ANÁLISIS DEL RETO 4

Simón Peña Alarcón - 202512907- s.penaa2@uniandes.edu.co

Juan Sebastian Chacón Ochoa-202513196- j.chacono@uniandes.edu.co

Manuel Santiago Figueroa-202511697- m.figueroag@uniandes.edu.co

Requerimiento <<1>>

Descripción

Este requerimiento recibe dos puntos GPS (origen y destino) y busca en el grafo los nodos más cercanos a esas coordenadas. Luego hace un recorrido DFS desde el nodo origen para saber si existe un camino hacia el nodo destino.
Si existe ruta, la reconstruye, calcula la distancia total y también verifica en qué punto aparece por primera vez un tag específico.
Finalmente arma una lista con detalles de los primeros y últimos puntos del recorrido.

Entrada	catalog: diccionario con todas las estructuras del modelo gps_origen: tupla con latitud y longitud gps_destino: tupla con latitud y longitud tag_id: identificador del individuo que se quiere buscar
Salidas	Un diccionario con: mensaje primer nodo que tiene ese tag distancia total de la ruta total de puntos detalles de primeros y últimos puntos la ruta completa
Implementado (Sí/No)	Si se implementó por Juan chacón

Análisis de complejidad

Pasos	**Complejidad**
1. Recorrer la lista de vértices para encontrar el nodo más cercano al origen y destino	O(n)
2. Ejecutar DFS desde el nodo origen	O(n + m) (n nodos, m arcos)
3. Reconstruir la ruta usando pilas	O(n)
4. Recorrer la ruta para buscar el primer nodo que tenga el tag deseado	O(n * k) (k es tamaño promedio de lista de tags)
5. Calcular distancia entre puntos consecutivos	O(n)

6. Construir la lista de detalles de los primeros 5 nodos y los últimos 5	O(k)
TOTAL	***O(n + m)***

Análisis

Este requerimiento se encarga de encontrar los nodos más cercanos a las coordenadas de origen y destino dentro del grafo. Luego ejecuta un DFS desde el nodo origen para verificar si existe un camino hacia el nodo destino. Si hay ruta, la reconstruye con pilas, calcula la distancia total y busca en qué punto aparece por primera vez el tag dado. También arma una lista con información de algunos nodos de la ruta. Como resultado retorna el mensaje final, la ruta encontrada, el primer nodo con el tag, la distancia total y los detalles.

La complejidad del requerimiento es principalmente lineal. Primero recorre todos los vértices para encontrar los nodos más cercanos, lo cual es O(n). Después ejecuta el DFS, que cuesta O(n + m). La reconstrucción de la ruta y el recorrido para buscar el tag toman O(n), y revisar los tags de cada nodo puede llegar a O(n * k), aunque k suele ser pequeño. En general el orden total se puede aproximar como O(n + m). En las pruebas se comprobó que el requerimiento funciona bien, encuentra los nodos correctos y calcula la ruta y la información adicional sin problemas.

Requerimiento <<2>>

Descripción

Este requerimiento identifica los nodos más cercanos a las coordenadas de origen y destino dentro del grafo usando la distancia haversine. Luego ejecuta un recorrido BFS desde el nodo origen para verificar si existe un camino hacia el nodo destino. Si la ruta existe, la reconstruye, calcula la distancia total entre nodos consecutivos y determina cuál fue el último nodo del recorrido que estuvo dentro del radio indicado a partir del punto de origen. Finalmente retorna toda la información asociada a ese trayecto.

Entrada	catalog: estructuras del modelo gps_origen: latitud y longitud del punto inicial gps_destino: latitud y longitud del punto final radio: distancia máxima para determinar los nodos dentro del rango
Salidas	Mensaje de estado Último nodo dentro del radio Distancia total del recorrido Total de puntos en la ruta La ruta reconstruida
Implementado (Sí/No)	Si se implementó por Simón Peña Alarcón

Análisis de complejidad

Pasos	Complejidad
1. Recorrer todos los vértices para encontrar los nodos más cercanos al origen y destino	O(n)
2. Ejecutar BFS desde el nodo origen	O(n + m)
3. Reconstruir la ruta usando pilas	O(n)
4. Recorrer la ruta para identificar el último nodo dentro del radio	O(n)
5. Calcular la distancia total del recorrido	O(n)
Total	**O(n + m)**

Análisis

Este requerimiento tiene un comportamiento principalmente lineal, ya que primero recorre todos los vértices para identificar los nodos más cercanos y luego ejecuta BFS, que es O(n + m). La reconstrucción de la ruta, la verificación de qué nodos caen dentro del radio y el cálculo de la distancia total también recorren la ruta una vez, por lo que son procesos lineales. En general la complejidad final queda en O(n + m). Durante las pruebas se observó que el requerimiento identifica correctamente los nodos más cercanos, reconstruye rutas válidas cuando existen y calcula sin problema tanto el último nodo dentro del radio como la distancia total del trayecto.

