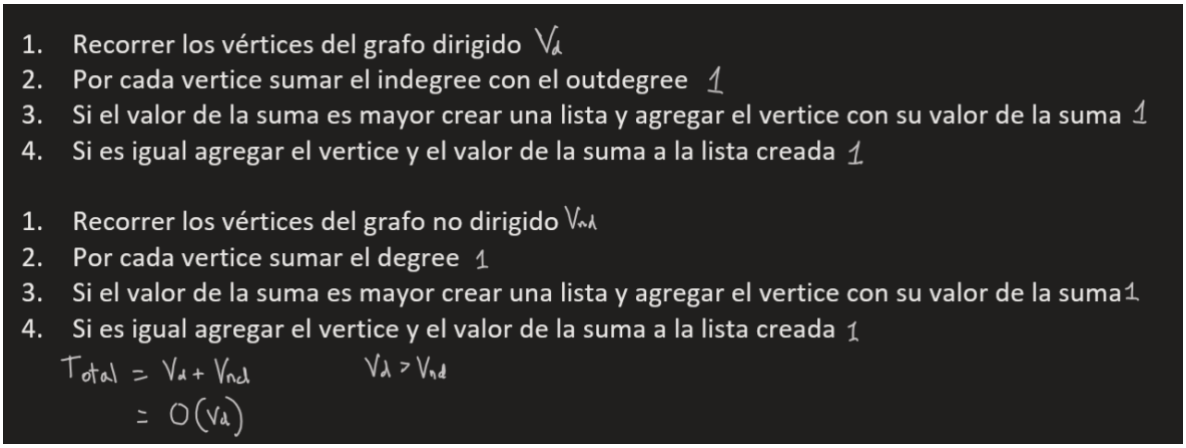
Camilo Ortiz Cruz 201821615 c.ortizc@uniandes.edu.co

Kevin Fernando Gómez Camargo 202015120 k.gomezc@uniandes.edu.co

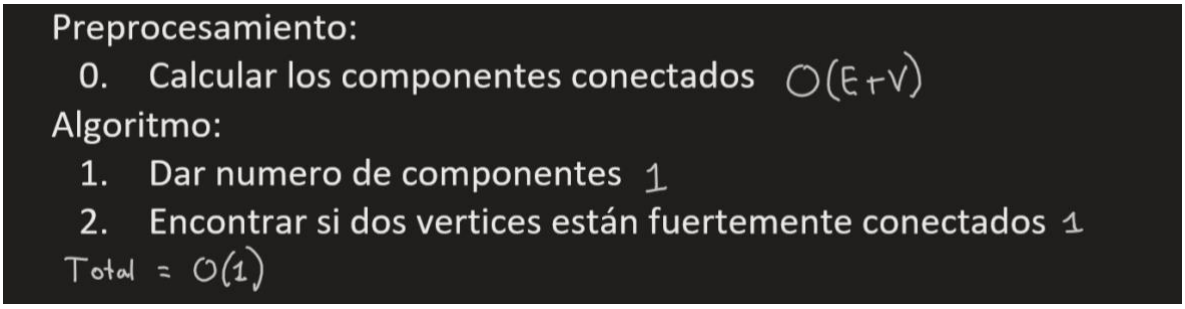
Análisis de complejidad temporal en notación Big O

Requerimiento 1



Como se puede ver la complejidad final es el número de vértices del grafo dirigido, esto se debe a que el grafo dirigido siempre va contener mas o los mismos vértices que el grafo no dirigido y por ende en notación O el termino mayor es el que queda, dado que este requerimiento era buscar un máximo consideramos que esta complejidad es muy buena ya que para encontrar el valor mas grande se debe recorrer todos los elementos por tanto consideramos que estamos cerca de los más optimo.

Requerimiento 2



Como se puede observar la complejidad dio constante, esto se debe a que el calculo de los componentes fuertemente conectados (KosarajuSCC) se realizo en la carga de datos dado que este no necesita de ningún parámetro del usuario, dado que esta en el catalogo entonces las funciones necesarias para responder el requerimiento solo son connectedComponents y stronglyConnected y estas son constantes, consideramos que este es la solución óptima.

