Integrantes	
Laura Maya (Pares)	201423854
Carlos Peñaloza (Impares)	201531973
Repositorio Bitbucket	
https://bitbucket.org/proyecto_201620_sec_1_team_14/proyecto_3_201620_sec_1_team_14	

Introducción

Realice una breve explicación de que hicieron y cómo llegaron a la solución propuesta.

Para llegar a la solución utilizamos diferentes estructuras tales como grafos, colas de prioridad indexadas, listas encadenada, pilas, colas y tablas de Hash. Creamos un grafo que contiene vuelos en sus vértices y en los arcos las posibles conexiones entre estos vuelos teniendo en cuenta las ciudades de origen y destino, los dias en que operan estos vuelos y las horas de llegada y salida de los mismos. Esto con el fin de conocer las conexiones que se pueden hacer entre vuelos y asi encontrar camino entre las diferentes ciudades. Ademas de esto hicimos un grafo para conectar las ciudades y de esta manera fuera mas facil conocer que ciudades tienen conexion con las otras. Para guardar la informacion en general usamos tablas de hash en las cuales almacenamos las aerolineas, las ciudades y los vuelos que cargamos de los archivos.

Requerimientos Funcionales

Identifique los 3 requerimientos funcionales principales para el sistema propuesto y documentos siguiendo el siguiente formato:

Nombre	R1. Crear catálogo de vuelos
Resumen	Crea un nuevo catálogo con los vuelos que vienen en un archivo dado.
Entradas	
Archivo con la información de los vuelos.	
Resultados	
Se han leído los archivos y agregado los datos a las estructuras de datos del proyecto.	
Complejidad temporal	
La complejidad temporal de este algoritmo es O(n^2) ya que es lo que se demora en leer los datos y agregarlos a las estructuras de datos.	

Nombre	R2. Agregar una aerolínea al catálogo de vuelos.

Resumen	Insertar en el catálogo, un nueva aerolínea dada su información.
Entradas	
Información de la aerolínea a agregar con sus respectivos datos.	
Resultados	
Se ha agregado al catálogo de vuelos una nueva aerolínea.	
Complejidad temporal	
Para este requerimiento se estima una complejidad temporal de $O(1)$ pues a partir de la información ingresada, el catálogo debe únicamente insertar dicha aerolínea a sus estructuras, las cuales en su mayoría, la complejidad de inserción es constante.	

Nombre	R3. Eliminar una aerolínea del catálogo de vuelos.
Resumen	Eliminar una aerolínea del catálogo de vuelos.
Entradas	
Nombre de la aerolínea a eliminar.	
Resultados	
Se ha eliminado del catálogo y sus respectivas estructuras una aerolínea junto con su información.	
Complejidad temporal	
Para este requerimiento se estima una complejidad de $O(n)$ donde n es el número de vuelos de dicha aerolínea, pues a partir de la información ingresada, el catálogo debe únicamente eliminar de sus estructuras el objeto aerolínea con su información.	

Nombre	R4. Agregar y eliminar ciudades autorizadas para realizar vuelos autorizados.
Resumen	Agrega y elimina una ciudad de acuerdo a la información que llega por parámetro.
Entradas	
Nombre de la ciudad a eliminar o agregar.	
Resultados	
Se eliminó una ciudad junto a todos los arcos-vuelos que salen y entran a la misma. Se agrega una ciudad a una aerolínea y se crean todos los arcos correspondientes.	

Complejidad temporal
Para eliminar una ciudad dada se calcula una complejidad temporal de $O(n)$ donde n es el número de vuelos que salen y llegan a la ciudad. Asimismo, para eliminar se estima una complejidad de $O(n)$ pues por vuelo de dicha ciudad se debe agregar el arco correspondiente.

Nombre	R5. Agregar un vuelo al catálogo de vuelos
Resumen	Agregar un vuelo al catálogo de vuelos con la información dada por el usuario
Entradas	
Número de vuelo, aerolínea, ciudad de origen, ciudad destino, hora de salida, hora de llegada, tipo de avión, el cupo de vuelo, y los días de operación.	
Resultados	
Se agregó un nuevo vuelo en las estructuras de datos del programa.	
Complejidad temporal	
La complejidad temporal de agregar un vuelo es O(n+m) donde n es el número de colisiones ocurridas al buscar un puesto en la hash para el nodo y m es el número de vuelos de las ciudades origen y destino del vuelo agregado.	

