

目录

1. 需求分析.....	1
2. 项目亮点.....	1
3. 概要设计.....	1
4. 详细设计.....	1
4.1 判断顶点在二维数组中的位置.....	1
4.2 创建无向网.....	2
4.3 自动创建无向网.....	4
4.4 找出权值最小的边的数组下标.....	4
4.5 普里姆算法.....	5
5. 用户手册.....	7
5.1 界面.....	7
5.2 手动输入.....	7
5.3 自动生成.....	9
6. 心得体会.....	10
7. 附录	11

1. 需求分析

- (1) 利用普利姆算法求网的最小生成树；
- (2) 以文本形式输出生成树中各条边以及他们的权值。

2. 项目亮点

- (1) 使用 system 来命令 Graphviz 绘制网以及最小生成树的图，更加直观地展现了结果。
- (2) 本程序考虑了多种错误输入情况，具有很高的鲁棒性。
- (3) 设计了手动输入网和自动生成网两种功能，满足不同的需求。

3. 概要设计

普利姆算法在找最小生成树时，将顶点分为两类，一类是在查找的过程中已经包含在树中的（假设为 A 类），剩下的是另一类（假设为 B 类）。

对于给定的连通网，起始状态全部顶点都归为 B 类。在找最小生成树时，选定任意一个顶点作为起始点，并将之从 B 类移至 A 类；然后找出 B 类中到 A 类中的顶点之间权值最小的顶点，将之从 B 类移至 A 类，如此重复，直到 B 类中没有顶点为止。所走过的顶点和边就是该连通图的最小生成树，算法运行的时间复杂度为 $O(n^2)$ 。

4. 详细设计

4.1 判断顶点在二维数组中的位置

Listing 1: LocateVex

```
1 int LocateVex(MGraph G,VertexType v)//判断顶点在二维数组中的位置
2 {
3     for(int i=0;i<G.vexnum;i++)
4     {
5         if(G.vexs[i]==v) return i;
6     }
7     return -1;
8 }
```

4.2 创建无向网

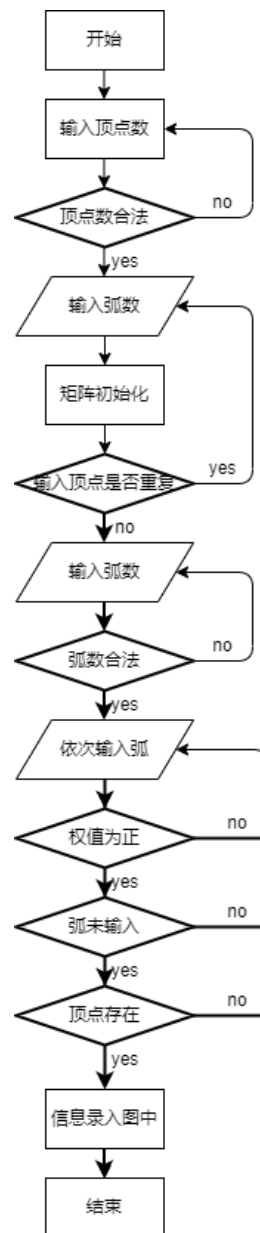


图 1: 创建流程

Listing 2: Create

```

1 void Create(MGraph *G)//创建无向网
2 {
3     int i,j,k,v1,v2,w,m,n,flag;
4     A:printf("请输入顶点数:");scanf("%d",&(G->vexnum));
5     if(G->vexnum<=1||G->vexnum>30){printf("顶点数输入错误!\n");goto A;}//顶点个数不合法
6     B:flag=0;
7     for(i=0;i<G->vexnum;i++)//输入顶点
8     {

