ANÁLISIS DEL RETO

Laura Calderón, 202122045, l.calderonm
Manuela Lizcano, 202122826, m.lizcanoc
Mariana Pineda Miranda, 202123330, m.pinedam

Ambientes de pruebas

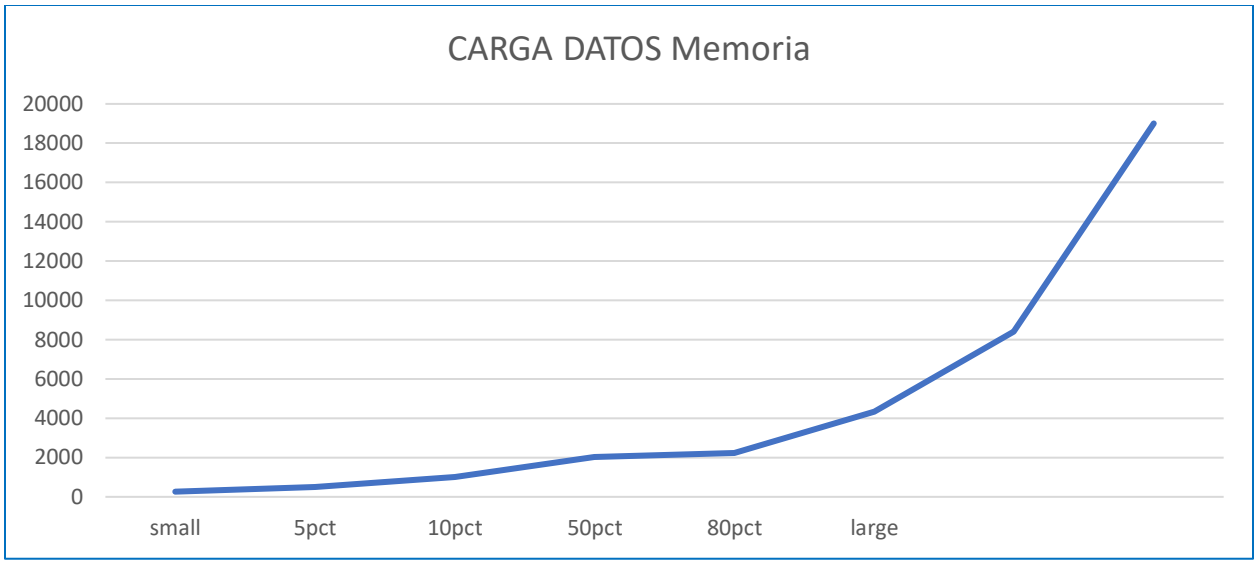
	Máquina 1	Máquina 2	Máquina 3
Procesadores	3,1 GHz Intel Core i5 de dos núcleo	2,00 GHz Ryzen 7 con gráfico de radeon	1,8 GHz Dual-Core Intel Core i5
Memoria RAM (GB)	8 GB	8 GB	8 GB
Sistema Operativo	MacOs	Windows 10	MacOs

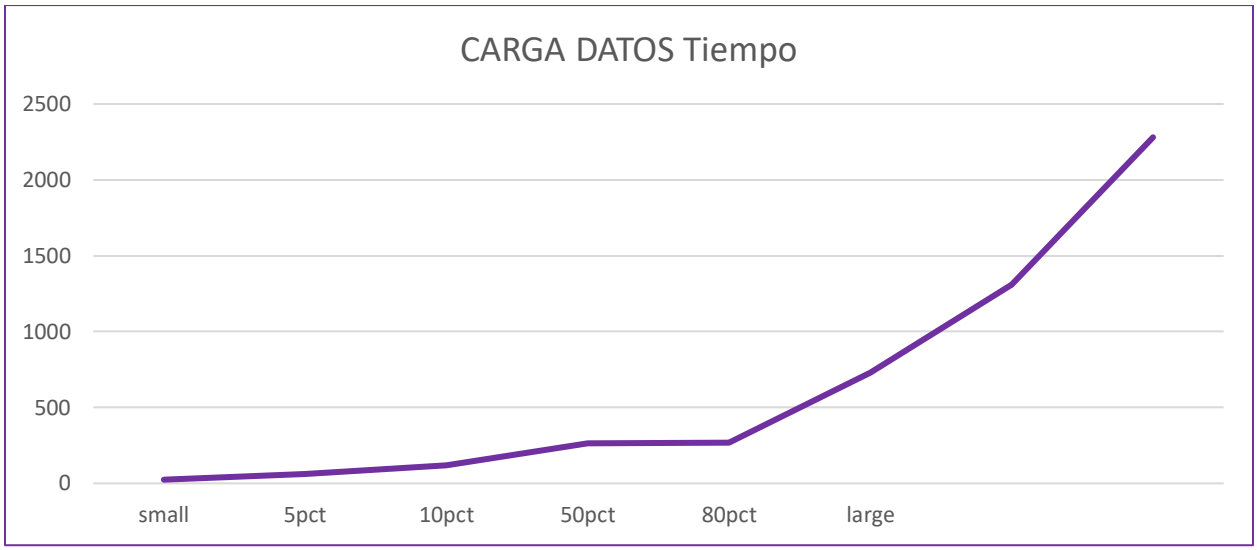
Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

