**ANÁLISIS DEL RETO**

Juan Camilo Cancelado, 202410123, j.canceladod@uniandes.edu.co

# **Requerimiento 1**

## **Descripción**

Dados dos Ids de ubicaciones A y B se requiere saber si existe un camino simple para conectarlos.

|  |  |
| --- | --- |
| **Entrada** | Grafo origen destino |
| **Salidas** | Cantidad de puntos que conectan los vercies |
| **Implementado (Sí/No)** | Si |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1: Inicializar estructuras auxiliares y variables | O(1) |
| Paso 2: Ejecutar recorrido en profundidad desde el nodo de origen | O(V + E) |
| Paso 3: Reconstruir el camino desde el origen hasta el destino | O(L) |
| Paso 4: Recorrer el camino para recolectar domiciliarios y restaurantes | O(L \* D) |
| Paso 5: Construir la secuencia del camino como cadena de texto | O(L) |
| Paso 6: Armar la fila de resultados para la respuesta | O(1) |
| **TOTAL** | **O(V + E + L \* D)** |

## **Pruebas Realizadas**

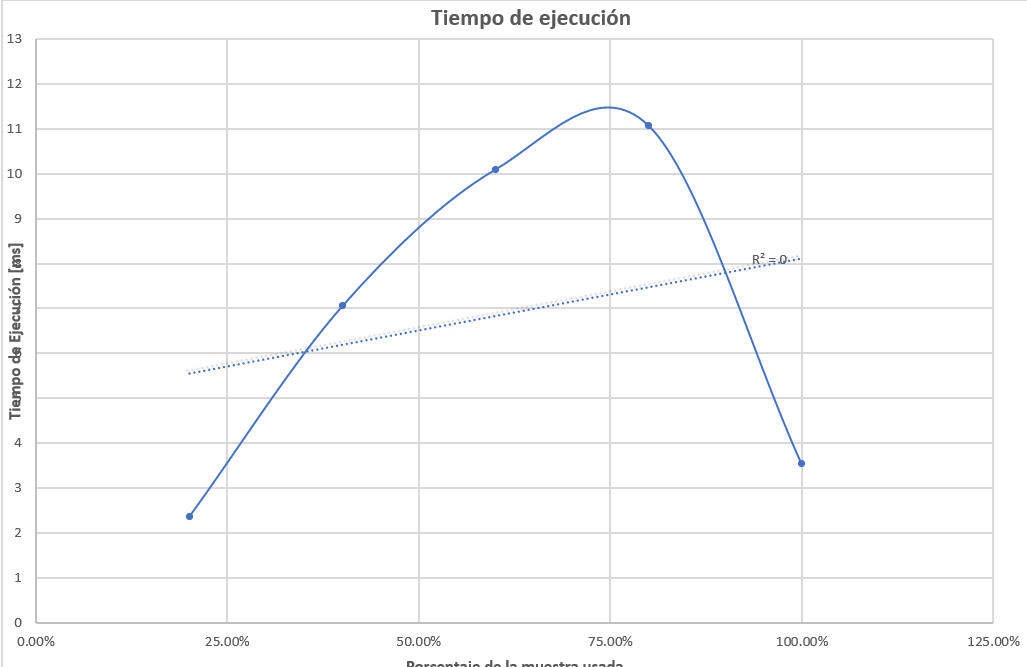
Dato: origen = 22.310526\_73.170937 Destino = 22.440526\_73.300937

|  |  |
| --- | --- |
| Procesadores | Intel core 17 7th gen |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 10 |

### **Tablas de datos**

|  |  |
| --- | --- |
| **Tiempo de ejecución ms** | |
| **Porcentaje de la muestra [pct]** | **Tiempo de ejecución (ms)** |
| 20.00% | 2.37 |
| 40.00% | 7.06 |
| 60.00% | 10.1 |
| 80.00% | 11.08 |
| 100.00% | 3.54 |

### **Graficas**



## **Análisis**

Puede que la gráfica tenga una forma extraña pero, aunque en DFS en este caso sea de distinta magnitud en cada base de datos, uno se da cuenta que el cambio de caminos entre cada base (teniendo en cuenta nodos específicos) es relativamente baja, por lo cual, el cambio temporal entre bases no será muy grande.  
Por otro lado, para encontrar un camino simple usando DFS se pueden escoger muchos caminos distintos, el bajo abrupto de tiempo puede ser por un Edge añadido que acorte un posible camino, por lo cual tiene sentido esta baja

# **Requerimiento 3**

.