```

```

9     printf("请输入第%d个顶点:",i+1);
10    scanf("%d",&(G->vexs[i]));
11    for(j=0;j<G->vexnum;j++)//矩阵初始化
12    {
13        G->arcs[i][j].adj=INFINITY;
14        G->arcs[i][j].info=NULL;
15    }
16 }
17 for(i=0;i<G->vexnum;i++)//顶点输入重复
18 {
19     for(j=i+1;j<G->vexnum;j++)
20     {
21         if(G->vexs[i]==G->vexs[j]) {flag=1;break;}
22     }
23 }
24 if(flag) {printf("顶点输入重复!\n");goto B;}
25 C:printf("请输入弧数:");scanf("%d",&(G->arcnum));
26 if(G->arcnum<1|G->arcnum>G->vexnum*(G->vexnum-1)/2) {printf("弧数不合法!\n");goto
    C;}//弧数不合法
27 printf("请依次输入弧(顶点A 顶点B 弧值):\n");
28 for(k=0;k<G->arcnum;k++)
29 {
30     D:flag=0;
31     printf("第%d条弧:",k+1);
32     scanf("%d %d %d",&v1,&v2,&w);
33     for(i=0;i<G->arcnum;i++)//弧已输入
34     {
35         if(v1==v[i][0]&&v2==v[i][1]||v1==v[i][1]&&v2==v[i][0])
36         {
37             flag=1;
38             break;
39         }
40     }
41     if(flag) {printf("%d-%d的弧已输入!\n",v1,v2);goto D;}
42     if(w<=0) {printf("弧值输入错误!\n");goto D;}//弧值非正
43     m=LocateVex(*G,v1);n=LocateVex(*G,v2);
44     if(m==-1|n==-1) {printf("顶点输入错误!\n");goto D;}
45     if(m==n) {printf("顶点%d重复!\n",v1);goto D;}
46     v[k][0]=v1;v[k][1]=v2;v[k][2]=w;
47     G->arcs[n][m].adj=w;
48     G->arcs[m][n].adj=w;
49 }
50 }

```

4.3 自动创建无向网

用户首先输入顶点数 n ，在检验合法性后自动生成从 1 到 n 的自然数顶点；然后用户输入网中边的权最大值，程序生成 $\frac{n(n-1)}{2}$ 条边及对应的权值，这样就创建了一个顶点数为 n 的随机的无向网。

Listing 3: create

```

1 void create(MGraph *G)//自动创建无向网
2 {
3     int a,b,i,j,k=0,m,n;
4     A:printf("请输入顶点数:");scanf("%d",&(G->vexnum));
5     if(G->vexnum<=1||G->vexnum>30) {printf("顶点数输入错误!\n");goto A;}//顶点个数不合法
6     B:printf("请输入生成随机数最大值:");scanf("%d",&b);
7     if(b<=0||b>INFINITY) {printf("输入超出范围!");goto B;}
8     for(i=0;i<G->vexnum;i++)
9     {
10         G->vexs[i]=i+1;
11         for(j=0;j<G->vexnum;j++)//矩阵初始化
12         {
13             G->arcs[i][j].adj=INFINITY;
14             G->arcs[i][j].info=NULL;
15         }
16     }
17     G->arcnum=G->vexnum*(G->vexnum-1)/2;
18     for(i=1;i<=G->vexnum;i++)
19     {
20         for(j=i+1;j<=G->vexnum;j++)
21         {
22             a=rand()%(b+1)+1; //生成1~b之间的随机整数
23             m=LocateVex(*G,i);n=LocateVex(*G,j);
24             v[k][0]=i;v[k][1]=j;v[k][2]=a;k++;
25             G->arcs[n][m].adj=a;
26             G->arcs[m][n].adj=a;
27         }
28     }
29 }

```

4.4 找出权值最小的边的数组下标

Listing 4: minimum

```

1 int minimum(MGraph G,closedge c)//从辅助数组中找出权值最小的边的数组下标
2 {
3     int i,min=INFINITY,index=-1;
4     for(i=0;i<G.vexnum;i++)

```

```

5     {
6         if(c[i].lowcost>0&& c[i].lowcost<min)
7         {
8             min=c[i].lowcost;
9             index=i;
10        }
11    }
12    return index;
13 }

