Universidad de los Andes
Facultad de Ingeniería

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

ABET
Engineering Accreditation Commission

ANÁLISIS DEL RETO

Estudiante 1, código 1, email 1

Juan Pablo Baldion Castillo, 202214765, j.baldion@uniandes.edu.co

Kenet Adrián Martínez Moya, 202211092, ka.martinezm1@uniandes.edu.co

Requerimiento 1

Descripción

En este requerimiento se nos pide buscar un camino posible entre dos estaciones, para esto, ya que se tienen los datos cargados en un grafo que representa toda la red de buses de Barcelona se hizo un recorrido con el algoritmo DFS (Depth First Search) el cual nos da una posible ruta entre un nodo y otro.

Entrada	origen: ID del nodo de origen (Code-Bus) destino: ID del nodo de destino (Code-Bus)
Salidas	Ruta entre la estación de origen y la estación de destino La distancia total que se recorre entre el origen y el destino El número de estaciones que se toman El número de transbordos que se toman
Implementado (Sí/No)	Sí

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Declaración y asignación de variables	O(16)
Recorrido de la ruta	O(9N)
TOTAL	***O(9N + 16)***

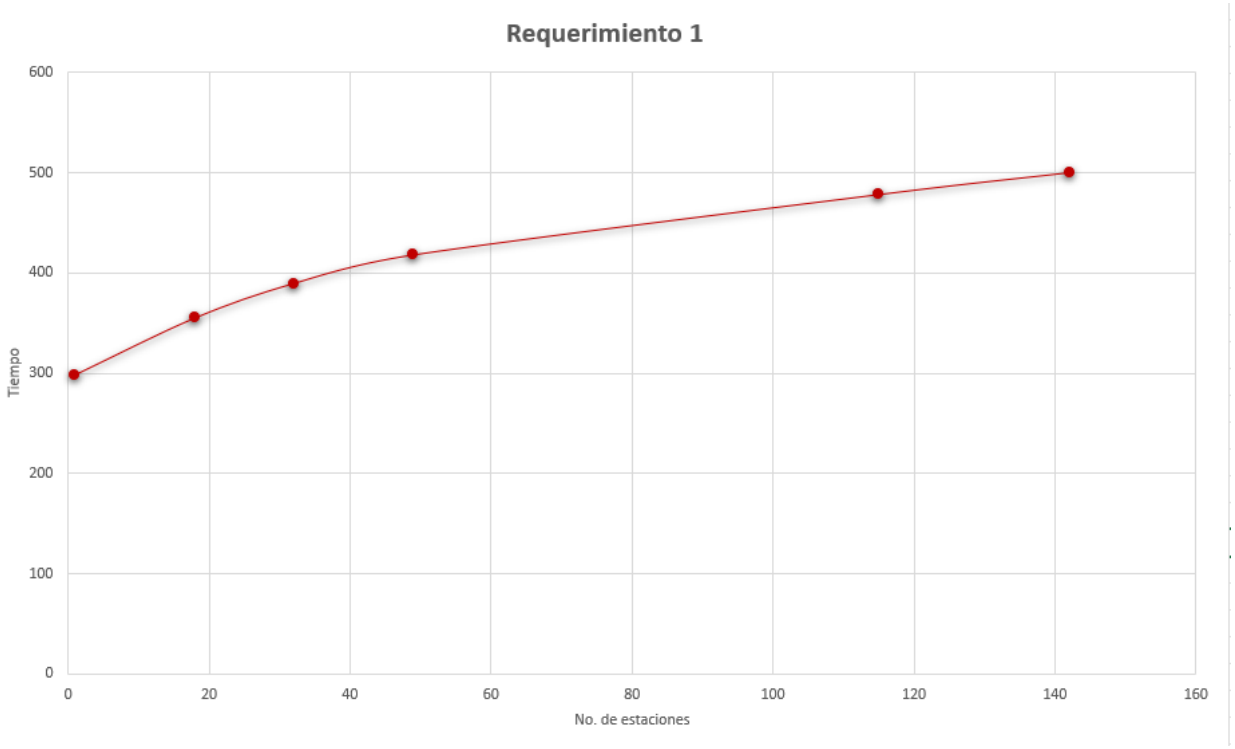
Pruebas Realizadas

AMD Ryzen 5 5500U 2.10GHz	16.0 GB	Windows 11 64 bits

Entrada	**No. de datos encontrados**	**Tiempo (s)**

57-109 58-109	1	298.36
57-109 911-109	18	355.14
57-109 1636-109	32	389.58
57-109 59-11	49	418.18
57-109 1168-L86	115	478.35
57-109 865-110	142	500.22

