Departamento de Ingeniería de Sistemas y Computación
Estructuras de Datos y Algoritmos
ISIS-1225

ANÁLISIS DEL RETO 3

Carol Florido, 202111430, c.florido.@uniandes.edu.co — req 3

Laura Carretero, 202214922, l.carretero.@uniandes.edu.co — req 4

Jhostin Sánchez, 202214064, ja.sanchezm12.@uniandes.edu.co — req 5

Requerimiento 1

Descripción: Encontrar los videojuegos publicados en un rango de tiempo para una plataforma (G)

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	• Plataforma (Platforms). • Límite inferior de fecha de lanzamiento (Release_Date). • Límite superior de fecha de lanzamiento (Release_Date).
Salidas	• El número total de videojuegos disponibles en la plataforma. • El número de videojuegos disponibles en el rango de fechas de publicación para la plataforma. • Los 3 primeros y últimos registros disponibles en dicho rango.
Implementado (Sí/No)	Si, Grupal

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Adquirir el hash de game	O(1)
2. Adquirir del hash de game, el hash de plataformas_req1	O(1)
3. Adquirir del hash de plataformas_req1, el árbol correspondiente a la plataforma buscada	O(1)
4. Adquirir los elementos que se encuentran dentro del rango de tiempo buscado	O(logn) n=Elementos del árbol
TOTAL	***O(logn)***

Requerimiento 2

Descripción: Encontrar los 5 registros con menor tiempo para un jugador en específico (G)

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	Jugador
Salidas	Tabla con los 5 registros con menor tiempo
Implementado (Sí/No)	Si, Grupal

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Adquirir el hash de category	O(1)
2. Del hash de category, se adquiere el hash de jugadoresreq2	O(1)
3. Del hash de jugadoresreq2 se adquiere el árbol del jugador buscado	O(1)
5. Los valores del arbol se transforman a listas con la función .values()	O(n) n = Elementos en el árbol
TOTAL	***O(n)***

Requerimiento 3

Descripción: Conocer los registros más veloces en un rango de intentos (I)

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	• Límite inferior del número de intentos para romper el récord (Num_Runs). • Límite superior del número de intentos para romper el récord (Num_Runs).
Salidas	• El número de registros que cumplen con los criterios del rango de búsqueda. • Los 3 primeros y últimos registros disponibles en dicho rango.
Implementado (Sí/No)	Si, Carol Florido - 202111430

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Adquirir el hash de category	O(1)
2. Adquirir el árbol dentro del hash de category que tiene como llaves el número de runs de los intentos	O(1)
3. Adquirir una lista con los datos que se encuentran en el árbol dentro del rango indicado por el usuario.	O(logk) k = Elementos en el árbol
4. Iterar sobre los valores de la lista	O(p) p = cantidad de listas dentro de la lista
5. Los valores anteriores corresponden a listas, las cuales son organizadas a través de un merge sort.	O(slogs) s= Cantidad de elementos en la lista
6. Devolver la lista resultante	O(1)
TOTAL	***O(p*slogs)***

Requerimiento 4

Descripción: Conocer los registros más lentos dentro de un rango de fechas (I)

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	• Límite inferior de la fecha: hora en que se obtuvo el récord (Record_Date_0) • Límite superior de la fecha: hora de que se obtuvo el récord (Record_Date_0).
Salidas	• El número de registros que cumplen con los criterios del rango de búsqueda. • Los 3 primeros y últimos registros disponibles en dicho rango
Implementado (Sí/No)	Si, Laura Carretero - 202214922

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Adquirir el hash de “category”	O(1)
2. Adquirir el árbol dentro del hash de category que tiene como llaves el “Record_Date_0	O(1)
3. Adquirir la información dentro del árbol según los rangos de “Record_Date_0” dados por el usuario.	O(logn)
4. Iterar sobre los valores de la lista	O(p) p = cantidad de listas dentro de la lista
5. Los valores son organizados con un heapsort	O(nlogn) n = cantidad de elementos en la lista
6. Devolver la lista resultante	O(1)
TOTAL	**O(n(logn))**

