

**软件基础实习设计报告**

**题目**  高校专用通信网络建设

**班级**  软件一班

**姓名**  何雨彤

**学号**  420109070318

**指导老师** 马立肖、张有华、张敬敏

王楠、李霞、李宁

**完成日期** 2022 年 1 月 5 日

信息工程学院

# 

一、需求分析

1.问题描述：中国移动公司正在积极推广4G通信应用，计划在河北省高校之间建立一个专用通信网络，为其规划一个投资最省的通信线路架设方案。本程序的主要功能包括构造通信网络、编辑通信网络、生成最佳方案、输出通信网络、保存通信网络。编辑通信网络的具体功能包括添加、删除高校，添加、删除高校间的线路，修改高校名，修改线路成本。

2.输入的形式和输入值的范围：在添加、删除高校时，输入的形式为一个以"回车符"为结束标志的字符串，不允许出现非法字符。在添加、删除高校间的线路，修改高校名时，输入的形式为两个以"回车符"为结束标志的字符串，不允许出现非法字符。在修改线路成本时，输入的形式为两个以"回车符"为结束标志的字符串以及一个整形数。

3.输出的形式：以图形和文本两种形式输出生成树中各条边以及他们的权值。

4.程序所能达到的功能包括：(1)构造通信网络；(2)编辑通信网络（添加、删除高校，添加、删除高校间的线路，修改高校名，修改线路成本）；(3)生成最佳方案；(4)输出通信网络；(5)保存通信网络；(6)结束。

5.测试数据：（1）输入1构造通信网络；（2）输入2编辑通信网络：输入1添加高校，输入石家庄铁道大学；输入2删除高校，输入河北大学；输入3添加高校间的线路，输入石家庄铁道大学、河北地质大学、15.0，以回车分隔；输入4删除高校间的线路，输入河北地质大学、河北大学，以回车分隔；输入5修改高校名，输入河北大学、石家庄铁道大学，以回车分隔；输入6修改线路成本，输入河北科技大学、河北地质大学、15.0，以回车分隔；结果应得到正确的通信网络；（3）输入3生成最佳方案；结果应得到正确的最佳方案；（4）输入4输出通信网络；（5）输入5保存通信网络；（6）输入0结束。

二、概要设计

为实现上述程序功能，应以无向图表示通信网络。为此，需要的抽象数据类型是图。

1. 图的抽象数据类型如下：

ADT Graph{

数据对象：V是具有相同特性的数据元素的集合，称为顶点集

数据关系：

R={VR}

VR={<v, w>|v, w∈V且P(v, w)<v, w>表示从v到w的弧，谓词P(v, w)定义了弧<v, w>的意义或信息}

基本操作：

CreateGraph (&G,V,VR)

初始条件:V是图的顶点集,VR是图中弧的集合。

操作结果:按V和 VR的定义构造图G。

DestroyGraph(&G)

初始条件:图G存在。

操作结果:销毁图G。

LocateVex(G,u)

初始条件:图G存在，u和G中顶点有相同特征。

操作结果:若G中存在顶点u,则返回该顶点在图中的位置;否则返回其他信息。

GetVex(G, v)

初始条件:图G存在,v是G中某个顶点。

操作结果:返回v的值。

PutVex(&G,v, value)

初始条件:图G存在,V是G中某个顶点。

操作结果:对v赋值value。

FirstAdjVex(G, v)

初始条件:图G存在，v是G中某个顶点。

操作结果:返回 v的第一个邻接顶点。若v在G中没有邻接顶点，则返回“空”。

NextAdjVex(G,v,w)

初始条件:图G存在,v是G中某个顶点,w是v的邻接顶点。

操作结果:返回v的（相对于w的)下一个邻接顶点。若w是v的最后一个邻接点，则返回“空”。

InsertVex(&G,v)

初始条件:图G存在,v和图中顶点有相同特征。

操作结果:在图G中增添新顶点v。

Deletevex(&G,v)

初始条件:图G存在,v是G中某个顶点。

操作结果:删除G中顶点v及其相关的弧。

InsertArc(&G, v,w)

初始条件:图G存在,v和w是G中两个顶点。

操作结果:在G中增添弧<v,w>,若G是无向图，则还增添对称弧<w,v>。

DeleteArc(&G, v,w)

初始条件:图G存在，v和w是G中两个顶点。

操作结果:在G中删除弧<v,w>，若G是无向图，则还删除对称弧<w,v>。

DFSTraverse(G)

初始条件:图G存在。

操作结果:对图进行深度优先遍历，在遍历过程中对每个顶点访问一次。

BFSTraverse(G)