```

4.5 普里姆算法

将绘制图的程序内嵌到普利姆算法中, 创建一个 graphviz 文件, 在使用普利姆算法输出最小生成树时, 首先将最小生成树的边写入, 边的颜色为蓝; 之后写入其余的边, 边无特定颜色。这样画出的图即包含整个网, 又能明显看出最小生成树, 同时在生成的 graphviz 文件中也可以看到所有的信息, 一举三得。

Listing 5: MiniSpanTree_PRIM

```

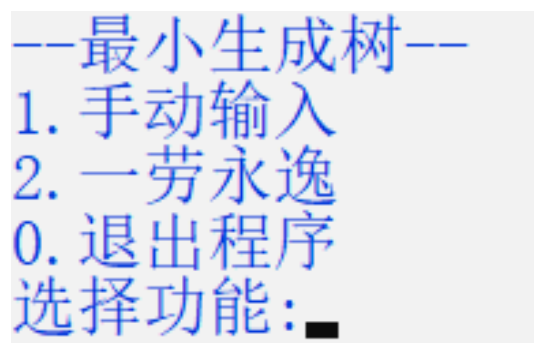
1 void MiniSpanTree_PRIM(MGraph G,VertexType u)//普里姆算法
2 {
3     int i,j,k,m,num=G.arcnum,n[G.vexnum-1][2];
4     char name[100],file[100],command[200]="dot ";
5     printf("请输入文件名:");
6     fflush(stdin);gets(name);strcpy(file,name);
7     strcat(file,".png");strcat(name,".gv");
8     FILE *fp=fopen(name,"w+");
9     fprintf(fp,"graph\n{\n/\\best\n");
10    k=LocateVex(G,u);
11    for(i=0;i<G.vexnum;i++)//辅助数组初始化
12    {
13        if(i!=k)
14        {
15            close[i].adjvex=k;
16            close[i].lowcost=G.arcs[k][i].adj;
17        }
18    }
19    close[k].lowcost=0;//初始
20    for(i=1;i<G.vexnum;i++)//选择其余G.vexnum-1个顶点
21    {
22        k=minimum(G,close);//求出T的下一个节点: 第k顶点
23        m=close[k].adjvex;
24        n[i-1][0]=G.vexs[m];n[i-1][1]=G.vexs[k];
25        fprintf(fp,"v%d--v%d [label=\"%d\"
26                color=blue];\\n",G.vexs[m],G.vexs[k],G.arcs[m][k].adj);//输出生成树的边
27        close[k].lowcost=0;//第k顶点并入U集

```

```
27     for(j=0;j<G.vexnum;j++)
28     {
29         if(G.arcs[k][j].adj<close[j].lowcost)//新顶点并入U后重新选择最小边
30         {
31             close[j].adjvex=k;
32             close[j].lowcost=G.arcs[k][j].adj;
33         }
34     }
35 }
36 for(i=0;i<num;i++)//筛选其他弧
37 {
38     for(j=0;j<G.vexnum-1;j++)
39     {
40         if(v[i][0]==n[j][0]&&v[i][1]==n[j][1]||v[i][0]==n[j][1]&&v[i][1]==n[j][0])
41         {
42             for(m=i;m<num;m++)
43             {
44                 for(k=0;k<3;k++)
45                     v[m][k]=v[m+1][k];
46             }
47             num--;
48             i--;
49             break;
50         }
51     }
52 }
53 fprintf(fp,"//other\n");
54 for(i=0;i<num;i++)//输出其他弧
55     fprintf(fp,"v%d--v%d [label=\"%d\"];\n",v[i][0],v[i][1],v[i][2]);
56 printf("成功生成最小生成树!\n");
57 fprintf(fp,"}");
58 fclose(fp);
59 strcat(command,name);strcat(command," -Ksfdp -Tpng -o ");strcat(command,file);
60 system(command);
61 }
```

