**Análisis de Resultados**

**Reto 4 Grupo 4**

**Integrantes**

Ashlee Yin Romero, a.yin@uniandes.edu.co 202421132

Daniel Galindo, d.galindot@uniandes.edu.co, 202414673

Tomás Lozano, t.lozanoc@uniandes.edu.co 202422837

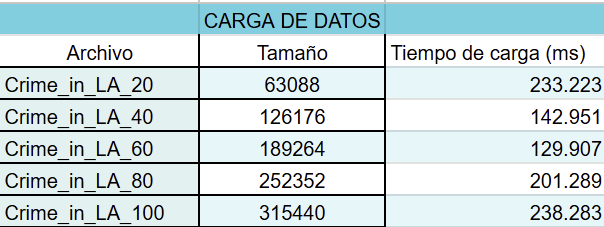
**Especificaciones**

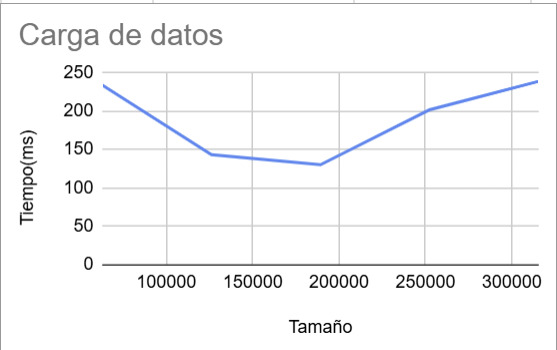
La medición del tiempo de las funciones fueron tomadas en un computador MacBook Air 2015, viejita pero bien cuidada, con un procesador 1,6 GHz Dual-Core Intel Core i5 y memoria de 8 GB 1600 MHz DDR3. Todos los tiempos fueron tomados en milisegundos.

### **Carga de Datos**

**Descripción:** Cargar y almacenar los datos en estructuras eficientes.  
**Complejidad:** La carga de datos tiene una complejidad O(V) + O(E), donde V son los nodos y E son todos los arcos.

**Análisis:**Durante la carga, los datos se procesan e insertan en un grafo no dirigido implementado con mapas de hashing (para nodos) y listas de adyacencia (para aristas), optimizando búsquedas en O(1) promedio. Las coordenadas se normalizan truncando a 4 decimales, y los tiempos de viaje se promedian para evitar aristas paralelas, mientras que un diccionario de historial registra los pedidos por domiciliario para generar conexiones adicionales eficientemente (O(M), donde M es el número de pares consecutivos).

****

****

### 

### 

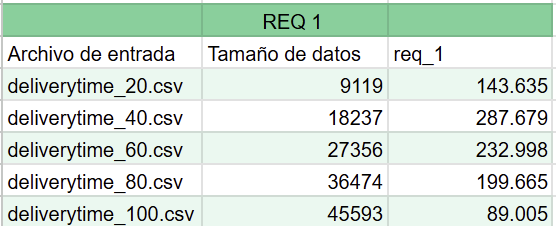
### 

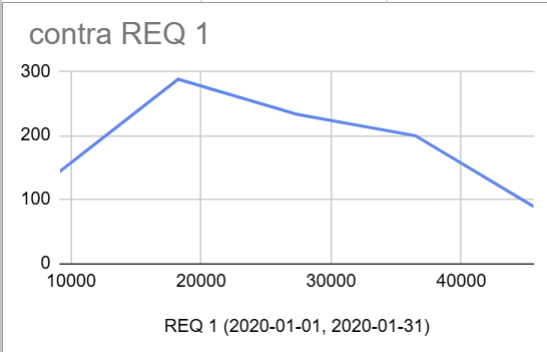
### **Requerimiento 1 (Grupal)**

**Descripción:** Identificar un camino simple entre dos ubicaciones geográficas

**Complejidad:** O(V + E)

**Análisis:**Se implementará BFS (Breadth-First Search) para encontrar el camino más corto en términos de número de aristas (puntos intermedios).