初始条件:图G存在。

操作结果:对图讲行广度优先遍历，在遍历过程中对每个顶点访问一次。

}ADT Graph

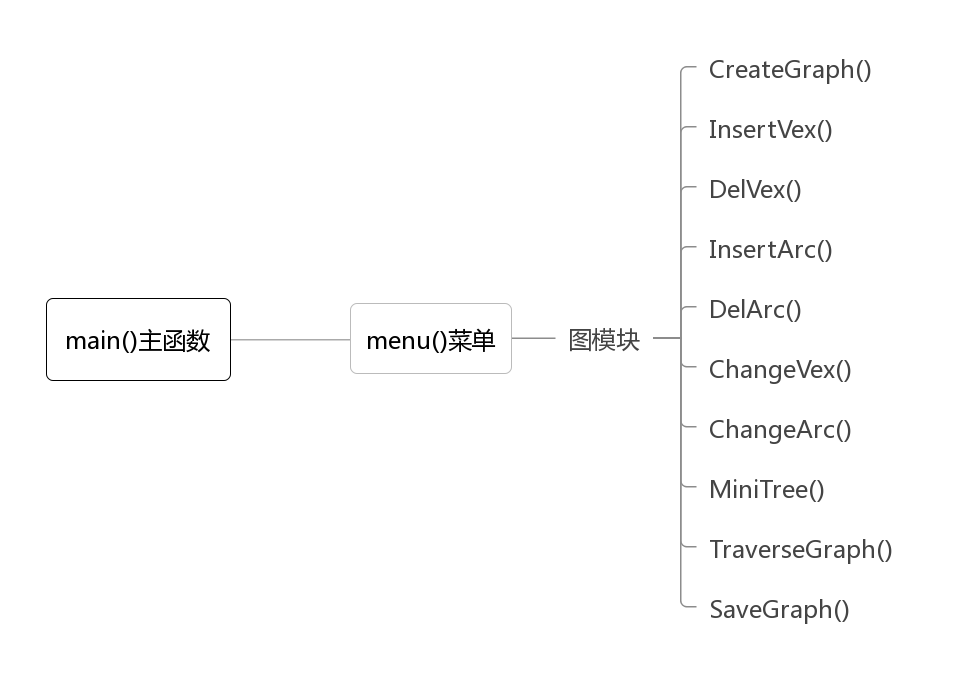
2. 本程序包含三个模块

1) 主程序模块：对菜单模块进行调用

2) 菜单模块：接收用户输入并调用相关函数；

3) 图模块：实现对通信网络相关的具体操作；

各模块之间的调用关系如下：

图 2-1 模块调用关系图

1. 详细设计
2. 数据类型定义

## typedef struct Edge

## {

## int from;地点一

## int to;//地点二

## int weight;//权值

## }Edge;//边

## typedef struct ALGraph

## {

## int vertex;//顶点数量

## int edge;//边的数量

## Edge \*p;//边集数组

## int \*m;//顶点数组

## }ALGraph;//图

## char schoolName[MAX\_PEAK\_NUM][MAX\_STR\_LEN];//学校名称

## int AdjMatrix[MAX\_PEAK\_NUM][MAX\_PEAK\_NUM];//邻接矩阵

2. 基本操作

## InitAdj()

## 操作结果：初始化邻接矩阵。

## DeleteEdge(ALGraph \*G,int i)

## 初始条件：图G存在

## 操作结果：删除第i条边。

## DeleteRoad(ALGraph \*G);

## 初始条件：图G存在。

## 操作结果：删除指定道路。

## DeleteVertex(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：删除指定学校

## EditEdge(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：修改道路权值

## EditName(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：修改学校名称

## AddEdge(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：添加道路

## AddVertex(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：添加学校

## PrintEdgeSet(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：打印邻接矩阵。

## PrintSchool(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：打印学校名称及序号

## CreatALGraph(ALGraph \*G)

## 操作结果：初始化学校及道路

## MinTree\_Kruskal(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：用克鲁斯卡尔算法求最小生成树，并输出最佳线路方案。

## LoadFile(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：读取文件。

## SaveFile(ALGraph \*G)

## 初始条件：图G存在。

## 操作结果：保存文件。

3. 详细设计

## void InitAdj()

## {//初始化邻接矩阵

## for(int i=0;i<MAX\_PEAK\_NUM;i++)

## {

## for(int j=0;j<MAX\_PEAK\_NUM;j++)

## {

## if(i!=j)

## AdjMatrix[i][j] = MAX\_EDGE\_LEN;

## else

## AdjMatrix[i][j] = 0;

## }

## }

## }

## void DeleteEdge(ALGraph \*G,int i)

## {//删除第i条边

## for(;i<G->edge-1;i++)

## G->p[i] = G->p[i+1];

## G->edge--;

## }

## void DeleteRoad(ALGraph \*G)

## {//删除指定道路

## int from,to,i;

## if(G->edge<1)

## {

## printf("请先添加道路！");

## return;

## }

## printf("请输入想要删除的道路信息(两个学校序号)：");

## scanf("%d%d",&from,&to);

## if(from>G->vertex||from<0||to>G->vertex||to<0||from==to)

## {

## printf("输入了错误的序号！\n");

## return;

## }

## i = SameEdge(G,from,to);

## if(i>-1)

## {

## char flag;

## printf("\n道路信息：%s————%s\t%d\n确定要删除这条道路吗？(Y确认/任意键否)\n", schoolName[G->p[i].from], schoolName[G->p[i].to], G->p[i].weight);

