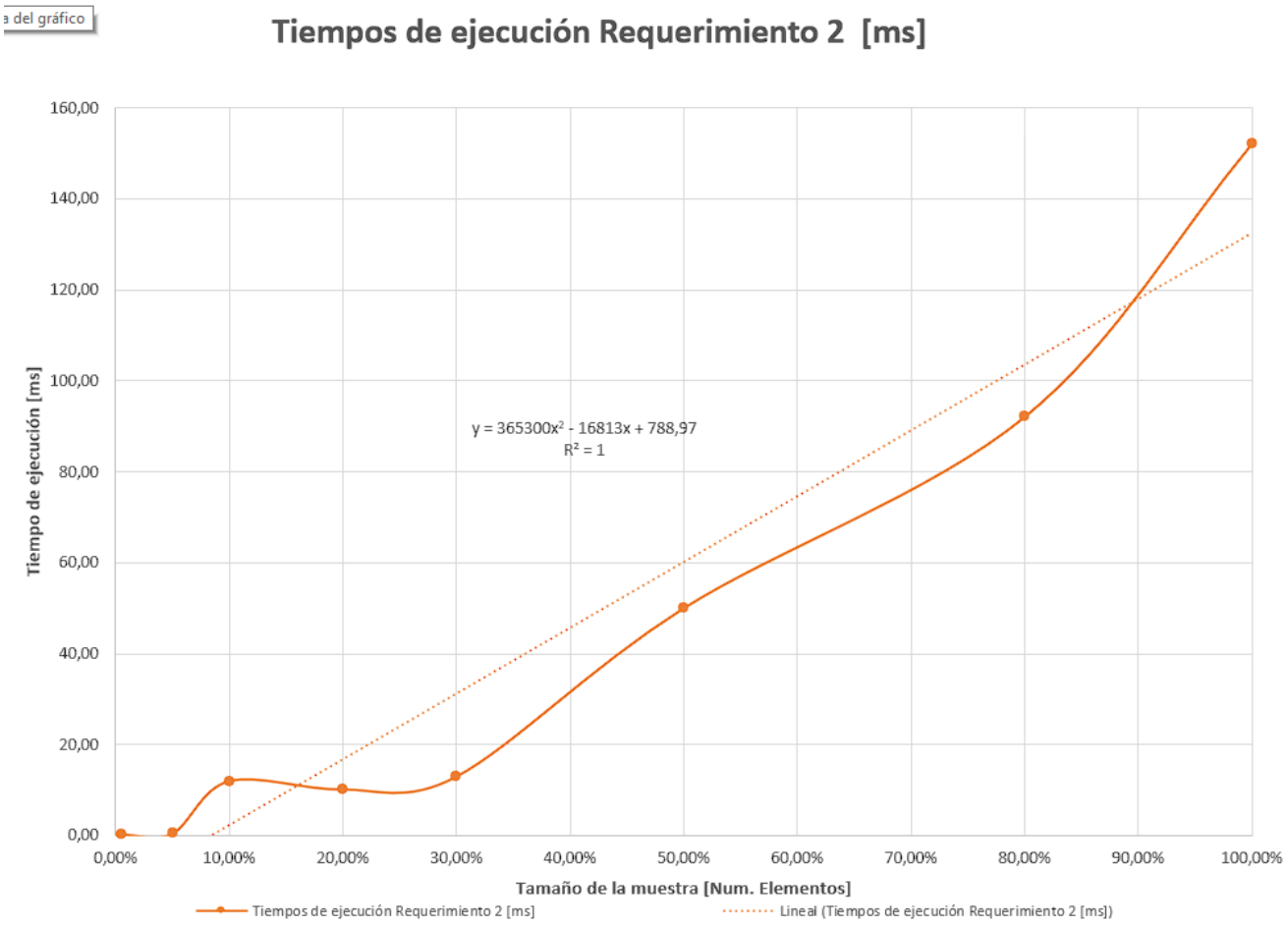
Tiempo [Milisegundos] vs porcentaje muestra

Tabla memoria utilizada:

Porcentaje de la muestra	REQ 2 [kB]
0.5%	7.17
5%	7.17
10%	2.12
20%	13.84
30%	19.23
50%	33.12
80%	51.08
100%	63.61