5. 用户手册

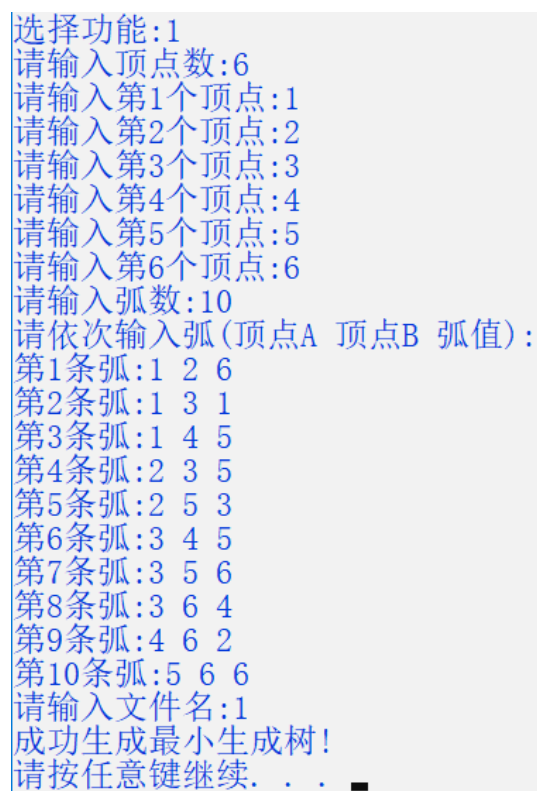
5.1 界面



```
---最小生成树---
1. 手动输入
2. 一劳永逸
0. 退出程序
选择功能: █
```

图 2: 用户界面

5.2 手动输入



```
选择功能:1
请输入顶点数:6
请输入第1个顶点:1
请输入第2个顶点:2
请输入第3个顶点:3
请输入第4个顶点:4
请输入第5个顶点:5
请输入第6个顶点:6
请输入弧数:10
请依次输入弧(顶点A 顶点B 弧值):
第1条弧:1 2 6
第2条弧:1 3 1
第3条弧:1 4 5
第4条弧:2 3 5
第5条弧:2 5 3
第6条弧:3 4 5
第7条弧:3 5 6
第8条弧:3 6 4
第9条弧:4 6 2
第10条弧:5 6 6
请输入文件名:1
成功生成最小生成树!
请按任意键继续. . . █
```

