ANÁLISIS DEL RETO

*Estudiante 1 Juan Goyeneche, 202320863,* [*j.goyeneches@uniandes.edu.co*](mailto:j.goyeneches@uniandes.edu.co)

*Estudiante 2 Mariana González, 202311308,* [*m.gonzalezp23@uniandes.edu.co*](mailto:m.gonzalezp23@uniandes.edu.co)

*Estudiante 3 Juan Jose Pineda, 202225439, jj.pineda@uniandes.edu.co*

# Requerimiento <<1>>

## Descripción

Este requerimiento se encarga de encontrar el camino entre dos aeropuertos.

|  |  |
| --- | --- |
| **Entrada** | Punto de origen y de destino |
| **Salidas** | -Tiempo de ejecución  -Distancia entre el punto de origen y el de destino  -Numero de aeropuertos que se visitan  -Del camino encontrado se da aeropuerto de origen, vuelos aeropuerto de destino y el tiempo del trayecto |
| **Implementado (Sí/No)** | Si. Implementado por todos. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtener el elemento (get) aeropuerto | O(log n) |
| (Iteración en lista) | O(n) |
| Hacer el sort | O(n log n) |
| Conversion Haversine | O(n) |
| algoritmo | O(E log V) |
| ***TOTAL*** | O(n log n) |

## Análisis

Este requerimiento obtiene 2 aeropuertos, uno de origen y otro de destino, seguido a esto, obtiene la ruta desde el aeropuerto de origen hasta el de destino. Finalmente, hace la ruta desde el aeropuerto de origen hasta el de destino.

# Requerimiento <<2>>

## Descripción

Este requerimiento se encarga de retornar el camino con menor número de escalas de un lugar de origen a un destino, ingresados por parámetro

|  |  |
| --- | --- |
| **Entrada** | Latitud origen, longitud origen, latitud destino, longitud destino |
| **Salidas** | -distancia total del recorrido  -Aeropuertos visitados  -Información del recorrido |
| **Implementado (Sí/No)** | Si. Implementado por Juan Goyeneche y Mariana González y Juan Jose Pineda. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 (BFS) | O(V + E) |
| Paso 1 (Dijkstra) | O(E logV) |
| Paso 3 (getEdge) | O(1) |
| ***TOTAL*** | ***O (***E logV)***)*** |

## Análisis

La función distancia\_a\_aero tiene como parámetros la lista de todos los aeropuertos y una tupla que contiene la latitud y la longitud de la ubicación que se ingresa por parámetro; para utilizar el método haversine y obtener la distancia de la ubicación al aeropuerto más cercano. Ahora bien, en la función req\_2 se utiliza la funcion mencionada dos veces para poder obtener la distancia del lugar de origen al aeropuerto más cercano, al igual que la del destino. Cuando ya se tiene el aeropuerto de origen se utiliza dijkstra y pathTo para encontrar los caminos hacia el vértice destino. Si existe un vuelo directo se extrae la distancia utilizando getEdge. Finalmente, se suman las distancias para obtener la distancia total del recorrido.

# Requerimiento <<3>>

## Descripción

Este requerimiento se encarga de planificar una red de trayectos para cubrir la mayor cantidad de aeropuertos posibles con vuelos comerciales desde el aeropuerto con más importancia.

|  |  |
| --- | --- |
| **Entrada** | No hay parametros de entrada |
| **Salidas** | -Tiempo de ejecución  -aeropuerto más importante y su concurrencia  -distancia total de los trayectos sumada  -número de trayectos posibles partiendo desde el aeropuerto de mayor importancia  -secuencia de trayectos y su información |
| **Implementado (Sí/No)** | Si, Mariana González |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| mp.get | O(1) |
| Dijkstra | O(E logV) |
| For pesos | O(n) |
| mp.get | O(1) |
| ***TOTAL*** | ***O(E logV + n)*** |

## Análisis

Este requerimiento en primera instancia busca cual es el aeropuerto más concurrido según el número de vuelos que salen y llegan de este. Después del mapa se extrae la información de ese aeropuerto para poder utilizar el algoritmo Dijkstra y encontrar los vértices que están conectados al aeropuerto de mayor importancia. Una vez se tengan las conexiones a los vértices se saca el peso de estos, para obtener la distancia total que se puede recorrer.