Graficas



Requerimiento 2

Descripción

En este requerimiento se nos pide buscar el camino con menos paradas entre dos estaciones, para esto se recorre el grafo que representa la red de buses de Barcelona con el algoritmo BFS (Breadth First Search) el cual nos devuelve el camino con menor longitud (en términos de grafos) entre 2 estaciones.

Entrada	origen: ID del nodo de origen (Code-Bus)

	destino: ID del nodo de destino (Code-Bus)
Salidas	Ruta entre la estación de origen y la estación de destino La distancia total que se recorre entre el origen y el destino El número de estaciones que se toman El número de transbordos que se toman
Implementado (Sí/No)	Sí

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

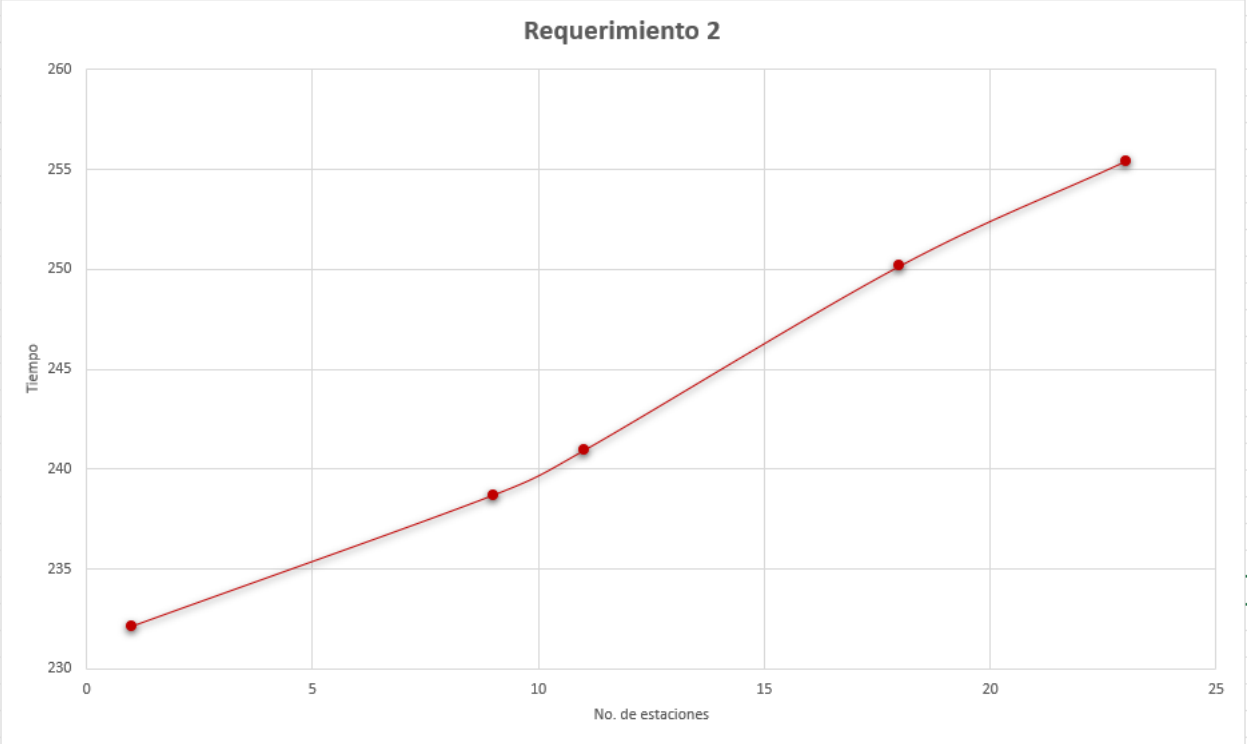
Pasos	Complejidad
Declaración y asignación de variables	O(14)
Recorrido de la ruta	O(10N)
TOTAL	***O(10N + 14)***

