Trabajo Practico Taller de Algoritmos 2

# Catedra “A”

# Profesor: Juan Carlos Sapolski

# Alumno: Alejo Valentín Galetto

**Problemática**

Para este trabajo practico se utilizó una problemática del aeropuerto de Denver. El mismo actualmente cuenta con un total de 150 aerolíneas trabajando diariamente con más de 1500 rutas nacionales y 350 internacionales, necesita optimizar sus rutas a través de establecer vuelos óptimos y confiables con el fin de que sus usuarios lleguen más rápido a sus destinos y por mucho menos dinero.

Para resolver esto, se están realizando una serie de pruebas en la aerolínea Good Flym, donde se estableció un cronograma de vuelos con distintos destinos buscando implementar un sistema que permita determinar el camino más corto dentro de las múltiples opciones para llegar a un destino determinado.

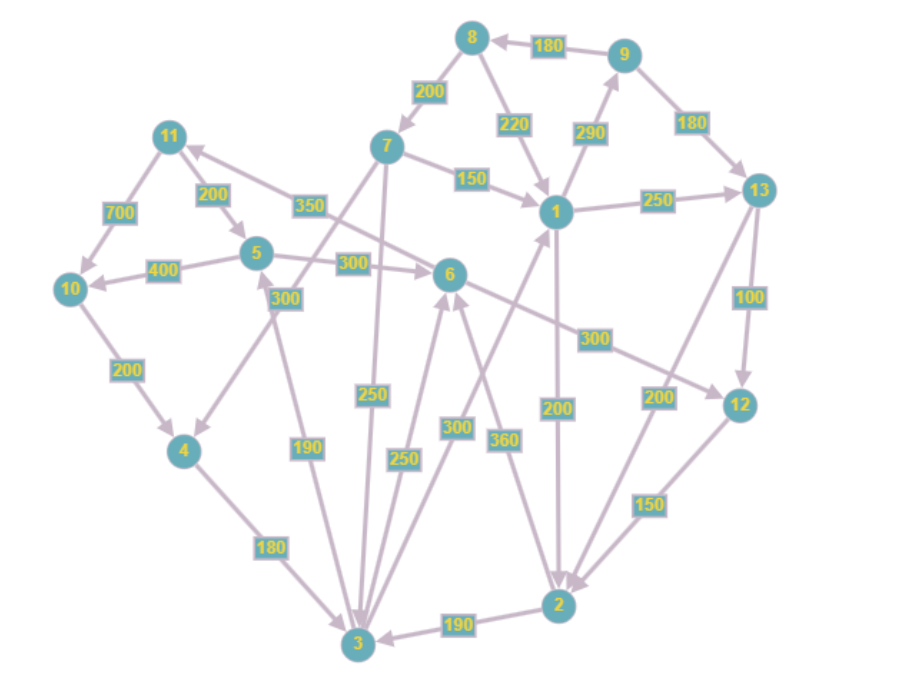
Se presentó un esquema de los distintos vuelos con sus respectivas distancias en (km) en una de las aerolíneas más colapsadas:

Destinos = { d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,11,d12,d13 }

Vuelos = {(d1,d2,200),(d1,d13,250),(d1,d9,290),(d2,d6,360),(d2,d3,190),(d3,d6,250), (d3,d5,190),(d3 ,d1,300),(d4,d3,180),(d5,d6,300),(d5,d10,400),(d6,d11,350),(d6,d12,300), (d7,d4,300),(d7,3,250),(d7,d1,150),(d8,d7,200),(d8,d1,220),(d9,d8,180),(d9,d13,180),

(d10,d4,200),(d11,d 10,700),(d11,d5,200), (d12,d2,150),(d13,d12,100),(d13,d2,200)}