# Requerimiento <<4>>

## Descripción

Este requerimiento se encarga de retornar el número de ofertas que se realizan en un pais (dado por parámetro) y la información de estas ofertas. Primero se filtra el mapa countries tomando en cuenta el pais y se recorre la lista. Después las ofertas que cumplen con las fechas se agregan a una lista para ser ordenadas cronológicamente. Finalmente se retornan las ofertas deseadas

|  |  |
| --- | --- |
| **Entrada** | Numero de ofertas, ciudad, tipo de trabajo (remoto, oficina, parcialmente remoto) |
| **Salidas** | Total de ofertas ofrecidas en la ciudad para la ubicación deseada, las N ofertas y su respectiva Info |
| **Implementado (Sí/No)** | Si, Juan Goyeneche |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtener el elemento (get) | O(log n) |
| (Iteración en lista) | O(n) |
| Obtener el tamaño (size) | O(1) |
| Addlast (addLast) | O(1) |
| ***Hacer el sort*** | O(n log n) (promedio) |
| ***TOTAL*** | ***O(n log n)*** |

## Pruebas Realizadas

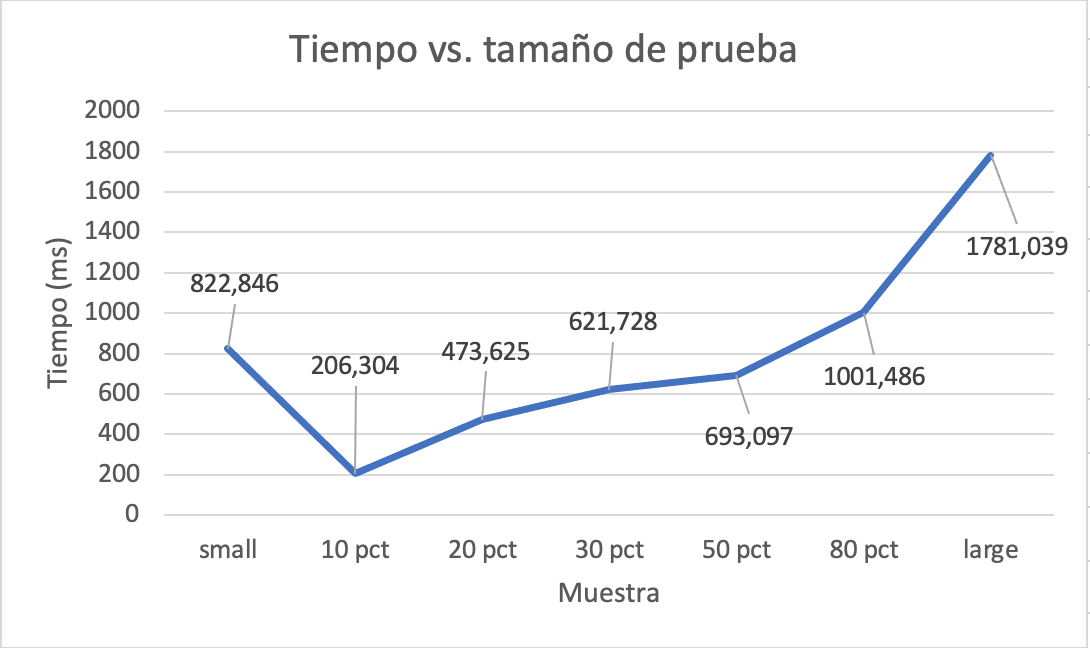
Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada 5, remote, berlin.

|  |  |
| --- | --- |
| **Procesadores** | **AMD Ryzen 7 4800HS with Radeon Graphics** |
| **Memoria RAM** | 8 GB |
| **Sistema Operativo** | Windows 10 |

|  |  |
| --- | --- |
| **Muestra** | **Tiempo (ms)** |
| small | 822.846 |
| 10 pct | 206.304 |
| 20 pct | 473.625 |
| 30 pct | 621.728 |
| 50 pct | 693.097 |
| 80 pct | 1001.486 |
| large | 1781.039 |

### Gráficas

Las gráficas con la representación de las pruebas realizadas.