图 3: 手动输入一张网

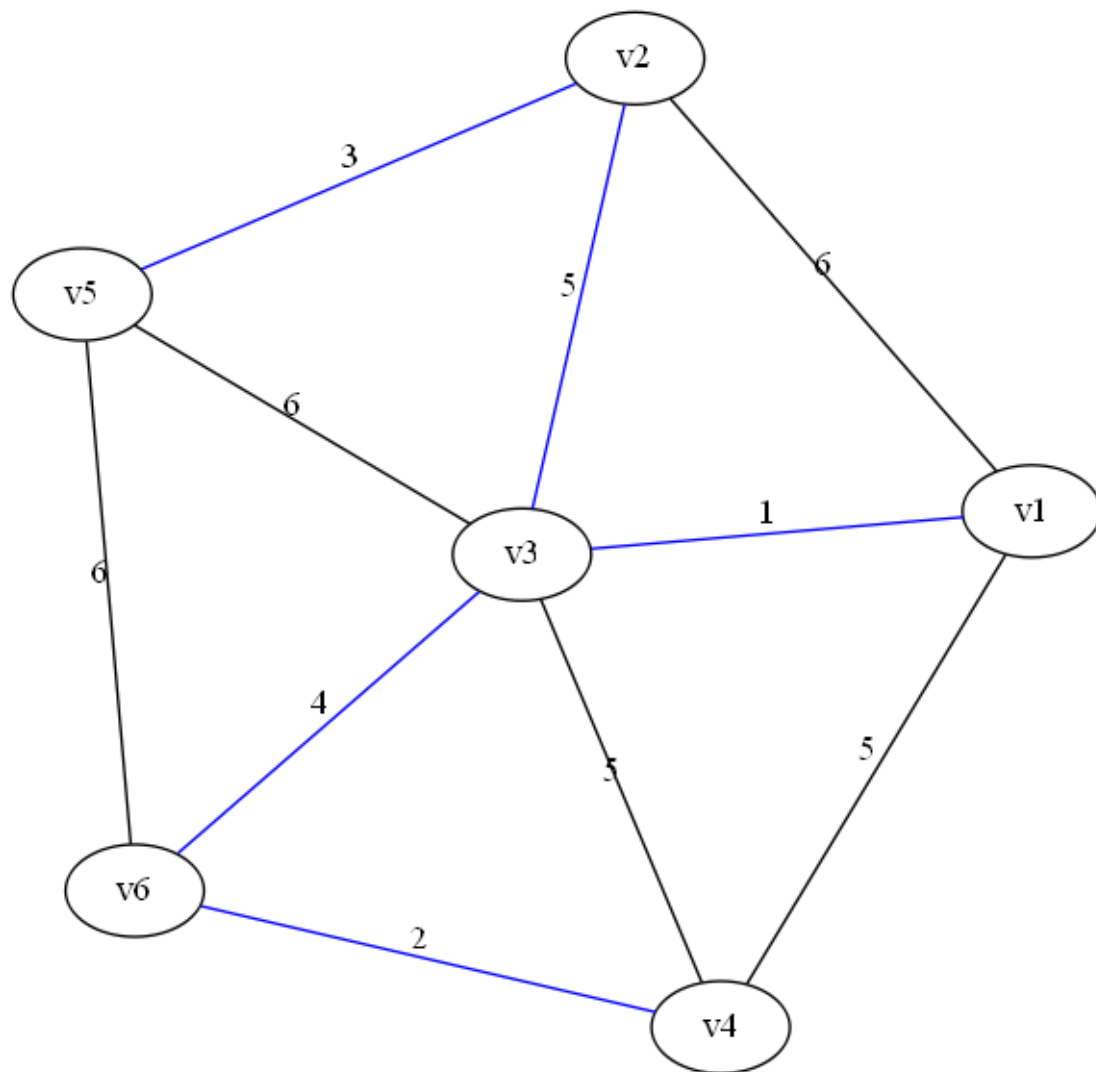


图 4: 功能 1 的最小生成树

```

1 graph
2 {
3 //best
4 v1--v3 [label="1" color=blue];
5 v3--v6 [label="4" color=blue];
6 v6--v4 [label="2" color=blue];
7 v3--v2 [label="5" color=blue];
8 v2--v5 [label="3" color=blue];
9 //other
10 v1--v2 [label="6"];
11 v1--v4 [label="5"];
12 v3--v4 [label="5"];
13 v3--v5 [label="6"];
14 v5--v6 [label="6"];
15 }
    
```

5.3 自动生成

```
--最小生成树--
1. 手动输入
2. 一劳永逸
0. 退出程序
选择功能:2
请输入顶点数:5
请输入生成随机数最大值:50
请输入文件名:2
成功生成最小生成树!
请按任意键继续. . .
```

