华北电力大学

**实 验 报 告**

|

|

**实验名称** 图

**课程名称**  数据结构综合实践

|

|

专业班级： 信息2202 学生姓名：徐梓乔

学 号： 220221100327 成 绩：

|  |
| --- |
| 一、实验内容  给定纽约市附近的一幅简单地图，图中的顶点为城市，无向边代表两个城市的连通关系，边上的权为两城市之间的距离。   1. 对该图进行深度优先和广度优先搜索，并输出搜索序列（图的搜索问题）。   （2）在分析这张图后可以发现，任一对城市都是连通的  第一个问题是：要用公路把所有城市连接起来，如何设计可使得工程的总造价最少（最小生成树问题）。  第二个问题是：要开车从一个城市到另外一个城市求其最短距离以及驱车路线（最短路径问题）。  二、实验方法与步骤  实验思路：  该问题是一图为基础进行了两个问题的解决，最小生成树问题和最短路径问题。  代码如下：  第一部分:图的搜索  private void SetGraph1()  {  string[] array = File.ReadAllLines(".\\data\\graph.txt");  List<string> list = new List<string>();  List<Edge2> list2 = new List<Edge2>();  for (int i = 1; i < array.Length; i++)  {  if (!string.IsNullOrEmpty(array[i]))  {  string[] array2 = array[i].Split(',');  Edge2 edge = new Edge2();  edge.Source = array2[0];  edge.Target = array2[1];  edge.Weight = double.Parse(array2[2]);  list.Add(edge.Source);  list.Add(edge.Target);  list2.Add(edge);  }  }  string[] array3 = list.Distinct().ToArray();  \_alg1 = new AdGraph(array3.Length);  for (int j = 0; j < \_alg1.VertexCount; j++)  {  \_alg1[j] = array3[j];  }  for (int k = 0; k < list2.Count; k++)  {  \_alg1.AddEdge(list2[k].Source, list2[k].Target, list2[k].Weight);  \_alg1.AddEdge(list2[k].Target, list2[k].Source, list2[k].Weight);  }  }  private void btnDFS\_Click(object sender, EventArgs e)  {  string startVertexName = cboxCity1.Text;  rtxtDFS.Clear();  rtxtDFS.Text = \_alg1.DfsTraversal(startVertexName);  }  private void btnBFS\_Click(object sender, EventArgs e)  {  string startNodeName = cboxCity1.Text;  rtxtBFS.Clear();  rtxtBFS.Text = \_alg1.BfsTraversal(startNodeName);  }  第二部分:最小生成树  private void SetGraph2()  {  \_alg2 = new AdGraph(9);  for (int i = 0; i < \_alg2.VertexCount; i++)  {  \_alg2[i] = string.Concat(dgvTopo.Rows[i].Cells[0].Value, ".", dgvTopo.Rows[i].Cells[1].Value);  }  \_alg2.AddEdge(\_alg2[0], \_alg2[6], 1.0);  \_alg2.AddEdge(\_alg2[0], \_alg2[2], 1.0);  \_alg2.AddEdge(\_alg2[1], \_alg2[2], 1.0);  \_alg2.AddEdge(\_alg2[1], \_alg2[3], 1.0);  \_alg2.AddEdge(\_alg2[1], \_alg2[8], 1.0);  \_alg2.AddEdge(\_alg2[3], \_alg2[5], 1.0);  \_alg2.AddEdge(\_alg2[4], \_alg2[5], 1.0);  \_alg2.AddEdge(\_alg2[5], \_alg2[8], 1.0);  \_alg2.AddEdge(\_alg2[6], \_alg2[7], 1.0);  \_alg2.AddEdge(\_alg2[6], \_alg2[8], 1.0);  }  private void btnSpan\_Click(object sender, EventArgs e)  {  rtxtSpanTree.Clear();  SpanTreeNode[] array;  if (rdoPrim.Checked)  {  string vName = cboxCity2.Text;  array = \_alg1.MiniSpanTree(vName);  }  else  {  array = \_alg1.MiniSpanTree();  }  double num = 0.0;  for (int i = 0; i < array.Length; i++)  {  string text = "(" + array[i].ParentName + "," + array[i].SelfName + ") Weight:" + array[i].Weight + "\n";  rtxtSpanTree.Text += text;  num += array[i].Weight;  }  lblPower.Text = "最小生成树的权值：" + num.ToString(CultureInfo.InvariantCulture);  }  第三部分:拓扑排序  private void SetDataGridView()  {  dgvTopo.ColumnCount = 3;  dgvTopo.Columns[0].HeaderText = "课程号";  dgvTopo.Columns[1].HeaderText = "课程名";  dgvTopo.Columns[2].HeaderText = "先修课程";  dgvTopo.RowCount = 9;  dgvTopo.Rows[0].Cells[0].Value = "v0";  dgvTopo.Rows[1].Cells[0].Value = "v1";  dgvTopo.Rows[2].Cells[0].Value = "v2";  dgvTopo.Rows[3].Cells[0].Value = "v3";  dgvTopo.Rows[4].Cells[0].Value = "v4";  dgvTopo.Rows[5].Cells[0].Value = "v5";  dgvTopo.Rows[6].Cells[0].Value = "v6";  