## Análisis

Este requerimiento tiene que obtener un nodo del árbol que tenga la ciudad y la ubicación deseada para después hacer un sort, por ello tiene una complejidad de O(n log n). Primero, este tiene que obtener el nodo de un árbol (RBT) que sea la ciudad dada, por ello tiene una complejidad de O(log n). Esto debido a lo que se hace después de obtener los trabajos en esa ciudad, es buscar de todos los elementos de esa lista que tenga la misma ubicación, después agregarlos a una lista y finalmente, organizar esa lista y agregar a una sublista. Debido a esto, en el peor de los casos es necesario recorrer toda la lista, seguido a esto agregar a una nueva lista y finalmente, organizar esta lista, agregar a una sublista y devolver el resultado.

Viendo el comportamiento de la gráfica, podemos darnos cuenta de que si tiene un compartimiento similar a O(n log n) ya que a medida que crece la muestra, el tiempo crece de manera similar (n log n), manteniendo una relación con el aumento del tiempo y el aumento de los datos. A diferencia del anterior requerimiento, esta si mantiene una relación con (n log n) puede ser debido a que el orden es por fecha y al obtener un elemento del árbol por una ubicación especifica el orden se ve afectado, por otro lado, en el req anterior no usábamos una xp especifica y por ello todos los elementos del nodo deseado eran validos bajando así la complejidad.

# Requerimiento <<5>>

## Descripción

Este requerimiento se encarga de retornar el aeropuerto mas importante según la concurrencia militar y los posibles trayectos partiendo desde este aeropuerto.

|  |  |
| --- | --- |
| **Entrada** | None |
| **Salidas** | -Tiempo de ejecución  -aeropuerto más importante y su concurrencia  -distancia total de los trayectos sumada  -número de trayectos posibles partiendo desde el aeropuerto de mayor importancia  -secuencia de trayectos y su info |
| **Implementado (Sí/No)** | Si, Juan Goyeneche |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtener el elemento (get) | O(1) |
| (Iteración en lista) | O(n) |
| Obtener el tamaño (size) | O(1) |
| algoritmo | O(E log V) |
| ***Hacer el sort*** | O(n log n) (promedio) |
| ***TOTAL*** | ***O(n log n)*** |

## Análisis

Este requerimiento obtiene el aeropuerto con más entradas y salidas de vuelos militares, seguido a esto, obtiene la ruta desde el aeropuerto mas relevante hasta los posibles aeropuertos a los que puede llegar en el país. Además, se obtiene la cantidad total de km entre todos los posibles trayectos. Debido a esto se usa el grafo dirigido donde los pesos son las distancias en km entre los aeropuertos.

# Requerimiento <<6>>

Este requerimiento se encarga de retornar el aeropuerto más importante según la concurrencia comercial y los trayectos de distancia a mínima a los M aeropuertos comerciales más importantes.

## Descripción

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Cantidad de aeropuertos mas importantes |
| **Salidas** | -Tiempo de ejecución  -aeropuerto mas importante y su concurrencia  -secuencia de todos los trayectos posibles y su info de cada aeropuerto por el que pasa. |
| **Implementado (Sí/No)** | Si. Implementado por Todos |

## Análisis de complejidad

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtener el elemento (get) | O(1) |
| (Iteración en lista) | O(n) |
| Obtener el tamaño (size) | O(1) |
| algoritmo | O(E log V) |
| *Hacer sublista* | O(n) |
|  |  |
| ***TOTAL*** | ***O(n log n)*** |

## Análisis

Este requerimiento obtiene el aeropuerto con más entradas y salidas de vuelos comerciales, seguido a esto, obtiene los M aeropuertos con más entradas y salidas comerciales que usuario desee. Finalmente, construye una ruta de menor distancia posible desde el aeropuerto con más importancia hasta los M aeropuertos que desea el usuario. Es por esto que se usa el grafo dirigido donde los pesos son las distancias en km entre los aeropuertos.

# Requerimiento <<7>>

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

## Descripción

## Este requerimiento se encarga de mostrar la gráfica de la propiedad elegida del país y el año ingresados por parámetro. Lo primero que se hace es filtrar el mapa y el arbol RBT para obtener una lista con las ofertas validas. Se hace un if por cada una de las tres propiedades, para filtrar la ciudad, el nivel de experiencia o la habilidad. Se crea un diccionario en el cual las llaves son los datos de la propiedad y los valores son el número de veces que se repiten en la lista de ofertas validas.