Nombre	R6. Calcular y actualizar las tarifas de los vuelos de acuerdo a la fórmula dada.
Resumen	Para que cada arco maneje un peso característico se debe calcular el mismo de acuerdo a la fórmula dada.
Entradas	
Para calcular dicho peso es necesario tener: tarifa de un minuto de vuelo por cada aerolínea, duración del vuelo en minutos, número de sillas del vuelo, número máximo de sillas por vuelos y el día de la semana al cual pertenece el vuelo.	
Resultados	
Valor numérico que representa el peso del arco para un vuelo en específico dependiendo de ciertos atributos.	
Complejidad temporal	
La complejidad de este requerimiento es $O(1)$ pues únicamente debe acceder a los datos de cada vuelo al momento de lectura y usarlos para realizar operaciones básicas de cálculo.	

Nombre	R7. Informar las ciudades que están conectadas entre sí pero no con el resto del país.
Resumen	Encuentra y retorna la información de los componentes dentro del grafo.
Entradas	
Resultados	
Se retorna la información de las ciudades conectadas del grafo	
Complejidad temporal	
La complejidad temporal de este requerimiento es O(V+E) utilizando el algoritmo de Kosaraju.	

Nombre	R8. Informar las ciudades que están conectadas entre sí pero no con el resto del país para cada aerolínea.
Resumen	Encuentra y retorna la información de los componentes dentro del grafo
Entradas	
Resultados	
Se retorna la información de las ciudades conectadas del grafo	
Complejidad temporal	
La complejidad temporal de este requerimiento es O(V+E) utilizando el algoritmo de Kosaraju.	

Nombre	R9. Calcular e imprimir el MST (para cada componente conectado) para vuelos nacionales a partir de una ciudad específica utilizando como peso el tiempo de vuelo.
Resumen	Se encuentra el Minimum Spanning Tree para una ciudad dadas las restricciones.
Entradas	
Nombre de la ciudad de la que se encontrará el MST	

Resultados
Se calculó e imprimió el MST
Complejidad temporal
La complejidad temporal de generar el MST utilizando el algoritmo de Edmonds es O(E log(V))

Nombre	R10. Calcular e imprimir el MST (para cada componente conectado) para vuelos nacionales, para una aerolínea específica, a partir de una ciudad específica utilizando como peso el costo de vuelo.
Resumen	Se encuentra el Minimum Spanning Tree para una ciudad dadas las restricciones.
Entradas	
Nombre de la ciudad de la que se encontrará el MST y nombre de la aerolínea.	
Resultados	
Se calculó e imprimió el MST	
Complejidad temporal	
La complejidad temporal de generar el MST utilizando el algoritmo de Edmonds es O(E log(V))	

Nombre	R11. Calcular e imprimir el MST (para cada componente conectado) para vuelos nacionales a partir de una ciudad específica utilizando como peso el tiempo de vuelo.
Resumen	Se encuentra el Minimum Spanning Tree para una ciudad dadas las restricciones.
Entradas	
Nombre de la ciudad de la que se encontrará el MST y nombre de la aerolínea.	
Resultados	
Se calculó e imprimió el MST	
Complejidad temporal	
La complejidad temporal de generar el MST utilizando el algoritmo de Edmonds es O(E log(V))	

Nombre	R12. Calcular e imprimir el itinerario de menor costo para cada aerolínea dada una ciudad de origen, una de destino y un día particular (si es posible)
Resumen	Se utilizará el algoritmo de Dijkstra para generar un itinerario de menor costo para algún trayecto dada una fecha de salida
Entradas	
Ciudad de origen, ciudad destino, fecha salida	
Resultados	
Se creó un itinerario de menos costo y se imprimió en orden ascendente por precio.	
Complejidad temporal	
La complejidad temporal utilizando el algoritmo de Dijkstra es de O(E log (V))	