## getchar();

## flag = getchar();

## if(flag!='Y'&&flag!='y')

## {

## printf("删除终止！\n");

## return;

## }

## DeleteEdge(G,i);

## AdjMatrix[from][to] = MAX\_EDGE\_LEN;

## AdjMatrix[to][from] = MAX\_EDGE\_LEN;

## printf("删除成功！\n");

## return;

## }

## printf("不存在该条道路！\n");

## }

## void DeleteVertex(ALGraph \*G)

## {//删除指定学校

## if(G->vertex<1)

## {

## printf("请先初始化学校及道路信息！\n");

## return;

## }

## int order,i,j=0,k,l;

## bool flag = false;

## char s;

## printf("已输入的学校有：\n");

## PrintSchool(G);

## printf("请输入想要删除的学校序号：");

## scanf("%d",&order);

## if(order>=0&&order<G->vertex)

## {

## int edgearr[G->edge];

## for(i=0;i<G->edge;i++)

## {

## if(G->p[i].from==order||G->p[i].to==order)

## {

## if(!flag)

## printf("连接 %s 的道路有：\n",schoolName[order]);

## printf("\t%s——%s\t%d\n", schoolName[G->p[i].from], schoolName[G->p[i].to], G->p[i].weight);

## edgearr[j] = i;

## j++;

## flag = true;

## }

## }

## if(!flag)

## printf("没有与 %s 连接的道路。\n",schoolName[order]);

## printf("确认要删除吗？(Y.确认 / 任意键.否)\n");

## getchar();

## s = getchar();

## if(s!='Y'&&s!='y')

## {

## printf("删除终止！\n");

## return;

## }

## for(k=order;k<G->vertex-1;k++)//删除名称

## strcpy(schoolName[k],schoolName[k+1]);

## for(k=0;k<G->vertex-1;k++)//删除矩阵中的顶点

## for(l=0;l<G->vertex-1;l++)

## {

## if(k<order&&l>=order)

## AdjMatrix[k][l] = AdjMatrix[k][l+1];

## else if(k>=order&&l<order)

## AdjMatrix[k][l] = AdjMatrix[k+1][l];

## else if(k>=order&&l>=order)

## AdjMatrix[k][l] = AdjMatrix[k+1][l+1];

## }

## if(flag)//删除所连接的边

## for(k=0;k<j;k++)

## DeleteEdge(G,edgearr[k]-k);

## for(k=0;k<G->edge;k++)//修正边所连顶点下标

## {

## if(G->p[k].from>=order)

## G->p[k].from--;

## if(G->p[k].to>=order)

## G->p[k].to--;

## }

## G->vertex -= 1;

## printf("删除完成！\n");

## }

## else

## printf("该学校不存在！\n");

## }

## int SameEdge(ALGraph \*G,int from,int to)

## {//根据顶点查找边的下标

## for(int i=0;i<G->edge;i++)

## if((G->p[i].from==from&&G->p[i].to==to) || (G->p[i].from==to && G->p[i].to == from))

## return i;

## return -1;

## }

## void EditEdge(ALGraph \*G)

## {//修改道路权值

## int from,to,weight,i;

## if(G->edge<1)

## {

## printf("请先添加道路！\n");

## return;

## }

## printf("请输入想要更改的道路信息(两个学校序号和道路权值)：");

## scanf("%d%d%d",&from,&to,&weight);

## if(from>G->vertex||from<0||to>G->vertex||to<0||from==to)

## {

## printf("输入了错误的序号！");

## return;

## }

## i = SameEdge(G,from,to);

## if(i>-1)

## {

## G->p[i].weight = weight;

## AdjMatrix[from][to] = weight;

## AdjMatrix[to][from] = weight;

## printf("更改成功！\n");

## return;

## }

## printf("不存在该条道路！\n");

## }

## void EditName(ALGraph \*G)

## {//修改学校名称

## int order,k;

## char name[MAX\_STR\_LEN];

## bool same;

## if(G->vertex<1)

## {

## printf("请先添加学校！