两景点所有路径模块

### 代码

int v[Maxi][Maxi],visiting[Maxi];

int cnt; //计数器

void pathl(Mgraph \*g,int i,int j,int k) /\*确定路径上第k+1个顶点的序号\*/

{

int s;

if(v[cnt][k]==j)/\*找到一条路径\*/

{

cnt++;

for(s=0;s<k;s++)/\*输出一条路径\*/

{

v[cnt][s]=g->vexs[v[cnt-1][s]].adjvex;

}

v[cnt][s]=g->vexs[v[cnt-1][s]].adjvex;

/\*路径的条数值加1\*/

}

s=0;

while(s<g->n)

{

if(s!=i)/\*保证找到的是简单路径\*/

{

if(g->edges[v[cnt][k]][s]!=FINITY&&visiting[s]==0) /\*当vk与vs之间有边存在且vs未被访问过\*/

{

visiting[s]=1;/\*置访问标志位为1,即已访问的\*/

v[cnt][k+1]=s;/\*将顶点s加入到v数组中\*/

pathl(g,i,j,k+1);/\*递归调用之\*/

visiting[s]=0;/\*重置访问标志位为0，即未访问的，以便该顶点能被重新使用\*/

}

}

s++;

}

}

void SearchAllpath(Mgraph g,int i,int j)/\*查询两个景点间的所有路径\*/

{

int k;

v[0][0]=i;

cnt=0;/\*初始化路径的条数\*/

for(k=0;k<g.n;k++)

{

visiting[k]=0;/\*初始化各顶点的访问标志位，即都为未访问过的\*/

}

pathl(&g,i,j,0);/\*通过调用path函数，找到从vi到vj的所有路径并输出\*//\*用来输出两个景点间的所有路径\*/

}

## 多景点最佳路径模块

### 代码

int st\_All[M];

int top\_all;

int print\_floyd\_All(Mgraph g,int bgn,int fnl,path\_f p,dist\_f d,int top,st\_f s)

{

int pre;

s[top]=fnl;

pre=p[bgn][fnl];

while(pre!=-1)

{

s[++top]=pre;

pre=p[bgn][pre];

}

s[++top]=bgn;

return top;

}

int OverFloyd(Mgraph g,int AllPath[M],int Max,path\_f p,dist\_f d,st\_f s)

{

int j=Max;

int i=Max-1;

int sum=0,top=0;

while(i>=0)

{

top=print\_floyd\_All(g,AllPath[i],AllPath[j],p,d,top,s);

sum=sum+d[AllPath[i--]][AllPath[j--]];

}

return sum;