数据结构

2022

实 验 报 告

实验项目名称： 普里姆算法求最小生成树和求有向图的简单路径

班级： 7

学号： 2021302181140

姓名： 应晓宇

指导教师： 沈志东

实验时间： 2022.5.12

实验八： 普里姆算法求最小生成树和求有向图的简单路径

一、实验要求

（1）独立完成实验

（2）撰写实验报告

二、实验环境

硬 件： CPU： AMD RYZEM 5900HX GPU： RTX 3070

操作系统：windows 10

软件：visual studio2022

三、实验步骤及思路

（1）题目分析：实验要求使用普里姆算法，那就主要编写普里姆算法。在此之前，先编写邻接矩阵的算法，然后在主函数中调用。

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#define INF 32767
#define MAXV 100
typedef char InfoType;
typedef struct
{
    int no;
    InfoType info;
}Vertextype;
typedef struct
{
    int edges[MAXV][MAXV];
```

```
    int n,e;
    Vertextype vex[MAXV];
}MatGraph;
typedef struct ANode
{
    int adjvex;
    struct ANode *nextarc;
    int weight;
}ArcNode;
typedef struct Vnode
{
    InfoType info;
    int count;
    ArcNode *firstarc;
}VNode;
typedef struct
{
    VNode adjlist[MAXV];
    int n,e;
}AdjGraph;
void CreateMat(MatGraph g,int A[MAXV][MAXV],int n,int e)
{
    int i,j;
    g.n=n;
    g.e=e;
    for(i=0;i<g.n;i++)
        for(j=0;j<g.n;j++)
            g.edges[i][j]=A[i][j];
}
void DispMat(MatGraph g)
{
    int i,j;
    for(i=0;i<g.n;i++){
        for(j=0;j<g.n;j++)
            if(g.edges[i][j]!=INF)
                printf("%4d",g.edges[i][j]);
            else
                printf("%4s","INF");
            printf("\n");
    }
}
void Prim(MatGraph g,int v)
{
    int lowcost[MAXV],min,n=g.n;
```

```
    int closest[MAXV],i,j,k;
    for(i=0;i<n;i++){
        lowcost[i]==g.edges[v][i];
        closest[i]=v;
    }
    for(i=1;i<n;i++){
        min=INF;
        for(j=0;j<n;j++)
            if(lowcost[j]!=0&&lowcost[j]<min){
                min=lowcost[j];
                k=j;
            }
        printf("  边(%d,%d)权为:%d\n",closest[k],k,min);
        lowcost[k]=0;
        for(j=0;j<n;j++)
            if(g.edges[k][j]!=0&&g.edges[k][j]<lowcost[j]){
                lowcost[j]=g.edges[k][j];
                closest[j]=k;
            }
    }
}
int main()
{
    int v=3;
    MatGraph g;
    int
A[MAXV][MAXV]={{0,5,8,7,INF,3},{5,0,4,INF,INF,INF},{8,4,0,5,INF,9},{7,INF,5,0,5,6},{INF,INF,INF,5,0,
1},{3,INF,9,6,1,0}};
    int n=6,e=10;
    CreateMat(g,A,n,e);
    printf("图 G 的邻接矩阵:\n");
    DispMat(g);
    printf("普里姆算法求解结果:\n");
    Prim(g,0);
    system("pause");
    return 1;
}
```

（2）题目分析：采用回溯深度优先搜索方法，计算路径，路径长度，最短路径。同样先编写邻接表的算法。