Requerimiento 5

Descripción: Conocer los registros más recientes para un rango de tiempos récord

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	• Límite inferior de la duración para el mejor tiempo registrado (Time_0). • Límite superior de la duración para el mejor tiempo registrado (Time_0).
Salidas	• El número de registros que cumplen con los criterios del rango de búsqueda. • Los 3 primeros y últimos registros disponibles en dicho rango.
Implementado (Sí/No)	Si, Jhostin Sánchez - 202214064

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Se accede al rbt del Hash	O(2)
2. Se saca el rango del rbt	O(n)
3. Se organizan los heaps según el criterio	O(k*n log(n))
4. Se recorren 6 seis elementos de la lista y Se recorren 6 seis elementos de la sublista por cada lista	O(6*6=36)
5. Se imprime la tabla	O(1)
TOTAL	**O(n log(n))**

Requerimiento 6

Descripción: Diagramar un histograma de propiedades para los registros de un rango de años (G)

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	● Límite inferior del año de lanzamiento (Release_Date). ● Límite superior del año de lanzamiento (Release_Date). ● Número de segmentos en que se divide el rango de propiedad en el histograma (N) ● Número de niveles en que se dividen las marcas de jugadores en el histograma (x). ● La opción de consultar las siguientes propiedades de los registros: ○ El mejor tiempo registrado (Time_0) ○ El segundo mejor tiempo (Time_1) ○ El tercer mejor tiempo (Time_2). ○ El tiempo promedio registrado (el promedio de Time_0, Time_1, y Time_2). ○ El número de intentos registrados (Num_Runs).
Salidas	● El número total de registros consultados. ● El número total de registros incluidos en el conteo para graficar el histograma. ● Valor mínimo y valor máximo dentro del rango del histograma.

	• El histograma con la distribución de la propiedad dividido por rango y niveles.
Implementado (Sí/No)	Si, Grupal

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Adquirir el hash de category	O(1)
2. Adquirir el árbol dentro del hash de category que tiene como llaves el “Release_Date”	O(1)
3. Adquirir la información dentro del árbol según los rangos de “Release_Date” dados por el usuario.	O(logk) k= Elementos en el árbol
5. Se crea una lista de tipo “ARRAY_LIST” donde luego se añadirán los datos	O(n)
4. Se hace un for para recorrer los datos de la lista de “Release_Date”	O(p) p = cantidad de elementos dentro de la lista
5. Se hace un for para recorrer los datos de la lista anterior	O(m) m = cantidad de elementos dentro de la lista
6. Se realizan condicionales para ejecutar una opción dependiendo de la opción a consultar por el usuario	O(1)
7. Dentro de las opciones realiza otro if para validar la existencia de los “Num_Runs”, “Time_0, “Time_1” o “Time_2” antes de incluirlo en el conteo	O(1)
8. Si se valida su existencia, se añaden a la lista que creamos anteriormente con la operación de “addLast”	O(1)
9. En el caso de los promedio de “Time”, se crea una lista natal de python, a la cual, con un .append se le añadirán los tiempos 0,1 y 2 para luego realizar una operación sobre esta para hallar los promedios.	O(1)

10. Los elementos dentro de la lista se sumarán y dividirán por la cantidad de elementos dentro de la lista, y cada promedio será un diccionario.	O(1)
11. Dichos diccionarios se agregaran a la lista anteriormente creada con la operación “addLast”	O(1)
12. A la lista que tiene los datos se le hace una operación de ordenamiento, en este caso “merge sort” de manera que queden organizados de menor a mayor	O(nlog(n)) n= cantidad de elementos en la lista
13. Se retorna dicha lista a una función en el view que se encargará de graficarla según los segmentos y niveles dados por el usuario.	
TOTAL	***O(nlog(n))***

Requerimiento 7

Descripción: Encontrar el TOP N de los videojuegos más rentables para retransmitir (G)

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	• Plataforma de interés (Platforms). • TOP N de los videojuegos para retrasmitir (N)
Salidas	• El número total de videojuegos disponibles en dicha plataforma. • Los N videojuegos más rentables para transmitir.
Implementado (Sí/No)	Si, Grupal

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Se crea un nuevo árbol	O(1)
2. Se obtiene el hash de game	O(1)
3. Se obtiene el hash que tiene como llaves las plataformas de los juegos	O(1)