Pruebas Realizadas

AMD Ryzen 5 5500U 2.10GHz	16.0 GB	Windows 11 64 bits

Entrada	No. de datos encontrados	Tiempo (s)
57-109 58-109	1	232.14
57-109 1636-109	9	238.71
57-109 1256-V9	11	240.94
57-109 911-109	18	250.19
57-109 59-11	23	255.43

Graficas



Requerimiento 3 – Juan Pablo Jerez Zarate

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Parámetros necesarios para resolver el requerimiento.
Salidas	Respuesta esperada del algoritmo.
Implementado (Sí/No)	Si se implementó y quien lo hizo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1	O(...)
Paso 2	O(...)
Paso ….	O(...)
TOTAL	***O(...)***

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).

Entrada	Tiempo (s)

Graficas

Las gráficas con la representación de las pruebas realizadas.

Requerimiento 4 – Juan Pablo Baldion Castillo

Descripción

En este requerimiento se pide encontrar una ruta con distancia mínima entre dos puntos geográfico, para esto primero se busca cuáles son las estaciones más cercanas a cada uno de los puntos geográficos que el usuario nos da, esto se hace recorriendo todas las estaciones y hallando las distancias con el punto geográfico y metiéndolas en un minPQ, ya obtenidas las estaciones más cercanas a esos puntos, aplicamos el algoritmo Dijkstra para encontrar la ruta con distancia mínima entre esas estaciones obteniendo así la distancia mínima entre los dos puntos geográficos.

Entrada	origen: Punto geográfico de origen (latitud longitud) destino: Punto geográfico de destino (latitud longitud)
Salidas	Ruta entre la estación más cercana al punto geográfico de origen y la estación más cercana al punto geográfico de destino. La distancia total que se recorre entre las estaciones La distancia entre el punto geográfico de origen y la distancia más cercana La distancia entre el punto geográfico de destino y la distancia más cercana El número de estaciones que se toman El número de transbordos que se toman
Implementado (Sí/No)	Sí

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Declaración y asignación de variables	O(19)
Recorrido de las estaciones	O(9V)

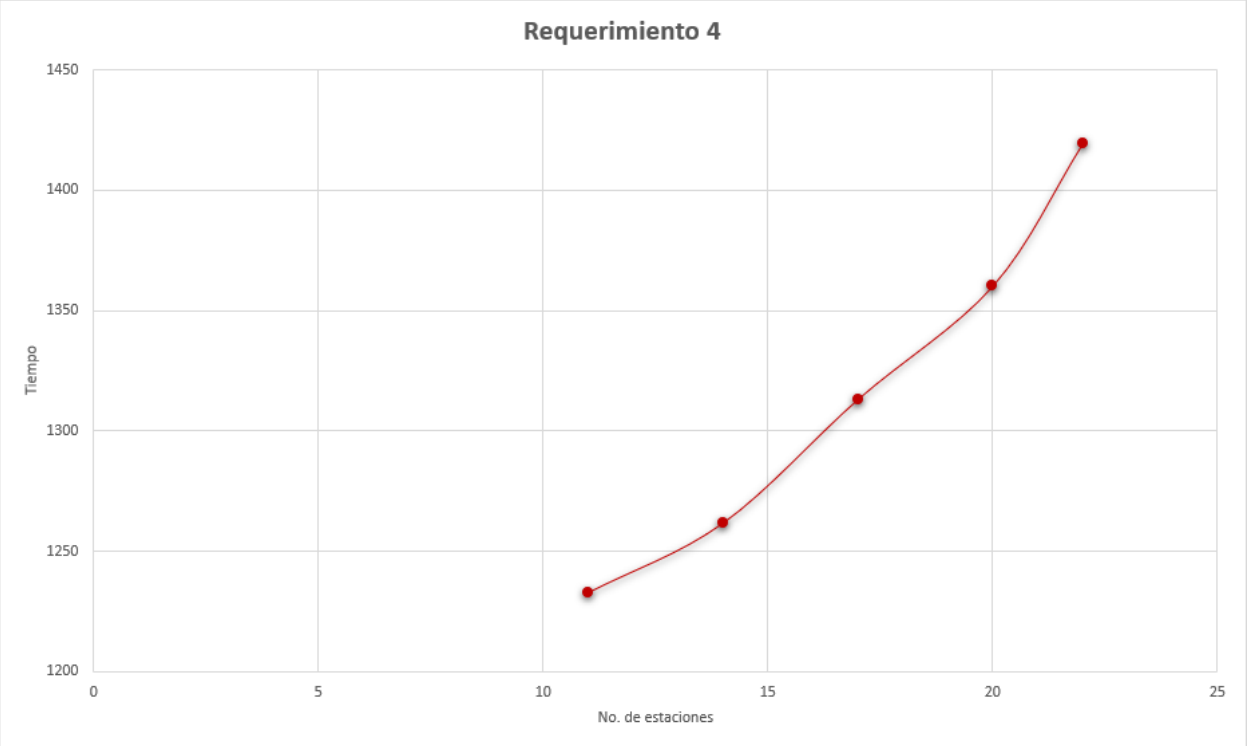
Recorrido de las rutas	O(N)
TOTAL	***O(9V+N+19)***