Carga de datos

Carga de Catálogo

Factor de Carga	Consumo de Datos [kB]	Tiempo de Ejecución Real @LP [ms]
small	445,66	98,15
5pct	3253,09	479,36
10pct	5956,01	1061,89
20pct	10690,79	1864,73
30pct	15077,55	2890,94
50pct	23216,07	4484,52
80pct	34377,75	6656,32
large	41428,56	7921,40





Requerimiento 1

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Estación de inicio, estación final y catálogo de datos
Salidas	El camino recorrido de una estación a otra, el total de distancia recorrida, el total de estaciones que se recorrieron, el numero de transbordos.
Implementado (Sí/No)	Si se implementó.

Análisis de complejidad

Pasos	**Complejidad**

Paso 1. Se asigna al indice ‘path’ el recorrido en bfs. Se decidio que se recorreria con DFS debido a que solo se requiere econtrar un camino sin ninguna restricción.	O(E+V)
Paso 2. Se encuentra un camino con la función hasPathTo.	O(E+V)
Paso 3. Si es verdadero se busca el camuno por medio de la función pathTo, la cual devuelve el posible camino.	O(E+V)
Paso 4. Teniendo en cuenta el camino encontrado, se recorre el camino y se coinciden los vertices usados para poder encontrar la distancia total.	O(n)
Paso 5. Para saber el total de paradas se toma el tamaño de la lista.	O(n)
TOTAL	***O(E+V)***

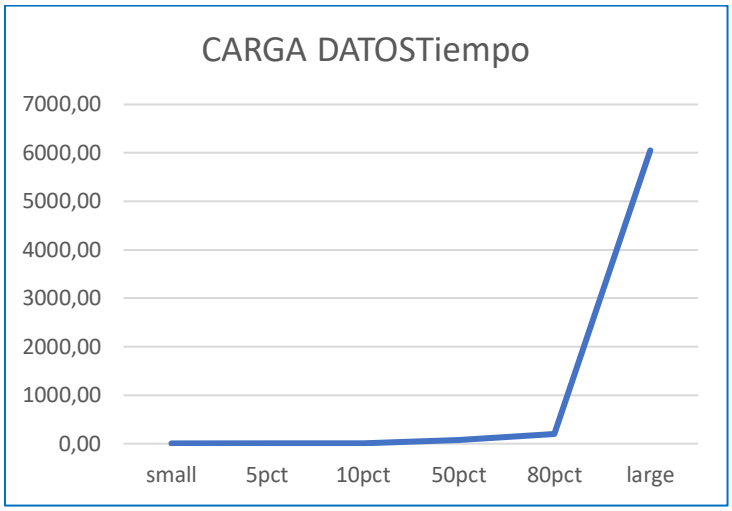
Pruebas Realizadas – Con máquina 3

Se realizaron las pruebas de tiempo de ejecución para este requerimiento con varios de los archivos variando su tamaño, cargando los datos por medio de un catálogo en el que se creó una lista con las películas y diferentes índices dependiendo de los requerimientos. Los datos se organizaron en un grafo el cual incluia los vertices y los arcos correspondientes a cada una de las estaciones las cuales conectaban la ruta de bus de Barcelona. Se ejecutaron las pruebas en tres computadores. Las pruebas se realizaron con los diferentes archivos con diferentes tamaños. Además, se usó la librería nativa de python time. Para realizar estas pruebas se utilizó como parametros, en primer lugar como estación de salida 1280-H16 y como estación de llegada 1856-H16 en cada una de las pruebas, ademas se repitió la prueba tres veces para tener un valor más exacto del resultado.

