Dijkstra Algorithm

Dijkstra 算法

INFORMATION ENGINEERING UNIVERSITY

中国人民解放军战略支援部队信息工程大学—李翔讲师

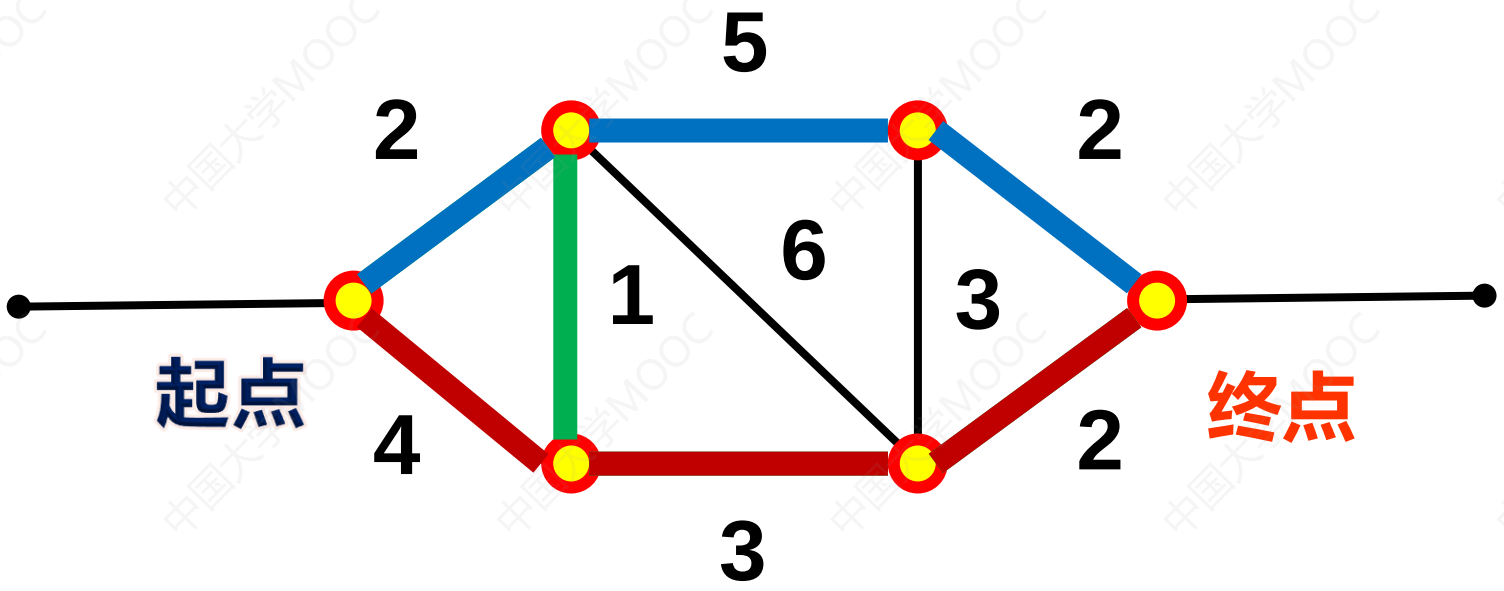
PLA Strategic Support Force Information Engineering University——Lecturer. Xiang Li

- 长期从事地理信息系统与地理空间数据库的教学与科研工作。
- 研究方向：地理信息辅助定位、网络空间数据建模等。讲授课程包括《地理空间数据库》、《地理信息数据处理程序设计》、《地理信息系统设计与开发》等。
- 获全国高校GIS专业青年教师讲课竞赛特等奖，获战略支援部队讲课比赛三等奖，主持和参与国家“十三五”重点研发计划、河南省科技攻关、部门科研课题等6项，发表学术和教学论文20余篇，授权发明专利5项，软著2项。

