REPORTE DE PRÁCTICA

| Nombre Completo | Machado Sanchez Javier |
| --- | --- |
| Número de la Unidad a Evaluar | 4 |
| Tema a Evaluar | Implementando Grafos |
| Fecha de Entrega | 28 de Octubre del 2023 |

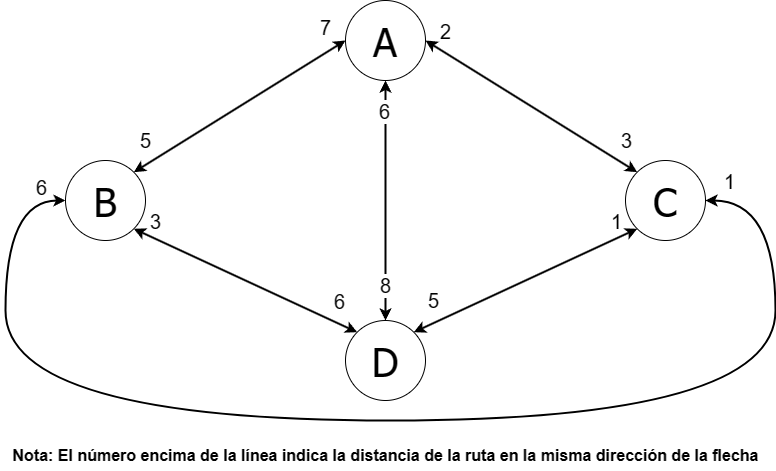
**Programa Fuente**

| using System;  using System.Collections.Generic;  using System.Linq;  namespace P31\_Implementando\_Grafos  {  internal class Program  {  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  public static int Contador\_Rutas = 0;    //Declaración de clase nodo  public class Node  {  public string Ciudad { get; set; }  public List<Camino> Caminos { get; set; }  //Constructor  public Node()  {  Caminos = new List<Camino>();  }  }  //Fin de la clase nodo  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio struct Recorrido  public struct Camino  {  public Node Node { get; set; }  public int Distancia { get; set; }  //Fin struct Recorrido  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Declaración de clase camino  public class Ruta  {  public List<Node> Nodes { get; set; }  public int TotalDistancia { get; set; }  //Constructor  public Ruta()  {  Nodes = new List<Node>();  TotalDistancia = 0;  }  }  //Fin clase camino  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio Main  static void Main(string[] args)  {  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio Variables locales  char opcion;  //Inicio Declaración de nodos  Node NodoA = new Node() { Ciudad = "California" };  Node NodoB = new Node() { Ciudad = "Kansas" };  Node NodoC = new Node() { Ciudad = "Philadelphia" };  Node NodoD = new Node() { Ciudad = "Miami" };  //Fin Declaración de nodos  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  List<Node> grafo = new List<Node>() //Declaración de lista que contiene los nodos o ciudades  {  NodoA, NodoB, NodoC, NodoD  };  //Fin variables locales  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Console.Title = "P30 Implementando Grafos";  do  {  Console.Write("\n\t\t.: MENÚ DE OPCIONES :." +  "\n\n\t| Estas son las ciudades a recorrer |" +  "\n\n\t[ I ] - Indicar Rutas" +  "\n\t[ A ] - California" +  "\n\t[ B ] - Kansas" +  "\n\t[ C ] - Philadelphia" +  "\n\t[ D ] - Miami" +  "\n\t[ S ] - Salir del Programa" +  "\n\n\tIngrese la letra de la ciudad origen deseada: ");  if (char.TryParse(Console.ReadLine(), out opcion) == true)  {  Console.Write($"\n\n\tLa opción \"{opcion}\" se ha capturado correctamente" +  "\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  switch (char.ToUpper(opcion))  {/// comentario azul  // comentario verde  case 'I':  {  Console.WriteLine("\n\t\t.: CAPTURANDO RUTAS :.");  //Ruta A a B  Console.Write($"\n\tCaptura la ruta de {NodoA.Ciudad} a {NodoB.Ciudad}: ");  NodoA.Caminos.Add(new Camino() { Node = NodoB, Distancia = int.Parse(Console.ReadLine()) });  //Ruta A a C  Console.Write($"\n\tCaptura la ruta de {NodoA.Ciudad} a {NodoC.Ciudad}: ");  NodoA.Caminos.Add(new Camino() { Node = NodoC, Distancia = int.Parse(Console.ReadLine()) });  //Ruta A a D  Console.Write($"\n\tCaptura la ruta de {NodoA.Ciudad} a {NodoD.Ciudad}: ");  NodoA.Caminos.Add(new Camino() { Node = NodoD, Distancia = int.Parse(Console.ReadLine()) });  //Ruta B a D  Console.Write($"\n\tCaptura la ruta de {NodoB.Ciudad} a {NodoD.Ciudad}: ");  NodoB.Caminos.Add(new Camino() { Node = NodoD, Distancia = int.Parse(Console.ReadLine()) });  //Ruta B a C  Console.Write($"\n\tCaptura la ruta de {NodoB.Ciudad} a {NodoC.Ciudad}: ");  NodoB.Caminos.Add(new Camino() { Node = NodoC, Distancia = int.Parse(Console.ReadLine()) });  //Ruta B a A  Console.Write($"\n\tCaptura la ruta de {NodoB.Ciudad} a {NodoA.Ciudad}: ");  NodoB.Caminos.Add(new Camino() { Node = NodoA, Distancia = int.Parse(Console.ReadLine()) });  //Ruta C a D  Console.Write($"\n\tCaptura la ruta de {NodoC.Ciudad} a {NodoD.Ciudad}: ");  NodoC.Caminos.Add(new Camino() { Node = NodoD, Distancia = int.Parse(Console.ReadLine()) });  //Ruta C a A  Console.Write($"\n\tCaptura la ruta de {NodoC.Ciudad} a {NodoA.Ciudad}: ");  NodoC.Caminos.Add(new Camino() { Node = NodoA, Distancia = int.Parse(Console.ReadLine()) });  //Ruta C a B  Console.Write($"\n\tCaptura la ruta de {NodoC.Ciudad} a {NodoB.Ciudad}: ");  NodoC.Caminos.Add(new Camino() { Node = NodoB, Distancia = int.Parse(Console.ReadLine()) });  //Ruta D a B  Console.Write($"\n\tCaptura la ruta de {NodoD.Ciudad} a {NodoB.Ciudad}: ");  NodoD.Caminos.Add(new Camino() { Node = NodoB, Distancia = int.Parse(Console.ReadLine()) });  //Ruta D a A  Console.Write($"\n\tCaptura la ruta de {NodoD.Ciudad} a {NodoA.Ciudad}: ");  NodoD.Caminos.Add(new Camino() { Node = NodoA, Distancia = int.Parse(Console.ReadLine()) });  //Ruta D a C  Console.Write($"\n\tCaptura la ruta de {NodoD.Ciudad} a {NodoC.Ciudad}: ");  NodoD.Caminos.Add(new Camino() { Node = NodoC, Distancia = int.Parse(Console.ReadLine()) });  Console.Write("\n\n\tRutas capturadas correctamente" +  "\n\tPresione la tecla <INTRO> para regresar al menú...