OBSERVACIONES DEL RETO

Anderson Arévalo – 202014997 – a.arevalom – Requisito 3
Manuel Pérez – 202021826 – m.perezp2 – Requisito 2

Análisis de Complejidades:

Requisito 1: La complejidad de este algoritmo en Big O es de O(N log N). Esto se debe a que el algoritmo únicamente utiliza funciones en las que, con las estructuras de datos utilizadas, la complejidad es de O(1), lo que da la complejidad a la función es el algoritmo de ordenamiento. Sin embargo, se espera que la complejidad no alcance estos niveles, ya que el algoritmo de ordenamiento se aplica en una lista mucho más pequeña que la original.

Requisito 2: La complejidad de este algoritmo en Big O es de O(N log N). Esto se debe a que el algoritmo únicamente utiliza funciones en las que, con las estructuras de datos utilizadas, la complejidad es de O(1) , lo que da la complejidad a la función es el algoritmo de ordenamiento. Sin embargo, se espera que la complejidad no alcance estos niveles, ya que el algoritmo de ordenamiento se aplica en una lista mucho más pequeña que la original.

Requisito 3: La complejidad de este algoritmo en Big O es de O(N)en el peor caso que correspondería al evento en el que todos los datos cargados tienen la popularidad digitada por el usuario, En los casos promedio la función tiene algoritmos de consulta que corresponden a O(1) y son las acciones que mas se repiten. Emplea un algoritmo de ordenamiento. Sin embardo este solo se aplica a la sublista creada y filtrada por la popularidad por ende es casi despreciable. Adicionalmente emplea un algoritmo adicional getTracksInfo que tiene un ciclo for en su interior. Sin embargo, este solo se aplica a la lista de autores de la canción que se va a imprimir por consola y se repetirá en su peor caso 6 veces para obtener los autores de las 3 primeras canciones y las 3 ultimas canciones. En conclusión el algoritmo presenta una complejidad de O(1) que es levemente aumentado por los ciclos mencionados. E incluso la complejidad que generan los ciclos, el ordenamiento y la recopilación de la información puede considerarse como despreciable.

Requisito 4: La complejidad de este algoritmo en Big O es de O(N) en su peor caso, que corresponde a que todos los datos ingresados correspondan al mismo autor ingresado por parámetro y todas las canciones estén disponibles en el país ingresado por parámetro. Es un algoritmo similar al anterior en el que la mayoría de las funciones se realizan en O(1) al ser funciones de consulta. Sin embargo esta complejidad puede aumentar levemente por los algoritmos de TracksByCountry y getTrackInfo que tienen ciclos for en su interior para traer las canciones de un país determinado y recopilar la información solicitada respectivamente. Sin embargo estos recorridos se realizan en sublistas por ejemplo el TracksByCountry realiza el recorrido únicamente en las canciones que pertenecen al artista ingresado por parámetro y el GetTrackInfo trae la información de la canción a imprimir en la terminal por lo cual son complejidades despreciables y se puede considerar que tiene una complejidad de O(1).

Requisito 5: La complejidad de este algoritmo para el caso de Big O es bastante elevada, O(N^2) para ser exactos. Esto partiendo de la base del peor caso, sin embargo, se espera que en las pruebas de tiempos el algoritmo tarde menos puesto a que las listas sobre las que se hacen los tres recorridos son más pequeñas que si fuera un O(N^2) sobre las listas originales.

Requisito 6: La complejidad de este algoritmo en Big O es de O(N) en el peor caso que corresponde a que todas las canciones ingresadas correspondan al artista ingresado por parámetro y estén disponibles en el país ingresado por el usuario. En general el algoritmo tiene una complejidad de O(1) dado que la gran mayoría de funciones que emplea son de consulta. Sin embargo, en la verificación de canciones por país se emplea TracksByCountry que tiene un ciclo en su interior y puede generar un aumento en la complejidad. Sin embargo el ciclo se realiza en una sublista que contiene únicamente las canciones del artista solicitado por el usuario. Por último hay un ciclo que trae la información de la canción que se debe imprimir en la terminal. En general las complejidades generadas por los ciclos son despreciables dado que solo actúan en sublistas y eso dejaría únicamente las funciones de consulta que tiene complejidad O(1).