Dijkstra 算法

Dijkstra Algorithm

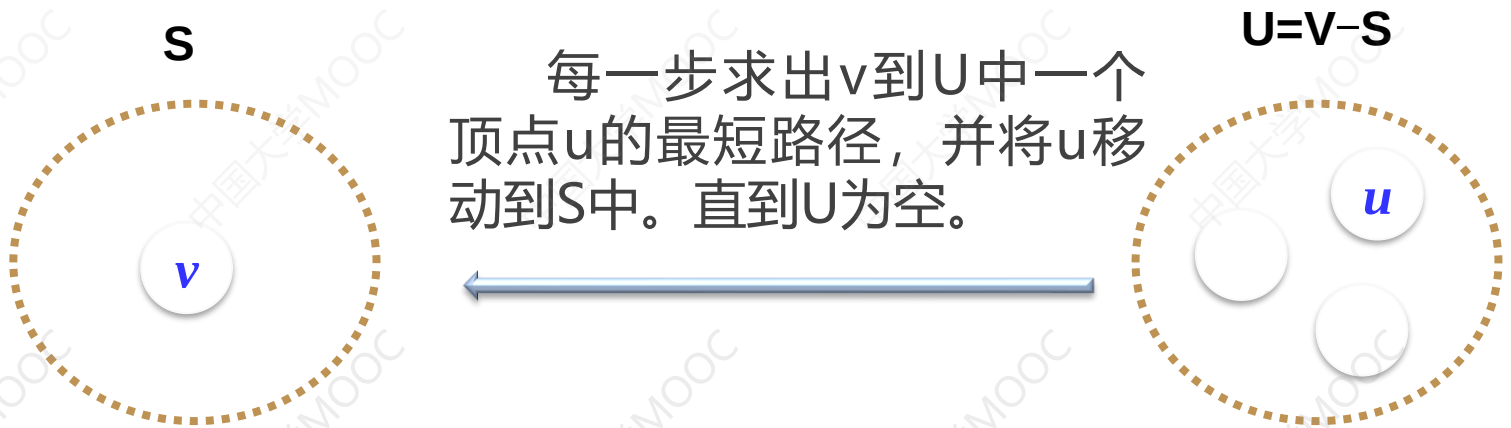
网络中求最短路径



Dijkstra 算法

Dijkstra Algorithm

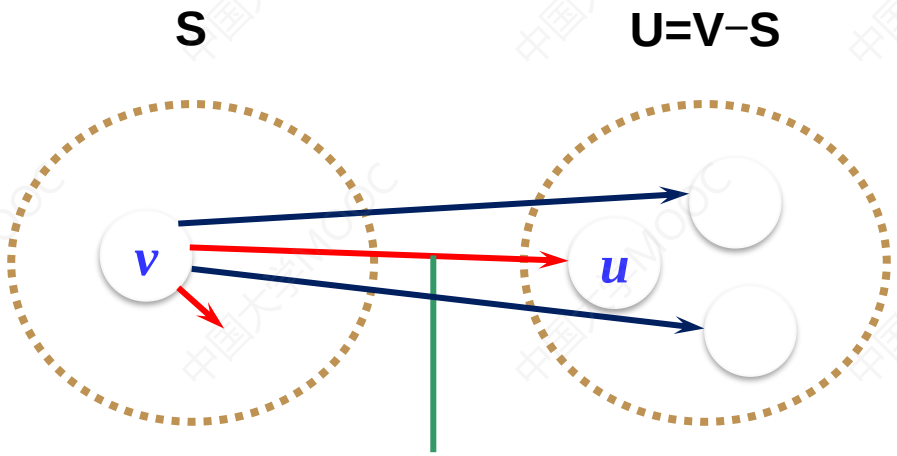
- 第1组为已求出最短路径的顶点集合（用S表示，初始时S中只有一个源点，以后每求得一条最短路径*v*，…，*u*，就将*u*加入到集合S中，直到全部顶点都加入到S中，算法就结束了）。
- 第2组为其余未求出最短路径的顶点集合（用U表示）。