**Diagrama** donde se puedan evidenciar los vértices, aristas y relaciones del esquema de viajes planteado

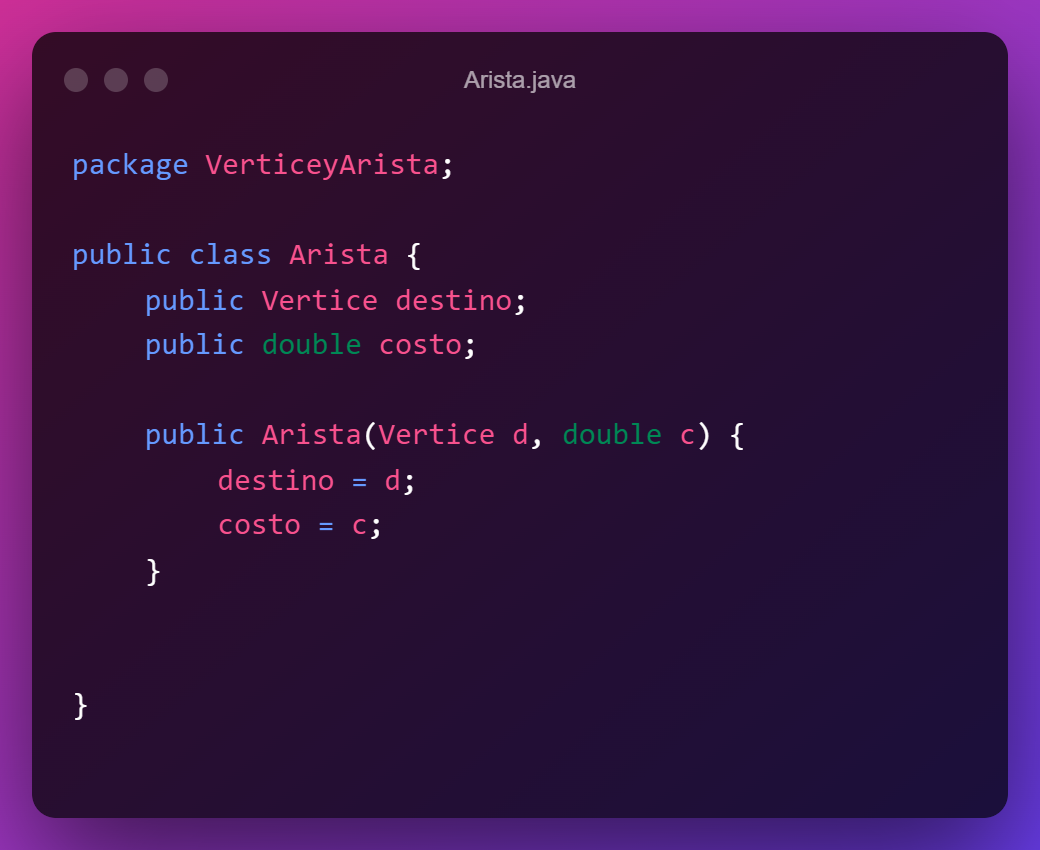


**Matriz de adyacencia del grafo**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 1 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 290 | 0 | 0 | 0 | 250 |
| 2 | 0 | 0 | 190 | 0 | 0 | 360 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 300 | 0 | 0 | 0 | 190 | 250 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 180 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 400 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 350 | 300 | 0 |
| 7 | 150 | 0 | 250 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 220 | 0 | 0 | 0 | 0 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 180 | 0 | 0 | 0 | 0 | 180 |
| 10 | 0 | 0 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 200 | 0 | 0 | 0 | 0 | 700 | 0 | 0 | 0 |
| 12 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 |

**Implementación del tipo abstracto de datos grafo, representando el modelo obtenido en la matriz de adyacencias**

En la **clase Arista** se encuentran dos variables y un método. La variable **destino** que toma como valor al nodo que se dirige la arista, la otra variable llamada **costo** es la encargada de indicar el costo de la arista.

****

En la **clase Vertice** se encuentran las siguientes variables:

* nombre: contiene el nombre que se le asignará al vértice
* adyacente: las aristas que salen del vértice y se conectaran a otros
* filaNC: Contiene una lista con los vértices que no pertenecen a un ciclo
* dist: toma el valor de la distancia hacia el vértice que se analizará
* prev: vértice previo
* visitado: valor booleano que se utilizará durante dijkstra

**Texto

Descripción generada automáticamente**

**Paquetes y librerías importadas**

**Texto

Descripción generada automáticamente**

En la **clase Grafo** encontraremos distintas variables y métodos importantes que serán explicadas durante las siguientes paginas

**VerticeMap:** Es un HashMap que contendrá los distintos vértices que son parte del grafo

**addVertice(String VerticeNombre,String VerticeNombre2):**  Este método es el encargado de agregar los distintos vértices pertenecientes al grafo, primero busca que ese vértice no este ingresado a través de los valor parámetro y si este no está, se insertara.