Memoria consumida [kB] vs porcentaje muestra

Gráfica:



Comparación con estimación: Como en el requerimiento anterior, los tiempos de ejecución parecen variar aleatoriamente, sin embargo, en ciertos intervalos de datos es posible apenas notar un comportamiento lineal tal y como se estimó. Lo que acontece en las pruebas reales, es que los datos no favorecen situaciones cercanas al peor caso y, además, los algoritmos suelen variar su ejecución cada vez que son invocados, modificando así el camino entre dos estaciones. No obstante, se cree que la estimación realizada anteriormente es buena, pues cumple el propósito de una notación Big O, es decir, acota por encima a las mediciones.

Requerimiento 3:

Tabla tiempos de ejecución:

Porcentaje de la muestra	REQ 3 [ms]

0.5%	27,10
5%	46,78
10%	99,55
20%	201,92
30%	289,55
50%	494,7
80%	2239,51
100%	4143,34

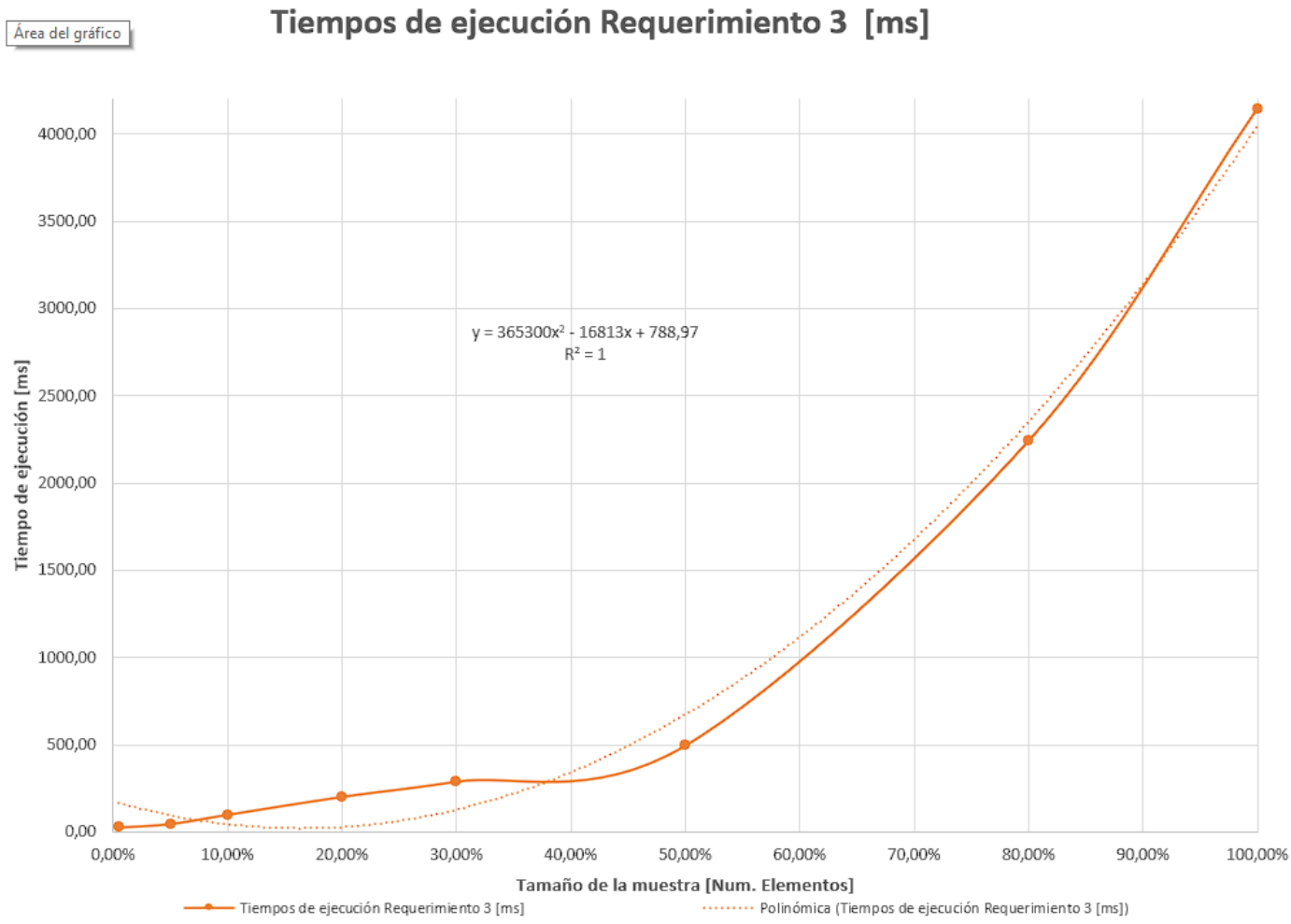
Tiempo [Milisegundos] vs porcentaje muestra