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case 'A':  {  Console.WriteLine("\n\t\t\t\t.: MOSTRANDO RUTAS :." +  "\n\n\t| Estas son las rutas para recorrer todas las ciudades desde California |" +  "\n\t| Las rutas más óptimas o cortas se muestran primero |\n");  var algoritmo = new Algoritmo(grafo, 5, NodoA);  algoritmo.Correr();  Console.WriteLine($"{algoritmo.Tomatodaslasrutas}");  Console.Write("\n\tPresione la tecla <INTRO> para regresar al menú...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case 'B':  {  Console.WriteLine("\n\t\t\t\t.: MOSTRANDO RUTAS :." +  "\n\n\t| Estas son las rutas para recorrer todas las ciudades desde Kansas |" +  "\n\t| Las rutas más óptimas o cortas se muestran primero |\n");  var algoritmo = new Algoritmo(grafo, 5, NodoB);  algoritmo.Correr();  Console.WriteLine($"{algoritmo.Tomatodaslasrutas}");  Console.Write("\n\tPresione la tecla <INTRO> para regresar al menú...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case 'C':  {  Console.WriteLine("\n\t\t\t\t.: MOSTRANDO RUTAS :." +  "\n\n\t| Estas son las rutas para recorrer todas las ciudades desde Philadelphia |" +  "\n\t| Las rutas más óptimas o cortas se muestran primero |\n");  var algoritmo = new Algoritmo(grafo, 5, NodoC);  algoritmo.Correr();  Console.WriteLine($"{algoritmo.Tomatodaslasrutas}");  Console.Write("\n\tPresione la tecla <INTRO> para regresar al menú...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case 'D':  {  Console.WriteLine("\n\t\t\t\t.: MOSTRANDO RUTAS :." +  "\n\n\t| Estas son las rutas para recorrer todas las ciudades desde Miami |" +  "\n\t| Las rutas más óptimas o cortas se muestran primero |\n");  var algoritmo = new Algoritmo(grafo, 5, NodoD);  algoritmo.Correr();  Console.WriteLine($"{algoritmo.Tomatodaslasrutas}");  Console.Write("\n\tPresione la tecla <INTRO> para regresar al menú...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  case 'S':  {  Console.Write("\n\t\t:. SALIDA :." +  "\n\n\tGracias por utilizar nuestro programa" +  "\n\n\tPresione la tecla <Esc> para salir...");  while (Console.ReadKey().Key != ConsoleKey.Escape) ;  }  break;  default:  {  Console.Write("\n\t\t.: OPCIÓN INVALIDA :." +  "\n\n\tPor favor ingrese una opción existente" +  "\n\n\tPresione la tecla <INTRO> para continuar...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  break;  }  }  else  {  Console.Write("\n\n\tLa opción ingresada no se ha podido capturar" +  "\n\tPresione la tecla <INTRO> para regresar al menú...");  while (Console.ReadKey().Key != ConsoleKey.Enter) ; Console.Clear();  }  } while (char.ToUpper(opcion) != 'S');  }  //Fin Main  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio clase Algoritmo  public class Algoritmo  {  //Campos privados  private List<Node> \_grafo { get; set; }  private int \_n;  private Node \_origen { get; set; }  private List<Ruta> \_solucion { get; set; }  //Métodos  public string Tomatodaslasrutas  {  get  {  Program.Contador\_Rutas = 0;  string resultado = "";  foreach (var ruta in \_solucion)  {  Program.Contador\_Rutas++;  resultado += $"\n\t{Program.Contador\_Rutas}.- ";  foreach (var node in ruta.Nodes)  {  resultado += node.Ciudad + ", ";  }  resultado += " " + ruta.TotalDistancia + "\n\t";  }  return resultado;  }  }  public void Correr()  {  \_solucion = new List<Ruta>();  for (int i = 0; i < \_n; i++)  {  \_solucion.Add(Generada());  }  \_solucion = \_solucion.OrderBy(d => d.TotalDistancia).ToList();  }  private Ruta Generada()  {  var sol = new Ruta();  sol.Nodes.Add(\_origen);  Node Current = \_origen;  for (int i = 0; i < \_grafo.Count - 1; i++)  {  Node next = null;  do  {  next = NextNode(Current);  } while (sol.Nodes.Contains(next));  sol.Nodes.Add(next);  sol.TotalDistancia+= Current.Caminos.Where(d => d.Node.Ciudad == next.Ciudad).First().Distancia;  Current = next;  }  sol.Nodes.Add(\_origen);  return sol;  }  private Node NextNode(Node Current)  {  int next\_node = new Random().Next(0, \_grafo.Count - 1);  return Current.Caminos[next\_node].Node;  }  //Fin Métodos  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Inicio Constructor  public Algoritmo(List<Node> grafo, int n, Node origen)  {  \_grafo = grafo;  \_n = n;  \_origen = origen;  }  //Fin Constructor  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  }  //Fin clase Algoritmo  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  }  }  } |
| --- |