Especificaciones de las Pruebas:

Las condiciones óptimas y en las que se harán las pruebas son con LINEAR_PROBING como estructura de datos y con el factor de carga de 0.5. Cuando se usan listas, se usan ARRAY_LIST con algoritmo de ordenamiento merge_sort, pues al revisar los datos se observo que estaban desordenados.

Las pruebas fueron hechas con las mismas entradas que se daban en los ejemplos del reto.

Tamaño de la Muestra	Req. 1	Req. 2	Req. 3	Req. 4	Req. 5	Req.6
0,5%	0,55	1,75	12,19	1274,68	10,52	484,98
5%	2,66	3,72	18,89	1290,97	334,56	450,74
10%	8,26	1,88	19,39	1374,02	629,38	710,34
20%	13,15	2,45	177,3	1415,46	1067,22	443,54
30%	17,13	3,61	108,32	1251,2	1172,44	582,33
50%	32,54	3,77	411,08	1594,27	2074,87	473,92
80%	46,02	3,87	536,25	1684,39	2641,18	596,35
100%	49,17	3,48	809,27	1891,25	2811,47	628,64

El tiempo registrado en las tablas se encuentra en ms.

Las pruebas fueron hechas con las mismas entradas que se daban en los ejemplos del reto.

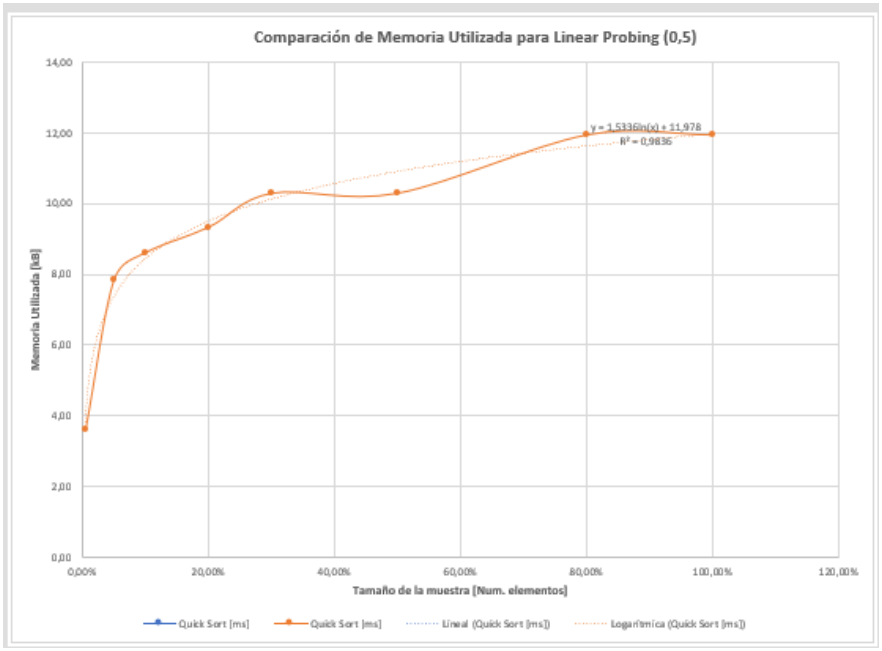
Tamaño de la Muestra	Req. 1	Req. 2	Req. 3	Req. 4	Req. 5	Req.6
0,5%	3,61	3,70	10,24	4528,9	2,71	2,975
5%	7,84	5,19	10,33	4529,77	7,98	4,28
10%	8,61	5,19	10,33	4529,77	8,88	4,28

20%	9,33	5,38	11,2	4528,9	8,57	4,81
30%	10,29	5,19	5,44	4529,04	9,66	2,54
50%	10,29	5,19	12,74	4529,04	9,69	5,32
80%	11,94	5,73	16,95	4529,05	9,69	7,31
100%	11,94	5,73	20,61	4529,06	9,69	8,62

La memoria registrada en las tablas se encuentra en kB.