```
#include<stdio.h>
#include<malloc.h>
#define INF 32767
```

```c
#define MAXV 100
typedef char InfoType;
typedef struct
{
    int no;
    InfoType info;
}Vertextype;
typedef struct
{
    int edges[MAXV][MAXV];
    int n,e;
    Vertextype vex[MAXV];
}MatGraph;
typedef struct ANode
{
    int adjvex;
    struct ANode *nextarc;
    int weight;
}ArcNode;
typedef struct Vnode
{
    InfoType info;
    int count;
    ArcNode *firstarc;
}VNode;
typedef struct
{
    VNode adjlist[MAXV];
    int n,e;
}AdjGraph;
void CreateAdj(AdjGraph *G,int A[MAXV][MAXV],int n,int e)
{
    int i,j;
    ArcNode *p;
    G=(AdjGraph*)malloc(sizeof(AdjGraph));
    for(i=0;i<n;i++)
        G->adjlist[i].firstarc=NULL;
    for(i=0;i<n;i++)
        for(j=n-1;j>=0;j--)
            if(A[i][j]!=0&&A[i][j]!=INF){
                p=(ArcNode*)malloc(sizeof(ArcNode));
                p->adjvex=j;
                p->weight=A[i][j];
                p->nextarc=G->adjlist[i].firstarc;
```

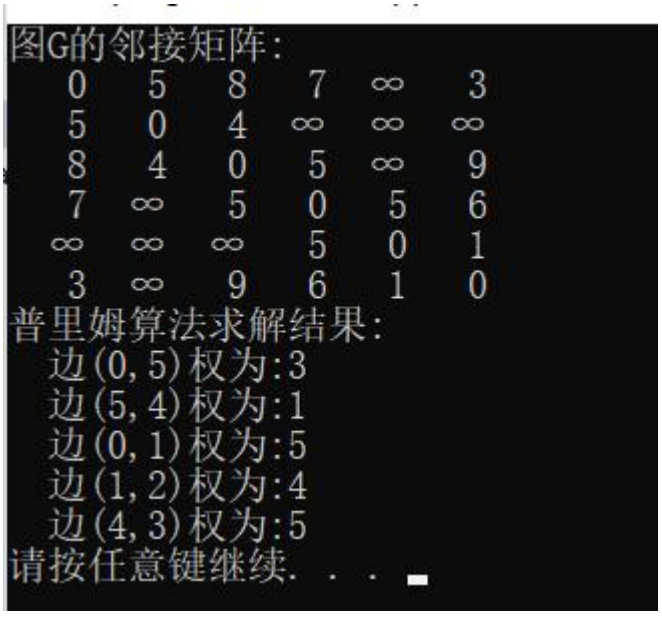
```
                G->adjlist[i].firstarc=p;
            }
        G->n=n;
        G->e=e;
}
void DispAdj(AdjGraph *G)
{
    ArcNode *p;
    for(int i=0;i<G->n;i++){
        p=G->adjlist[i].firstarc;
        printf("%3d:",i);
        while(p!=NULL){
            printf("%3d[%d]->",p->adjvex,p->weight);
            p=p->nextarc;
        }
        printf("^\n");
    }
}
void DestroyAdj(AdjGraph *G)
{
    ArcNode *p,*pre;
    for(int i=0;i<G->n;i++){
        pre=G->adjlist[i].firstarc;
        if(pre!=NULL){
            p=pre->nextarc;
            while(p!=NULL){
                free(pre);
                pre=p;
                p=p->nextarc;
            }
            free(pre);
        }
    }
    free(G);
}
int visited[MAXV];
void PallAll1(AdjGraph *G,int u,int v,int path[],int d)
{
    ArcNode *p;
    int j,w;
    d++;
    path[d]=u;
    visited[u]=1;
    if(u==v&&d>0){
```

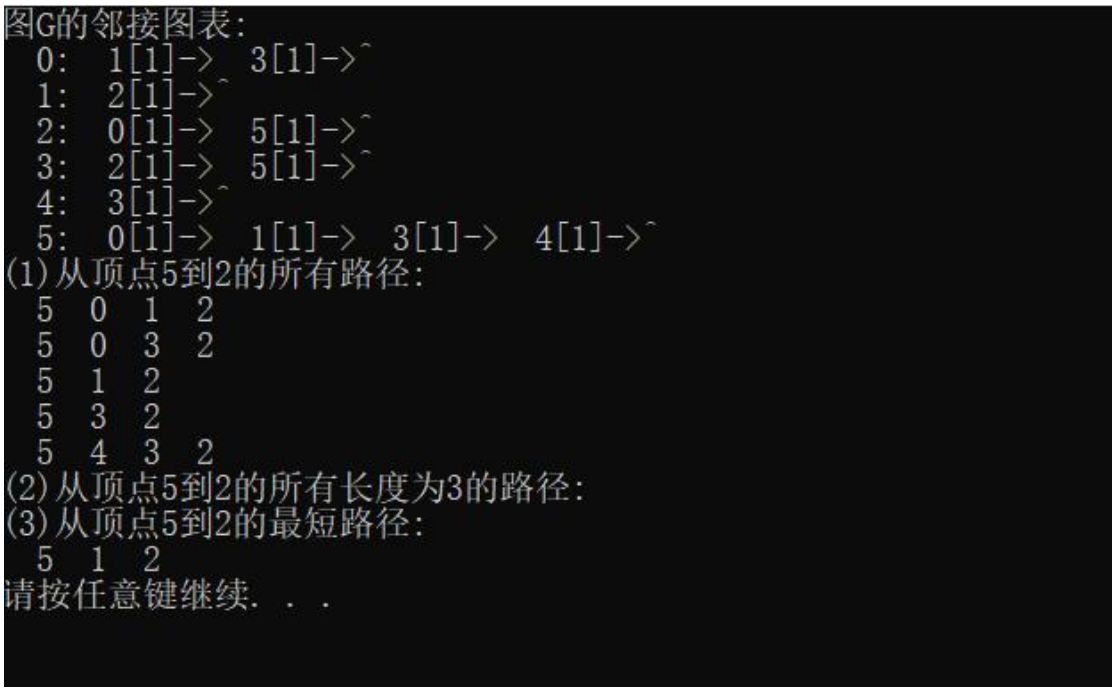
```
        for(j=0;j<=d;j++)
            printf("%3d",path[j]);
        printf("\n");
    }
    p=G->adjlist[u].firstarc;
    while(p!=NULL){
        w=p->adjvex;
        if(visited[w]==0)
            pathAll1(G,w,v,path,d);
        p=p->nextarc;
    }
    visited[u]=0;
}
void pathAll2(AdjGraph *G,int u,int v,int l,int path[],int d)
{
    int w,i;
    ArcNode *p;
    visited[u]=1;
    d++;
    path[d]=u;
    if(u==v&&d==1){
        for(i=0;i<=d;i++)
            printf("%3d",path[i]);
        printf("\n");
    }
    p=G->adjlist[u].firstarc;
    while(p!=NULL){
        w=p->adjvex;
        if(visited[w]==0)
            PallAll2(G,w,v,l,path,d);
        p=p->nextarc;
    }
    visited[u]=0;
}
int ShortPath(AdjGraph *G,int u,int v,int path[])
{
    struct{
        int vno;
        int level;
        int parent;
    }qu[MAXV];
    int front=-1,rear=-1,k,lev,i,j;
    ArcNode *p;
    visited[u]=1;
```