Tablas de datos

REQUERIMIENTO 1	
Tamaño archivo	**Tiempo**
small	5,32
5pct	5,83
10pct	6,72
50pct	81,06
80pct	204,32
large	6052,72

Graficas



Análisis

Teniendo en cuenta los resultado obtenidos por la pruebas y el analisis hecho sobre la complejidad del algoritmo, llegamos a la conclusión que se tiene una complejidad O(E+V) debido a que en cada archivo se aumenta tanto el numero de vertices como el numero de arcos la grafica tiende a ser exponencial. De igual manera, de este algoritmo podemos decir que no es el mas eficiente debido a que con BFS solo se asegura que exista un camino, sin embargo, el camino puede tomar de muchas estaciones las cuales son innecesarias de recorrer.

Requerimiento 2

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Estacion de inicio, estacion de fin y catalogo de datos
Salidas	El camino con el menor numero de paradas desde la estacion de inicio y estacion final, el total de distancia recorrida, el total de paradas que en este caso debe ser el minimo, el total de transbordos.
Implementado (Sí/No)	Sí se implementó

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1. Se asigna al indice ‘path’ el recorrido en bfs. Se decidio que se recorreria con BFS debido que se quiere encontrar el camino con el menor numero de paradas. El algoritmo DFS recorre el menor numero de vertices.	O(E+V)
Paso 2. Se encuentra un camino con la función hasPathTo.	O(E+V)
Paso 3. Si es verdadero se busca el camuno por medio de la función pathTo, la cual devuelve el posible camino.	O(E+V)

Paso 4. Teniendo en cuenta el camino encontrado, se recorre el camino y se coinciden los vertices usados para poder encontrar la distancia total.	O(E)
Paso 5. Para saber el total de paradas se toma el tamaño de la lista.	O(1)
TOTAL	***O(E+V)***

Pruebas Realizadas – Con máquina 3

Se realizaron las pruebas de tiempo de ejecución para este requerimiento con varios de los archivos variando su tamaño, cargando los datos por medio de un catálogo en el que se creó una lista con las películas y diferentes índices dependiendo de los requerimientos. Los datos se organizaron en un grafo el cual incluia los vertices y los arcos correspondientes a cada una de las estaciones las cuales conectaban la ruta de bus de Barcelona. Se ejecutaron las pruebas en tres computadores. Las pruebas se realizaron con los diferentes archivos con diferentes tamaños. Además, se usó la librería nativa de python time. Para realizar estas pruebas se utilizó como parametros, en primer lugar como estación de salida 1280-H16 y como estación de llegada 1856-H16 en cada una de las pruebas, ademas se repitió la prueba tres veces para tener un valor más exacto del resultado.

Tablas de datos

REQUERIMIENTO 2	
Tamaño archivo	**Tiempo**
small	1,34
5pct	7,56
10pct	8,36
50pct	50,22
80pct	238,05
large	425,99

Graficas

Análisis

Se puede observar que realizando el analisis de complejidad se llega a una complejidad temporal O(V+E), esto tambien se puede ver evidenciado en la grafica de los datos debido a que se cuando se aumenta el tamaño de uno de los archivos tambien se hace con el otro. A comparación del algoritmo usado en el requerimiento 1 ese algoritmo es mas optimo debido a que DFS recorre el grafo y pasa por el menor numero de estaciones, lo que significa que va solo a las estaciones necesarias.

Requerimiento 3

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Catalogo de datos
Salidas	Se desea encontrar el numero de componentes conectados, los 5 componentes con el mayor numero de vertices y el total de vertices en estos componentes.
Implementado (Sí/No)	Si se implementó. Hecho por Mariana Pineda Miranda.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Se asigna al indice ‘components’ del catalogo el algoritmo de Kosaraju aplicado al grafo.	O(E+V)
Paso 2: Para encontrar el numero de componentes de usa la función ‘connectedComponents’	O(E+V)
Paso 3: Se cuenta el total de elementos en cada uno de los componentes conectados, para asi encontrar los 5 con mas elementos.	O(n)
Paso 4: Se crea un mapa en el que se añaden cada uno de los componentes identificados por el numero de componente al que pertenecen.	O(1)
Paso 5: Se añanden cada uno de los vertices al mapa en el cual la llave es el nuemero de componente mientras que el valor es una lista de los componentes.	O(V)
Paso 6: Se imprimen los 5 componentes con mas elementos. Y sus correspondientes primeras 3 y ultimas 3 estaciones.	O(V)
TOTAL	***O(E+V)***