\n");

## return;

## }

## printf("可更改的学校有：\n");

## PrintSchool(G);

## printf("请输入想要更改的学校序号：");

## scanf("%d",&order);

## if(order>G->vertex||order<0)

## {

## printf("该序号不存在！");

## return;

## }

## getchar();

## do{

## same = false;

## printf("将 %s 重命名为：",schoolName[order]);

## scanf("%s",name);

## for(k=0;k<G->vertex;k++)

## {

## if(!strcmp(schoolName[order],name))

## {

## printf("\n该学校已存在！\n");

## same = true;

## break;

## }

## }

## }while(same);

## strcpy(schoolName[order],name);

## printf("命名完成！\n");

## }

## void AddEdge(ALGraph \*G)

## {//添加道路

## int addnum,i,a,b,c,flag;

## bool pass=false;

## if(G->vertex<2)

## {

## printf("学校数量不够！\n");

## return;

## }

## else if((G->vertex\*(G->vertex-1))/2-G->edge<1)

## {

## printf("道路已满，不可再添加！\n");

## return;

## }

## printf("\n已输入的学校有：\n");

## PrintSchool(G);

## while(!pass)

## {

## printf("请输入添加几条道路：");

## scanf("%d",&a);

## if(a>(G->vertex\*(G->vertex-1))/2-G->edge)

## printf("道路数量过多！\n");

## else

## pass = true;

## }

## addnum = a;

## for(i=G->edge;i<G->edge+addnum;i++)

## {

## printf("添加第%d条道路(共%d条)的信息(两个学校序号和道路权值)：",i-G->edge+1,addnum);

## scanf("%d%d%d",&a,&b,&c);

## if(a<0||a>=G->vertex||b<0||b>G->vertex||c>=1000)

## {

## printf("数据输入错误，请重新输入！\n");

## i--;

## continue;

## }

## flag = SameEdge(G,a,b);

## if(flag>-1)

## {

## printf("该道路已存在！\n");

## i--;

## continue;

## }

## G->p[i].from = a;

## G->p[i].to = b;

## G->p[i].weight = c;

## AdjMatrix[a][b] = c;

## AdjMatrix[b][a] = c;

## }

## G->edge += addnum;

## printf("添加操作完成！\n");

## }

## void AddVertex(ALGraph \*G)

## {//添加学校

## int addnum,i,k;

## char name[MAX\_STR\_LEN];

## printf("请输入要添加学校的个数：");

## scanf("%d",&addnum);

## getchar();

## G->m = (int \*)realloc(G->m,sizeof(int)\*(G->vertex+addnum));

## for(i=G->vertex;i<G->vertex+addnum;i++)

## {

## printf("请输入添加的第%d个学校名称（共%d个）：", i-G->vertex+1, addnum);

## scanf("%s",name);

## for(k=0;k<G->vertex;k++)

## {

## if(!strcmp(schoolName[k],name))

## {

## printf("\n该学校已存在！\n");

## i--;

## break;

## }

## }

## strcpy(schoolName[k],name);

## }

## G->vertex += addnum;

## printf("添加操作完成！\n");

## }

## void PrintEdgeSet(ALGraph \*G)

## {//打印邻接矩阵

## int i,j;

## int p = G->vertex;

## if(p>0)

## {

## printf("\t\t");

## for(i=0;i<p;i++)

## printf("(%d)\t",i);

## printf("\n");

## for(i=0;i<p;i++)

## {

## printf("(%d)%s",i,schoolName[i]);

## for(j=0;j<p;j++)

## {

## if(AdjMatrix[i][j]<MAX\_EDGE\_LEN)

## printf("\t%d",AdjMatrix[i][j]);

## else

## printf("\t∞");

## }

## printf("\n");

## }

## }

## else

## printf("\t无数据\n");

## }

## void PrintSchool(ALGraph \*G)

## {//打印学校名称及序号

## for(int i=0;i<G->vertex;i++)

## {

## printf("%d.