第一题

算法：

void dijkstra(int graph[][V\_SIZE],int n,int start,int dist[])

{

int\* path=(int\*)malloc(sizeof(int)\*n);

int\* longest=(int\*)malloc(sizeof(int)\*n);

int\* mark=(int\*)malloc(sizeof(int)\*n);

int min,v,i,j;

array\_fill(mark,n, 0);

array\_fill(dist,n, inf);

for(i=0; i<n; i++)

{

dist[i]=graph[start][i];

if(i!=start&&dist[i]<inf)

path[i]=start;

else

path[i]=-1;

}

mark[start]=1;

while(1)

{

min=0;

v=-1;

for(i=0; i<n; i++)

{

if(!mark[i])

{

if(dist[i]>min)

{

min=dist[i];

v=i;

}

}

}

if(v==-1)

break;

mark[v]=1;

for(i=0; i<n; i++)

{

if(!mark[i]&&

graph[v][i]!=inf&&

dist[v]\*graph[v][i]>dist[i])

{

dist[i]=dist[v]+graph[v][i];

path[i]=v;

}

}

}

printf("起点\t终点\t可靠距离\t对应路径 \n");

for(i=0; i<n; i++)

{

if(i==start)

continue;

array\_fill(longest,n, 0);

printf("%d\t",start);

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

printf("%d\t",dist[i]);

int k=0;

longest[k]=i;

while(path[longest[k]]!=start)

{

k++;

longest[k]=path[longest[k-1]];

}

k++;

longest[k]=start;

for(j=k; j>0; j--)

{

printf("%d->",longest[j]);

}

printf("%d\n",longest[0]);

}

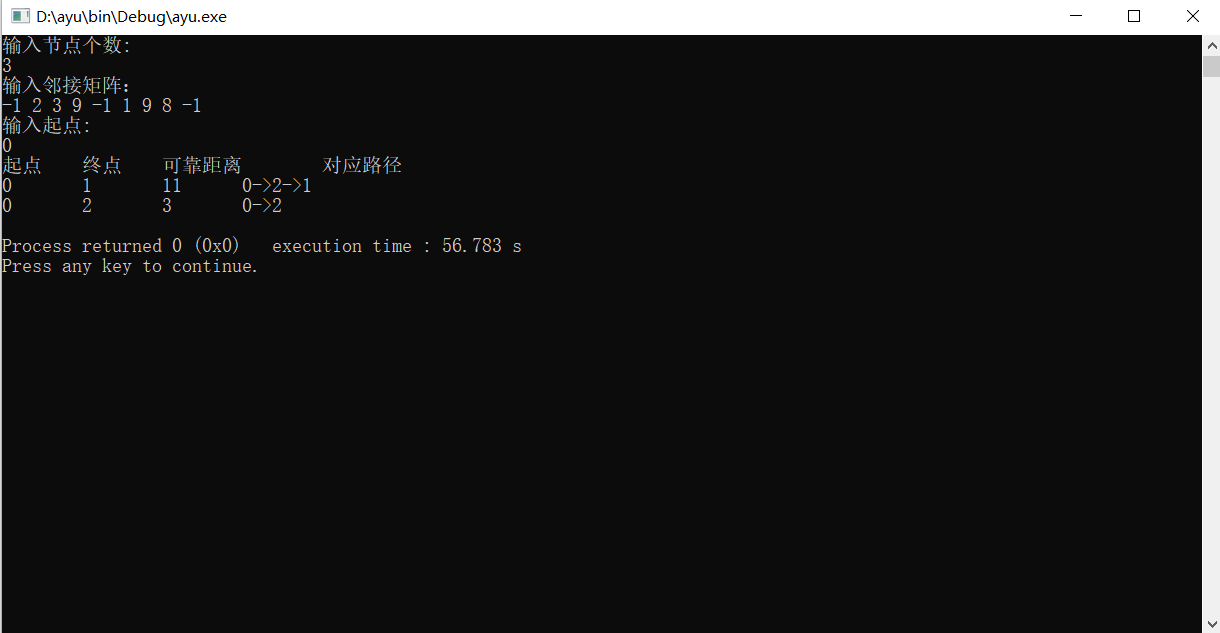
free(path);

free(longest);

free(mark);

return ;

