

# ANÁLISIS DEL RETO

*Luis Alejandro Rodríguez Arenas, Cód. 202321287, la.rodrigueza12@uniandes.edu.co*

*Santiago Rojas, Cod. 202420928, s.rojasg23@uniandes.edu.co*

## Requerimiento <<1>>

### Descripción

Detecta el camino recorrido por un individuo entre dos zonas geográficas.

Busca los puntos migratorios más cercanos al origen y destino, verifica que el individuo pase por ellos, arma un grafo solo con sus movimientos y reconstruye la ruta en orden temporal, calculando la distancia total y generando una tabla de la trayectoria.

<b>Entrada</b>	Catálogo, latitud y longitud de origen y destino, identificador del individuo (tag).
<b>Salidas</b>	Diccionario con mensaje, id de origen y destino, id del individuo, distancia total recorrida, número de puntos en la ruta, ruta completa y ruta resumida (5 primeros y 5 últimos), y tiempo de ejecución.
<b>Implementado (Sí/No)</b>	Sí se implementó y lo hizo Luis Alejandro

### Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Buscar vértice más cercano al origen	$O(V)$
Buscar vértice más cercano al destino	$O(V)$
Recorrer eventos del individuo y construir grafo	$O(N)$
Ejecutar DFS sobre el grafo individual	$O(N)$

Compactar ruta y construir tablas de salida	$O(N)$
<b>TOTAL</b>	<b><math>O(V+N)</math></b>

## Análisis

El costo se reparte entre buscar los vértices más cercanos en todo el catálogo ( $O(V)$ ) y procesar los eventos del individuo ( $O(N)$ ).

En conjunto, el requerimiento crece como  $O(V + N)$ .

## Requerimiento <<2>>

### Descripción

Encuentra un camino entre dos puntos migratorios usando BFS y analiza qué parte de la ruta queda dentro de un radio dado alrededor del origen.

Busca los vértices más cercanos a las coordenadas dadas, ejecuta BFS en el grafo de distancias, reconstruye el camino, suma la distancia total, halla el último vértice dentro del área de interés y arma resúmenes de los primeros y últimos vértices de la ruta.

<b>Entrada</b>	Catálogo, latitud y longitud de origen y destino, radio de interés (km).
<b>Salidas</b>	Diccionario con éxito o error, id de origen y destino, último vértice dentro del radio, distancia total, número de puntos de la ruta, primeros 5 y últimos 5 vértices, y tiempo de ejecución.
<b>Implementado (Sí/No)</b>	Si se implementó y lo hizo Santiago Rojas

### Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Buscar vértice más cercano al origen	$O(V)$
Buscar vértice más cercano al destino	$O(V)$
Ejecutar BFS en graph_dist	$O(V + E)$
Reconstruir el camino	$O(L)$
Calcular distancias y último nodo dentro del radio	$O(L)$
Construir listas 5+5	$O(L)$

<b>TOTAL</b>	<b>O(V + E)</b>
--------------	-----------------

## Análisis

El paso dominante es BFS sobre el grafo, con costo  $O(V + E)$ .

Las demás operaciones dependen solo de la longitud del camino ( $L$ ) y no cambian el orden global, por lo que la complejidad se mantiene en  $O(V + E)$ .

## Requerimiento <<4>>

### Descripción

Calcula un corredor hídrico “óptimo” a partir de un punto dado, usando un árbol de expansión mínima (MST) sobre el grafo hídrico.

Encuentra el vértice más cercano al origen, ejecuta Prim sobre `graph_agua`, obtiene el MST del componente alcanzable, calcula cuántos puntos y cuántos individuos participan, la distancia total del corredor a las fuentes hídricas y arma tablas con los puntos del corredor.