4. Se obtiene la lista que está dentro de la llave que coincide con la plataforma ingresada por el usuario	O(1)
5. Se recorre cada uno de lo juegos dentro de la lista	O(k) k = Cantidad de juegos dentro de la lista de la plataforma
6. A través de un split se adquiere el año del juego	O(1)
7. Con un condicional se transforma al formato requerido	O(1)
8. Se aplican una serie de condicionales sobre el año para conocer la Antiquity del juego	O(1)
9. Se saca logaritmo natural del total_runs del juego	O(1)
10. Del hash de category se adquiere el hash que tiene como llaves el Game_Id	O(1)
11. Se adquiere la lista que está dentro de la llave que coincide con la del juego	O(1)
12. se recorren cada uno de los intentos dentro de la lista hallada anteriormente.	O(p) p = Intentos dentro de la lista
13. Se saca el promedio de los tiempos y se agregan a una nueva lista.	O(1)
14. Se recorren los elementos de la lista anterior para calcular el revenue	O(m) m = Cantidad de elementos dentro de la lista
15. Se adquiere del hash de categoría, el hash que tiene como llaves las plataformas de los juegos dentro de los intentos.	O(1)
16. Se adquiere la lista dentro de la llave que coincide con la plataforma de interes	O(1)
17. Se recorren cada uno de los elementos dentro de la lista anterior	O(n) n = Cantidad de elementos dentro de la lista
18. Con un condicional se verifica que su Misc sea igual a Falso, si esto se cumple se suma 1 a un contador	O(1)

19. Dentro del hash de category se adquiere el hash que tiene como valores el Game_Id	O(1)
20. Se adquiere la lista que se encuentra dentro de la llave del Game_Id del juego	O(1)
21. Se hace un for para recorrer los datos dentro de la lista anterior	O(g) g= Cantidad de intentos dentro de la lista
22. Se hace un condicional para comprobar que el misc sea falso y que la plataforma coincida con la indicada, si esto se cumple se suma 1 a otro contador	O(1)
23. Se hace una división con los dos contadores	O(1)
24. Se asignan los valores encontrados a nuevas llaves dentro del diccionario del juego	O(1)
25. Se agrega el juego al árbol de acuerdo a el Stream Revenue encontrado	O(logn) n = elementos en el árbol
total	***O(k)***

Requerimiento 8 (Bono)

Descripción: Graficar la distribución de intentos por país en un rango de años de publicación (B)

Breve descripción de cómo abordaron la implementación del requerimiento

Entrada	Lim_inferior, Lim_superior, año
Salidas	folium.map
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
1. Accedemos al Hash con la info	O(1)
2. Accedemos al rbt	O(1)
3. Accedemos al los rangos inf, sup	O(n)
4. Se organiza los datos en un hash por país	O(N*n), N=size
5. Se recorre la información del hash para el cluster	O(n)
6. Se guarda y se abre el mapa	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Ambientes de pruebas

	Máquina
Procesadores	11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GH
Memoria RAM (GB)	16 GB
Sistema Operativo	Windows 10 Pro – 64 bits

Tabla 1. Especificaciones de la máquina donde se ejecutaron las pruebas de rendimiento.