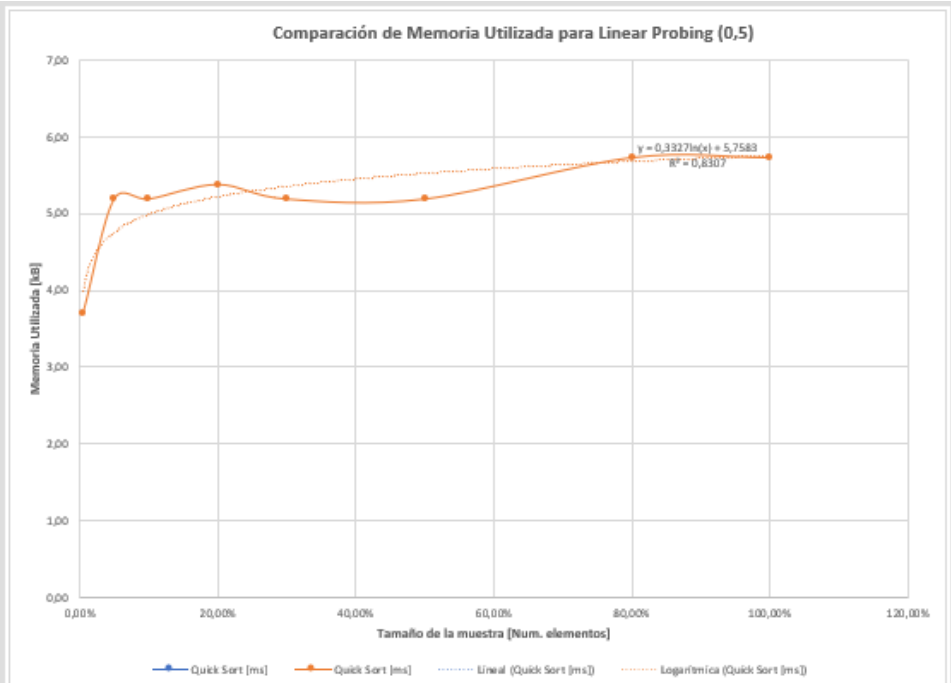
Gráficas Ordenes de Crecimiento de los Requisitos

Requisito 1:

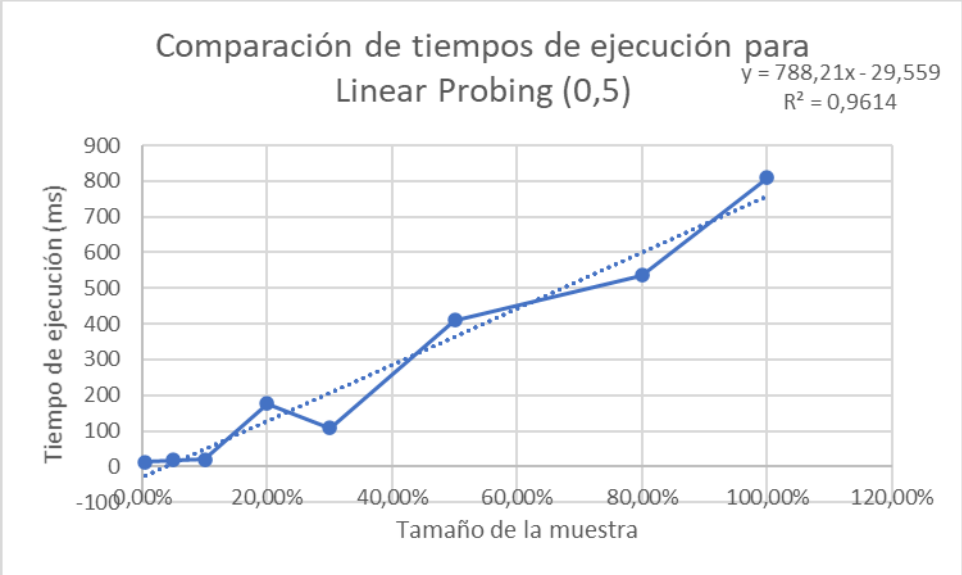

Comparación de Memoria Utilizada para Linear Probing (0,5)
Memoria Utilizada [kB]
Tamaño de la muestra [Num. elementos]
y = 1,5336ln(x) + 11,978
R² = 0,9836
Quick Sort [ms]
Quick Sort [ms]
Lineal (Quick Sort [ms])
Logarítmica (Quick Sort [ms])