Nombre	R13. Calcular e imprimir el itinerario de menor costo dada una ciudad de origen, una de destino y un día particular (si es posible), pueden haber distintas aerolíneas en el itinerario
Resumen	Se utilizará el algoritmo de Dijkstra para generar un itinerario de menor costo para algún trayecto dada una fecha de salida
Entradas	
Ciudad de origen, ciudad destino, fecha salida.	
Resultados	
Se creó un itinerario de menos costo y se imprimió.	
Complejidad temporal	
La complejidad temporal utilizando el algoritmo de Dijkstra es de O(E log (V))	

Nombre	R14. Saliendo de una ciudad en particular, calcular e imprimir la ruta de costo mínimo para ir a todas las otras ciudades cubiertas por una aerolínea.
Resumen	Hacer uso de Edmond´s Algorithm con el fin de encontrar la ruta más óptima en términos de costo para ir a todas las ciudades con una misma aerolínea.

Entradas
Nombre de la ciudad origen.
Resultados
Información impresa en consola de la ruta óptima para recorrer todas las ciudades en el costo mínimo.
Complejidad temporal
La complejidad calculada para realizar Dijkstra Algorithm es O(E log (V)).

Nombre	R15. Saliendo de una ciudad en particular, calcular e imprimir la ruta de costo mínimo en término de tiempo para ir a todas las otras ciudades cubiertas por una aerolínea.
Resumen	Hacer uso de Edmond´s Algorithm con el fin de encontrar la ruta más óptima en términos de tiempo para ir a todas las ciudades con una misma aerolínea.
Entradas	
Nombre de la ciudad origen.	
Resultados	
Nombre de la ciudad origen.	
Complejidad temporal	
La complejidad calculada para realizar Dijsktra Algorithm es O(E log (V)).	

Explicación/Justificación de las estructuras de datos:

Tabla de Hash: La utilización de tablas de Hash es útil para el proyecto ya que estas permiten buscar e insertar datos con una complejidad temporal de O(1), sin importar el tamaño de la estructura, siempre y cuando se utilice una función de Hash adecuada.

Cola de prioridad: Las colas de prioridad son importantes ya que estas permiten ordenar la lista según se desee (mayor a menor o viceversa) mientras se están leyendo y agregando los datos, lo cual reduce la complejidad temporal requerida en el proyecto anterior al tener que leer y ordenar por aparte antes de poder procesar los datos.

Grafo: El grafo es la estructura principal del proyecto ya que este nos permitirá establecer las conexiones entre vuelos y entre ciudades. De esta manera si conocemos esta información podemos establecer el camino que debemos tomar para llegar de una ciudad a

otra, el tiempo que nos tomara y que vuelos debemos coger si buscamos la mínima duración o el mínimo costo.

Cola De Prioridad Indexada: La cola de prioridad indexada es importante ya que se necesita para hacer los algoritmos que nos permitirán encontrar el camino más corto sea en duración o en costo entre dos ciudades. Al ser indexadas podemos actualizar los valores que existen en la cola si lo necesitamos y así iremos tomando el nodo adyacente que más nos convenga considerando todos los caminos posibles para continuar y no solo los iniciales.