Pruebas Realizadas

AMD Ryzen 5 5500U 2.10GHz	16.0 GB	Windows 11 64 bits

Entrada	No. de datos encontrados	Tiempo (s)
41.359 2.1394 41.3876 2.13789765	11	1232.79
41.359 2.1394 41.343342 2.13244	14	1261.71
41.359 2.1394 41.32 2.13	17	1313.06
41.359 2.1394 41.35234567 2.1394876	20	1360.16
41.359 2.1394 41.000009 2.99987	22	1419.36

Graficas



Requerimiento 5 – Kenet Adrián Martínez Moya

Descripción

Lo que se hizo fue aplicar el algoritmo BFS al grafo que mapea los datos siendo el vértice de orígen, el vértice entrado como parámetro. La implementación de la librería de este algoritmo arroja un mapa donde las llaves son los vértices que se pueden alcanzar desde un vértice source dado como parámetro, y los valores asociados son diccionarios que contienen, entre otras cosas, la distancia, en términos de cuántos vértices intermedios hay, entre el vértice de origen y uno visitado. La idea fue recorrer una lista con todos los vértices del grafo y para cada uno, ver si está en el mapa que el BFS arrojó; si está, se comprueba que el número de vértices intermedios entre él y el vértice de origen no sobrepasen el número de conexiones entrado como parámetro; si se cumple esa última condición, se agrega a un diccionario la información del mismo para que luego sea mostrado por el view

Entrada	Código de la estación desde la cual se harán las búsquedas y el número máximo de conexiones para llegar a cierta estación desde dicha estación.
Salidas	Un reporte consolidado que incluya la información de las estaciones “alcanzables” con el número de conexiones dado con la siguiente información: El identificador de la estación. La geolocalización de la estación (latitud y longitud) Longitud del camino desde la estación origen a la estación alcanzada.
Implementado (Sí/No)	Sí

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Llamar al grafo que da la información de las conexiones entre cada una de las estaciones.	O(1)
Aplicar BFS al grafo mencionado	O(V+E), donde V es el número de vértices y E el número de arcos
Recorrido de la lista que contiene el nombre de todas las paradas con conexiones a la estación entrada por parámetro. En este ciclo hay dos condicionales, uno para filtrar las estaciones de transbordo, y uno para filtrar las estaciones que toman más de cierto número de conexiones para llegar hasta ella.	O(V+3n)), donde V es el número de vértices en el grafo, y n es el número de estaciones que cumplen con la condición de conexiones entrada por parámetro. En el peor de los casos: O(V)
TOTAL	***O(1+2V+E+3n)***