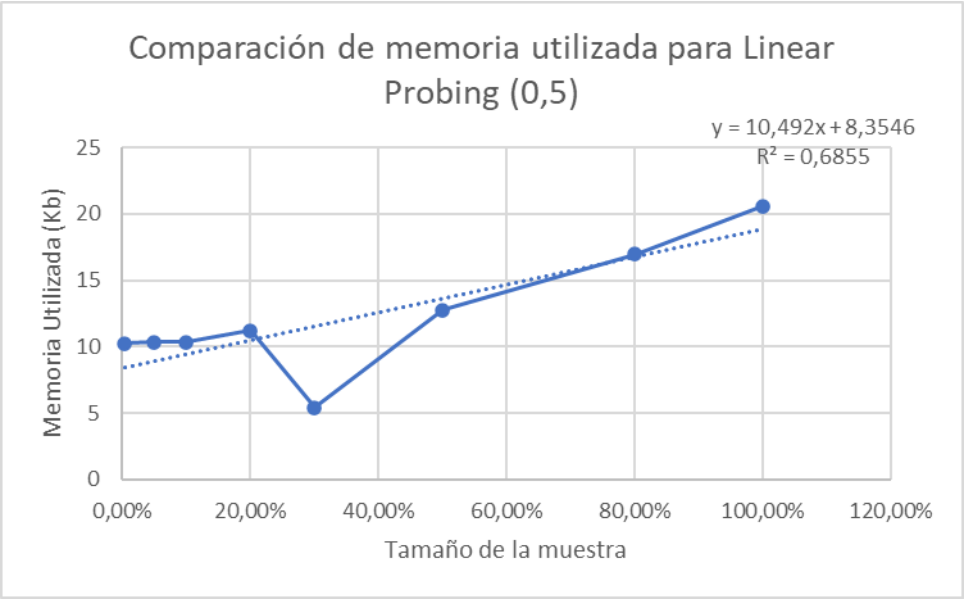
Requisito 2:


Comparación de tiempos de ejecución para Linear Probing (0,5)
Tiempo de Ejecución [ms]
Tamaño de la muestra [Num. elementos]
y = 0,3488ln(x) + 3,6909
R² = 0,4657
0,5 Load Factor [ms]
0,5 Load Factor [kB]
Logarítmica (0,5 Load Factor [ms])
Polinómica (0,5 Load Factor [kB])

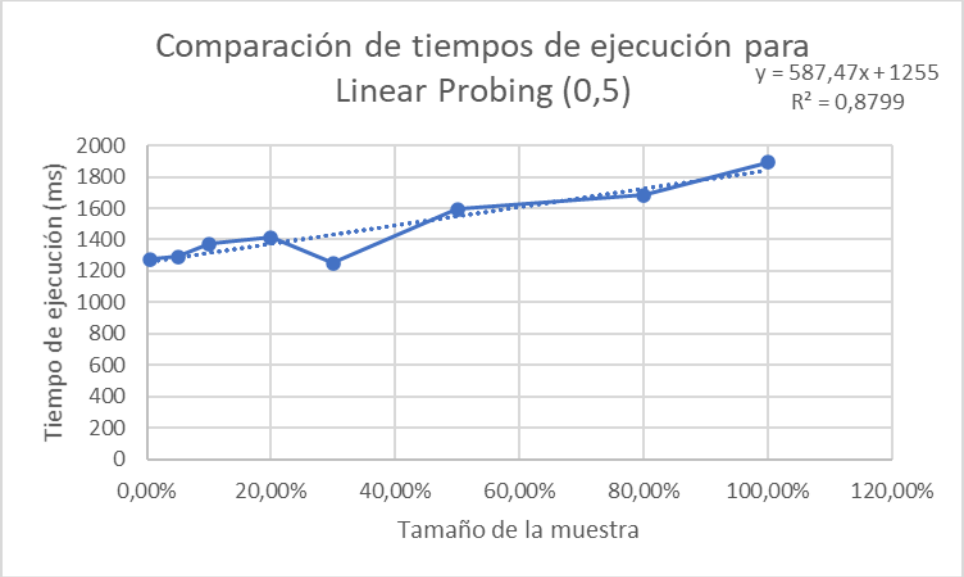

Comparación de Memoria Utilizada para Linear Probing (0,5)
Memoria Utilizada [kB]
Tamaño de la muestra [Num. elementos]
Quick Sort [ms]
Quick Sort [ms]
Lineal (Quick Sort [ms])
Logarítmica (Quick Sort [ms])

