**Trabajo practico especial Programacion-03**

**Alumnas: Venere Maia, Flamminio Brenda**

**Introducción:** La idea del trabajo especial era desarrollar un sistema aéreo, que cuente con la información correspondiente de los aeropuertos con sus aerolíneas y rutas. El sistema también debe contar con un servicio de reservas por día.

El usuario especificara sus necesidades y preferencias al realizar un viaje, nuestro programa debe satisfacer estas peticiones, como por ejemplo volar de un país a otro sin utilizar una aerolínea en particular, también saber la cantidad de kilómetros por recorrer, si es un vuelo con escalas o no, ofrecer todas las rutas disponibles desde un aeropuerto a otro y si hay asientos disponibles para determinado vuelo.

**Objetivo:** Conocer y trabajar con la estructura de datos brindada (Grafo) expandir nuestros conocimientos para poder saber cómo funciona y aplicarlo.

**Tareas involucradas:**

Lo primero que hicimos fue interiorizarnos un poco mas con grafos, conocer su estructura así poder definir como lo vamos a manipular. Seguimos leyendo y releyendo el enunciado así captar toda la información requerida y modelar la estructura interna acorde a los datos recolectados.

**Modelado del problema:**

Lo primero que identificamos fue la estructura del grafo donde encontramos que las aristas iban a ser nuestras rutas, y los nodos(vértices) serian los aeropuertos con su lista de adyacencias. En el grafo van a encontrarse varios aeropuertos que van a estar conectados por las rutas donde ellas además van a tener la información del vuelo, es decir cantidad de kilómetros y que aerolíneas realizan esos vuelos y su respectiva cantidad de asientos libres. Utilizamos la lista de adyacencia para guardar sus vecinos.

ARISTAS-RUTAS :

En esta clase tendremos dos HashMap uno de ellos guarda la información de todas las aerolíneas de esa ruta y el otro guarda todas las reservas de las aerolíneas de la ruta.

ArrayList (String) que solo contiene los nombres de dichas aerolíneas.

Un dato de tipo float que guarda la información de los kilómetros de ese recorrido.

Dos datos tipo AEROPUERTO donde uno es el aeropuerto origen y el otro aeropuerto destino de esta ruta en particular.

VERTICE-AEROPUERTO: En esta clase contamos con una ArrayList donde se guardan la lista de aristas(rutas) que serian adyacentes del aeropuerto.

Luego están los datos de tipo String que son País, Cuidad y nombre del aeropuerto. Luego tenemos la variable también String color que la vamos a utilizar cuando hagamos el dfs con backtraking para implementar el servicio 2.

RESERVA: Esta clase cuenta con un String con el nombre de la aerolínea, un INT donde se guardan la cantidad de asientos reservados. Dos variables tipo Aeropuerto un es el origen y la otra va ser el destino.

GRAFO: En esta clase se encuentran las ArrayList de los aeropuertos y las rutas. Desde aquí se controla gran parte del sistema.

**Inconvenientes – desventajas:** El primer inconveniente que tuvimos fue apresurarnos a modelar la estructura interna(clases) con los datos del enunciado sin abrir los archivos de texto también brindados, ya que luego nos dimos cuenta que había datos de los archivos de texto que no iban a encajar en nuestra estructura y decidimos repensar las clases y amoldarlas a las datos brindados.

Una desventaja fue utilizar el HashMap de aerolíneas en la clase ruta, no podíamos obtener la key del HashMap necesaria para una de nuestras funciones “asientosDisponibles” optamos por añadir además del HashMap una ArrayList de aerolíneas donde solo contenga los nombres (String)

También tuvimos problemas al abrir los archivos, donde no pudimos poner una ruta que no sea absoluta , no pudimos añadir los archivos de rutas y reservas, NULL POINTER EXCEPTION.

**SERVICIO 1:** Creamos una función donde al aeropuerto origen ingresado le preguntamos a su lista de adyacencia (rutas) si contiene al aeropuerto de destino. Si lo contiene este nos devuelve la cantidad de kilómetros que los separa, y también verifica que existan vuelos disponibles con la aerolínea que le usuario nos solicitó, si esto se cumple nos retorna la cantidad de asientos disponibles.

**SERVICIO 2:**

En este caso implementamos un DFS rediseñado con backtraking , donde íbamos buscando por todo el grafo todos los vuelos disponibles sean con o sin escala para un viaje de un aeropuerto origen a uno destino solicitado por un cliente donde además debíamos controlar no incluir en la lista de retorno la aerolínea que nuestro cliente pidió expresamente que no desea tomar. Debemos informarle al cliente , la lista de aerolíneas que puede utilizar para realizar el viaje , la cantidad de escalas si es la que las hay, y la cantidad de kilómetros por recorrer.

**SERVICIO 3:**

Esta implementado casi de la misma manera que el servicio 1 solo que tuvimos que agregar la condición de averiguar por una lista de aeropuertos origen(país a) y otra de aeropuertos destino(país b), uniendo así cada origen con su vuelvo directo al destino accesible para poder mostrar todos los vuelos disponibles, es decir que cuenten con asientos libres en el vuelo, y hacerle saber al cliente los kilómetros del recorrido.

**CONCLUSION :** Nos hubiera gustado en lo personal poderle dedicar mas horas de trabajo y consultas a este proyecto ya que no alcanzamos nuestras expectativas en cuanto a la entrega a realizar, nos quedamos con ganas de implementar mejor los servicios, y ver todos los servicios funcionando, al tener el error con la carga de los archivos hay muchas cosas que no pudimos corroborar por ende tampoco arreglarlas como nos hubiera gustado.