## **Descripción**

Se quiere saber el domiciliario que más domicilios ha atendido en una ubicación geográfica específica.

|  |  |
| --- | --- |
| **Entrada** | Punto geográfico |
| **Salidas** | • El tiempo en que se demora el algoritmo en encontrar la solución. • El domiciliario más popular según su cantidad pedidos en el punto geográfico de consulta.  • La cantidad de pedidos atendidos por el domiciliario con el punto geográfico de consulta.  • Tipo del vehículo que más repitió el domiciliario |
| **Implementado (Sí/No)** | si |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Paso** | **Complejidad** |
| 1. Verificar si el punto existe en el grafo | O(1) |
| 2. Obtener el mapa de domiciliarios del nodo | O(1) |
| 3. Buscar el domiciliario con más pedidos en ese punto | O(D) |
| 4. Buscar el tipo de vehículo más usado por ese domiciliario | O(T) |
| 5. Armar la fila de resultados para la respuesta | O(1) |
| **Total** | **O(D + T)** |

## **Pruebas Realizadas**

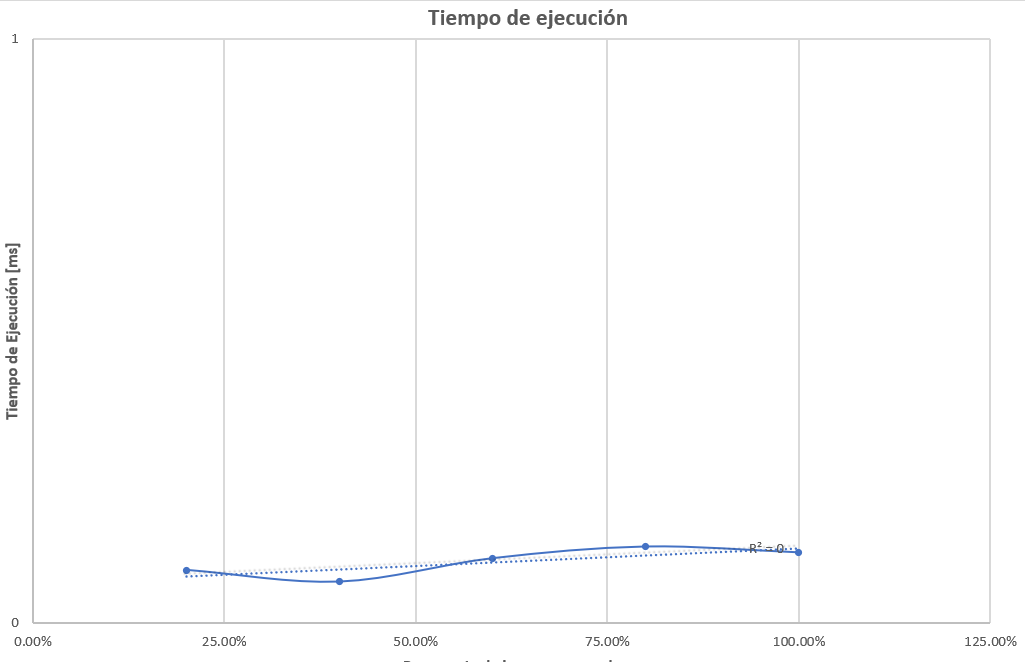
Punto geográfico = 21.1751\_72.8043

|  |  |
| --- | --- |
| Procesadores | Intel core 17 7th gen |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 10 |

### **Tablas de datos**

|  |  |
| --- | --- |
| **Tiempo de ejecución ms** | |
| **Porcentaje de la muestra [pct]** | **Tiempo de ejecución (ms)** |
| 20.00% | 0.09 |
| 40.00% | 0.07 |
| 60.00% | 0.11 |
| 80.00% | 0.13 |
| 100.00% | 0.12 |

### **Graficas**



## **Análisis**

Considerando en la tabla D como domiciliarios y T como tipos de vehículo, esta función está muy bien optimizada ya que hubo un cambio en el load data que permite el acceso a esta información de una manera casi constante.