Pruebas Realizadas

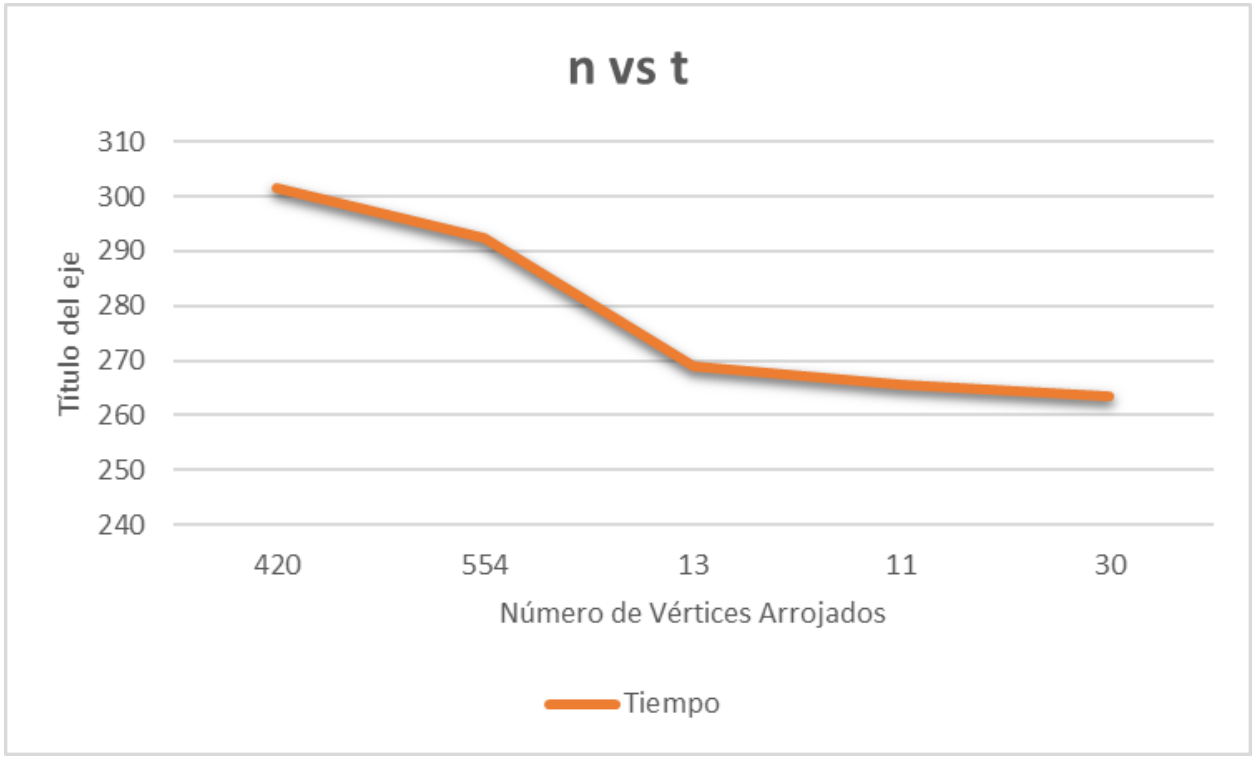
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	Numero de Vértices Arrojados	Tiempo (ms)
57-109, 10 conexiones	420	301.5126
911-109, 10 conexiones	554	292.38
2306-V7, 10 conexiones	13	268.9266
59-11, 10 conexiones	11	265.5491
737-19, 10 conexiones	30	263.5934

Estos resultados son acordes a la fórmula BIG O descrita, ya que a pesar de que el número de vértices arrojados son bastante distintos, el tiempo es bastante parecido, y esto se da ya que V y E, los parámetros constantes y los más importantes en la fórmula, son constantes.

Graficas

Las gráficas con la representación de las pruebas realizadas.



Requerimiento 6

Descripción

En este requerimiento se nos pide buscar el camino con mínima distancia entre una estación de origen y un vecindario de destino, para esto hicimos un mapa que clasifica las estaciones por vecindario, con esto cuando el usuario nos pide una ruta hasta un vecindario específico, consultamos este mapa y obtenemos sólo las estaciones que pertenecen a dicho vecindario, ya obtenidas estas estaciones con el algoritmo Dijkstra obtenemos la distancia de la ruta mínima entre la estación de origen y cada una de estas estaciones para luego introducir esas distancias en un minPQ y al final obtener la ruta con la distancia mínima.