}运行截图：



第二题

算法：void DFSTraverse(ALGraph G)//DFS遍历

{

int v,count=0;

for (v=0;v<G.vexnum;++v)

visited[v]=FALSE;

for (v=0;v<G.vexnum;++v)

if(!visited[v])

{

DFS(G,v);

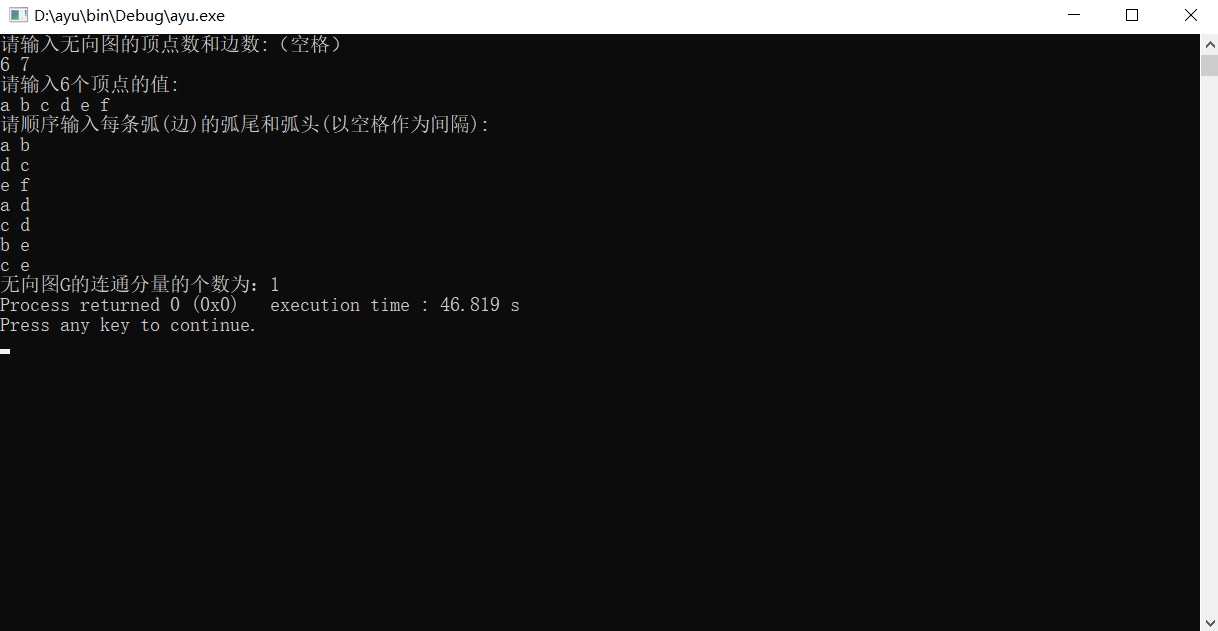
count++;//对连通分量进行计数

}

printf("无向图G的连通分量的个数为：%d",count);

}

运行截图：



第一题源代码：

#include <stdio.h>

#include <stdlib.h>

#define inf 65535

#define V\_SIZE 10

void array\_fill(int \* array, int len, int val)

{

int i;

for (i = 0; i < len; i++)

{

array[i] = val;

}

}

void dijkstra(int graph[][V\_SIZE],int n,int start,int dist[])

{

int\* path=(int\*)malloc(sizeof(int)\*n);

int\* longest=(int\*)malloc(sizeof(int)\*n);

int\* mark=(int\*)malloc(sizeof(int)\*n);

int min,v,i,j;

array\_fill(mark,n, 0);

array\_fill(dist,n, inf);

for(i=0; i<n; i++)

{

dist[i]=graph[start][i];

if(i!=start&&dist[i]<inf)

path[i]=start;

else

path[i]=-1;

}

mark[start]=1;

while(1)

{

min=0;

v=-1;

for(i=0; i<n; i++)

{

if(!mark[i])

{

if(dist[i]>min)

{

min=dist[i];

v=i;

}

}

}

if(v==-1)

break;

mark[v]=1;

for(i=0; i<n; i++)

{

if(!mark[i]&&

graph[v][i]!=inf&&

dist[v]\*graph[v][i]>dist[i])

{

dist[i]=dist[v]+graph[v][i];

path[i]=v;