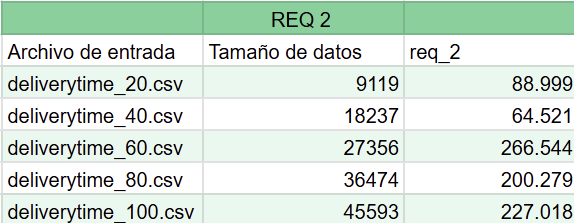


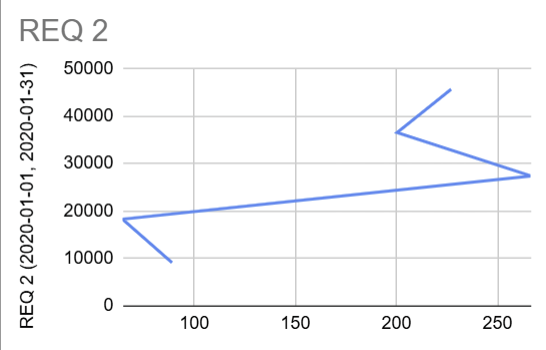
### **Requerimiento 2 (Grupal)**

**Descripción:** Camino con menos puntos intermedios para un domiciliario

**Complejidad:** O((V' + E') log V')

**Análisis:**Se filtrará el grafo para incluir solo nodos visitados por el domiciliario (usando el diccionario de historial). Aplicar Dijkstra sobre este subgrafo para minimizar el peso total (tiempo de viaje).



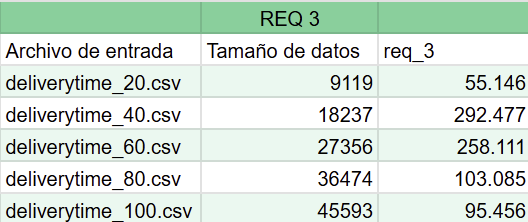


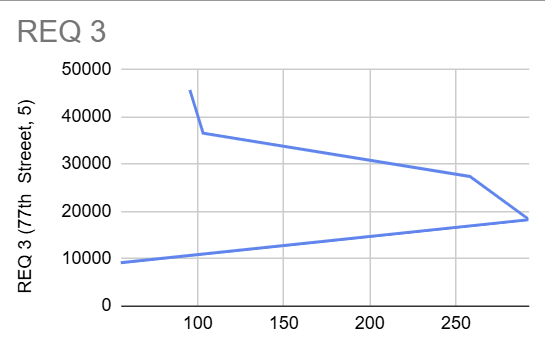
### 

### **Requerimiento 3 (Individual)**

**Descripción**: Identificar el domiciliario con mayor cantidad de pedidos asociados a un nodo específico (restaurante o destino).  
**Complejidad**: O(D), donde D = número de domiciliarios distintos en el nodo.  
**Análisis**:

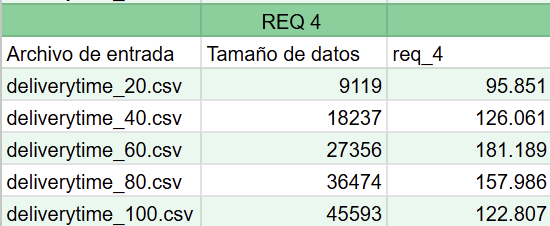
1. Acceso al nodo en el grafo en O(1) (mapa de hashing).
2. Conteo de frecuencias de domiciliarios en su lista domiciliarios (puede contener duplicados).
3. Uso de un mapa auxiliar para conteo en O(D).

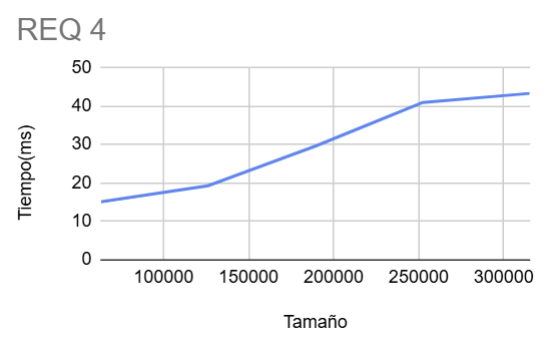




### **Requerimiento 4 (Individual)**

**Descripción**: Encontrar domiciliarios que hayan visitado ambos puntos geográficos.  
**Complejidad**: O(D₁ + D₂), donde Dᵢ = domiciliarios únicos en cada punto.  
**Análisis**: Obtención de listas de domiciliarios de A y B en O(1). Intersección con sets para eliminar duplicados.