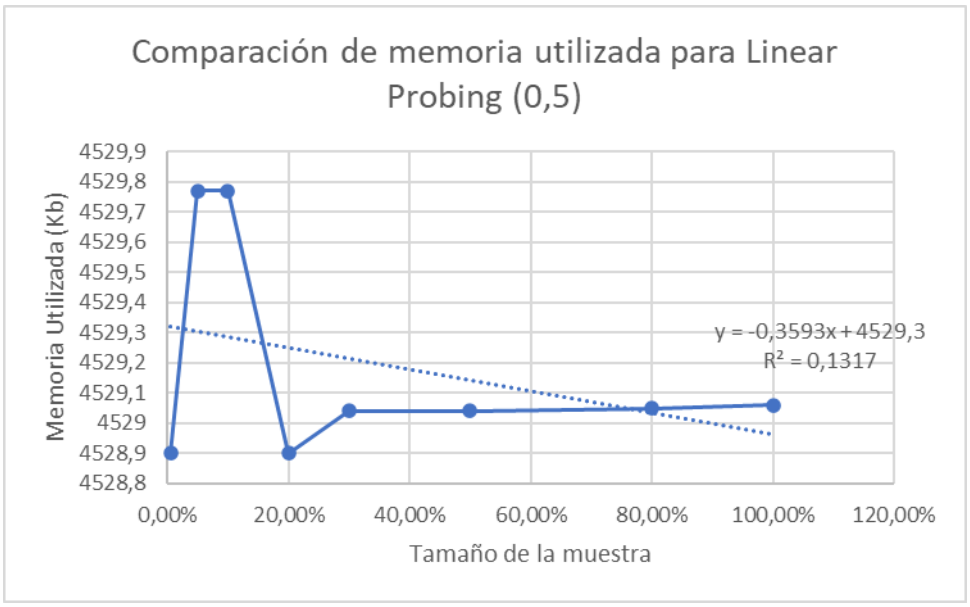
Requisito 3:


Comparación de tiempos de ejecución para Linear Probing (0,5)
y = 788,21x - 29,559
R² = 0,9614
Tiempo de ejecución (ms)
Tamaño de la muestra


Comparación de memoria utilizada para Linear Probing (0,5)
y = 10,492x + 8,3546
R² = 0,6855
Memoria Utilizada (Kb)
25
20
15
10
5
0
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra

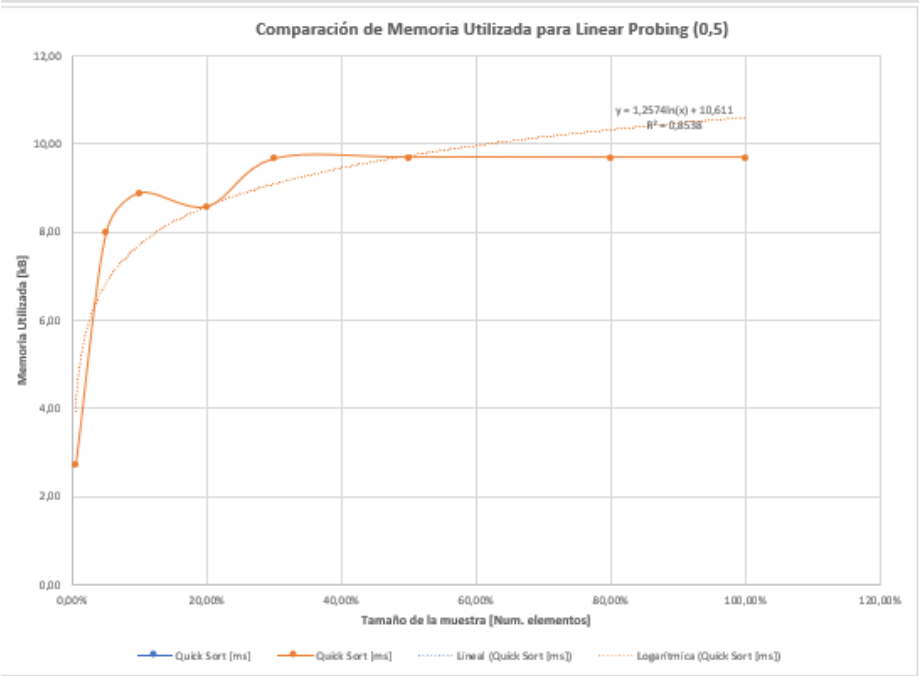
Requisito 4:


Comparación de tiempos de ejecución para Linear Probing (0,5)
y = 587,47x + 1255
R² = 0,8799
Tiempo de ejecución (ms)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra

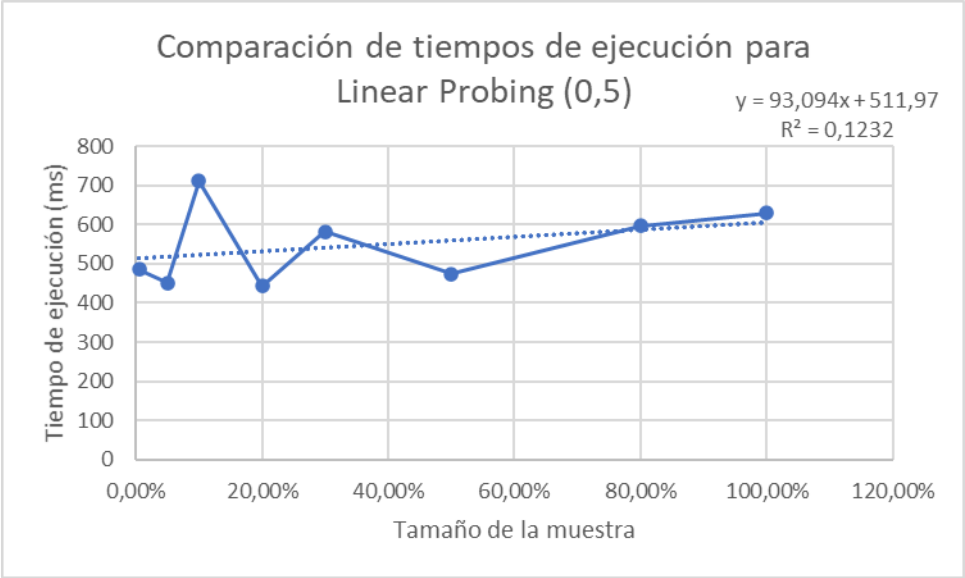

Comparación de memoria utilizada para Linear Probing (0,5)
Memoria Utilizada (Kb)
4529,9
4529,8
4529,7
4529,6
4529,5
4529,4
4529,3
4529,2
4529,1
4529
4528,9
4528,8
y = -0,3593x + 4529,3
R² = 0,1317
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra

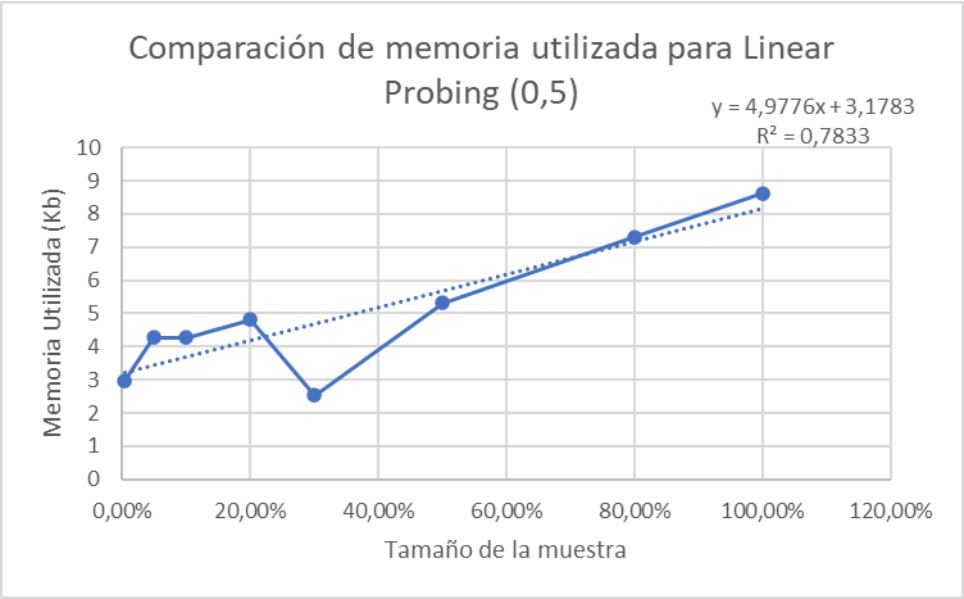
Requisito 5:


Comparación de tiempos de ejecución para Linear Probing (0,5)
Tiempo de Ejecución [ms]
3000
2500
2000
1500
1000
500
0
-500
-1000
y = 544,23ln(x) + 2317,3
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra [Num. elementos]
0,5 Load Factor [ms]
0,5 Load Factor [kB]
Logarítmica (0,5 Load Factor [ms])
Polinómica (0,5 Load Factor [kB])


Comparación de Memoria Utilizada para Linear Probing (0,5)
y = 1,2574ln(x) + 10,611
R² = 0,8538
Memoria Utilizada [kB]
12,00
10,00
8,00
6,00
4,00
2,00
0,00
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra [Num. elementos]
Quick Sort [ms]
Quick Sort [ms]
Lineal (Quick Sort [ms])
Logarítmica (Quick Sort [ms])

Requisito 6:


Comparación de tiempos de ejecución para Linear Probing (0,5)
y = 93,094x + 511,97
R² = 0,1232
Tiempo de ejecución (ms)
800
700
600
500
400
300
200
100
0
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra



Análisis de Resultados:

Requisito 1: En este reto, el requisito 1 tiene una complejidad de O(1), a pesar de que en el análisis de complejidades se advirtió que en el peor de los casos la complejidad sería de O(N logN), también se advirtió que llegar a este peor caso era sumamente improbable. En este caso la lista que se ordena es tan pequeña que el algoritmo de ordenamiento no termina afectando la complejidad del resto del algortimo. La complejidad de este requisito mejora por bastante la complejidad del primero, pues ya no se necesita un recorrido para nada.

En el reto 1, el requisito 1, arrojó una función muy parecida a lo que es una función lineal donde la operación más costosa era un recorrido a una lista en la primera parte del código.

Requisito 2: En este reto, el requisito 2 tiene una complejidad de O(1), a pesar de que en el análisis de complejidades se advirtió que en el peor de los casos la complejidad sería de O(N logN), también se advirtió que llegar a este peor caso era sumamente improbable. En este caso la lista que se ordena es tan pequeña que el algoritmo de ordenamiento no termina afectando la complejidad del resto del algortimo. La complejidad en este reto es mejor que en el primero, pues antes era logarítmico, y además era muy dependiente de la sub-lista que debía crear. Cabe aclarar que aunque en las gráficas el comportamiento sea similar al logarítmico, esta información es despreciable, ya que la diferencia entre el tiempo de carga del archivo más pequqño y el más grande es de 2ms.

En el reto 1, el requisito 2 dependía del ejemplo con el que se corra el algoritmo. Esto se puede explicar porque la operación más costosa es crear una sub-list, por lo que dependiendo de la entrada que se ingrese, la sublista puede ser muy corta o ser cercana al tamaña de la lista inicial. En las gráficas podemos ver como el algoritmo siempre tuvo un crecimiento similar al logarítmico,

al principio con un comportamiento un poco mejor que el logarítmico y para las últimas pruebas, un poco sobre la función logarítmica.