Requerimiento <<3>>

Descripción

Este requerimiento realiza un DFS sobre el grafo para verificar si desde el nodo inicial se puede recorrer sin problemas. Si el DFS no detecta ciclos, entonces ejecuta un DFO para obtener el orden de recorrido en postorden inverso. Luego toma cada vértice del recorrido, calcula cuántos pájaros tiene según su lista de tags, extrae los primeros y últimos tags, y cuenta el total de pájaros en todo el recorrido. También arma una lista con información resumida de los primeros y últimos vértices procesados. Finalmente retorna la cantidad total de vértices, la cantidad total de pájaros y la información pedida.

Entrada	catalog: contiene el grafo del nicho biológico.
Salidas	Número total de vértices recorridos. Número total de pájaros (suma de los tags de todos los vértices). Lista con los primeros vértices procesados. Lista con los últimos vértices procesados.
Implementado (Sí/No)	Si se implementó por Manuel Santiago Figueroa

Análisis de complejidad

Pasos	Complejidad
1. Ejecutar DFS sobre el grafo	O(n + m)
2. Ejecutar DFO para obtener el postorden inverso	O(n + m)

3. Recorrer la pila del postorden inverso y procesar cada vértice	O(n)
4. Contar tags de cada vértice	O(k) por vértice (k es número de tags)
Total	**O(n + m)**

Análisis

Este requerimiento depende principalmente del DFS y el DFO, que tienen un costo O(n + m) ya que recorren todos los nodos y todas las aristas. Después se procesa cada vértice para contar sus tags y construir la información que se va a retornar, lo cual agrega un costo O(n*k), aunque en la práctica k suele ser pequeño. Por eso el orden total se mantiene cercano a O(n + m). En las pruebas se observó que el requerimiento funciona correctamente cuando no hay ciclos y que calcula sin problemas el total de pájaros, el total de vértices y las listas pedidas. Si se detectan ciclos, el requerimiento devuelve un mensaje adecuado.

Requerimiento <<4>>

Descripción

Este requerimiento recibe una coordenada (lon, lat) y busca en el grafo hídrico el vértice más cercano a ese punto. Luego ejecuta el algoritmo de Prim para construir el árbol de expansión mínima a partir de ese vértice. Con ese MST calcula la distancia total recorrida, el número de vértices marcados dentro del MST y el total de individuos únicos presentes en esos vértices. Después construye un árbol de hijos usando el campo edge_from del MST y realiza un recorrido en anchura para obtener el orden de la ruta. Finalmente arma una lista con información detallada de los primeros cinco y los últimos cinco nodos del recorrido.

Entrada	catalog: estructuras del modelo que contienen el grafo hídrico. lon: longitud del punto base. lat: latitud del punto base.
Salidas	Distancia total del MST. Total de vértices incluidos en el MST. Total de individuos únicos encontrados. Lista con los primeros cinco nodos procesados. Lista con los últimos cinco nodos procesados.
Implementado (Sí/No)	Si se implementó por el grupo

Análisis de complejidad

Pasos	**Complejidad**
1. Recorrer todos los vértices para encontrar el más cercano al punto dado	O(n)
2. Ejecutar Prim sobre el grafo hídrico	O(m log n)
3. Contar vértices marcados dentro de la tabla visited	O(n)

4. Recorrer cada vértice marcado y procesar sus tags	O(n * k)
5. Construir el árbol de hijos en un mapa	O(n)
6. Hacer un recorrido BFS sobre el árbol	O(n)
7. Construir la información de primeros cinco y últimos cinco	O(n)
Total	**O(m log n +n**

Análisis

Este requerimiento mezcla tres procesos principales: localizar el vértice más cercano, ejecutar Prim y luego recorrer la estructura generada para extraer información. El primer paso es lineal porque revisa todos los vértices del grafo. El algoritmo de Prim utiliza estructuras con prioridad y por eso su complejidad es O(m log n), que es la parte más costosa del requerimiento. Después se revisan todos los vértices marcados del MST para contar individuos y armar las listas, lo cual agrega un costo lineal y depende del número de tags. Finalmente se recorre el árbol de hijos con BFS para generar la ruta, que también es lineal. En las pruebas se confirmó que el requerimiento encuentra correctamente el vértice inicial más cercano, construye un MST válido, cuenta bien los individuos únicos y entrega la información solicitada en los primeros y últimos nodos del recorrido.