**Tabla del grafo**

| **Letra** | **Origen** | **Destino** | **Letra** | **Distancia** |
| --- | --- | --- | --- | --- |
| A | California | Kansas | B | 5 |
| A | California | Philadelphia | C | 3 |
| A | California | Miami | D | 8 |
| B | Kansas | Miami | D | 6 |
| B | Kansas | Philadelphia | C | 1 |
| B | Kansas | California | A | 7 |
| C | Philadelphia | Miami | D | 5 |
| C | Philadelphia | California | A | 2 |
| C | Philadelphia | Kansas | B | 6 |
| D | Miami | Kansas | B | 3 |
| D | Miami | California | A | 6 |
| D | Miami | Philadelphia | C | 1 |

**Representación del grafo**



**Ventana de Entrada de datos**

|  |
| --- |

**Ventana de Resultados**

|  |
| --- |

**EXPLICA BREVEMENTE QUÉ TE PARECIÓ EL PROBLEMA, QUE SE TE DIFICULTO Y COMO LO SOLUCIONASTE.**

| No fue difícil agregar la modificación para insertar las rutas manualmente, sin embargo, para eso quería utilizar una matriz que guardara los valores ingresados por el usuario, así podría capturar por medio de un ciclo todos los valores. El problema es que el programa me estaba lanzando una excepción en la declaración del arreglo, incluso sí aún no había estado insertando nada en él. El IDE no marcaba ningún error de sintaxis, porque la declaración estaba bien hecha. Esa declaración del arreglo la copie y la pegué en otro proyecto y no había ningún problema, pero al implementar el código completo aparecía de nuevo la excepción. Por falta de tiempo decidí dejar de lado el ciclo y poner varios console.write para capturar las rutas.  También descubrí una opción de Visual Studio 2022 llamada “Live Share” donde puedes crear una sesión para que otros desarrolladores se conecten a tu proyecto de forma remota. Se me hizo interesante, ya que podía programar desde mi computadora y desde mi laptop en tiempo real, parecido a cuando hay varias personas en un docs de drive.  Aproveché esta opción para pedirle a un amigo que me ayudara a escribir los Console.Write para que nos tocaran de 6 y 6 y así terminar más rápido. |
| --- |