TALLER DE ALGORITMOS Y ESTRUCTURAS DE DATOS II

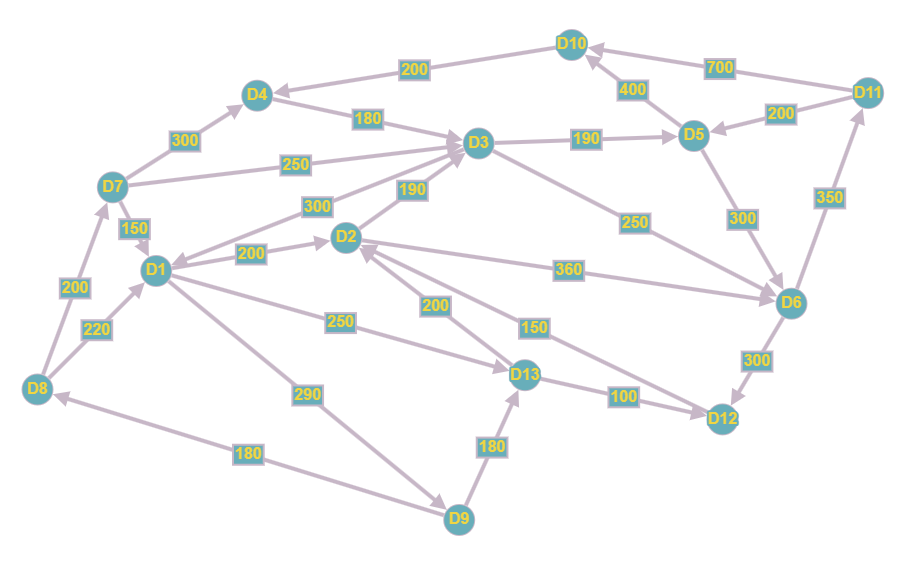
# Trabajo Practico Nº1

Situación problemática Si bien es relativamente fácil para los seres humanos ver mapas de caminos y rutas, y entender las relaciones entre diferentes lugares, una computadora no tiene tal conocimiento. Sin embargo, también podemos pensar en un mapa de rutas y trabajarlo como un grafo. Para cumplir los objetivos del trabajo práctico se ha propuesto la siguiente situación problemática: En el aeropuerto internacional de Denver, que 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 ello se ha propuesto hacer una serie de pruebas en la aerolínea Good Fly, se establece un cronograma de vuelos con sus distintos destinos, buscando implementar un sistema que permita determinar el camino más corto dentro de las múltiples opciones para llegar a un destino determinado. A su vez también es importante poder agregar nuevos destinos y en ese caso la aerolínea debe tener la posibilidad de poder expandirse. Se presenta 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, d3,250),(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)}

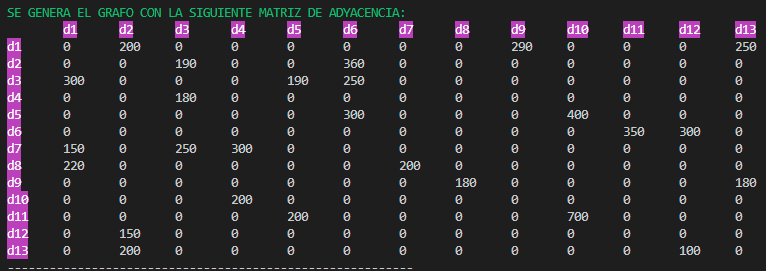
| Consignas de la actividad |  |
| --- | --- |
| En relación con los objetivos de la unidad y de acuerdo con el esquema asignado se piden las siguientes consignas:   * Elaborar un diagrama donde se evidencien los vértices, las aristas y relaciones del esquema de viajes anteriormente planteado. * Determinar la matriz de adyacencias del grafo. * Implementar en el lenguaje de programación Java el tipo abstracto de datos grafo, representando el modelo obtenido en la matriz de adyacencias. * Implementar en el lenguaje de programación Java el algoritmo del camino mínimo sin pesos dentro del esquema y mostrar los resultados por pantalla. * Implementar un algoritmo en el lenguaje Java que determine los grafos acíclicos que tiene el diagrama y mostrar los resultados por pantalla. * Elaborar un algoritmo en el lenguaje Java donde implemente las operaciones del TDA árbol utilizando el modelo de grafo como referencia. * Elaborar una función en lenguaje de programación Java, que permita recorrer el árbol obtenido de manera recursiva. * Implementar un algoritmo en lenguaje Java que, dado un vértice A y un vértice B, determine el camino más corto (algoritmo de Dijkstra) entre ambos y mostrar los resultados por pantalla.  **Grafo** | |



#### Matriz de Adyacencia

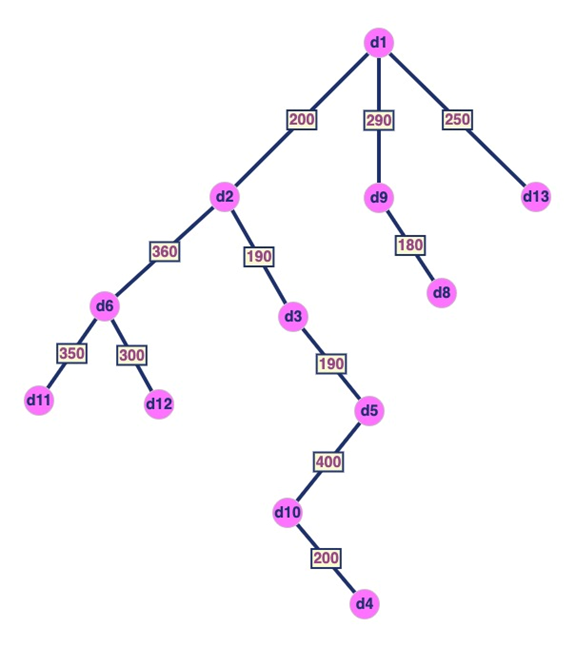
|  | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** | **D10** | **D11** | **D12** | **D13** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **D1** | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 290 | 0 | 0 | 0 | 250 |
| **D2** | 0 | 0 | 190 | 0 | 0 | 360 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D3** | 300 | 0 | 0 | 0 | 190 | 250 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D4** | 0 | 0 | 180 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D5** | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 400 | 0 | 0 | 0 |
| **D6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 350 | 300 | 0 |
| **D7** | 150 | 0 | 250 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D8** | 220 | 0 | 0 | 0 | 0 | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 180 |
| **D9** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 180 | 0 | 0 | 0 | 0 | 0 |
| **D10** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D11** | 0 | 0 | 0 | 200 | 200 | 0 | 0 | 0 | 0 | 700 | 0 | 0 | 0 |
| **D12** | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D13** | 0 | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 |

El codigo realiza la misma matriz de adyacencia:



* **Grafo.java**
  + **Para realizar la Matriz de Adyacencia:**
    - Se Crea la clase Grafo
    - Se inicia declarando N vertices
    - Dicha matriz se inicializa con 0 datos

#### TDA ARBOL



* **ArbolTDA.java**
  + **Para realizar el calculo del Arbol TDA:**
    - Primero se crea el Nodo Raiz
    - Recorremos todos los nodos hijos