Requerimiento <<5>>

Descripción

Este requerimiento recibe dos puntos GPS y selecciona si el recorrido se hará en el grafo hídrico o en el grafo de distancias. Primero busca los vértices más cercanos al origen y al destino. Con esos dos nodos ejecuta el algoritmo de Dijkstra para encontrar el camino óptimo entre ellos. Si existe camino, reconstruye la ruta, calcula el número total de puntos y segmentos y arma una lista con la información de cada nodo del recorrido. Para cada nodo también obtiene los primeros tres y últimos tres tags y calcula la distancia hacia el siguiente nodo. Finalmente devuelve un resumen con los primeros cinco y los últimos cinco nodos de la ruta.

Entrada	catalog: estructuras del modelo. pto_origen: coordenadas del punto de inicio. pto_destino: coordenadas del punto final. seleccion: indica si usar graph_water o graph_distance.
Salidas	Mensaje del estado del recorrido. Costo total del camino óptimo. Total de puntos y segmentos de la ruta. Primeros cinco nodos detallados. Últimos cinco nodos detallados. La ruta completa con información adicional.
Implementado (Sí/No)	Si se implementó por el grupo

Análisis de complejidad

Pasos	Complejidad
1. Recorrer todos los vértices para encontrar los nodos más cercanos	O(n)
2. Ejecutar Dijkstra desde el nodo origen	O(m log n)
3. Reconstruir la ruta con la pila	O(n)
4. Recorrer la ruta y procesar cada nodo	O(n * k)
5. Calcular distancia al siguiente nodo usando edges	O(n)
6. Obtener primeros cinco y últimos cinco nodos	O(1)
Total	**O(m log n)**

Análisis

El requerimiento combina la búsqueda lineal del nodo más cercano con el algoritmo de Dijkstra, que es la parte más costosa con una complejidad O(m log n). Después de obtener la estructura del recorrido, reconstruir la ruta y procesar los nodos tiene un costo lineal, incluyendo la revisión de los tags de cada vértice. Como normalmente la cantidad de tags es pequeña, esto no afecta demasiado y la complejidad final queda dominada por Dijkstra. En las pruebas se comprobó que el requerimiento detecta correctamente los nodos más cercanos, encuentra la ruta óptima cuando existe y arma sin problema la información de cada nodo, además de separar los primeros y últimos cinco. También calcula correctamente la distancia entre nodos consecutivos.

Requerimiento <<6>>

Descripción

Este requerimiento identifica todas las subredes hídricas aisladas dentro del grafo de distancias. Para hacerlo, recorre todos los vértices y, para cada vértice no visitado, realiza un recorrido DFS manual con una pila para construir una subred completa. Después calcula información importante de esa subred, como su total de individuos, límites de latitud y longitud, y también construye una lista con detalles de los primeros tres y los últimos tres nodos. Cada subred se guarda con su identificador y finalmente, al terminar, se ordenan todas las subredes por el tamaño y se devuelven las cinco más grandes.

Entrada	catalog: estructura que contiene el grafo de distancias.
Salidas	total_subredes: cantidad total de subredes encontradas. top_subredes: lista con las cinco subredes más grandes, cada una con detalles de puntos, rangos y número total de individuos.
Implementado (Sí/No)	Si se implementó por el grupo

Análisis de complejidad

Pasos	Complejidad
1. Recorrer todos los vértices del grafo	O(n)
2. Para cada vértice no visitado, ejecutar DFS con pila	O(n + m)
3. Procesar cada nodo de la subred (contar individuos, calcular límites)	O(n * k)
4. Construir lista con primeros tres y últimos tres nodos	O(n)
5. Ordenar todas las subredes por tamaño	O(s log s) (s = número de subredes)
Total	**O(n + m)**

Análisis

El comportamiento principal del requerimiento es encontrar componentes conectados usando DFS, por lo que la complejidad está dominada por recorrer todo el grafo una sola vez, resultando en O(n + m). Después de identificar cada subred, se recorren todos sus nodos para contar individuos, calcular límites y armar las listas de resumen, lo cual agrega un costo lineal por subred. Luego ordenar las subredes cuesta O(s log s), aunque el número de subredes es mucho menor que el total de nodos. En general el requerimiento se comporta muy bien y mantiene una complejidad lineal respecto al tamaño del grafo. En las pruebas se confirmó que detecta todas las subredes correctamente, calcula bien la información de cada una y retorna adecuadamente las cinco subredes más grandes.