# **Requerimiento 4**

## **Descripción**

Dado dos ubicaciones geográficas A y B, se requiere el listado de domiciliarios en común que están en el camino simple con menos puntos intermedios entre los puntos geográficos A y B.

|  |  |
| --- | --- |
| **Entrada** | • Punto geográfico A (id del punto) origen desde donde se iniciará búsqueda.  • Punto geográfico B (id del punto) destino de la búsqueda. |
| **Salidas** | • El tiempo en que se demora el algoritmo en encontrar la solución. • Secuencia de las ubicaciones del camino simple con menos puntos intermedios entre A y B  • El listado de domiciliarios que se encuentra en común en este camino. |
| **Implementado (Sí/No)** | Si |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Paso** | **Complejidad** |
| 1. Ejecutar BFS desde el nodo de origen | O(V + E) |
| 2. Reconstruir el camino desde el destino usando el mapa de visitados | O(L) |
| 3. Recorrer el camino para recolectar domiciliarios | O(L \* D) |
| 4. Calcular la intersección de domiciliarios en todos los nodos | O(D \* L) |
| 5. Construir la secuencia del camino como cadena de texto | O(L) |
| 6. Armar la fila de resultados para la respuesta | O(1) |
| **Total** | **O(V + E + L \* D)** |

## **Pruebas Realizadas**

Origen = 17.450851\_78.379347

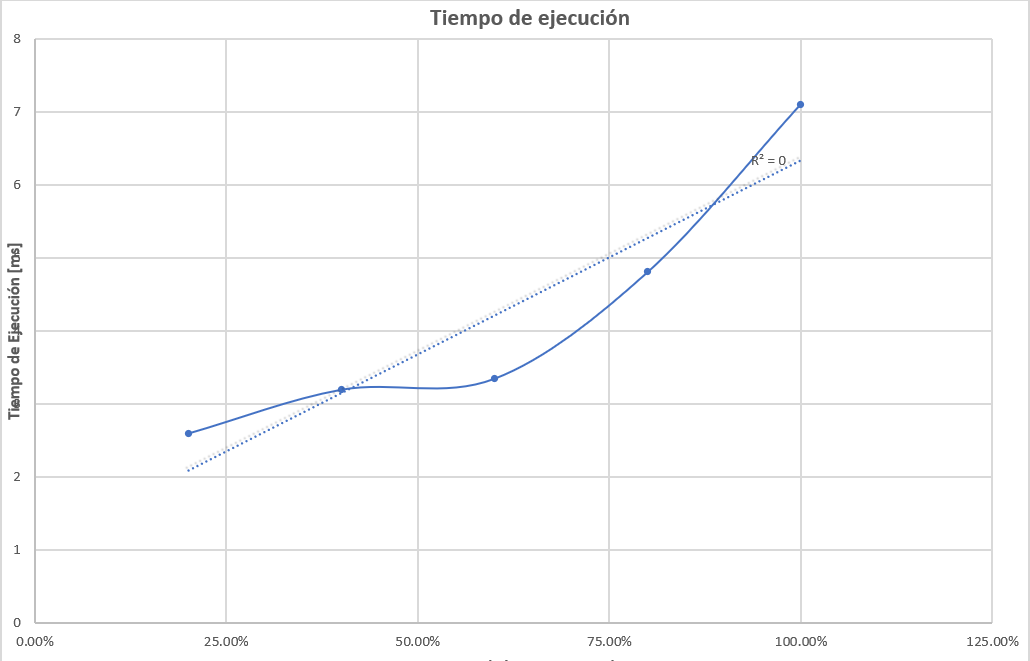
Destino = 17.460851\_78.389347

|  |  |
| --- | --- |
| Procesadores | Intel core 17 7th gen |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 10 |

### **Tablas de datos**

|  |  |
| --- | --- |
| **Tiempo de ejecución ms** | |
| **Porcentaje de la muestra [pct]** | **Tiempo de ejecución (ms)** |
| 20.00% | 2.6 |
| 40.00% | 3.2 |
| 60.00% | 3.35 |
| 80.00% | 4.81 |
| 100.00% | 7.11 |

### **Graficas**



## **Análisis**

El análisis de complejidad muestra que el tiempo de ejecución depende de factores como la cantidad de nodos, aristas y domiciliarios por nodo (O(V + E + L \* D)). Si estos valores crecen mucho al aumentar la base de datos, el tiempo puede incrementarse rápidamente, lo que explica el comportamiento casi exponencial observado en las mediciones.