Entrada	origen: Estación de origen (Code-Bus). destino: Nombre del vecindario destino.
Salidas	Ruta entre la estación de origen y la estación de destino. La distancia total que se recorre entre el origen y el destino. El número de estaciones que se toman. El número de transbordos que se toman. La estación de destino del vecindario.
Implementado (Sí/No)	Sí.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Declaración y asignación de variables	O(22)
Recorrido de las estaciones de un vecindario	O(2N)
Recorrido de la ruta	O(N)
TOTAL	***O(3N + 22)***

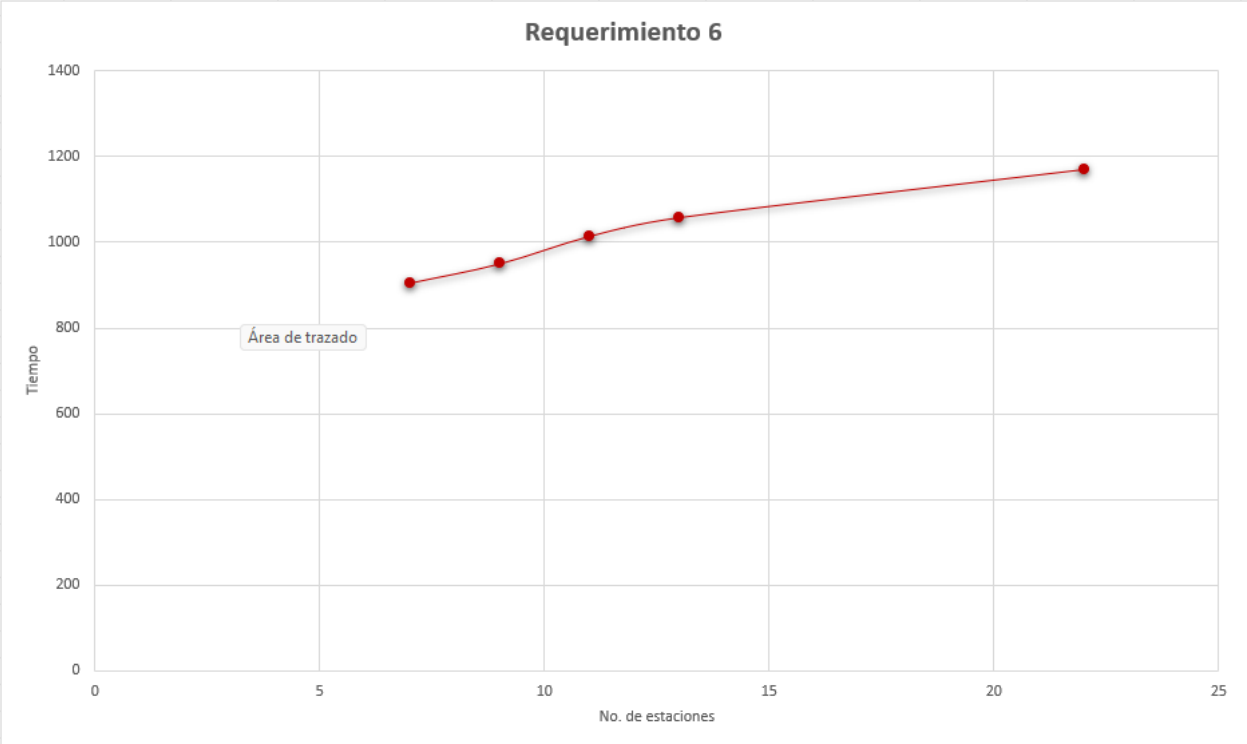
Pruebas Realizadas

AMD Ryzen 5 5500U 2.10GHz	16.0 GB	Windows 11 64 bits

Entrada	**No. de datos encontrados**	**Tiempo (s)**
57-109 la Bordeta	7	905.86
57-109 les Corts	9	950.95
57-109 Porta	11	1015.12

57-109 Pedralbes	13	1058.68
57-109 la Clota	22	1170.84

Graficas



Requerimiento 7

Descripción

En este requerimiento se nos pide encontrar un posible camino circular desde una estación de origen, para este requerimiento se recorre el grafo que representa la red de buses de Barcelona mediante el algoritmo Kosaraju para obtener los componentes fuertemente conectados del grafo, sacamos solo los vértices fuertemente conectados a la estación de origen, luego elegimos uno al azar, comprobamos que no sea un vértice adyacente del origen, si no lo es y si el usuario quiere un camino largo sacamos una ruta entre el origen y ese vértice elegido mediante DFS y luego sacamos otro camino entre el vértice y el origen con BFS obteniendo así una ruta circular. Si el usuario quiere un camino corto se usan los algoritmos BFS y Dijkstra de la misma manera.