%s\t",i,schoolName[i]);

## if(i%2==1)

## printf("\n");

## }

## printf("\n");

## }

## void CreatALGraph(ALGraph \*G)

## {//初始化学校及道路信息

## int i,a,b,c;

## bool pass=false;

## InitAdj();

## if(G->vertex>0)

## free(G->m);

## if(G->edge>0)

## free(G->p);

## while(!pass)

## {

## printf("输入学校数量和道路数量：");

## scanf("%d %d",&a,&b);

## if(b>(a\*(a-1))/2)

## printf("道路数量过多！\n");

## else

## pass = true;

## }

## G->vertex = a;

## G->edge = b;

## G->p = (Edge \*)malloc(sizeof(Edge)\*(G->edge+1));

## G->m = (int \*)malloc(sizeof(int)\*MAX\_PEAK\_NUM);

## for(i=0;i<MAX\_PEAK\_NUM;i++)

## G->m[i] = i;

## for(i=0;i<G->vertex;i++)

## {

## printf("请输入第%d个学校名称（共%d个）：",i+1,G->vertex);

## scanf("%s",schoolName[i]);

## }

## printf("\n已输入的学校有：\n");

## PrintSchool(G);

## for(i=0;i<G->edge;i++)

## {

## printf("请输入第%d条(共%d条)道路的信息(两个学校序号和道路权值)：", i+1,G->edge);

## scanf("%d%d%d",&a,&b,&c);

## if(a<0||a>=G->vertex||b<0||b>G->vertex||c>=1000)

## {

## printf("数据输入错误，请重新输入！\n");

## i--;

## continue;

## }

## G->p[i].from = a;

## G->p[i].to = b;

## G->p[i].weight = c;

## AdjMatrix[a][b] = c;

## AdjMatrix[b][a] = c;

## }

## printf("创建成功！\n");

## }

## void SwapEdge(ALGraph \*G,int i,int j)

## {//交换第i条和第j条边

## G->p[MAX\_PEAK\_NUM]=G->p[i];

## G->p[i]=G->p[j];

## G->p[j]=G->p[MAX\_PEAK\_NUM];

## }

## void Heapify(ALGraph \*G,int start,int end)

## {//重建堆

## int dad = start;

## int son = dad\*2+1;

## if(son>end)

## return;

## if((son+1<=end)&&(G->p[son].weight<G->p[son+1].weight))

## son++;

## if(G->p[dad].weight>G->p[son].weight)

## return;

## SwapEdge(G,dad,son);

## Heapify(G,son,end);

## }

## void HeapSort(ALGraph \*G)

## {//堆排序，权重从小到大存在边集数组中

## int sign,len = G->edge;

## for(sign=(len-1)/2;sign>=0;sign--)//初始化堆

## Heapify(G,sign,len-1);

## for(sign=len-1;sign>0;sign--)

## {

## SwapEdge(G,0,sign);

## Heapify(G,0,sign-1);

## }

## }

## int Find(int \*parent,int g)

## {//通过parent[]找到可连接的边

## while(parent[g]!=0)

## g=parent[g];

## return g;

## }

## int Finish(ALGraph \*G,int \*parent)

## {//判断生成树是否完成（顶点数-1等于边数）

## int i,n=0;

## for(i=0;i<G->vertex;i++)

## if(parent[i])

## n++;

## if(n==G->vertex-1)

## return 1;

## return 0;

## }

## void MinTree\_Kruskal(ALGraph \*G)

## {//克鲁斯卡尔算法求最小生成树

## int i,a,b,total=0;

## int parent[G->vertex];

## HeapSort(G);

## for(i=0;i<G->vertex;i++)//初始化parent[]

## parent[i]=0;

## for(i=0;i<G->edge;i++)

## {

## a=Find(parent,G->p[i].from);

## b=Find(parent,G->p[i].to);

## if(a!