dgvTopo.Rows[7].Cells[0].Value = "v7";  dgvTopo.Rows[8].Cells[0].Value = "v8";  dgvTopo.Rows[0].Cells[1].Value = "数学分析";  dgvTopo.Rows[1].Cells[1].Value = "高等代数";  dgvTopo.Rows[2].Cells[1].Value = "概率论与数理统计";  dgvTopo.Rows[3].Cells[1].Value = "离散数学";  dgvTopo.Rows[4].Cells[1].Value = "高级程序设计语言";  dgvTopo.Rows[5].Cells[1].Value = "数据结构与算法";  dgvTopo.Rows[6].Cells[1].Value = "常微分方程";  dgvTopo.Rows[7].Cells[1].Value = "数学物理方程";  dgvTopo.Rows[8].Cells[1].Value = "数值分析";  dgvTopo.Rows[0].Cells[2].Value = "无";  dgvTopo.Rows[1].Cells[2].Value = "无";  dgvTopo.Rows[2].Cells[2].Value = "v0,v1";  dgvTopo.Rows[3].Cells[2].Value = "v1";  dgvTopo.Rows[4].Cells[2].Value = "无";  dgvTopo.Rows[5].Cells[2].Value = "v3,v4";  dgvTopo.Rows[6].Cells[2].Value = "v0";  dgvTopo.Rows[7].Cells[2].Value = "v6";  dgvTopo.Rows[8].Cells[2].Value = "v1,v5,v6";  }  private void btnTopo\_Click(object sender, EventArgs e)  {  rtxtTopo.Text = \_alg2.TopoSort();  }  public string TopoSort()  {  string text = string.Empty;  int[] inDegressList = GetInDegressList();  LinkQueue<int> val = new LinkQueue<int>();  for (int i = 0; i < VertexCount; i++)  {  if (inDegressList[i] == 0)  {  val.EnQueue(i);  }  }  if (val.get\_Length() == VertexCount)  {  throw new Exception("此有向图无有向边.");  }  while (!val.IsEmpty())  {  int queueFront = val.get\_QueueFront();  val.DeQueue();  text = text + \_vertexList[queueFront].VertexName + "\n";  for (EdgeNode edgeNode = \_vertexList[queueFront].FirstNode; edgeNode != null; edgeNode = edgeNode.Next)  {  inDegressList[edgeNode.Index]--;  if (inDegressList[edgeNode.Index] == 0)  {  val.EnQueue(edgeNode.Index);  }  }  }  int j;  for (j = 0; j < VertexCount && inDegressList[j] == 0; j++)  {  }  return (j == VertexCount) ? text : "该AOV网有环.";  }  第四部分:最短路径  private void btnShortPath\_Click(object sender, EventArgs e)  {  string vName = cboxCity3.Text;  rtxtShortPath.Clear();  rtxtShortPath.Text = \_alg1.ShortestPath(vName);  }  public string ShortestPath(string vName)  {  int num = GetIndex(vName);  if (num == -1)  {  return string.Empty;  }  string text = string.Empty;  double[] array = new double[VertexCount];  string[] array2 = new string[VertexCount];  for (int i = 0; i < VertexCount; i++)  {  \_vertexList[i].Visited = false;  array[i] = double.MaxValue;  array2[i] = \_vertexList[num].VertexName;  }  array[num] = 0.0;  \_vertexList[num].Visited = true;  for (int j = 0; j < VertexCount - 1; j++)  {  for (EdgeNode edgeNode = \_vertexList[num].FirstNode; edgeNode != null; edgeNode = edgeNode.Next)  {  if (!\_vertexList[edgeNode.Index].Visited && array[num] + edgeNode.Weight < array[edgeNode.Index])  {  array[edgeNode.Index] = array[num] + edgeNode.Weight;  array2[edgeNode.Index] = array2[num] + " ->" + \_vertexList[edgeNode.Index].VertexName;  }  }  double num2 = double.MaxValue;  for (int k = 0; k < VertexCount; k++)  {  if (!\_vertexList[k].Visited && array[k] < num2)  {  num2 = array[k];  num = k;  }  }  \_vertexList[num].Visited = true;  }  for (int l = 0; l < VertexCount; l++)  {  text = text + array2[l] + ":" + array[l] + "\n";  }  return text;  }  三、实验结果          四、结论与总结  结论:图的遍历可以用广度优先搜索和深度优先搜索进行.从原图能够使用两种算法来得到最小生成树,一是Prim算法,另一个是Kruskar算法,能够利用拓扑排序进行课程规划.利用迪杰特斯拉算法解决最短路径问题.  总结:利用图的性质以及操作和一些算法能够解决有关城市或区域距离问题.拓扑排序就是一个构造AOV网拓扑的序列的过程.而拓扑序列就是把AOV网中的所有顶点排成一个线性序列,使每个活动的所有前驱活动都排在该活动的前边. |
|  |