Requisito 3: Se observo una reduccion sustancial del tiempo empleado por los algoritmos con respecto al reto 1 y el empleado en el reto 2. Pese a que el reto 1 tenia complejidad O(NlogN) y en este caso es O(N) se observa que el reto 2 en este requerimiento emplea menor tiempo que el reto 1, tiene una diferencia inmenza entre los tiempos empleados por ambos requerimientos. Con las graficas se ratifico que la complejidad empleada por el algoritmo es O(N) generado principalmente por la funcion de consulta que trae la informacion solicitada. Dado que en general solo se utilizan funciones de consulta que son O(1). Observando la informacion recopilada se observa que presenta un comportamiento inferior a O(N) que incluso puede sugerir una complejidad de O(1), sin embargo en la grafica no es muy evidente producto de posibles errores en la toma de datos o mal rendimiento en la maquina empleada. Aun asi la complejidad del algoritmo es O(1) por la mayoria de funciones que emplea que corresponden a consultas siendo levemente afectadas por los ciclos incorporador en funciones utilizadas y que pueden considerarse despreciable ya que actuan en sublistas.

Requisito 4: En el reto 2 observados una complejidad igual al reto 1 es decir O(N). Sin embargo se observa que a mayor cantidad de datos el reto 2 emplearia menor tiempo que el reto 1. Adicionalmente la pendiente del reto 1 es mucho mayor que la obtenida en la ecuacion luego de graficar el tiempo de ejecucion obtenido del reto 2. Lo que sugiere mayor eficiencia en tiempo del reto 2 a medida que los datos aumentan. De esta forma podemos concluir que el uso adecuado de una estructura de datos permite agilizar el tiempo empleado. Dado que en este caso usamos consultas en maps con complejidad O(1) y no O(N) como en el reto 1, nos permite disminuir el tiempo que utiliza el algoritmo para resolver el requerimiento. El algorimo empleado en este requerimiemto en su mayoria tiene complejidad O(1) a excepsion de la funcion empleada para recopilar la informacion solicitada que es O(N) y la obtencion de canciones por ciudad. Sin embargo estas 2 ultimas funciones se ejecutan en sublistas lo que las hace casi despreciables en la complejidad del algoritmo por lo que en conclusion la complejidad sera O(1) y para el peor caso en caso que todas las canciones sean del artista solicitado y esten en el pais ingresado tendra complejidad O(N). En los casos promedio la complejidad de los ciclos es casi despreciable como semuestra en las graficas que presenta graficas cercanas a las constantes, demostrando una complejidad de O(1) afectada por los ciclos empleados unicamente.

Requisito 5: En este reto, el requisito 5 tiene un comportamiento similar al logarítmico tal como se puede apreciar en las gráficas. Una vez más, en el análisis de complejidades se dijo que la complejidad podía ser de O(N^2), sin embargo, las listas sobre las que se hace el doble recorrido son muy pequeñas y esta complejidad no se ve reflejada. Aunque el comportamiento de la gráfica es logarítmico, debido a que la diferencia entre tiempos de ejecución es tan pequeña, se podría considerar que la complejidad de este algoritmo es constante. Este requisito fue el que tuvo una mayor mejora con respecto al reto 1 pues la complejidad paso de ser casi N^1.5 a log N o constante.

En el reto 1, el requisito 5 tiene unos resultados peculiares, pues en el análisis se explico que el Big O es de O(N^3). Sin embargo, ahí mismo se tuvo la hipótesis de que, debido a que el triple

recorrido (que es el que da la complejidad al algoritmo) se hace sobre una sub lista que generalmente es mucho más pequeña que la lista grande, nunca se alcanzará el límite superior que arroja Big O. En las gráficas podemos evidenciar que la complejidad es de O(N^1,5), una complejidad que si bien es alta, no está ni cerca de ser lo que pudo haber sido según Big O.

Requisito 6: En este caso no podemos compararlo con el reto 1 dado que no logramos el requerimiento 6. Mediante las gráficas constatamos que la complejidad es O(1). Por medio de las gráficas se muestran graficas casi constantes debido a que el algoritmo utiliza en su gran mayoría funciones de consulta que son de complejidad O(1) levemente puede ser afectada por los ciclos que hay en funciones incluidas pero que solo se aplican en sublistas que corresponden a las canciones del artista. Por lo que la complejidad generada por las funciones externas es despreciable. En conclusión el algoritmo propuesto tiene una complejidad de O(1) en los casos promedio. La complejidad puede ser O(N) en el peor caso que corresponde cuando todas las canciones cargadas correspondan al artista ingresado por la terminal y se encuentren disponibles en el país ingresado por el usuario. En general para los casos promedios tienen complejidad O(1).