=b)//如果a==b则表是a和b在同一颗生成树上

## {

## parent[a]=b;

## printf("\t%s——%s\t%d\n",schoolName[G->p[i].from],schoolName[G->p[i].to],G->p[i].weight);

## total += G->p[i].weight;

## }

## if(Finish(G,parent))//完成后返回

## {

## printf("\t道路权值总计：%d\n",total);

## return;

## }

## }

## printf("\t道路权值总计：%d\n",total);

## }

## void LoadFile(ALGraph \*G)

## {//读取文件

## int i,j,k;

## InitAdj();

## k = -1;

## FILE \*file = fopen("data.txt","r");

## if(file == NULL)

## return;

## fscanf(file,"%d %d\n",&G->vertex,&G->edge);

## G->p = (Edge \*)malloc(sizeof(Edge)\*(G->edge+1));

## G->m = (int \*)malloc(sizeof(int)\*MAX\_PEAK\_NUM);

## for(i=0;i<MAX\_PEAK\_NUM;i++)

## G->m[i] = i;

## for(i=0;i<G->vertex;i++)

## {

## fgets(schoolName[i],MAX\_STR\_LEN,file);

## schoolName[i][strlen(schoolName[i])-1] = 0;//消除行末的换行符

## }

## for(i=0;i<G->vertex;i++)//读取邻接矩阵

## {

## for(j=0;j<G->vertex;j++)

## {

## fscanf(file,"%d",&AdjMatrix[i][j]);

## if(AdjMatrix[i][j]>0&&AdjMatrix[i][j]<1000&&j>=i)

## {

## G->p[++k].from = i;

## G->p[k].to = j;

## G->p[k].weight = AdjMatrix[i][j];

## }

## }

## }

## fclose(file);

## }

## void SaveFile(ALGraph \*G)

## {//保存文件

## int i,j;

## FILE \*file = fopen("data.txt","w");

## if(file == NULL)

## {

## printf("文件打开失败！");

## return;

## }

## fprintf(file,"%d %d\n",G->vertex,G->edge);//存储顶点和边

## for(i=0;i<G->vertex;i++)//存储学校名称

## {

## fputs(schoolName[i],file);

## fputs("\n",file);

## }

## for(i=0;i<G->vertex;i++)//存储邻接矩阵

## {

## for(j=0;j<G->vertex;j++)

## {

## fprintf(file,"%d ",AdjMatrix[i][j]);

## }

## fputs("\n",file);

## }

## fclose(file);

## printf("保存完成！\n");

## }

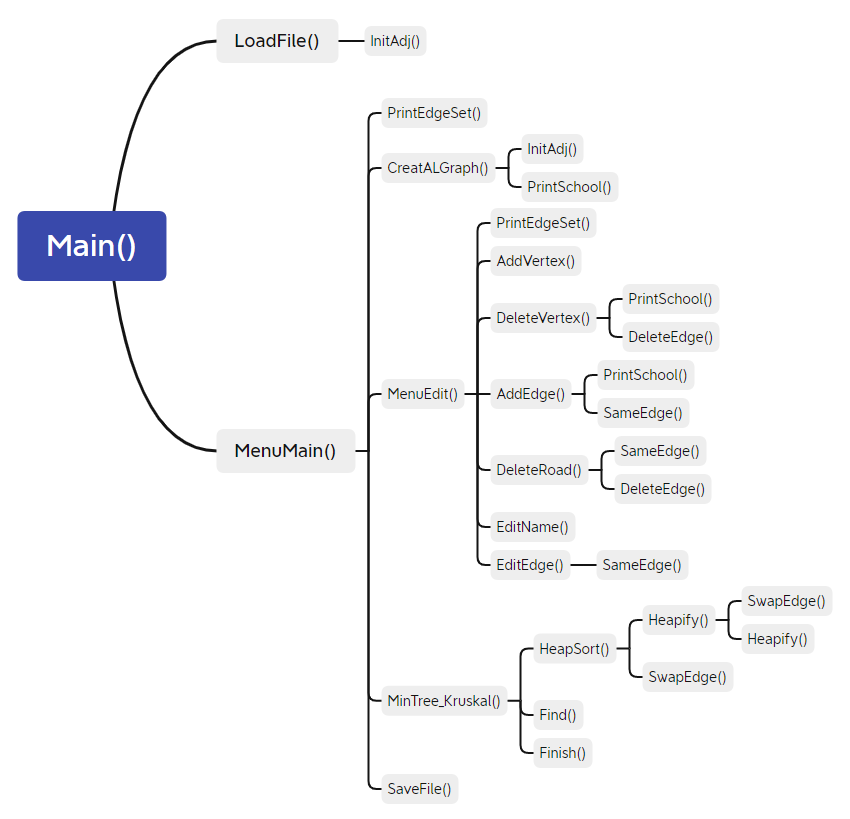
1. 函数的调用关系图

图3-1 函数关系调用图

1. 具体界面

4.1 主界面

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

欢迎使用高校专用通信网络建设系统

===========================================

当前数据：

无数据

===========================================

1.构造通信网络

2.编辑通信网络

3.生成最佳方案

4.保存通信网络

0.退出

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4.2 编辑界面

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

编辑学校及道路信息

===========================================

当前数据：

无数据

===========================================

1.添加高校

2.删除高校

3.添加高校间的线路

4.删除高校间的线路

5.修改高校名称

6.修改线路成本

0.返回

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 调试分析

1.在执行构造通信网络，即初始化学校及道路信息时，先输入顶点数和边数，从而判断边数是否为顶点数\*(顶点数-1)/2，确保在边数已经满的情况下仍要输入道路信息的情况不会发生。这种提前判断的意识需要时刻拥有，防止发现时再修改就为时已晚了。

2.在删除指定学校时，忘记了修改图中各线路所指向顶点的下标，由于考虑不周，导致浪费了大量时间用于调试。

3.由于对动态申请内存空间的运用不够熟练，忘记在结束进程前free()，导致进程频繁报错。

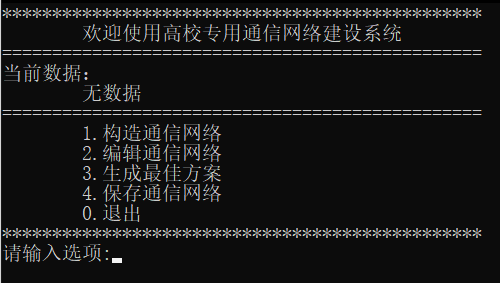
4.对文件的保存和读取掌握不够，在文件中调试如何交叉读取字符串和数字以及在已读取的字符串中去除末尾的换行符时，粗心大意，浪费了很多时间。

5. 算法的时空分析

由于InitAdj, DeleteVertex, PrintEdgeSet, CreatALGraph, LoadFile, SaveFile涉及到矩阵的遍历，时间复杂度都为O(n²)。DeleteRoad, EditName, EditEdge, AddVertex, AddEdge, PrintSchool的时间复杂度为O(n)，MinTree\_Kruskal的时间复杂度为O(logn)。