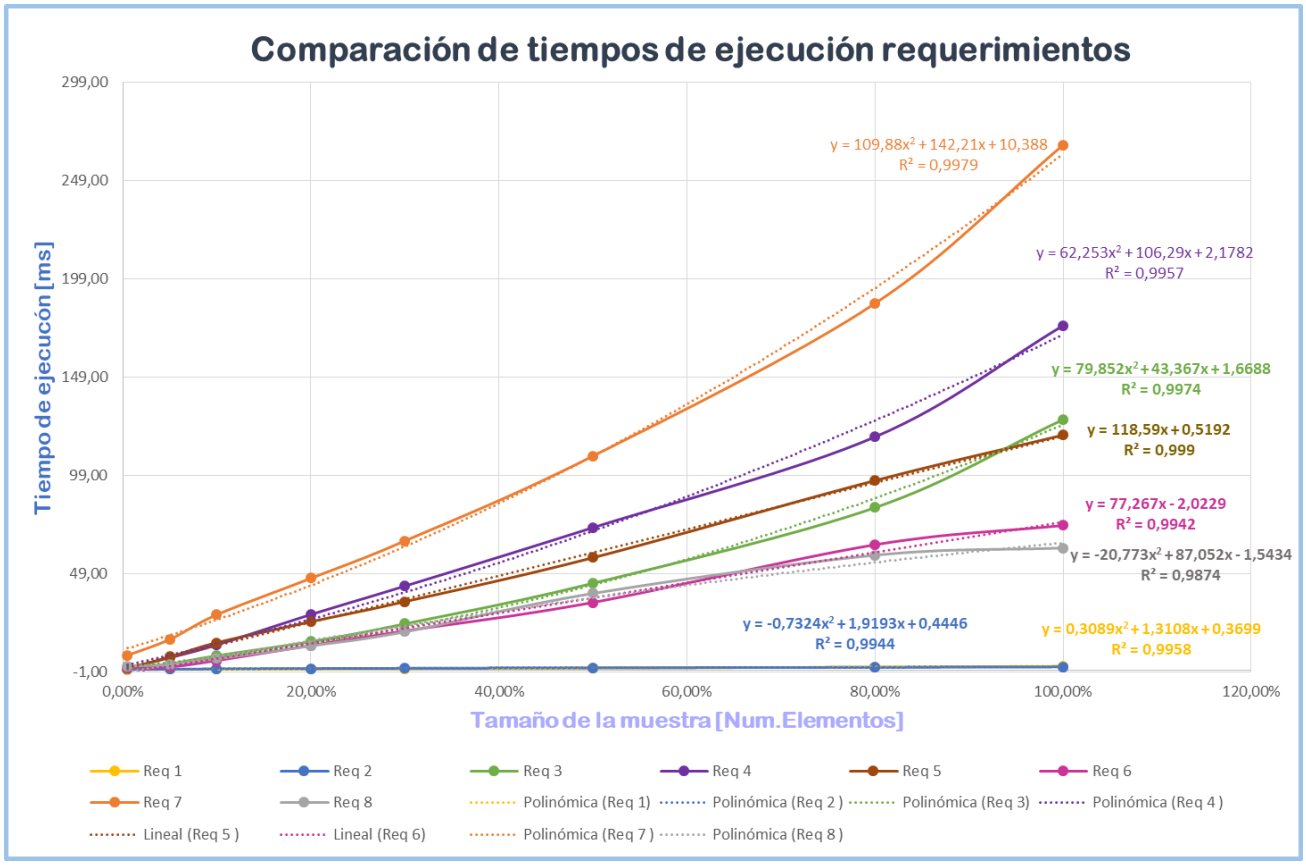
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