Pruebas Realizadas

Se realizaron las pruebas de tiempo de ejecución para este requerimiento con varios de los archivos variando su tamaño, cargando los datos por medio de un catálogo en el que se creó una lista con las películas y diferentes índices dependiendo de los requerimientos. Los datos se organizaron en un grafo el cual incluia los vertices y los arcos correspondientes a cada una de las estaciones las cuales conectaban la ruta de bus de Barcelona. Se ejecutaron las pruebas en tres computadores. Las pruebas se realizaron con los diferentes archivos con diferentes tamaños. Además, se usó la librería nativa de python time. En este caso no era necesario ingresar datos por parametro por lo que lo unico que se uso fue el catalogo de datos, ademas se repitió la prueba tres veces para tener un valor más exacto del resultado.

Tablas de datos

REQUERIMIENTO 3	
Tamaño archivo	**Tiempo**
small	2,70
5pct	52,19
10pct	101,15
50pct	694,48
80pct	1152,56
large	1371,42

Graficas

Análisis.

Se puede observar que la complejidad de este algoritmo es de O(V+E) y en la grafica se puede ver que se tiene un comportamiento que no llega a ser del todo exponencial, sin embargo tampoco es lineal. Se puede considerar que se tienen tiempos altos para este requerimiento debido a que se debe recorrer el grafo y entre mas vertices y arcos existan mayor sera la complejidad.

Requerimiento 4

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Los parámetros de entrada de esta función son el catálogo de datos, la longitud y latitud de la localización geográfica del origen del usuario y la longitud y latitud de la localización geográfica del destino del usuario.
Salidas	La respuesta de esta función es: La distancia de entre la localidad del origen del usuario hasta la estación de bus más cercana, la distancia total que tomara realizar el recorrido entre la estación de origen y la estación destino, la distancia entre la estación de destino y la localización destino del usuario, el número de estaciones que estan contenidas en el camino de solución junto con el número de transbordos que se deben realizar. Y, por último, debe contener para cada una de las estaciones que hacen parte del camino se debe incluir el identificador y la distancia respecto a la siguiente estación.
Implementado (Sí/No)	Si fue implementada. Realizada por Manuela Lizcano.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Se accede a la lista que se encuentra en el catálogo, la cual contiene las estaciones..	O(n)
Paso 2: Se hace un recorrido por todos los datos, con el objetivo de encontrar la estación más cercacanas a la localidad en donde se encuentra el usuario utilizando la función de haversine.	O(n)
Paso 3: Se crea una lista vacía, con el objetivo de buscar las localidades que pueden ser utilizadas como estación destino por la cercanía de estas estaciones con la localidad destino del usuario. Por esta razón, se realiza un recorrido por todos los datos y se utiliza la función de haversine para buscar las distancias más cortas.	O(n)
Paso 4: Se hace un recorrido por todos los datos de la lista creada en el paso 3, con el objetivo de mirar si existe un camino entre la estación encontrada en el paso 2 y las estaciones de la lista 3. En caso de que, si exista un camino, se hace un llamado de la función de pathStationBFS1, entonces se elige esa estación como destino. La funcion pathStationBFS1, utiliza un recorrido con BFS.	O(V+E)

Paso 5: Del paso 4, por el llamado de la función de pathStationBFS1, se crea una lista que contiene los identificadores de las estaciones por donde se realiza el recorrido. Entonces, se crea una lista con el objetivo que de esta contenga únicamente la información relacionada con las estaciones que pertenecen al recorrido.	O(n)
Paso 6: Se hace un recorrido por la lista del paso 5, para contabilizar el número de transbordos.	O(n)
Paso 7: Se hace un recorrido por la lista creada en el paso 5, esto con el objetivo de poder incluir el identificador de cada una de las estaciones y poder hacer calcular las distancias entre las estaciones llamado a la función de haversine.	O(n)
TOTAL	***O(V+E)***

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).