Entrada	origen: Estación de origen (Code-Bus). corto: Valor que determina si lo que se quiere es un camino circular corto o un camino circular largo.
Salidas	La ruta circular encontrada. La distancia total que se recorrida.

	El número de estaciones que se toman. El número de transbordos que se toman.
Implementado (Sí/No)	Sí

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

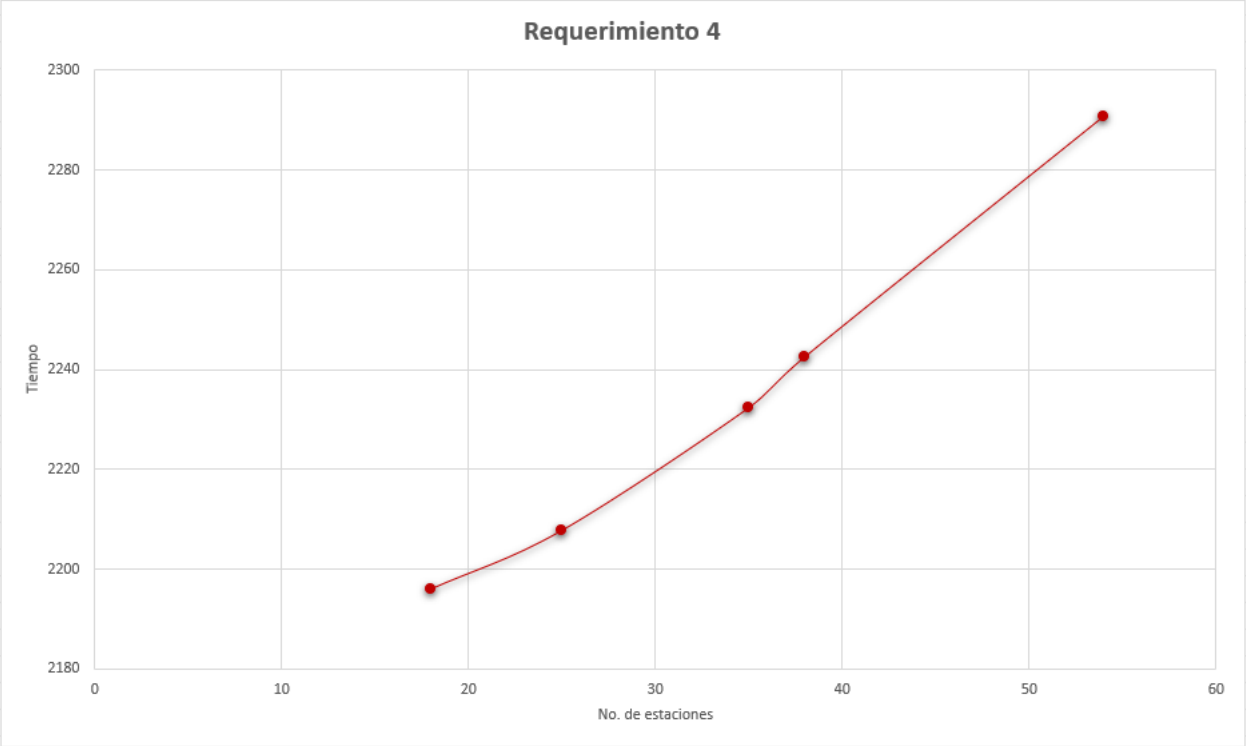
Pasos	Complejidad
Declaración y asignación de variables	O(25)
Recorrido del SCC	O(N)
Recorrido de la ruta de origen a destino	O(N)
Recorrido de la ruta de destino origen	O(N)
TOTAL	***O(3N + 25)***

Pruebas Realizadas

AMD Ryzen 5 5500U 2.10GHz	16.0 GB	Windows 11 64 bits

Entrada	No. de datos encontrados	Tiempo (s)
2242.47 Corto	18	2195.92
57-109 Corto	25	2207.66
737-19 Corto	35	2232.27
2306-V7 Corto	38	2242.47
59-11 Corto	54	2290.79

Graficas


Requerimiento 4
Tiempo
2300
2280
2260
2240
2220
2200
2180
0
10
20
30
40
50
60
No. de estaciones