# 测试结果

程序打开后进入主菜单页面如图5-1。具体操作如下：

图5-1 主菜单页面

1. 执行命令1：构造通信网络

输入： 河北地质大学

河北科技大学

华北理工大学

河北医科大学

河北师范大学

0 1 56

1 3 55

2 4 45

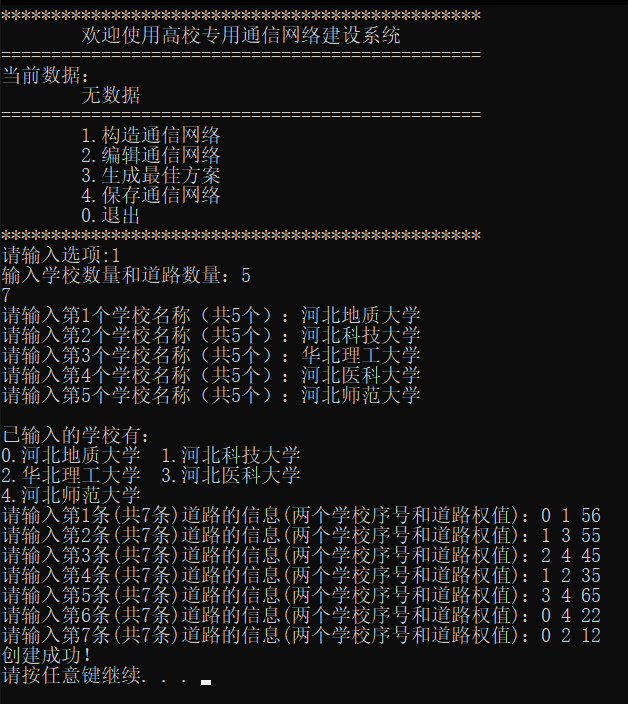
1 2 35

3 4 65

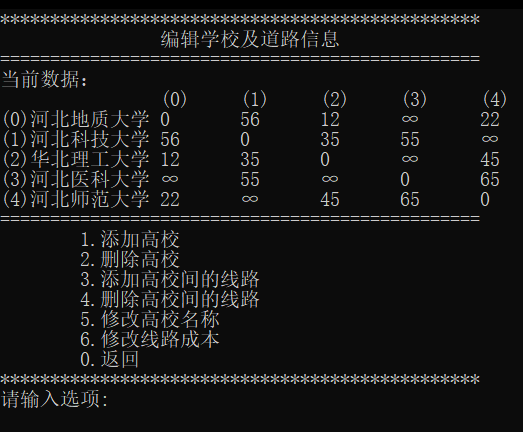
0 4 22

0 2 12

结果如图5-2。

图5-2 构造通信网络

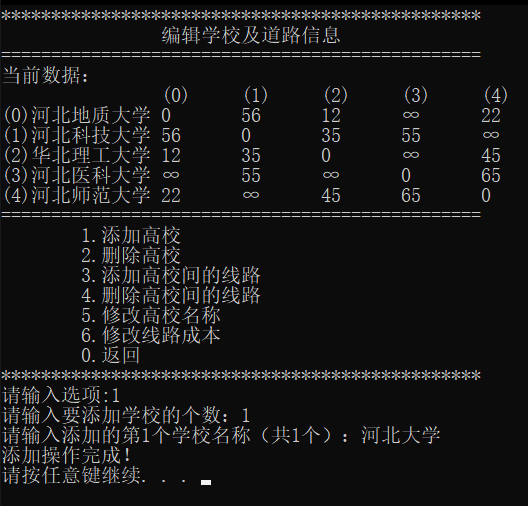
1. 执行命令2：编辑通信网络，进入编辑菜单页面如图5-3。

图5-3 编辑菜单页面

2.1 执行命令1：添加高校

输入：1 河北大学

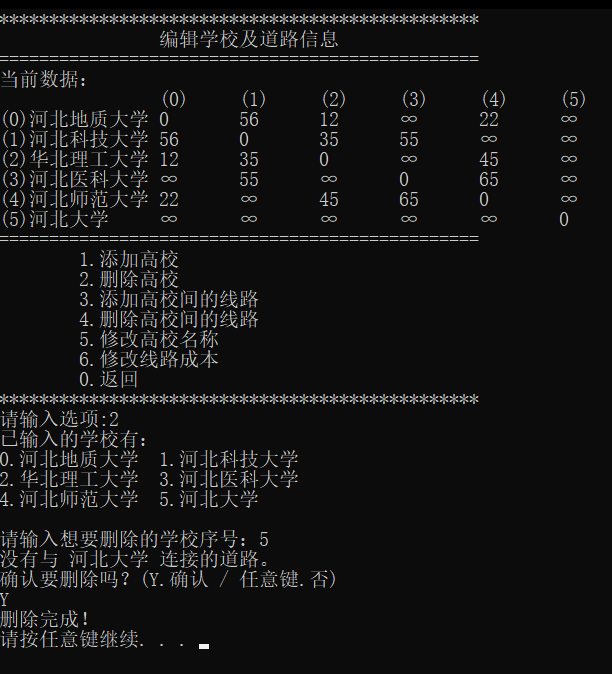
结果如图5-4。

图5-4 添加高校

2.2 执行命令2：删除高校

输入：5 Y

结果如图5-5。

图5-5 删除高校

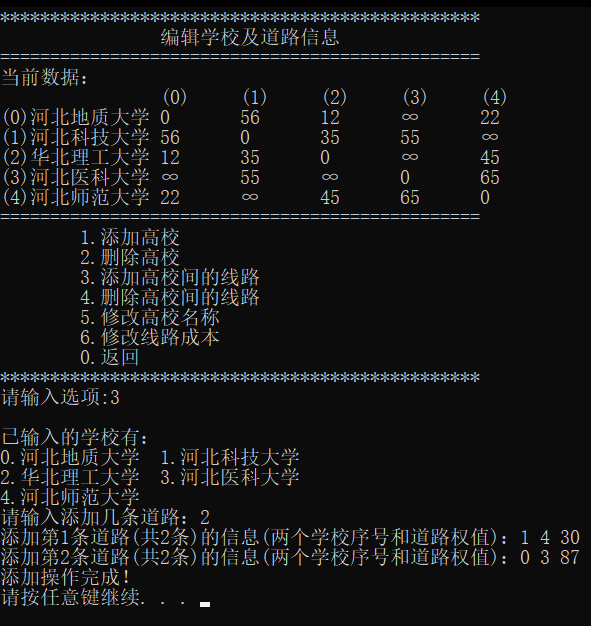
2.3 执行命令3：添加高校间的线路

输入： 2

1 4 30