Tabla memoria utilizada:

Porcentaje de la muestra	REQ 4 [kB]
0.5%	0.78
5%	3.67
10%	4.56
20%	16.84
30%	19.23
50%	39.12
80%	68.08
100%	89.61

Memoria consumida [kB] vs porcentaje muestra

Gráfica:



Comparación con estimación: Se evidencia un claro comportamiento polinómico acorde con las estimaciones. No obstante, existen algunas variaciones que es posible atribuir a la implementación de DISClib del algoritmo de Kosaraju. Recordemos que el comportamiento polinómico se da gracias a la organización del hash que maneja la implementación del algoritmo, y esta reorganización de los datos se asemeja a una doble iteración que favorece en la mayoría de los casos la aproximación al peor caso.

Requerimiento 4:

Tabla tiempos de ejecución:

Porcentaje de la muestra	REQ 4 [ms]
0.5%	23,19
5%	26,42
10%	68,45
20%	121,21
30%	229,53
50%	304,21
80%	639,28
100%	2226,67

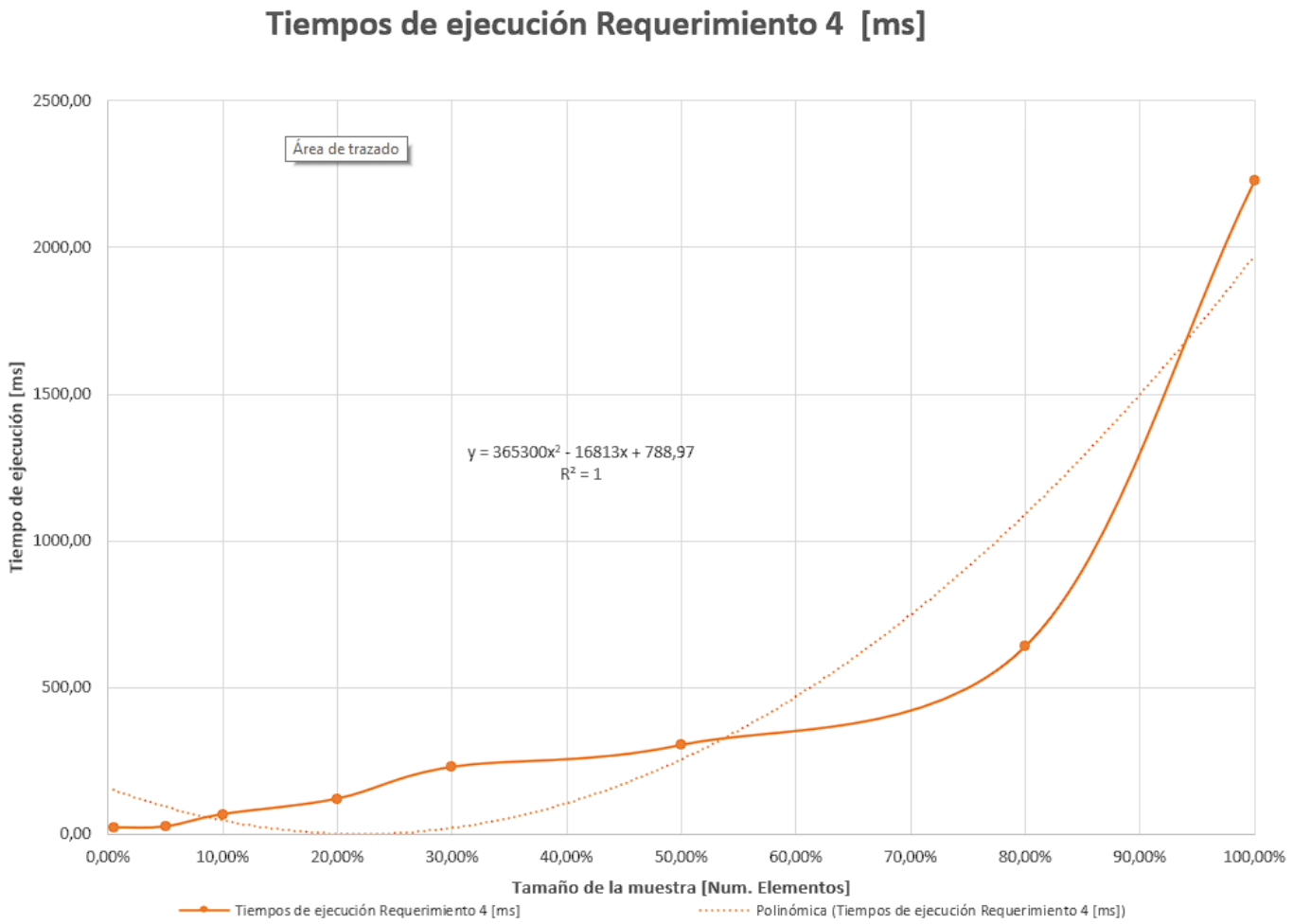
Tiempo [Milisegundos] vs porcentaje muestra

