**西南石油大学实验报告**

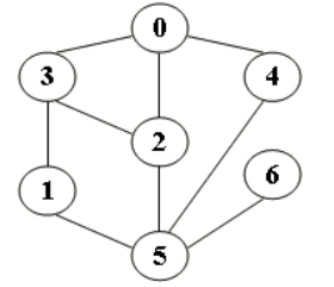
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目编号** | 001063000504 | **项目名称** | 图的应用 | **成绩** |  |
| **专业年级** | **软件工程2022级** | | | **指导教师** | **胡卫东** |
| **姓名** | **李浩楠** | **学号** | **202231060920** | **实验日期** | **2023/6/3** |

1. **实验目的**
2. 掌握图的定义。
3. 掌握图的相关操作。
4. 使用C/C++语言实现相关操作。
5. **实验工具**

PC微机、Windows、PTA或Dev C++。

1. **实验步骤**
2. 最小生成树（普里姆算法）。（程序填空题）
3. 最短路径（迪杰斯特拉算法）。（程序填空题）
4. 小岛计数。

伽马群岛由若干小岛构成，开发者在某些小岛间修建了水上通路，使得群岛大部分连通，但也不排除部分小岛仍为孤岛，创建伽马群岛的地图，例如：



编写函数数一数该群岛共有多少个小岛。

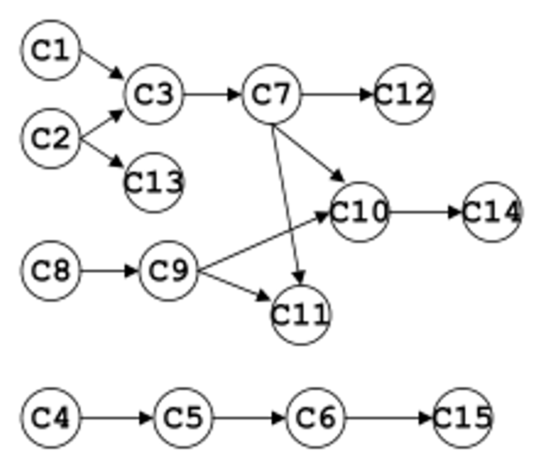
1. 图的广度遍历-邻接表实现。

本题要求实现邻接表存储图的广度优先遍历。

1. 拓扑排序。

已知课程的先后关系如下表所示：

  
可用如下有向图表示课程间的关系：

  
编写函数输出所有课程的拓扑序列。

1. **实验结果**
2. 测试用例：无

代码：#include <iostream>

#define MVNum 100

#define MaxInt 32767

using namespace std;

struct edge{

char adjvex;

int lowcost;

}closedge[MVNum];

typedef struct{

char vexs[MVNum];

int arcs[MVNum][MVNum];

int vexnum,arcnum;

}AMGraph;

int LocateVex(AMGraph G , char v);//实现细节隐藏

int CreateUDN(AMGraph &G);//实现细节隐藏

int Min(AMGraph G){

int i;

int index = -1;

int min = MaxInt;

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

if(closedge[i].lowcost<min

){

min = closedge[i].lowcost;

index = i;

}

}

return index;

}

void MiniSpanTree\_Prim(AMGraph G, char u){

int k , j , i;

char u0 , v0;

k =LocateVex(G, u);

for(j = 0; j < G.vexnum; ++j){

if(j != k){

closedge[j].adjvex = u

;

closedge[j].lowcost = G.arcs[k][j];

}

}

closedge[k].lowcost = 0

;

for(i = 1; i < G.vexnum; ++i){

k = Min(closedge)

;

u0 = closedge[k].adjvex;

v0 = G.vexs[k];

cout << u0 << "->" << v0 << endl;

closedge[k].lowcost = 0

;

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

if(G.arcs[k][j] < closedge[j].lowcost){

closedge[j].adjvex =G.vexs[k]

;

closedge[j].lowcost = G.arcs[k][j]

;

}

}

}

int main(){

AMGraph G;

CreateUDN(G);

char u;

cin >> u;

MiniSpanTree\_Prim(G , u);

return 0;

}

运行结果：无

1. 测试用例：无

代码：#include <iostream>

using namespace std;

#define MaxInt 32767

#define MVNum 100

typedef char VerTexType;

typedef int ArcType;

int \*D=new int[MVNum];

bool \*S=new bool[MVNum];

int \*Path=new int[MVNum];

typedef struct{

VerTexType vexs[MVNum];