}

}

}

printf("起点\t终点\t可靠距离\t对应路径 \n");

for(i=0; i<n; i++)

{

if(i==start)

continue;

array\_fill(longest,n, 0);

printf("%d\t",start);

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

printf("%d\t",dist[i]);

int k=0;

longest[k]=i;

while(path[longest[k]]!=start)

{

k++;

longest[k]=path[longest[k-1]];

}

k++;

longest[k]=start;

for(j=k; j>0; j--)

{

printf("%d->",longest[j]);

}

printf("%d\n",longest[0]);

}

free(path);

free(longest);

free(mark);

return ;

}

int main()

{

int valCount;

printf("输入节点个数:\n");

scanf("%d",&valCount);

printf("输入邻接矩阵：\n");

int newW[V\_SIZE][V\_SIZE];

int tempVal;

for(int i=0; i<valCount; i++)

{

for(int j=0; j<valCount; j++)

{

scanf("%d",&tempVal);

if(tempVal==-1)

{

newW[i][j] = inf;

}

else

newW[i][j] = tempVal;

}

}

int newDist[valCount];

int startIndex;

while(1)

{

printf("输入起点:\n");

scanf("%d",&startIndex);

if(startIndex>=0&&startIndex<valCount)

{

break;

}

}

dijkstra(newW,valCount,startIndex,newDist);

return 0;

}第二题源代码：

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#define MAX\_NAME 3

#define MAX\_VERTEX\_NUM 20

#define TRUE 1

#define FALSE 0

int visited[MAX\_VERTEX\_NUM];

typedef char VertexType[MAX\_NAME];

typedef struct ArcNode

{

int adjvex; // 该弧所指向的顶点的位置

struct ArcNode \*nextarc; // 指向下一条弧的指针

}ArcNode; // 表结点

typedef struct VNode

{

VertexType data;

ArcNode \*firstarc;

}VNode,AdjList[MAX\_VERTEX\_NUM];

typedef struct

{

AdjList vertices;

int vexnum,arcnum;

}ALGraph;

int LocateVex(ALGraph G,VertexType u)

{

int i;

for(i=0;i<G.vexnum;++i)

if(strcmp(u,G.vertices[i].data)==0)

return i;

return -1;

}

int CreateGraph(ALGraph \*G)

{

int i,j,k;

VertexType va,vb;

ArcNode \*p;

printf("请输入无向图的顶点数和边数:（空格）\n");

scanf("%d%d", &(\*G).vexnum, &(\*G).arcnum);

printf("请输入%d个顶点的值:\n",(\*G).vexnum);

for(i = 0; i < (\*G).vexnum; ++i)

{

scanf("%s", (\*G).vertices[i].data);

(\*G).vertices[i].firstarc = NULL;

}

printf("请顺序输入每条弧(边)的弧尾和弧头(以空格作为间隔):\n");

for(k = 0;k < (\*G).arcnum; ++k)

{

scanf("%s%s",va,vb);

i = LocateVex(\*G,va); // 弧尾

j = LocateVex(\*G,vb); // 弧头

p = (ArcNode\*)malloc(sizeof(ArcNode));

p->adjvex = j;

p->nextarc = (\*G).vertices[i].firstarc; // 表头

(\*G).vertices[i].firstarc = p;

p = (ArcNode\*)malloc(sizeof(ArcNode));

p->adjvex = i;

p->nextarc = (\*G).vertices[j].firstarc; // 表尾

(\*G).vertices[j].firstarc = p;

}

return 1;

}

void DFS(ALGraph G,int v)//从第v个结点递归遍历

{

ArcNode \*p;

visited[v] = TRUE;//访问第v个顶点

for(p=G.vertices[v].firstarc;p;p=p->nextarc)

if(!visited[p->adjvex]) DFS(G,p->adjvex);

}

void DFSTraverse(ALGraph G)//DFS遍历

{

int v,count=0;

for (v=0;v<G.vexnum;++v)

visited[v]=FALSE;

for (v=0;v<G.vexnum;++v)

if(!visited[v])

{

DFS(G,v);

count++;//对连通分量进行计数

}

printf("无向图G的连通分量的个数为：%d",count);

}

int main()

{

ALGraph f;

CreateGraph(&f);

DFSTraverse(f);

}