Tabla memoria utilizada:

Porcentaje de la muestra	REQ 4 [kB]
0.5%	0.48
5%	2.12
10%	2.12
20%	13.84
30%	19.23
50%	33.12
80%	51.08
100%	63.61

Memoria consumida [kB] vs porcentaje muestra

Gráfica:



Comparación con estimación: Es evidente que las mediciones varían ampliamente con un comportamiento bastante confuso, sin embargo, una vez más se tiene que la estimación para el peor caso es eficiente, dado que acota por arriba a las mediciones de cualquier forma. Está claro que el comportamiento cuadrático de las mediciones se atenúa más conforme la cantidad de datos crece, pero lo hace de una manera bastante pronunciada teniendo en cuenta que las iteraciones que representa la forma polinómica de la notación hacen referencia a ejecuciones del algoritmo de Dijkstra.

Requerimiento 6:

Tabla tiempos de ejecución:

Porcentaje de la muestra	REQ 6 [ms]
0.5%	22,12
5%	36,71
10%	88,21
20%	231,92
30%	519,55
50%	1003,09
80%	1804,04
100%	2495,21

Tiempo [Milisegundos] vs porcentaje muestra

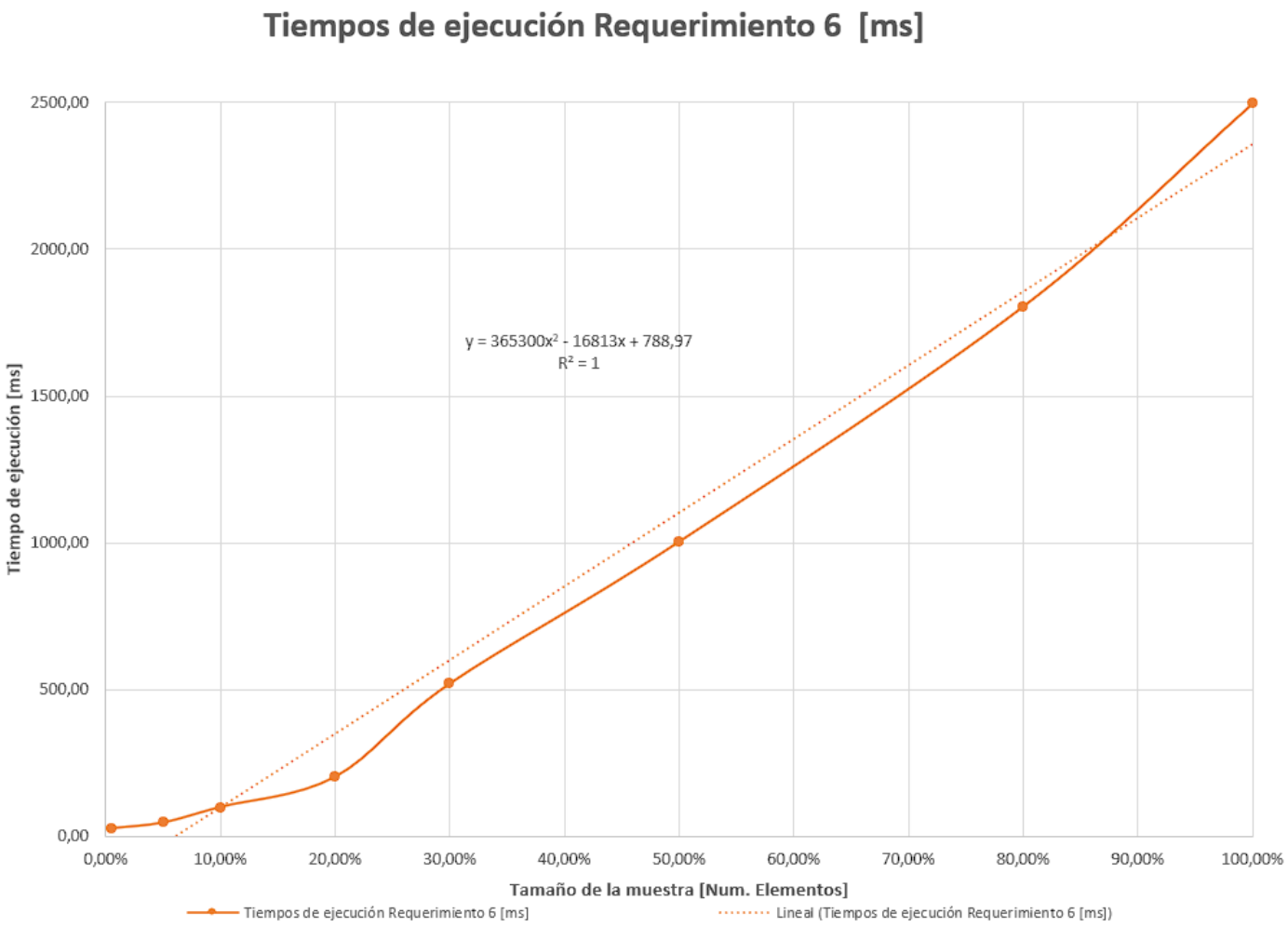
Tabla memoria utilizada:

Porcentaje de la muestra	REQ 6 [kB]
0.5%	0.48
5%	2.12

10%	2.12
20%	13.84
30%	19.23
50%	33.12
80%	51.08
100%	63.61

Memoria consumida [kB] vs porcentaje muestra

Gráfica:



Comparación con estimación: Es claro el comportamiento lineal logarítmico que se suponía. Esto dado que la ejecución del algoritmo de Dijkstra es la que más se repite en la función que satisface este requerimiento. Las únicas variaciones en los tiempos de ejecución las proporcionan las irregularidades en los datos.

Requerimiento 7:

Tabla tiempos de ejecución:

Porcentaje de la muestra	REQ 7 [ms]
0.5%	0
5%	16,42
10%	38,45
20%	41,21
30%	72,22
50%	110,02
80%	174,99
100%	244,66