图 5: 自动生成一张网

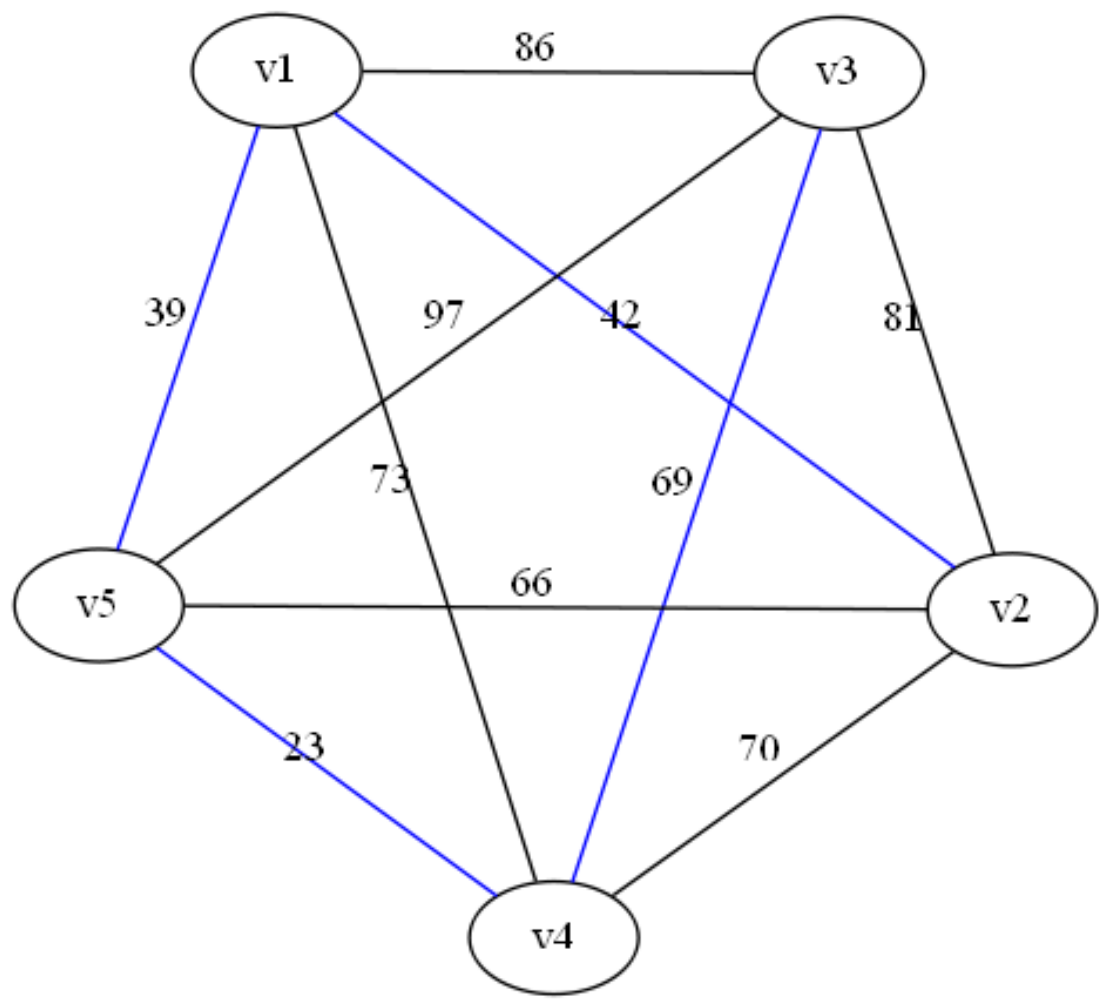


图 6: 功能 2 的最小生成树

```
2 {  
3 //best  
4 v1--v5 [label="39" color=blue];  
5 v5--v4 [label="23" color=blue];  
6 v1--v2 [label="42" color=blue];  
7 v4--v3 [label="69" color=blue];  
8 //other  
9 v1--v3 [label="86"];  
10 v1--v4 [label="73"];  
11 v2--v3 [label="81"];  
12 v2--v4 [label="70"];  
13 v2--v5 [label="66"];  
14 v3--v5 [label="97"];  
15 }
```

6. 心得体会

这次课程设计的心得体会通过实践我们的收获如下：

1. 在这次的最小生成树课程设计的过程中，我们更深刻地了解了普利姆算法的特点与用法。
2. 学会了使用 Graphviz。
3. 由于这次的课程设计相对简单，因此增强了程序的鲁棒性，为以后的 Debug 提供了思路。