<b>Entrada</b>	Catálogo y coordenadas de origen (lat, lon).
<b>Salidas</b>	Diccionario con mensaje, id del punto de origen, total de puntos del corredor, total de individuos, distancia total del MST, tabla completa de puntos, tabla 5+5 y tiempo de ejecución.
<b>Implementado (Sí/No)</b>	Sí se implementó y se hizo en grupo

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

<b>Pasos</b>	<b>Complejidad</b>
Buscar vértice más cercano al origen	$O(V)$
Ejecutar Prim sobre <code>graph_agua</code>	$O(E \log V)$
Obtener arcos del MST y construir conjunto de vértices	$O(V)$
Contar individuos (tags únicos) en los vértices del MST	$O(N)$
Construir tablas completas y 5+5	$O(V + N)$
<b>TOTAL</b>	<b><math>O(E \log V)</math></b>

## Análisis

El costo dominante es Prim sobre el grafo hídrico, con complejidad  $O(E \log V)$ .

El resto de recorridos sobre vértices y tags es lineal en  $V$  y  $N$ , por lo que no cambia el orden global.

## Requerimiento <<5>>

### Descripción

Encuentra la ruta migratoria más eficiente entre dos zonas, usando caminos mínimos con Dijkstra sobre uno de los dos grafos (distancias o agua).

Primero aproxima los puntos migratorios de origen y destino con Haversine, luego ejecuta Dijkstra desde el origen, reconstruye el camino mínimo, calcula el costo total, cuenta los puntos y segmentos y arma una tabla detallada de la ruta.

<b>Entrada</b>	Catálogo, latitud y longitud de origen y destino, tipo de grafo a usar ("dist" o "agua").
<b>Salidas</b>	Diccionario con mensaje, id de origen y destino, métrica utilizada, costo total del camino, número de puntos, número de segmentos, tabla completa de la ruta y tiempo de ejecución.
<b>Implementado (Sí/No)</b>	Sí se implementó y se hizo en grupo

### Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

<b>Pasos</b>	<b>Complejidad</b>
Buscar vértice más cercano al origen	$O(V)$
Buscar vértice más cercano al destino	$O(V)$
Ejecutar Dijkstra desde el origen	$O(E \log V)$
Reconstruir el camino mínimo	$O(L)$
Calcular distancias entre vértices consecutivos	$O(L)$
Construir la tabla de la ruta	$O(L)$
<b>TOTAL</b>	<b><math>O(E \log V)</math></b>

## Análisis

El algoritmo de Dijkstra es el paso dominante, con costo  $O(E \log V)$ .

Las operaciones adicionales dependen solo de la longitud de la ruta ( $L$ ) y no afectan el orden global, por lo que la complejidad total se mantiene en  $O(E \log V)$ .

## Requerimiento <<6>>

### Descripción

Identifica subredes hídricas aisladas (componentes conexas) en el grafo hídrico.

Recorre todos los vértices de `graph_agua`, lanza BFS desde los que aún no tienen componente asignada, forma subredes, las ordena por tamaño y selecciona las 5 más grandes.

Para cada una calcula el rango espacial (latitud y longitud mínimas y máximas), el total de individuos (tags únicos), una muestra de tags y los 3 primeros y 3 últimos puntos migratorios.

<b>Entrada</b>	Catálogo con <code>graph_agua</code> y <code>vertices_info</code>
<b>Salidas</b>	Diccionario con mensaje, total de subredes hídricas encontradas, lista de las 5 subredes más grandes con su información (id, número de vértices, rangos de lat/lon, total de individuos, muestra de tags, primeros y últimos puntos) y tiempo de ejecución.
<b>Implementado (Sí/No)</b>	Sí se hizo en grupo

### Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

<b>Pasos</b>	<b>Complejidad</b>
Obtener lista de vértices	$O(V)$
Recorrer vértices y ejecutar BFS sobre los no visitados	$O(V + E)$ en total
Construir lista de componentes	$O(V)$
Ordenar las componentes por tamaño	$O(N \log N)$
Recorrer vértices y tags de las subredes top	$O(N)$

Construir info de primeros/últimos puntos y estructura de salida	$O(N)$
<b>TOTAL</b>	<b><math>O(V + E + N \log N)</math></b>

## Análisis

El recorrido completo del grafo mediante BFS explora todos los vértices y arcos en  $O(V + E)$ .

Después, las componentes se ordenan en  $O(N \log N)$ , y el procesamiento de las subredes más grandes (cálculos y construcción de salidas) es lineal en  $N$ .

En conjunto, el orden de crecimiento del requerimiento es  $O(V + E + N \log N)$ .