**Codigo**

|  |
| --- |
| **Clase Grafo**  import java.util.ArrayList;  import java.util.Arrays;  import java.util.Collections;  import java.util.Comparator;  import java.util.List;  import java.util.PriorityQueue;  /\*\*  \*  \* @author gisse  \*/  public class Grafo {  //matriz de adyacencia  private final int cantidadVertices;//creo una lista  private final List<List<Arista>> listaAdyacencia;  //metodo para inicializar variables  public Grafo(int cantidadVertices) {  this.cantidadVertices = cantidadVertices;  this.listaAdyacencia = new ArrayList<>(cantidadVertices);  for (int i = 0; i < cantidadVertices; i++) {  this.listaAdyacencia.add(new ArrayList<>());  }  }  //constructor para definir e inicializar las variables  public void agregarArista(char origen, char destino, int peso) {  this.listaAdyacencia.get(origen - 'A').add(new Arista(destino - 'A', peso));//definir letras  }  //metodo para crear el dijkstra  public List<Character> dijkstra(char origen, char destino) {  PriorityQueue<Nodo> colaPrioridad = new PriorityQueue<>(cantidadVertices, Comparator.comparingInt(a -> a.peso));  int[] distancias = new int[cantidadVertices];  int[] previos = new int[cantidadVertices];//arreglo para cantidad de vertices  Arrays.fill(distancias, Integer.MAX\_VALUE);  Arrays.fill(previos, -1);  colaPrioridad.add(new Nodo(origen - 'A', 0));  distancias[origen - 'A'] = 0;//define origen  while (!colaPrioridad.isEmpty()) {  Nodo nodoActual = colaPrioridad.poll();  int u = nodoActual.vertice;  //para ver cantidad de aritas e ingresar dartos  for (Arista arista : listaAdyacencia.get(u)) {  int v = arista.destino;  int peso = arista.peso;  if (distancias[u] + peso < distancias[v]) {  distancias[v] = distancias[u] + peso;  previos[v] = u;  colaPrioridad.add(new Nodo(v, distancias[v]));  }  }  }  //lista de modo carcter busca la ruta y el destino  List<Character> ruta = new ArrayList<>();  for (int i = destino - 'A'; i != -1; i = previos[i]) {  ruta.add((char) (i + 'A'));  }  Collections.reverse(ruta);  return ruta; }  //metodo para vertice y peso  private static class Nodo {  private final int vertice;  private final int peso;  //nodo para el vertice y el peso  public Nodo(int vertice, int peso) {  this.vertice = vertice;  this.peso = peso;  }}  //metodo para realizar la suma y saber el peso del grafo y la ruta  public int sumaPesosAristas(List<Character> ruta) {  int suma = 0;  for (int i = 0; i < ruta.size() - 1; i++) {  char origen = ruta.get(i);  char destino = ruta.get(i + 1);  int peso = obtenerPesoArista(origen, destino);  suma += peso; }  return suma; }  //metodo para mostrar la matriz de adyacencia  public void mostrarMatrizAdyacencia() {  System.out.println("Matriz de Adyacencia es ");  for (int i = 0; i < cantidadVertices; i++) {  for (int j = 0; j < cantidadVertices; j++) {  int peso = obtenerPesoArista((char) ('A' + i), (char) ('A' + j));  System.out.print((peso == Integer.MAX\_VALUE ? "0" : peso) + " ");  }  System.out.println();  } }  //metodo para obtener el peso de las aristas  private int obtenerPesoArista(char origen, char destino) {  for (Arista arista : listaAdyacencia.get(origen - 'A')) {  if (arista.destino == destino - 'A') {  return arista.peso;  }  }  return Integer.MAX\_VALUE;  }  //metodo para inicializar las aristas  private static class Arista {  private final int destino;  private final int peso;  //constructor del metodo arustas  public Arista(int destino, int peso) {  this.destino = destino;  this.peso = peso;  }  }} |
| Clase main  public class GrafoBusqueda {  public static void main(String[] args) {  // TODO code application logic here  Grafo grafo = new Grafo(9); // Por ejemplo, un grafo con 6 vértices  // Agregar aristas al grafo con origen destino y peso  grafo.agregarArista('A', 'B', 10);  grafo.agregarArista('A', 'C', 15);  grafo.agregarArista('B', 'D', 12);  grafo.agregarArista('B', 'F', 15);  grafo.agregarArista('C', 'E', 10);  grafo.agregarArista('C', 'G', 8);  grafo.agregarArista('D', 'E', 5);  grafo.agregarArista('D', 'H', 7);  grafo.agregarArista('E', 'F', 7);  grafo.agregarArista('E', 'I', 8);  grafo.agregarArista('F', 'I', 6);  grafo.agregarArista('G', 'H', 12);  grafo.agregarArista('H', 'I', 10);  //muestra el recorr¡do del grafo  System.out.println("EL RECORRIDO ES " + grafo.dijkstra('A', 'I'));  //mustra el peso del grafo  System.out.println("EL PESO ES " + grafo.sumaPesosAristas(grafo.dijkstra('A', 'I')));  //muestra la matriz de adyacencia del garfo  grafo.mostrarMatrizAdyacencia();  }  } |