Para realizar las pruebas de tiempo de ejecución se utilizaron los elementos encontrados en los archivos de bus_edges y bus_stops. Este análisis se hizo por medio de los documentos de tamaño small, 5pct, 10pct, 50pct, 80pct y large. Para este requerimiento la información fue organizada por medio de listas y el uso las funciones de gráficos, las cuales incluían busquedas DFS y BFS. El ordenamiento de estos datos se dio por las conexiones existentes entre diferentes elementos de grafo. En el cálculo de los tiempos se utilizaron tres diferentes computadores, en donde se sacó el promedio de los resultados, esto se pudo realizar con las funciones de la librería de time de Python. Todas las pruebas para este requerimiento se realizaron con los mismos parámetros de entrada en donde la longitud de la ubicación inicial fue de 2.123432, la latitud de la ubicación inicial fue 41.2334, la longitud de la ubicación destino fue 2.134342 y por último la latitud de la ubicación destino fue de 41.23423.

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

REQUERIMIENTO 4	
Tamaño archivo	**Tiempo**
small	5.434
5pct	10.49
10pct	24.85
50pct	31.32
80pct	170.56
large	240.83

Graficas

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.
El requerimiento 4, cuenta con una complejidad de O(V+E), esto se da principalmente debido a que se realiza un llamado a una función pathStationBFS1, la cual realiza una búsqueda del camino más corto entre dos vértices, esto con el objetivo de encontrar el camino entre dos estaciones y poder asegurar un recorrido con menor distancia.

Requerimiento 5

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Estación de origen y número de conexiones permitidas
Salidas	Las estaciones con su latitud, longitud y longitud del camino
Implementado (Sí/No)	Sí se implementó, realizado por Laura Calderón.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1. Se hace un recorrido BFS	O(V + E)
Paso 2. Se hace un recorrido a los caminos posibles	O (n)
Paso 3. Se verifica que la llave sea diferente a None	O (1)
Paso 4. Si la llave es diferente a None es porque existe una estación y por lo tanto se evalúa si la distancia es menor al número ingresado	O (1)

Paso 5. Si cumple con las anteriores condiciones se asigna la estación como el catálogo en esa posición del recorrido y en esa llave específica.	O (V + E)
Paso 6. Luego, para obtener el número de conexiones, se obtiene el DistTo que da el algoritmo BFS.	O (1)
Paso 7. Se recorren los elementos del catálogo en donde se busca coincidir las estaciones con el fin de ir hallando las latitudes y longitudes correspondientes.	O (n)
TOTAL	***O (V + E)***

Pruebas Realizadas – Con máquina 1

Se realizaron las pruebas de tiempo de ejecución para este requerimiento con varios de los archivos variando su tamaño, cargando los datos por medio de un catálogo. En este se obtuvo una complejidad temporal de O (V + E), la cual corresponde al algoritmo de BFS, el cual es un algoritmo con un enfoque iterativo que se encarga de responder por cuál es el camino con menor número de saltos (en este caso de conexiones).

Tablas de datos

REQUERIMIENTO 5	
Tamaño archivo	**Tiempo**
small	10,23
5pct	19,21
10pct	67,93
50pct	120,44
80pct	198,72
large	208,91

Graficas

Análisis

En este requerimiento se obtuvo una complejidad temporal de O (V + E), la cual corresponde al algoritmo de BFS, el cual es un algoritmo con un enfoque iterativo que se encarga de responder por cuál es el camino con menor número de saltos (en este caso de conexiones).

Requerimiento 6

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	La entrada de esta función es: el identificador de la estación origen y el identificador del vecindario destino.
Salidas	La respuesta final de este requerimiento incluye: la distancia total que se va a tomar desde la estación origen y la estación destino del vecindario, el total de estaciones que estan contenidas en el recorrido y el número total de transbordos que se deben realizar. Por último, se incluye información relacionada con las estaciones del camino incluyendo el identificador de cada una de ellas, el identificador del vecindario y la distancia hasta la siguiente estación.
Implementado (Sí/No)	Sí se implemento

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Accedo a la lista que contiene la información de las estaciones, la cual se encuentra en el catálogo.	O(n)