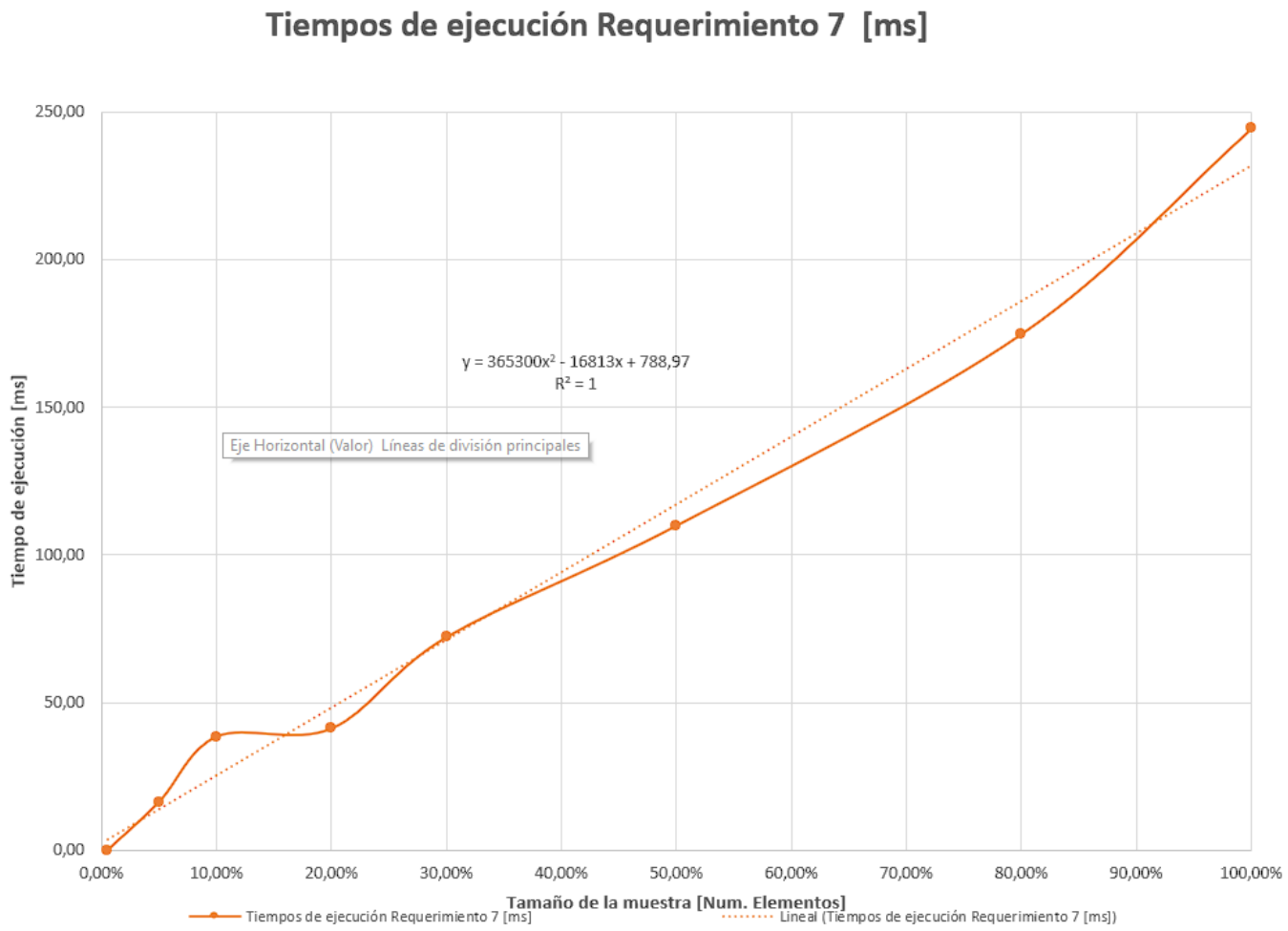
Tiempo [Milisegundos] vs porcentaje muestra

Tabla memoria utilizada:

Porcentaje de la muestra	REQ 7 [kB]
0.5%	73.77
5%	310.57
10%	2.12
20%	13.84
30%	19.23
50%	33.12
80%	51.08
100%	63.61

Memoria consumida [kB] vs porcentaje muestra

Gráfica:



Es evidente que las mediciones se acotan de manera más precisa por una función lineal. Esto sucede probablemente porque el peor de los casos no es posible encontrarlo en los datos. Recordemos que el peor de los casos sucedía cuando los adyacentes a la estación inicial son todos los vértices, y cada uno de ellos conforma un camino circular hacia el origen. De esta manera, se tiene que el comportamiento cuadrático predicho se pierde, y más bien, la cota superior se vuelve una función lineal propia de las ejecuciones del algoritmo DFS.