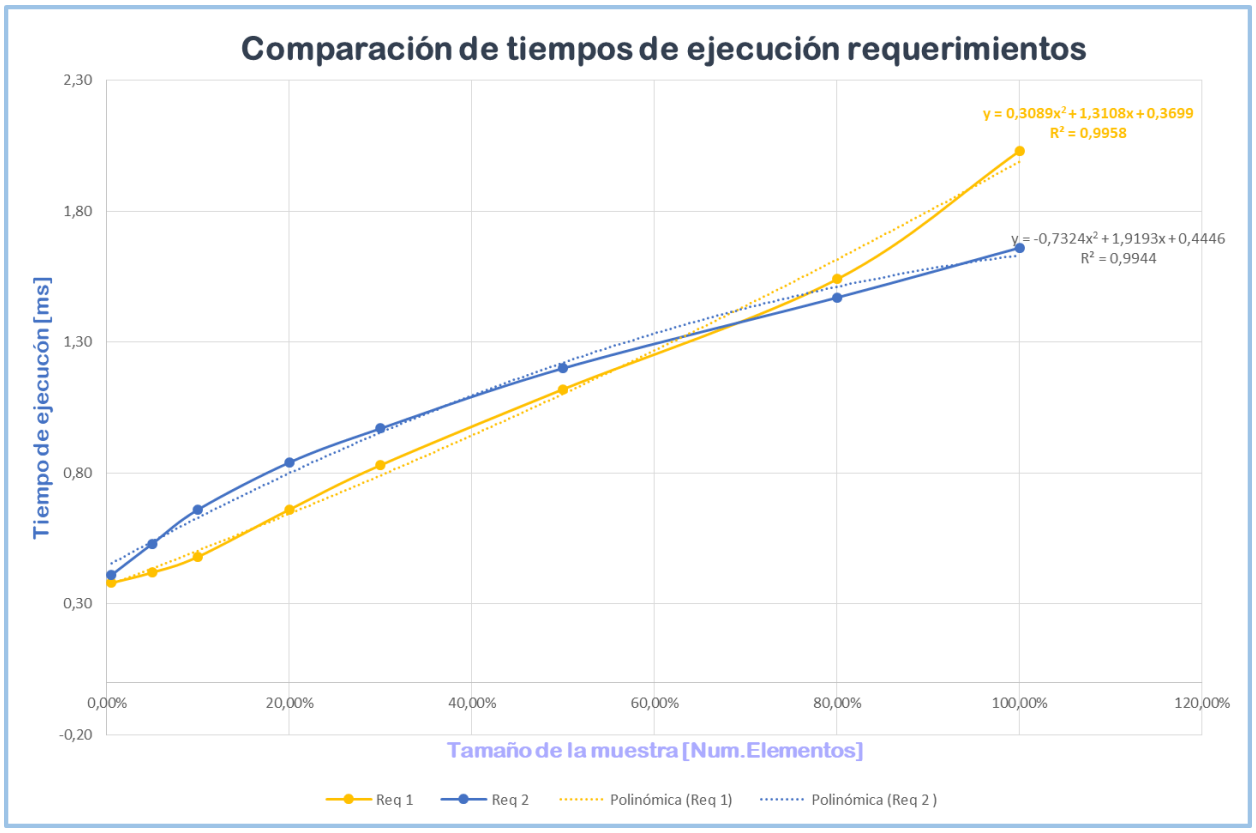
Tiempos de carga de requerimientos

Porcentaje de la muestra [pct]	Req 1	Req 2	Req 3	Req 4	Req 5	Req 6	Req 7	Req 8
0,50%	0,38	0,41	0,58	1,1	0,89	0,99	7,59	2,05
5,00%	0,42	0,53	3,23	6,4	6,78	1,68	15,66	3,09
10,00%	0,48	0,66	7,41	12,91	14,01	4,93	28,24	5,86
20,00%	0,66	0,84	14,55	28,32	24,68	12,57	46,77	12,38
30,00%	0,83	0,97	23,59	42,79	34,89	19,98	65,78	19,53
50,00%	1,12	1,2	44,23	72,54	57,23	34,49	108,93	39,05
80,00%	1,54	1,47	82,7	118,68	96,48	63,83	186,58	58,51
100,00%	2,03	1,66	127,51	175,29	119,62	73,67	267,13	62,2

GRÁFICA DE TIEMPOS DE EJECUCIÓN RETO 3



GRÁFICA DE REQ 1 Y 2



Análisis

Las pruebas que se desarrollaron fueron llevadas a cabo a través de un computador con procesador 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GH, una memoria RAM de 16.0 GB y un sistema operativo Windows 10 Pro - 64 bits.

Es posible observar como los requerimientos 1 y 2 son los que tuvieron los menores cambios en sus tiempos de ejecución a medida que el tamaño del archivo fue aumentando en su tamaño, esto es coherente puesto que dichos requerimientos son los que menores cantidades de operaciones desarrollan, ya que desde la carga de datos se puede obtener la gran mayoría de la información requerida. Por otro lado, los requerimientos individuales se comportaron de forma similar, lo cual es congruente puesto que todos tienen complejidades temporales cercanas entre sí. En cuanto al requerimiento 7, se puede observar que este fue el que obtuvo un mayor crecimiento temporal, esto puede explicarse en la gran variedad de “fors” que implementa su funcionamiento, los cuales crecen a medida que el tamaño de los archivos usados son mayores. De forma general, se puede evidenciar que el crecimiento temporal de los requerimientos si es acorde con lo analizado teóricamente. Sin embargo, es necesario considerar que dicho crecimiento va a variar dependiendo de la entrada que se use dentro del requerimiento.