```
    rear++;
    qu[rear].vno=u;
    qu[rear].level=0;
    qu[rear].parent=-1;
    while(front<rear){
        front++;
        k=qu[front].vno;
        lev=qu[front].level;
        if(k==v){
            i=0;
            j=front;
            while(j!=-1){
                path[lev-i]=qu[j].vno;
                j=qu[j].parent;
                i++;
            }
            return lev;
        }
        p=G->adjlist[k].firstarc;
        while(p!=NULL){
            if(visited[p->adjvex]==0){
                visited[p->adjvex]=1;
                rear++;
                qu[rear].vno=p->adjvex;
                qu[rear].level=lev+1;
                qu[rear].parent=front;
            }
            p=p->nextarc;
        }
    }
    return -1;
}
int main()
{
    int i,j;
    int u=5,v=2,l=3;
    int path[MAXV];
    AdjGraph *G;
    int A[MAXV][MAXV]={{0,1,0,1,0,0},{0,0,1,0,0,0},
                       {1,0,0,0,0,1},{0,0,1,0,0,1},
                       {0,0,0,1,0,0},{1,1,0,1,1,0}};
    int n=6,e=10;
    CreateAdj(G,A,n,e);
    printf("图 G 的邻接图表:\n");
```

```
    DispAdj(G);
    printf("(1)从顶点%d 到%d 的所有路径:\n",u,v);
    for(i=0;i<n;i++)    visited[i]=0;
    PallAll1(G,u,v,path,-1);
    printf("(2)从顶点%d 到%d 的所有长度为%d 的路径:\n",u,v,l);
    pathAll2(G,u,v,l,path,-1);
    printf("(3)从顶点%d 到%d 的最短路径:\n",u,v);
    for(i=0;i<n;i++) visited[i]=0;
    j=ShortPath(G,u,v,path);
    for(i=0;i<j;i++)
        printf("%3d",path[i]);
    printf("\n");
    DestroyAdj(G);
    system("pause");
    return 1;
}
```

四、实验结果及分析





五、总结

实验完成，结果正确。

问题：图这一章难度很高，学的不是很会。