|  |  |
| --- | --- |
| **Entrada** | Año relevante, código del país, propiedad de conteo (experticia, ubicación, habilidad) |
| **Salidas** | Número de ofertas publicadas durante ese año, número de ofertas utilizadas para hacer el gráfico, valor mínimo y máximo de la gráfica, gráfica, listado de ofertas laborales y su información requerida. |
| **Implementado (Sí/No)** | Si. Implementado por Juan Goyeneche y Mariana González |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| mp.get() | O(1) |
| me.getValue() | O(1) |
| om.get() | O(log n) |
| nodoRBT.getValue() | O(log n) |
| For experiencia | O(n) |
| For filtro ciudades | O(n) |
| For diccionario | O(n) |
| For lista ides | O(n) |
| me.get() skills | O(1) |
| me.getValue() skills | O(1) |
| ***TOTAL*** | ***O(log n)*** |

## Pruebas Realizadas

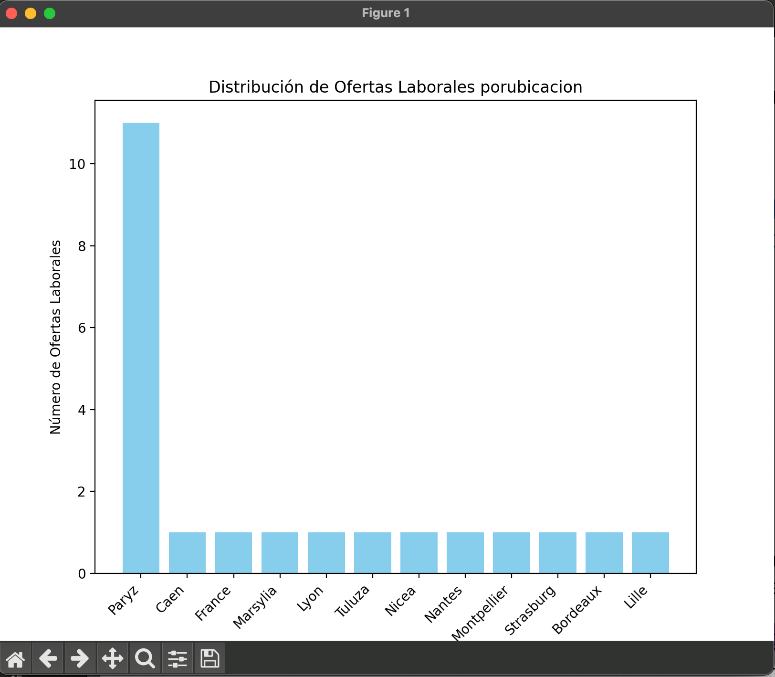
Las pruebas realizadas se realizaron en una maquina con estas especificaciones. Los datos de entrada fueron FR, 2023, ubicación

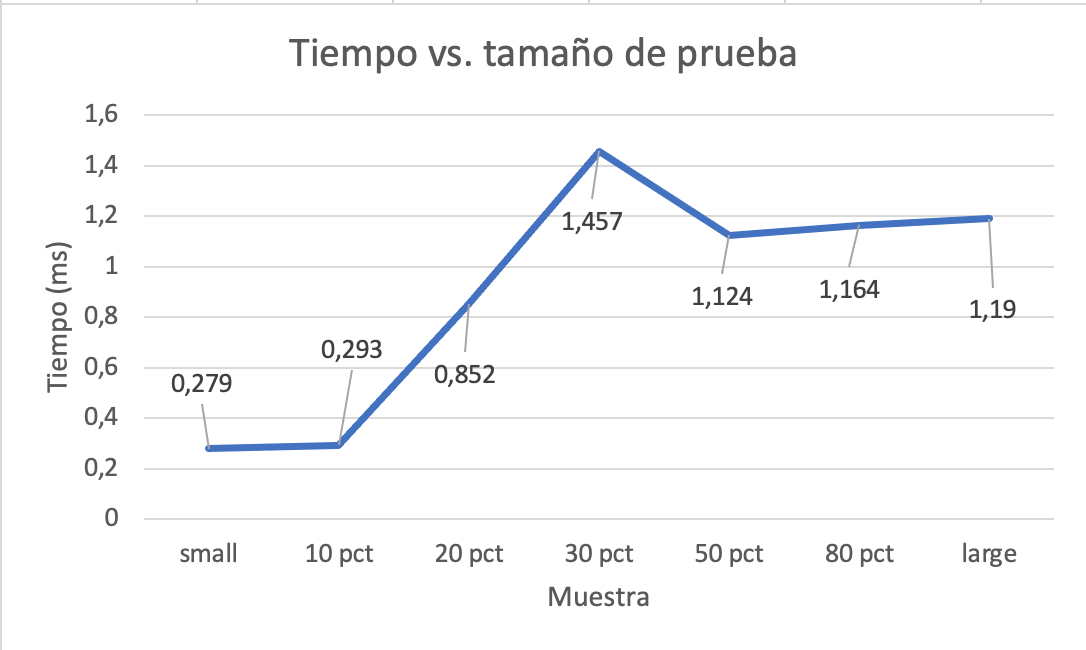
|  |  |
| --- | --- |
| **Procesadores** | **AMD Ryzen 7 4800HS with Radeon Graphics** |
| **Memoria RAM** | 8 GB |
| **Sistema Operativo** | MacOS Sonoma |