7. 附录

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4
5  #define VertexType int
6  #define VRType int
7  #define InfoType char
8  #define MAX_VERTEX_NUM 30 //最大顶点个数
9  #define INFINITY 99999 //最大值
10
11 typedef struct
12 {
13     VRType adj;          //顶点关系 1|0表示是否相邻
14     InfoType *info;      //该弧相关信息的指针
15 }ArcCell,AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
16 typedef struct
17 {
18     VertexType vexs[MAX_VERTEX_NUM]; //顶点向量
19     AdjMatrix arcs;                  //邻接矩阵
20     int vexnum,arcnum;               //图的当前顶点数和弧数
21 }MGraph;
22 typedef struct
23 {
24     VertexType adjvex; //权值最小边的起始点
25     VRType lowcost;    //该边权值
26 }closedge[MAX_VERTEX_NUM];
27
28 closedge close;//辅助数组，用于每次筛选出权值最小的边
29 int v[100][3]={0,0,0};//存储弧信息
30
31 int LocateVex(MGraph G,VertexType v)//判断顶点在二维数组中的位置
32 {
33     for(int i=0;i<G.vexnum;i++)
34     {
35         if(G.vexs[i]==v) return i;
36     }
37     return -1;
38 }
39
40 void Create(MGraph *G)//创建无向网
41 {
42     int i,j,k,v1,v2,w,m,n,flag;
43     A:printf("请输入顶点数:");scanf("%d",&(G->vexnum));
44     if(G->vexnum<=1||G->vexnum>30) {printf("顶点数输入错误!\n");goto A;}//顶点个数不合法

```

```

45     B:flag=0;
46     for(i=0;i<G->vexnum;i++)//输入顶点
47     {
48         printf("请输入第%d个顶点:",i+1);
49         scanf("%d",&(G->vexs[i]));
50         for(j=0;j<G->vexnum;j++)//矩阵初始化
51         {
52             G->arcs[i][j].adj=INFINITY;
53             G->arcs[i][j].info=NULL;
54         }
55     }
56     for(i=0;i<G->vexnum;i++)//顶点输入重复
57     {
58         for(j=i+1;j<G->vexnum;j++)
59         {
60             if(G->vexs[i]==G->vexs[j]) {flag=1;break;}
61         }
62     }
63     if(flag) {printf("顶点输入重复!\n");goto B;}
64     C:printf("请输入弧数:");scanf("%d",&(G->arcnum));
65     if(G->arcnum<1||G->arcnum>G->vexnum*(G->vexnum-1)/2) {printf("弧数不合法!\n");goto
        C;}//弧数不合法
66     printf("请依次输入弧(顶点A 顶点B 弧值):\n");
67     for(k=0;k<G->arcnum;k++)
68     {
69         D:flag=0;
70         printf("第%d条弧:",k+1);
71         scanf("%d %d %d",&v1,&v2,&w);
72         for(i=0;i<G->arcnum;i++)//弧已输入
73         {
74             if(v1==v[i][0]&&v2==v[i][1]||v1==v[i][1]&&v2==v[i][0])
75             {
76                 flag=1;
77                 break;
78             }
79         }
80         if(flag) {printf("%d-%d的弧已输入!\n",v1,v2);goto D;}
81         if(w<=0) {printf("弧值输入错误!\n");goto D;}//弧值非正
82         m=LocateVex(*G,v1);n=LocateVex(*G,v2);
83         if(m==-1||n==-1) {printf("顶点输入错误!\n");goto D;}
84         if(m==n) {printf("顶点%d重复!\n",v1);goto D;}
85         v[k][0]=v1;v[k][1]=v2;v[k][2]=w;
86         G->arcs[n][m].adj=w;
87         G->arcs[m][n].adj=w;
88     }
89 }
90

```

```

91 void create(MGraph *G)//自动创建无向网
92 {
93     int a,b,i,j,k=0,m,n;
94     A:printf("请输入顶点数:");scanf("%d",&(G->vexnum));
95     if(G->vexnum<=1||G->vexnum>30) {printf("顶点数输入错误!\n");goto A;}//顶点个数不合法
96     B:printf("请输入生成随机数最大值:");scanf("%d",&b);
97     if(b<=0||b>INFINITY) {printf("输入超出范围!");goto B;}
98     for(i=0;i<G->vexnum;i++)
99     {
100         G->vexs[i]=i+1;
101         for(j=0;j<G->vexnum;j++)//矩阵初始化
102         {
103             G->arcs[i][j].adj=INFINITY;
104             G->arcs[i][j].info=NULL;
105         }
106     }
107     G->arcnum=G->vexnum*(G->vexnum-1)/2;
108     for(i=1;i<=G->vexnum;i++)
109     {
110         for(j=i+1;j<=G->vexnum;j++)
111         {
112             a=rand()%(b+1)+1; //生成1~b之间的随机整数
113             m=LocateVex(*G,i);n=LocateVex(*G,j);
114             v[k][0]=i;v[k][1]=j;v[k][2]=a;k++;
115             G->arcs[n][m].adj=a;
116             G->arcs[m][n].adj=a;
117         }
118     }
119 }
120
121 int minimum(MGraph G,closedge c)//从辅助数组中找出权值最小的边的数组下标
122 {
123     int i,min=INFINITY,index=-1;
124     for(i=0;i<G.vexnum;i++)
125     {
126         if(c[i].lowcost>0&&c[i].lowcost<min)
127         {
128             min=c[i].lowcost;
129             index=i;
130         }
131     }
132     return index;
133 }
134
135 void MiniSpanTree_PRIM(MGraph G,VertexType u)//普里姆算法
136 {
137     int i,j,k,m,num=G.arcnum,n[G.vexnum-1][2];

```