Texto

Descripción generada automáticamente

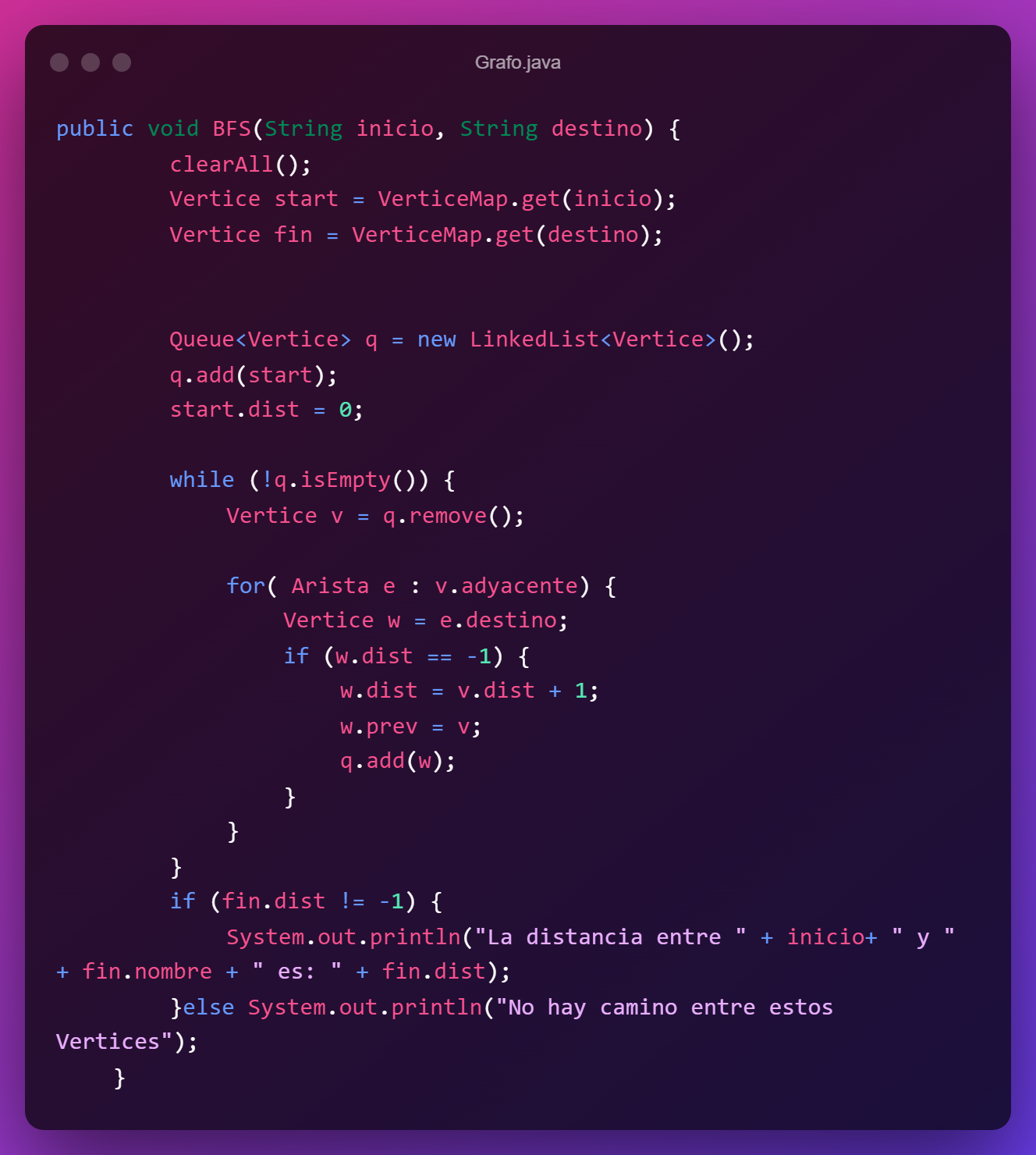
El método **seeGraph():** Es el encargado de mostrar el grafo por pantalla y la dirección de apuntado en cada vértice.

****

Texto

Descripción generada automáticamente

**Implementación del camino mínimo sin pesos**



**Texto

Descripción generada automáticamente**

**Implementación del detector de grafos acíclicos**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Implementación árbol TDA**

**Pantalla de computadora con letras

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Pantalla de computadora con letras

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Implementación recorrido de árbol recursivo**

**Texto

Descripción generada automáticamente**

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

**Implementación de Dijkstra recursiva**