0 3 87

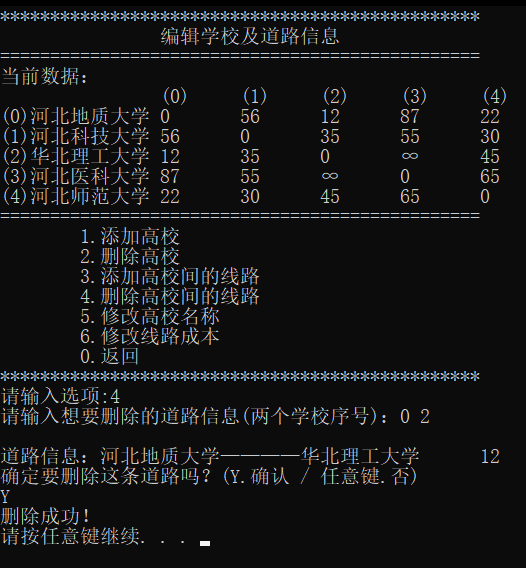
结果如图5-6。

图5-6 添加高校间的线路

2.4 执行命令4：删除高校间的线路

输入：0 2 Y

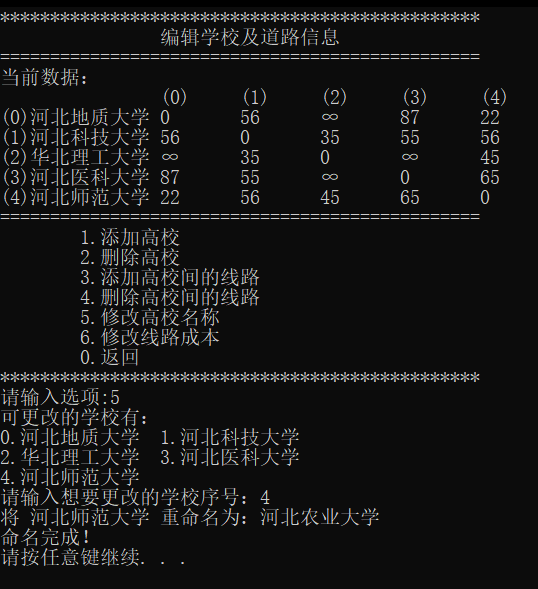
结果如图5-7。

图5-7 删除高校间的线路

2.5 执行命令5：修改高校名称

输入：4 河北农业大学

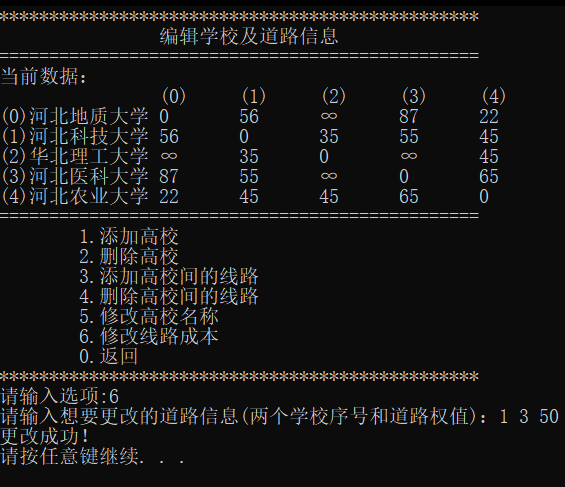
结果如图5-8。

图5-8 修改高校名称

2.6 执行命令6：修改线路成本

输入：1 3 50

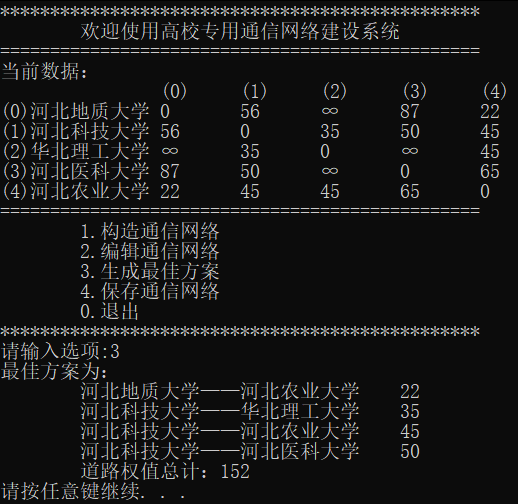
结果如图5-9。

图5-9 修改线路成本

2.7 执行命令0：返回主菜单

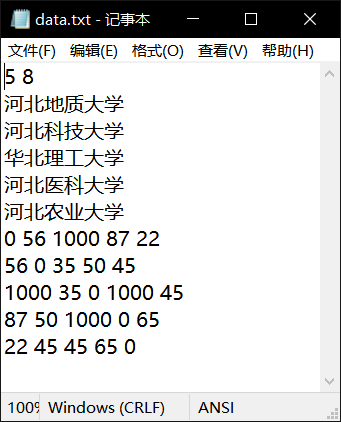
3.执行命令3：生成最佳方案

结果如图5-10。

图5-10 生成最佳方案

4.执行命令4：保存通信网络

保存结果如图5-11。

5-11 保存通信网络

5.执行命令0：退出程序

# 六、附录

main.cpp//主函数文件

graph.cpp//自定义函数文件

graph.h//头文件