ArcType arcs[MVNum][MVNum];

int vexnum,arcnum;

}AMGraph;

int LocateVex(AMGraph G , VerTexType v)//该实现细节隐藏

void CreateUDN(AMGraph &G);//该实现细节隐藏

void ShortestPath\_DIJ(AMGraph G, int v0){

int v , i , w , min;

int n = G.vexnum;

for(v = 0; v < n; ++v){

S[v] = false;

D[v] = G.arcs[v0][v];

if(D[v] < MaxInt) Path [v] = v0;

else Path [v] = -1;

}

S[v0]=true;

D[v0]=0;

for(i = 1;i < n; ++i){

min= MaxInt;

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

If(!S[w]&&D[w]<min)

{

v=w;

min=D[w];

}

S[v]=ture;

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

If(!S[w]&&(D[v]+G.arcs[v][w]<D[w])){

D[w]=D[V]+G,ARCS[V][W];

Path[w]=v;y}

}

}

void DisplayPath(AMGraph G , int begin ,int temp ){

if(Path[temp] != -1){

DisplayPath(G , begin ,Path[temp]);

cout << G.vexs[Path[temp]] << "->";

}

}

int main()

{

AMGraph G;

int i , j ,num\_start , num\_destination;

VerTexType start , destination;

CreateUDN(G);

cin >> start >> destination;

num\_start = LocateVex(G , start);

num\_destination = LocateVex(G , destination);

ShortestPath\_DIJ(G , num\_start);

DisplayPath(G , num\_start , num\_destination);

cout << G.vexs[num\_destination]<<endl;

return 0;

}

运行结果：无

1. 测试用例：ABCD#

A B

A C

# #

代码：int DFS(AMGraph G, int v)//以v为起点遍历图G（v所在的连通分量）

{

int w,res=1;

visited[v]=true;

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

{

if((G.arcs[v][w]!=0)&&(!visited[w]))

res+=DFS(G,w);

}

return res;

}

int DFSTraverse(AMGraph G)//遍历图G

{

int res=0,v;

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

visited[v]=false;

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

{

if(!visited[v])

res+=DFS(G,v);

}

return res;

}

运行结果：4

1. 测试用例：7 9

0 2

0 3

0 4

1 3

1 5

2 3

2 5

4 5

5 6

5

代码：void BFS(ALGraph \*G,int i)

{

visited[i]=TRUE;

printf(" %d",i);

int queue[1000],front=0,rear=0;

queue[rear++]=i;

VNode \*temp;

while(front<rear)

{

temp=G->vertices[queue[front++]].firstarc;

while(temp)

{

int p=temp->data;

if(!visited[p])

{

visited[p]=TRUE;

printf(" %d",p);

queue[rear++]=p;

}

temp=temp->firstarc;

}

}

}

运行结果：BFS from 5: 5 6 4 2 1 0 3

1. 测试用例：6,5

高等数学

计算机导论

C语言

英语

数据结构

数据库

0,2

1,2

1,5

2,3

3,4

代码：void topsort( ALGraph &G)

{

int i,v,w;

int cnt=0;//计数器初始化为0

EdgeNode \*ptr;

SeqStack st;

InitStack\_Sq(st);

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

{

if(G.adjlist[i].Indegree==0)

Push\_SeqStack(st,i);

}

while(!Empty\_Sq(st))

{

Pop\_SeqStack( st, v);//出栈一次，出栈元素放在v中

printf("%s ",G.adjlist[v].vertex);

cnt++;

ptr=G.adjlist[v].firstedge; //ptr指向第一个边结点

while(ptr!=NULL)//只要有边

{

w=ptr->adjvex;

G.adjlist[w].Indegree--;

if(G.adjlist[w].Indegree==0)

Push\_SeqStack(st,w);

ptr=ptr->next;

}

}

if(cnt<G.n)

printf("后续无法输出！\n");

}

运行结果：拓扑序列为：C1 后续无法输出！

1. **实验总结**
2. 图的存储方式有邻接矩阵和邻接链表两种，根据不同的需要选择合适的存储结构。
3. BFS和DFS是图的遍历的基本方法，能遍历任何形式的图，包括非连通图。
4. Prim算法可得到图的最小生成树，Kruskal算发可用于求得最短路径。
5. 拓扑排序可用来解决关键路径问题，也可用于判断图是否存在环，如果图存在环，则拓扑排序无法输出后续拓扑序列。