Dijkstra 算法

Dijkstra Algorithm

从U中选取一个距离v最小的顶点u，把u加入S中（该选定的距离就是**v ⇨ u**的最短路径长度）。

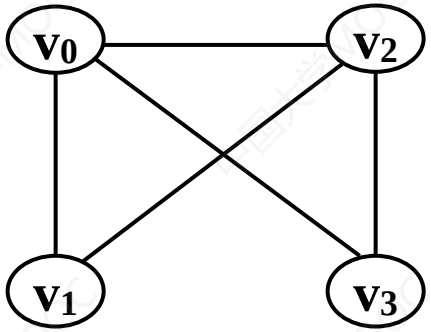


v与U中顶点u的边最小

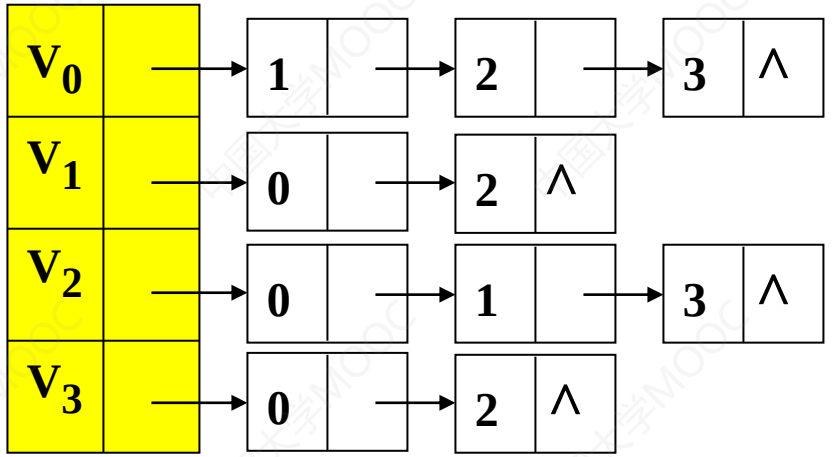
Dijkstra 算法

Dijkstra Algorithm

图的邻接表存储结构



邻 接 表

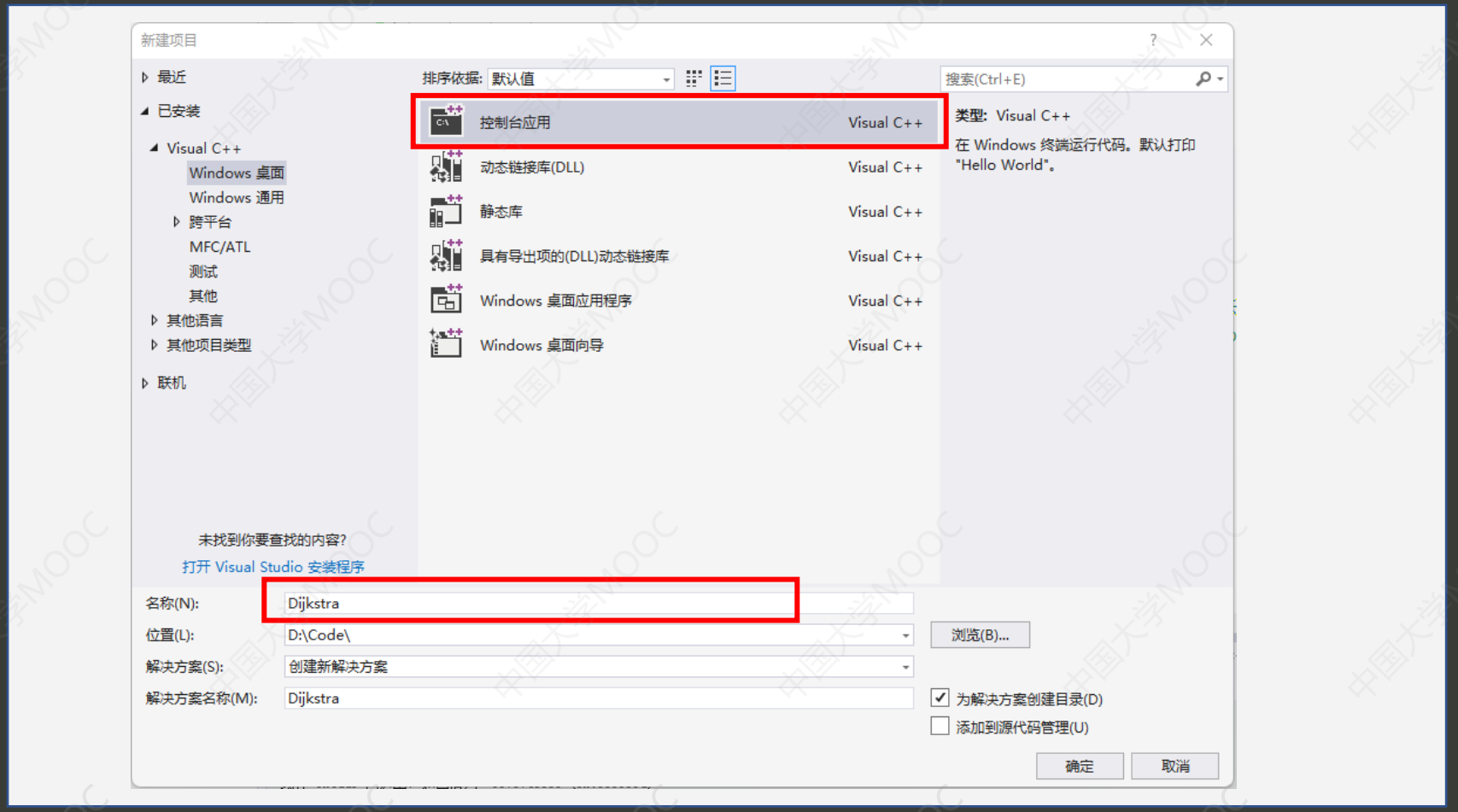


顶 点

边

Dijkstra 算法

Dijkstra Algorithm



Dijkstra 算法

Dijkstra Algorithm

在Dijkstra.cpp文件里，定义顶点结点结构体：

```
//定义顶点结点结构体
typedef struct vertexnode {
    vertexType boolval; /* 顶点结点域，这里存放的是该结点是否
                           找到其距源顶点最短路径的标记，