|  |  |
| --- | --- |
| **Muestra** | **Tiempo (ms)** |
| small | 0.279 |
| 10 pct | 0.293 |
| 20 pct | 0.852 |
| 30 pct | 1.457 |
| 50 pct | 1.124 |
| 80 pct | 1.164 |
| large | 1.190 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.





## Análisis

La estructura de datos para este requerimiento es un mapa que tiene como llaves los países y valor un árbol RBT, el cual a su vez tiene como llave los años y como valor un ARRAY\_LIST que contiene las ofertas de trabajo que son válidas según los parámetros dados. Primero se realizan los gets necesarios con complejidad O(1) y O(logn) para obtener la lista de la cual se basará la gráfica.

Cuando la propiedad deseada es experticia se crea el diccionario con cada una de las llaves (‘senior’, ‘mid’, ‘junior’) y se realiza un for para que según la experticia de cada oferta se agregue uno al valor de la llave necesaria.

Cuando la propiedad requerida es ubicación, se realiza un for para tener una lista sin las ubicaciones repetidas y por lo tanto no estén repetidas en el diccionario, se realiza otro for en el cual se crea el diccionario y por cada oferta de esa ciudad se agrega uno al valor.

Finalmente, cuando la propiedad deseada es habilidad, se filtra la lista para que los ides no estén repetidos; una vez con esta lista se genera un for para obtener la información de cada id del archivo skills , haciendo un get() y un getVlue(), ambos con complejidades O(1). Después se crea el diccionario y por cada oferta con ese ID se genera una llave y se le agrega uno a su valor.

Por último, se utiliza la librería matplotLib para generar los gráficos.

La complejidad final del requerimiento es O(log n) debido a que los get() de tablas de hash son O(1), mientras que los get() y getValue() de los arboles RBT son O(log n) siendo los de mayor complejidad.

# Requerimiento <<8>>

Este requerimiento se encarga de graficar los requerimientos anteriores

## Descripción

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Lista de los aeropuertos |
| **Salidas** | Un Grafico de el resultado de cada req |
| **Implementado (Sí/No)** | Si. Implementado por todos |

## Análisis de complejidad

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Iterar sobre la lista dada (depende del req) | O(n) |
| ***TOTAL*** | ***O(n)*** |

## Análisis

Este requerimiento tiene que obtener los elementos de un array\_list y mostrar su ubicación en el mapa, por ello tiene una complejidad de O(n). Esto se debe a que se itera sobre la lista propuesta y después con folium se muestran laeropuertos y sus rutas en el mapa.

# Carga de datos:

Este requerimiento se encarga de hacer un grafo el cual muestre los vuelos que se generan desde los aeropuertos y los vuelos que llegan a estos, teniendo como peso la distancia entre los dos o el tiempo de recorrido.

## Descripción

Breve descripción de como abordaron la implementación del requerimiento

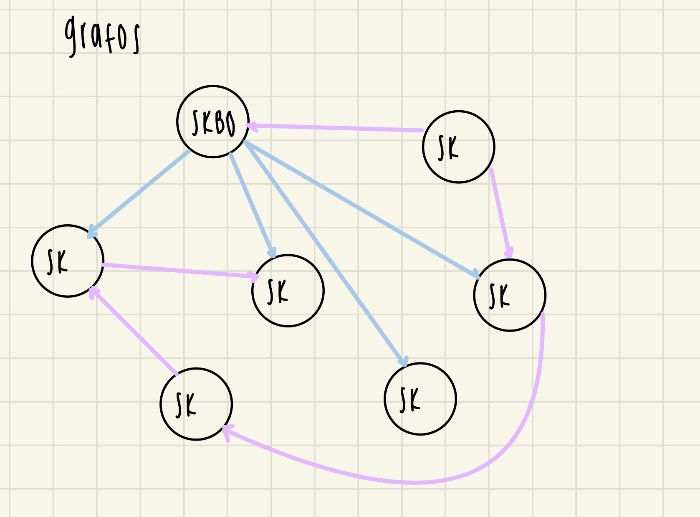
|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo |
| **Salidas** | Grafos con pesos los cuales son el tiempo o la distancia en km  -Mapa que tiene como llave el aeropuerto y como valor la información de este  -Mapa que tiene como llave el vuelo(origen-destino) y como valor la información de este |
| **Implementado (Sí/No)** | Si. Implementado por todos |