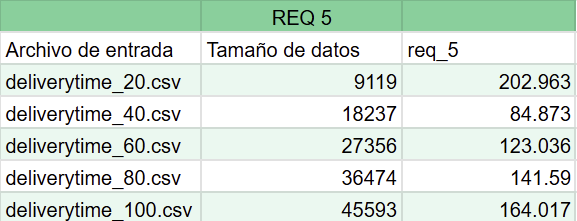


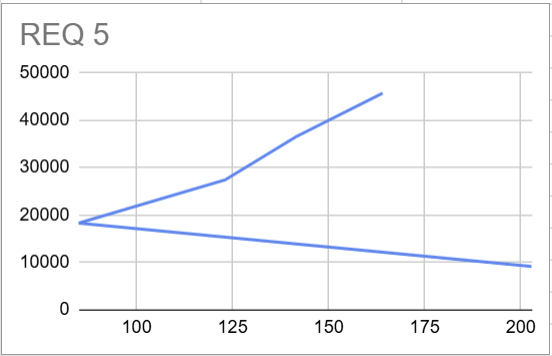


### **Requerimiento 5 (Individual)**

**Descripción**: Calcular el domiciliario con máxima distancia acumulada en sus primeros N cambios de ubicación desde un punto inicial.  
**Complejidad**: O(H), donde H = total de pedidos en el historial.  
**Análisis**:

1. Recorrer el historial del domiciliario: para cada pedido, calcular distancia Haversine entre destino actual y anterior.
2. Sumar las primeras N distancias válidas y seleccionar el máximo global.





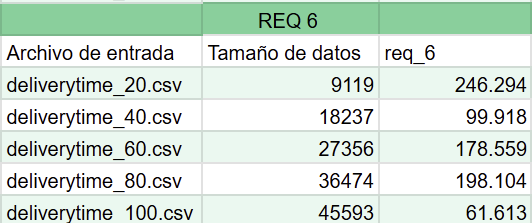
**Requerimiento 6**

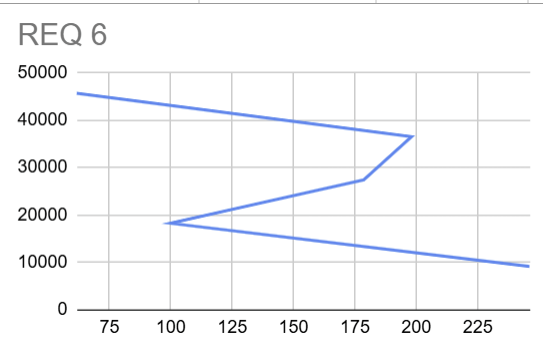
**Descripción:** Encontrar las rutas más rápidas desde un nodo inicial a todos los demás puntos del grafo.

**Complejidad:** O((V + E) log V)

**Análisis:**

Se ejecutará el algoritmo de Dijkstra desde el nodo geográfico inicial para calcular los caminos más cortos en términos de tiempo hacia todos los demás nodos del grafo completo. Se usará una cola de prioridad (heap) para optimizar la selección del siguiente nodo a expl





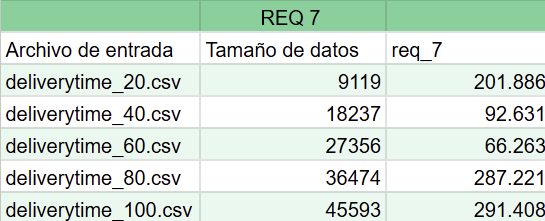
### **Requerimiento 7 (Grupal)**

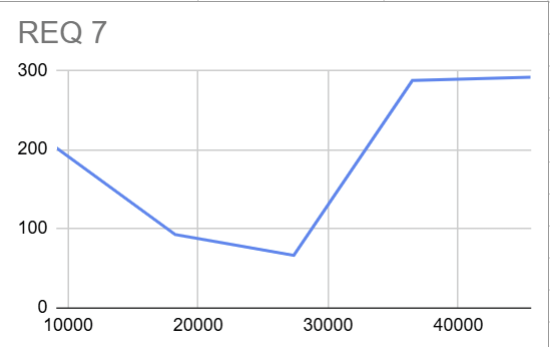
**Descripción:** Generar una subred óptima que conecte todos los nodos visitados por un domiciliario con el menor costo total de tiempo.

**Complejidad:** O((V' + E') log V')

**Análisis:**

Se filtrará el grafo completo para obtener un subgrafo con solo los puntos visitados por el domiciliario, usando su historial. Luego se aplicará el algoritmo de Prim o Kruskal para generar el árbol de recubrimiento mínimo (MST) que conecte todos los nodos del subgrafo desde un nodo inicial dado.





**Requerimiento 8 (BONO)**

**Descripción**: Visualizar el camino real recorrido por un domiciliario en el mapa, según su historial de visitas.

**Complejidad**: O(n), donde n es la cantidad de puntos en el historial.

**Análisis**:

Se utilizará el diccionario de historial para extraer los puntos geográficos visitados en orden. Cada punto se conectará con el siguiente mediante líneas sobre el mapa. La visualización se apoyará en coordenadas geográficas asociadas a cada nodo del grafo.