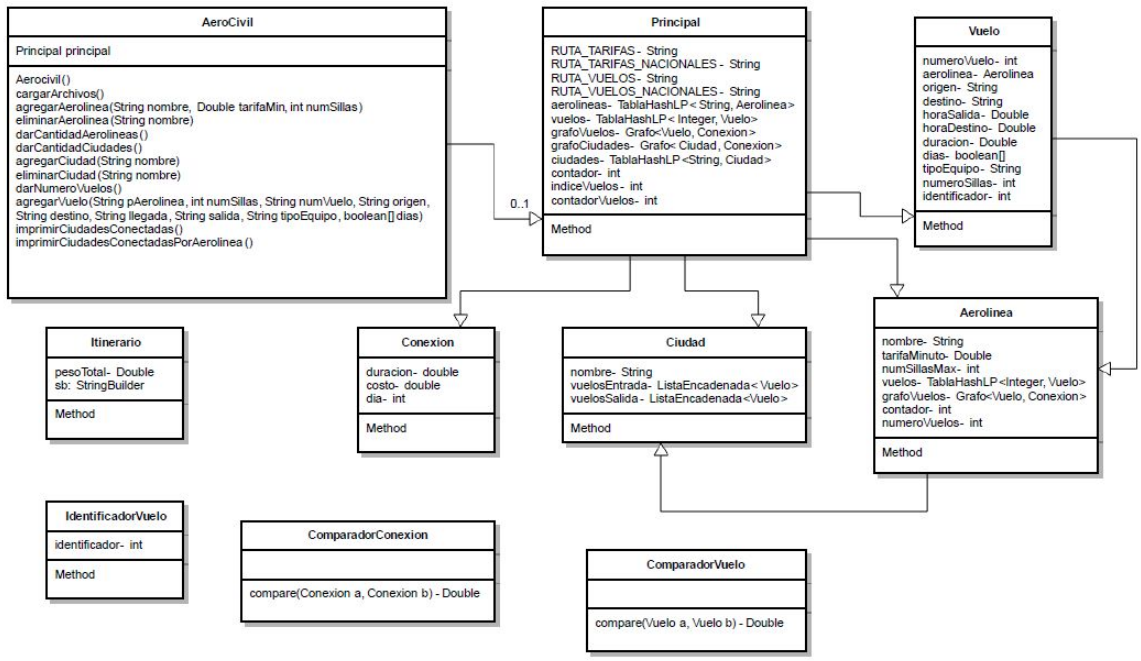
Lista Encadenada: Esta estructura la usamos principalmente para guardar las adyacencias de los vertice en el grafo y en algunos metodos como variables para guardar nuestros resultados.

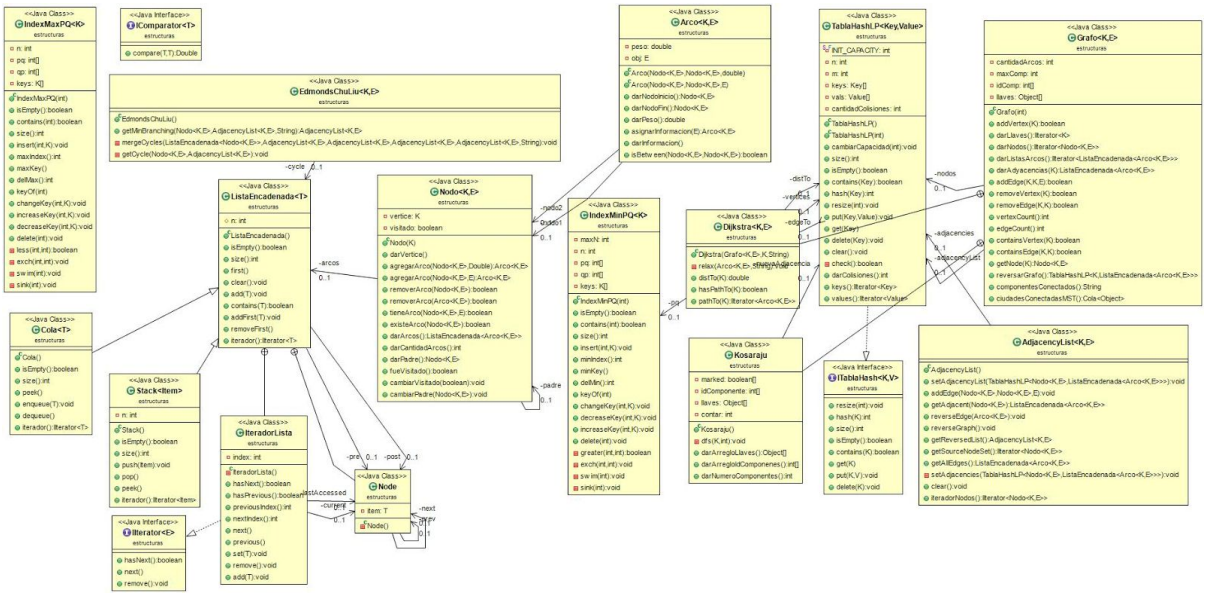
Pila: La pila la usamos para poder ejecutar correctamente el algoritmo de Dijkstra guardando la ruta entre dos vertices y poderla imprimir correctamente.