## Análisis de complejidad

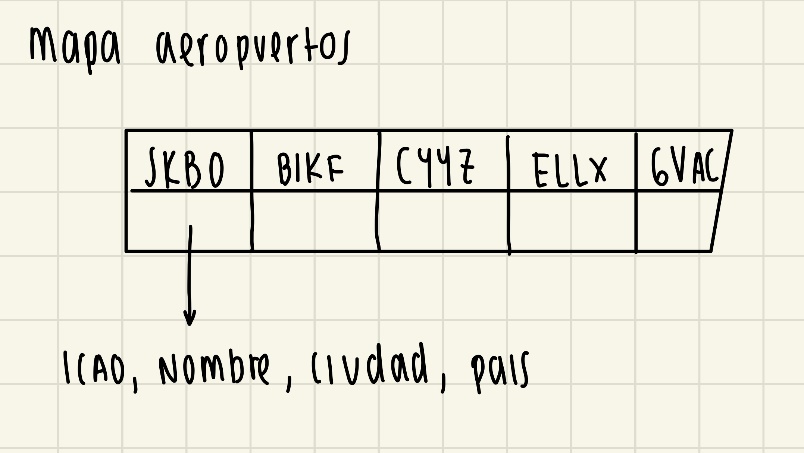
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Buscar si el elemento existe (isEmpty) | O(1) |
| Obtener el elemento (getElement) de los mapas | O(n) |
| Craer mapas | O(n) |
| (Conversión formato de km) | O(n) |
| Crear los diccionarios | O(n) |
| Añadir los vertices(aeropuertos) | O(n) |
| Añadir los arcos | O(n) |
| ***TOTAL*** | ***O(n)*** |

## Análisis

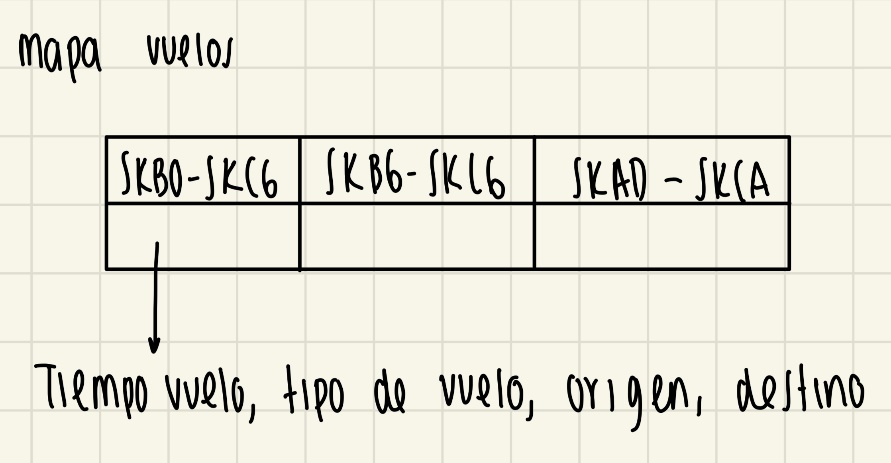
Se crean los grafos a partir de la info en mapas y sus vértices son los aeropuertos mientras que los arcos son los vuelos. En un grafo los pesos de los arcos son los km entre el aeropuerto de origen y destino y en el otro es el tiempo, se hace uso de grafos dirigidos. Los mapas son los aeropuertos y otro los vuelos.



*Imagen 1*: Representación gráfica de la estructura de los grafos



*Imagen 2*: Representación gráfica de la estructura del mapa de aeropuertos



*Imagen 3*: Representación gráfica de la estructura del mapa de vuelos