Paso 2: Se crea una lista vacía, y se realiza un recorrido por todos los elementos de la lista con el objetivo de añadir únicamente aquellos elementos cuyo "Neighborhood_Name" coincida con el vecindario destino que entro como parámetro a la función.	O(n)
Paso 3: Realizo un recorrido por todos los elementos de la lista creada en el paso 2, con el objetivo de encontrar la estación del vecindario más cercana a la estación origen. Esta comparación de distancia, lo realizo por medio de la ecuación de haversine.	O(n)
Paso 4: Realizo un llamado a la función pathStationBFS1 con el objetivo de encontrar el camino más corto entre la estación origen y la estación destino. La funcion pathStationBFS1, utiliza un recorrido con BFS.	O(V+E)
Paso 5: Realizo un recorrido por la lista de estaciones para incluir toda la información acerca las estaciones que hacen parte del recorrido.	O(n)
Paso 6: Se realiza un recorrido por todos los elementos de la lista creada en el paso 5, con el objetivo de mirar el número de transbordos.	O(n)
Paso 7: Se realiza un recorrido por toda la lista que fue creada en el paso 5, con el objetivo de poder incluir el identificador de cada una de las estaciones, el identificador del vecindario y las distancias que se deben recorrer entre cada una de las estaciones.	O(n)
TOTAL	***O(V+E)***

Pruebas Realizadas – Con máquina 4

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).

Las pruebas para el requerimiento 6, fueron realizadas mediante el uso de los elementos encontrados en los archivos de bus_edges y bus_stops. El análisis de tiempo de ejecución se realizó mediante diferentes tamaños de documentos, entre estos el tamaño small, 5pct, 10pct, 50pct, 80pct y large. Para la elaboración de este requerimiento, la información fue organizada principalmente por medio de las funciones de graficas de las librerías, entre las cuales existían búsquedas de DFS y BFS. Para el cálculo del tiempo se utilizaron funciones de la librería de time de python, y estas pruebas de tiempo fueron realizadas en los tres computadores y se sacó un promedio de los tiempos. Las pruebas para este requerimiento se realizaron con los mismos parámetros de entrada en donde la estación origen era 1470-75 y el vecindario destino fue el Raval.

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

REQUERIMIENTO 6	
Tamaño archivo	**Tiempo**
small	7.32
5pct	20.2
10pct	50.13
50pct	120.56
80pct	141.04
large	236.83

Graficas

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

El requerimiento 6 tiene una complejidad de O(V+E), esto se debe a que se llama a la función de pathStationBFS1, la cual realiza un recorrido por los vértices con el objetivo de encontrar el camino más corto existente en el gráfico.

Requerimiento 7

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Estación de inicio del camino, catálogo de información.

Salidas	Camino circular desde la estacion de inicio, distancia total recorrida, total de estaciones recorridas, total de trasbordos.
Implementado (Sí/No)	Si se implemento

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1. Se decidió que para encontrar un camino circular se usarían los componentes conectados. Por lo que se accedió el índice el cual contiene los componentes conectados 'components'	O(1)
Paso 2. Se tomaron cada uno de los elementos del componente por medio de recorrer la lista y coincidirlos con el id que entra por parámetro.	O(V)
Paso 3. Debido a que se pide un camino sin especificaciones se generó un numero aleatorio el cual define el total de estaciones que se aleja de la estación de partida.	O(1)
Paso 4. Se recorrio por medio de BFS para así tener el menor número de estaciones.	O(E+V)
Paso 5. Se encuentra el camino desde la estación hasta el vértice definido por el parametro y de igual manera el camino desde la estacion asignada por el aleatorio hasta el vertice de partida.	O(E+V)
Paso 6. Se añaden a una sola lista todos los vertices visitados para completar el camino.	O(V)
Paso 7. Se encuentra la distancia total por medio de coincidir los vertices de la ruta con las rutas establecidas.	O(E)
Paso 8. Para saber el total de paradas se toma el tamaño de la lista.	O(1)
TOTAL	***O(E+V)***

Pruebas Realizadas – Con máquina 3

Se realizaron las pruebas de tiempo de ejecución para este requerimiento con varios de los archivos variando su tamaño, cargando los datos por medio de un catálogo en el que se creó una lista con las películas y diferentes índices dependiendo de los requerimientos. Los datos se organizaron en un grafo el cual incluia los vertices y los arcos correspondientes a cada una de las estaciones las cuales conectaban la ruta de bus de Barcelona. Se ejecutaron las pruebas en tres computadores. Las pruebas se realizaron con los diferentes archivos con diferentes tamaños. Además, se usó la librería nativa de python time. Para realizar estas pruebas se utilizó como parametro la estación 1280-H16 en cada una de las pruebas, ademas se repitió la prueba tres veces para tener un valor más exacto del resultado.