```

138     char name[100],file[100],command[200]="dot ";
139     printf("请输入文件名:");
140     fflush(stdin);gets(name);strcpy(file,name);
141     strcat(file,".png");strcat(name,".gv");
142     FILE *fp=fopen(name,"w+");
143     fprintf(fp,"graph\n{\n/\\best\n");
144     k=LocateVex(G,u);
145     for(i=0;i<G.vexnum;i++)//辅助数组初始化
146     {
147         if(i!=k)
148         {
149             close[i].adjvex=k;
150             close[i].lowcost=G.arcs[k][i].adj;
151         }
152     }
153     close[k].lowcost=0;//初始
154     for(i=1;i<G.vexnum;i++)//选择其余G.vexnum-1个顶点
155     {
156         k=minimum(G,close);//求出T的下一个节点:第k顶点
157         m=close[k].adjvex;
158         n[i-1][0]=G.vexs[m];n[i-1][1]=G.vexs[k];
159         fprintf(fp,"v%d--v%d [label=\"%d\"\\n",G.vexs[m],G.vexs[k],G.arcs[m][k].adj);//输出生成树的边
160         close[k].lowcost=0;//第k顶点并入U集
161         for(j=0;j<G.vexnum;j++)
162         {
163             if(G.arcs[k][j].adj<close[j].lowcost)//新顶点并入U后重新选择最小边
164             {
165                 close[j].adjvex=k;
166                 close[j].lowcost=G.arcs[k][j].adj;
167             }
168         }
169     }
170     for(i=0;i<num;i++)//筛选其他弧
171     {
172         for(j=0;j<G.vexnum-1;j++)
173         {
174             if(v[i][0]==n[j][0]&&v[i][1]==n[j][1]||v[i][0]==n[j][1]&&v[i][1]==n[j][0])
175             {
176                 for(m=i;m<num;m++)
177                 {
178                     for(k=0;k<3;k++)
179                     v[m][k]=v[m+1][k];
180                 }
181                 num--;
182                 i--;
183                 break;

```

```
184     }
185 }
186 }
187 fprintf(fp, "//other\n");
188 for(i=0; i<num; i++)//输出其他弧
189     fprintf(fp, "v%d--v%d [label=\"%d\"]; \n", v[i][0], v[i][1], v[i][2]);
190 printf("成功生成最小生成树!\n");
191 fprintf(fp, "}");
192 fclose(fp);
193 strcat(command, name); strcat(command, " -Ksfdp -Tpng -o "); strcat(command, file);
194 system(command);
195 }
196
197 int main()
198 {
199     int choice;
200     MGraph G;
201     system("color F1");
202     A: system("cls");
203     printf("--最小生成树--\n1. 手动输入\n2. 一劳永逸\n0. 退出程序\n选择功能:");
204     scanf("%d", &choice);
205     switch(choice)
206     {
207         case 1: Create(&G); MiniSpanTree_PRIM(G, 1); system("pause"); goto A;
208         case 2: create(&G); MiniSpanTree_PRIM(G, 1); system("pause"); goto A;
209         case 0: break;
210         default: goto A;
211     }
212     system("pause");
213 }
```