Requerimiento 3

1. Encontrar aeropuerto mas cercano

Complejidad función findNearestAirport:

1. Sacar los valores que estén en el rango de latitud en el árbol de las latitudes de los aeropuertos $\log(lat) + \#Along$
2. Iterar sobre los valores del árbol de latitud, los cuales son arboles de longitud y los valores de este son aeropuertos. $\#Along$
3. Por cada arbol de longitud sacar los valores que estén en el rango de longitud $\log(long) + \#air$
4. Iterar sobre esos valores y calcular la distancia a la ciudad y escoger el que este a menos distancia $\#air$
5. De no encontrar ninguno incrementar el radio de búsqueda hasta encontrar o hasta que el radio sea de mas de 1000 km 100

$Total = \#Along * \#Air$

2. Calcular arbol de rutas mas cortas (Dijkstra) desde el aeropuerto de origen $E\log(V)$
3. Encontrar el path hacia el aeropuerto de destino $\#arcosOrigenDestino$
4. Calcular la distancia hacia el aeropuerto de destino 1

$Total = O(E\log(V))$

En el caso del requerimiento 3, la complejidad temporal la determina correr Dijkstra esto se debe a que el numero de arcos por el logaritmo de vértices es mayor a la complejidad de encontrar los aeropuertos, esto se debe a que en esencia la complejidad de encontrar el aeropuerto cercano es la de recorrer todos los aeropuertos que estén en el rango de búsqueda lo cual siempre será menor que todos los arcos del grafo multiplicado por el logaritmo del número de vértices.

Requerimiento 4

Preprocesamiento:

0. Calcular MST $E\log(V)$

Algoritmo:

1. Encontrar aeropuerto mas cercano $\#Along * \#Air$
2. Hacer BFS modificado $E+V$
3. Encontrar el valor del mayor numero de saltos 1
4. Encontrar el vertice que esta a esa distancia 1
5. Encontrar el path hacia el vertice mas lejano $\#arcosDestinoOrigen$
6. Encontrar cual es el vertice mas lejano donde le alcancen las millas para llegar $\#VerticesdesdeOrigen$
7. Si se encuentra un vertice posible calcular el path $\#arcosDestinOrigen$

$Total = O(E+V)$

Como se puede observar la complejidad del algoritmo está relacionada a realizar BFS, consideramos que este tiempo es razonable ya que para encontrar cual es el vértice que esta mas lejos de un origen ese necesario recorrer todos los vértices y arcos del árbol puesto que de otra forma no podríamos estar seguros y como podemos ver el resto de pasos del algoritmo solo tienen como complejidad un subconjunto de elementos de los arcos o los vértices, por lo cual E+V siempre será mayor.

Requerimiento 5

1. Recorrer todos los vertices V
2. Por cada vertice si el vertice es el pasado por parámetro recorrer todos los adyacentes a este $\#adj$
3. Si no es el pasado por parámetro revisar si los adyacentes contienen el vertice pasado por parámetro. $\#adj$

$Total = O(V * \#adj)$

El algoritmo tiene una complejidad que consideramos adecuada pues que para poder encontrar que aeropuertos son afectados se debe revisar todos los arcos entre el y todos los demás aeropuertos, sin embargo, dado que necesitamos saber los aeropuertos afectados entonces se debe revisar las listas de adyacencia de cada vértice del grafo y revisar si está el de interés, lo cual es exactamente lo que hace el algoritmo, algo importante a resaltar es que lo mas probable es que no se revisen las lista de adyacencia a totalidad si no que sean recorridos parciales pero su utiliza el peor caso posible.

Requerimiento 6 (bono)

En base a lo realizado en el requerimiento 3 es claro que la complejidad de este es similar ya que funciona de la misma forma solo que en vez de buscar el aeropuerto mas cercano en la base de datos se calcula utilizando el API de Amadeus, por ende la única diferencia es la complejidad de tiempo de la búsqueda de Amadeus, ya que no se conoce la implementación entonces la complejidad total será la misma del requerimiento 3 $O(E \cdot log(v) + c)$ pero con una constante que hace referencia a la complejidad desconocida del API, sin embargo, para simplificar se dirá que es $O(E \cdot log(v))$.

Requerimiento 7 (bono)

Req. 1

Para el requerimiento 1 la complejidad de crear el mapa es de O(E), donde E es el numero de arcos en cada grafo, esto se debe a que en el mapa se ponen los arcos como líneas y marcadores con los aeropuertos.

Req. 2

La complejidad de este requerimiento es la de recorrer todas las llaves del mapa de componentes conectados, lo cual en esencia es recorrer todos los vértices por ende la complejidad O(V).

Req. 3

Para este requerimiento solo es necesario recorrer el path retornado por la función del requerimiento 3, por tanto, la complejidad es O(P) siendo P el numero de elementos en el path.

Req. 4

La complejidad total de hacer este mapa es igual O(MB) siendo MB el numero de elementos en la rama más larga, hay que tener en cuenta que también se grafica el mapa de las ciudades a visitar entonces hay una complejidad adicional de C (C es el numero de ciudades que puede visitar la persona con las millas) pero dado que C como máximo es igual a la rama mas larga entonces la complejidad final es O(MB).

Req. 5

La complejidad total de crear este mapa es de O(A) siendo A el numero de aeropuertos afectados por cerrar un aeropuerto.