Tablas de datos

REQUERIMIENTO 7

Tamaño archivo	Tiempo
small	12,14
5pct	52,78
10pct	134,67
50pct	567,88
80pct	766,20
large	880,61

Graficas

Análisis

En este caso, se puede observar que los tiempos de ejecución aumentan de manera acelerada, esto se da debido a que para poder sacar los datos necesarios de este requerimiento se necesita recorrer algunos de los datos varias veces, sin embargo nunca se recorre toda la lista de datos mas de una vez. En el analisis de los pasos tambien se puede evidenciar lo anterior y se ve que al aumentar el tamaño de los archivos se aumenta tambien este tiempo de ejecución. Los tiempos de ejecución son aceptables por lo ya mencionado anteriormente.

Requerimiento 8 - BONO

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	El usuario debe ingresar el año de publicación sobre el cual quiere obtener los histogramas, junto con el límite inferior de la duración del mejor tiempo y el límite superior de la duración del mejor tiempo.
Salidas	El número total de registros de speedrun en dicho año y rango. Un mapa interactivo de clústeres que muestra todos los registros de speedrun en dicho año y rango de tiempos.
Implementado (Sí/No)	Si se implementó

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Se accede al índice “gamesbyTime”, que tiene los registros ordenados por el tiempo en segundos para el mejor intento realizado y al índice “”gamesByRelease_Date”.	O(1)
Paso 2: Se crean dos listas vacías, una que tendrá los id de los juegos y otra para los países.	O(1)
Paso 3: Se hace un recorrido inorder con el fin de ir agregando los elementos del árbol creado con el índice de tiempos dentro de una nueva lista.	O(n)
Paso 4: Se hace un recorrido inorder con el fin de ir agregando los elementos del árbol creado con el índice de fechas de publicación dentro de una nueva lista.	O(n)
Paso 5: Se crea un mapa del mundo utilizando la librería folium.	O(1)
Paso 6: Se recorre la lista de los juegos, en donde se saca el año de publicación del juego, se revisa si este es igual al año ingresado, y si si lo es, se agrega a la lista creada anteriormente los id de los juegos que corresponden al año solicitado.	O(n)
Paso 7: Se recorre la lista de los id de los juegos, la lista de las categorías creada al principio, y los elementos de la lista de dichas categorías.	O(n)
Paso 8: Se compara que el game id del registro a evaluar coincida con uno de los game id de la lista que se tenía, también que el tiempo sea mayor al límite inferior del tiempo y que el tiempo sea menor al límite superior del tiempo.	O(1)
Paso 9: Si lo anterior se cumple, se agrega a la lista vacía de países creada al inicio, el país del registro que cumple con dichos criterios.	O(1)

Paso 10: Se crea una variable a la cual se le asigna Nominatim (función que permite sacar coordenadas).	O(n)
Paso 11: Se recorre la lista final de países, en donde se utiliza la librería geopy para obtener la longitud y latitud dado un país específico.	O(n)
Paso 12: Se utiliza folium marker para ir marcando en el mapa las ubicaciones de los países de la lista, con las latitudes y longitudes brindadas por geopy.	O(n)
Paso 13: Se guarda el mapa en un archivo html.	O(n)
TOTAL	***O(n)***

Pruebas Realizadas – Con máquina 3

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).

Tablas de datos

REQUERIMIENTO 8 - BONO	
Tamaño archivo	**Tiempo**
small	1091.52
5pct	14865.47
10pct	30579.03
50pct	165314.744
80pct	297271.42
large	443144.54

Graficas

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.
La complejidad del requerimiento 8 es lineal O(n) debido a que no se recorrieron la totalidad de los datos brindados por los archivos de game y category. Las pruebas de ejecución se realizaron con el año 2017, y los tiempos 1000 y 5000. Con esto se obtuvo el mapa con las respectivas ubicaciones que permiten que el usuario conozca los países a los que pertenecen los juegos del año y del rango ingresado.