# **Requerimiento 6**

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

## **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

|  |  |
| --- | --- |
| **Paso** | **Complejidad** |
| Inicializar Dijkstra desde el origen | O((V + E) log V) |
| Recopilar ubicaciones alcanzables y buscar la de mayor tiempo | O(V) |
| Ordenar ubicaciones alfabéticamente | O(V log V) |
| Reconstruir el camino de mayor tiempo | O(L) |
| Construir la secuencia del camino como string | O(L) |
| Armar la fila de resultados | O(1) |
| **Total** | **O((V + E) log V + L)** |

## **Pruebas Realizadas**

Origen = 12.975377\_77.696664

Destino = 13.005377\_77.726664

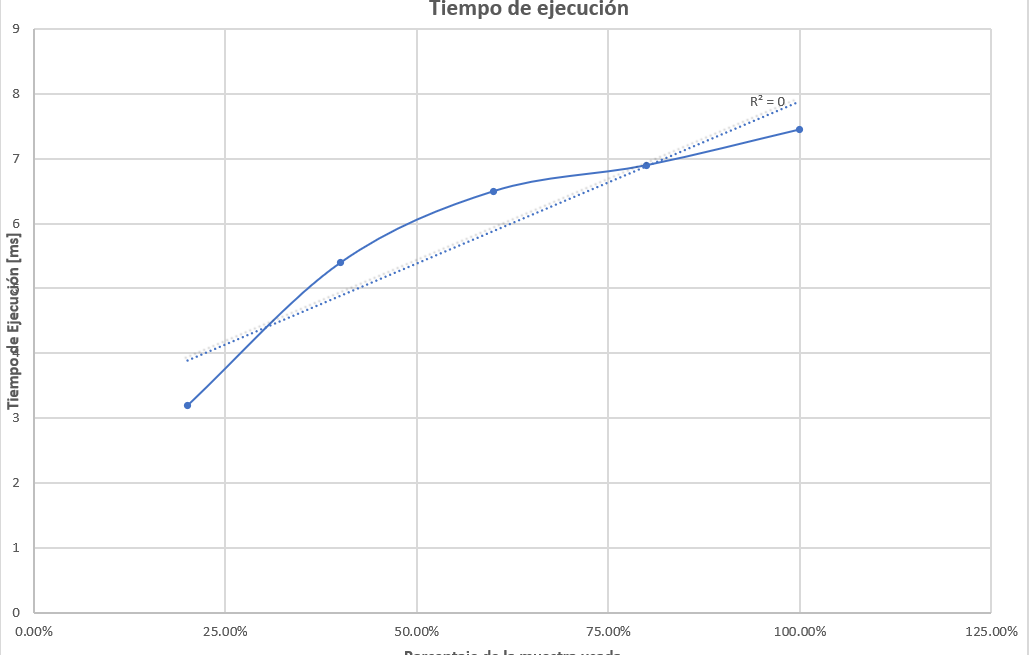
|  |  |
| --- | --- |
| Procesadores | Intel core 17 7th gen |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 10 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Tiempo de ejecución ms** | |
| **Porcentaje de la muestra [pct]** | **Tiempo de ejecución (ms)** |
| 20.00% | 3.2 |
| 40.00% | 5.4 |
| 60.00% | 6.5 |
| 80.00% | 6.9 |
| 100.00% | 7.45 |

### **Graficas**



## **Análisis**

El crecimiento moderado en la gráfica se debe a que el algoritmo principal de req\_6 (Dijkstra) tiene una complejidad O((V + E) log V), que es eficiente para grafos de tamaño medio. Además, el uso de estructuras como mapas y listas propias permite acceder y actualizar información rápidamente. Por eso, aunque el tiempo aumenta con la base de datos, no lo hace de forma abrupta, sino de manera controlada y predecible. El aumento de un logN puede ser por ruido en las pruebas.

