1. Dijikstra算法+DFS，最短路径（单源最短路径）

1）数据结构

图用邻接矩阵int G[maxn][maxn]存储；

源点到目标点的距离用数组int dist[maxn]存储；

int used[maxn]记录节点是否已经计算过；

Int pathnum[num]记录到第num个节点的最短路径有几条；

vector<int> pre记录最短路径中节点的前一个节点；

vector<int> path,tempath;记录最短路径，临时最短路径。

初始化：

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

{

G[i][j] = G[j][i] = INF;

}

used[i] = 0;

dist[i] = INF;

pathNum[i] = 0;

1. 算法

求最短路径长度。

void dij(int s){ //s是起始点。

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

{

int minx = maxn, p;

//挑选最短边

for (int j = 0; j < n; j++) if (minx>g[s][j] && !used[j]){ p = j; minx = g[s][j]; } if (minx==maxn)

return;

used[p] = 1; //p是当前与源点项链最短的边。

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

{

if (!used[j]&&g[p][j]!=maxn) //点j的最短路径未确定且p与j有边

{

if (g[s][p]+g[p][j]<g[s][j]) //过点p到j的路径长度与原路径长度比较

{

g[s][j] = g[s][p] + g[p][j];

pre[j].clear();

pre[j].push\_back(p);

pathnum[j]=pathnum[p];

}

else if (g[s][p]+g[p][j]==g[s][j])

{

pre[j].push\_back(p);

pathnum[j]=pathnum[j]+pathnum[p];

}

}

}

}

}

Dfs求源点到个目标点的路径

void dfs(int d){ //d是目标点

if (d==s) //从pre中不断取值，直到回到源点

{

cost = 0;

tem.push\_back(d);

for (int i = 0; i < tem.size()-1; i++)

{

cost += w[tem[i]][tem[i + 1]]; //统计路径上权值

}

if (cost<mincost)

{

mincost = cost;

path = tem; //多条路径选择花费最短的一条路径

}

tem.pop\_back();

return; //回到源点就结束。

}

tem.push\_back(d); //未回到源点将点加入路径

for (int i = 0; i < pre[d].size(); i++)

{

dfs(pre[d][i]); //递归求解路径中上一个节点

}

tem.pop\_back(); // !!!当前点的所有前驱节点都访问过了， //从路径中删除这个节点，然后回到上一级递 //归，访问上一个节点的另一个前驱节点。

}