Modelo del mundo

Presente de forma clara su modelo del mundo usando la notación UML y explique los diferentes elementos del qué consiste (para cada parte del proyecto).

Diagramas UML





Diseño, implementación y pruebas de las estructuras de datos

Indique cómo las estructuras que usted creó, las cuales deben expresarse en el modelo UML, satisfacen los requerimientos funcionales propuestos por el cliente al igual como los no funcionales.

Indique qué cambios se realizaron desde la primera entrega y el porqué de ellos.

Desde la primera entrega cambiamos la lista ya que el iterador no funcionaba correctamente y agregamos la pila ya que se necesitaba para desarrollar el algoritmo de Dijsktra. En cuanto al mundo

Análisis de complejidad

Cargar los archivos

La complejidad temporal calculada para este requerimiento es de O(n^2) ya que el método de leer el archivo CSV que contiene la información de los vuelos y las aerolineas, realiza un doble recorrido con el fin de conocer lo que contiene cada celda.

Agregar Aerolinea:
Para este requerimiento se estima una complejidad temporal de O(1) pues a partir de la información ingresada, el catálogo debe únicamente insertar dicha aerolínea a sus estructuras, las cuales en su mayoría, la complejidad de inserción es constante.

Eliminar Aerolinea
Para este requerimiento se estima una complejidad de O(n) ya que debe eliminar la aerolinea y todos los vuelos que pertenezcan a esta aerolinea, donde n es el número de vuelos de dicha aerolínea,

Agregar y eliminar ciudad
Para agregar o eliminar una ciudad se estima una complejidad O(n) ya que se necesitan recorrer los vuelos y asi agregar o eliminar los que salgan o lleguen a esta ciudad segun lo que se necesite

Agregar vuelo
Para agregar un vuelo se estima una complejidad O(1) ya que estos se almacenan en una tabla de hash

Requerimiento 7.
Para este requerimiento se utilizó el algoritmo de Kosaraju el cual tiene una complejidad temporal de O(E+V)siendo V el número de vértices del grafo y E el número de arcos que hay.

Requerimiento 8
Para este requerimiento se utilizó el algoritmo de Kosaraju para poder encontrar los componentes fuertemente conectados entre aerolíneas. Para poder resolver este requerimiento se creó un nuevo grafo auxiliar, que estaba conformado por las ciudades de donde salían los vuelos de cada aerolínea. Siendo así una complejidad O(V2(E+V)) debido a que se debe hacer un doble recorrido sobre las ciudades para poder conectarlas en el grafo. Donde E representa el número de arcos, V el número de vértices.

Requerimiento 9
Para este requerimiento se estima una complejidad de O(EV) debido a que se usa el algoritmo de Edmonds' para poder calcular el MST en el grafo dirigido.

Requerimiento 10
Para este requerimiento se calculó una complejidad de O(EV) donde E es el número arcos que se encuentran en el grafo y V es el número de vértices. Esta complejidad se debe a que se utilizó el algoritmo de Chu-Liu/Edmonds', el cual tiene dicha complejidad temporal.

Requerimiento 11
Para este requerimiento se utilizó el algoritmo de Chu-Liu/Edmonds' debido a que de esta forma se podía obtener la arborescencia del grafo dirigido de la forma más eficiente. Es por eso que calculamos una complejidad de O(EV) debido a que dicho algoritmo en esta implementación cumple dicha complejidad.

Requerimiento 12
La complejidad temporal para este requerimiento fue de aproximadamente O(Elog(V)) debido a que se utilizó el algoritmo de Dijkstra.

Requerimiento 13

La complejidad temporal del algoritmo de Dijkstra es O(E log (V)), por ello, podemos decir que la complejidad de este requerimiento va a ser semejante a O(E log(V)).

Requerimiento 14
Para este requerimiento se calculó una complejidad de d O(E log(V)) debido a que se utilizó el algoritmo de Dijkstra.

Requerimiento 15
La complejidad calculada para realizar Edmond´s Algorithm es O(E log (V)).