# **Requerimiento 7**

## **Descripción**

Se desea obtener una sub-red (Árbol de recubrimiento de Costo Mínimo en Tiempo) con una

ubicación geográfica inicial y que conecta a las ubicaciones por donde pasa un domiciliario

particular con costo minimo.

|  |  |
| --- | --- |
| **Entrada** | • Punto geográfico A (id del punto) origen desde donde se iniciará búsqueda.  • Id del domiciliario a buscar. |
| **Salidas** | • El tiempo que se demora el algoritmo en encontrar la solución.  • La cantidad de ubicaciones que definen la sub-red (incluyendo la ubicación inicial).  • Identificadores de las ubicaciones que definen la sub-red (ordenados alfabéticamente)  • Total en tiempo del Arbol de Recubrimiento de costo mínimo. |
| **Implementado (Sí/No)** | Si |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Paso** | **Complejidad** |
| Filtrar nodos donde estuvo el domiciliario | O(V) |
| Crear subgrafo con esos nodos y aristas | O(V + E) |
| Ejecutar Prim en el subgrafo | O(E log V) |
| Sumar pesos y procesar nodos del MST | O(N) |
| Ordenar ubicaciones alfabéticamente | O(N log N) |
| Armar la fila de resultados | O(1) |
| **Total** | **O(V + E + V log V)** |

## **Pruebas Realizadas**

Source = "12.9702\_77.6453"

Id dom = "BANGRES02DEL03"

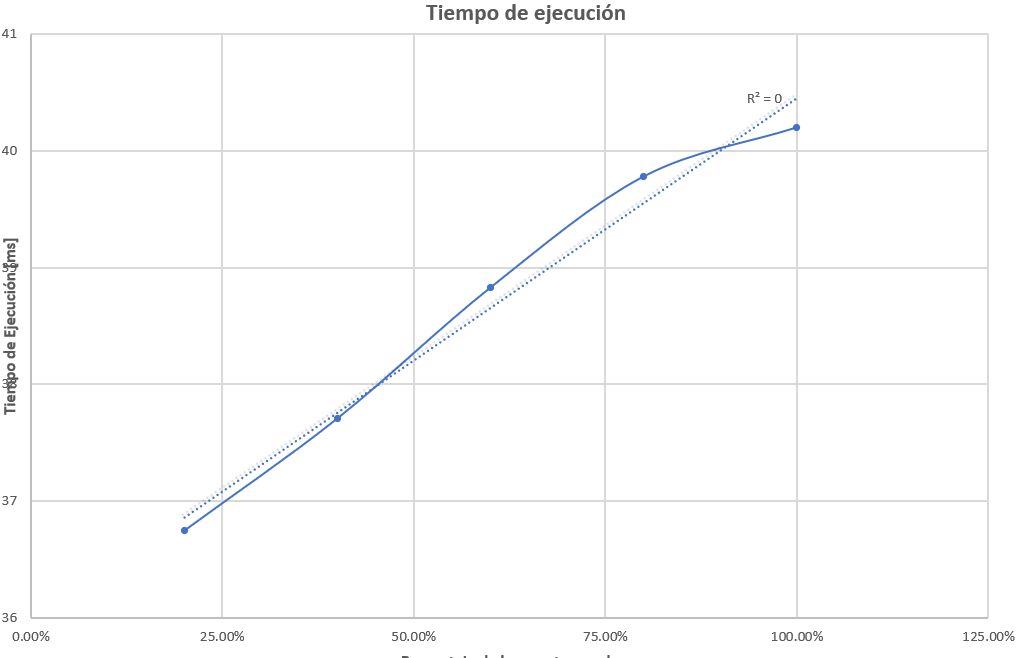
|  |  |
| --- | --- |
| Procesadores | Intel core 17 7th gen |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 10 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Tiempo de ejecución ms** | |
| **Porcentaje de la muestra [pct]** | **Tiempo de ejecución (ms)** |
| 20.00% | 36.75 |
| 40.00% | 37.71 |
| 60.00% | 38.83 |
| 80.00% | 39.78 |
| 100.00% | 40.2 |

### **Graficas**



## **Análisis**

El comportamiento creciente del tiempo en el requerimiento 7 se explica porque el algoritmo debe filtrar nodos, construir un subgrafo y ejecutar Prim, cuya complejidad es O(Elog E) usando una priority (lazy) queue. Así, al aumentar la cantidad de ubicaciones y conexiones del domiciliario, el tiempo de ejecución crece de forma eficiente pero notoria, reflejando el costo de procesar más nodos y aristas en el MST.