                           若找到最短路径，则该值为1，否则该值为0 */
    EdgeNode *firstedge; //边表头指针
} VertexNode;
```

Dijkstra 算法

Dijkstra Algorithm

在Dijkstra.cpp文件里，定义边表结点结构体：

```
//定义边表结点结构体
typedef struct edgenode {
    int adjvertex; //边表结点域
    infoType info; //边表结点权值，这里存放的是其父结点到该结点的距离
    struct edgenode *next; //指向下一个邻接点的指针域
} EdgeNode;
```

Dijkstra 算法

Dijkstra Algorithm

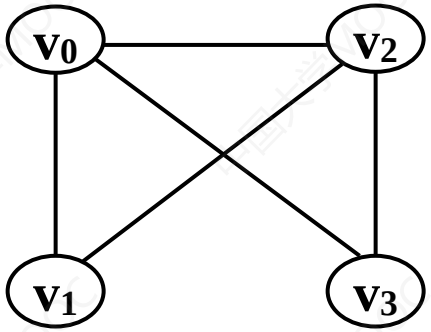
在Dijkstra.cpp文件里，定义邻接表：

```
typedef struct {
    VertexNode adjlist[MAX]; /*邻接表*/
    int vertexNum; /*顶点数*/
    int edgeNum; /*边数*/
} ALGraph; //adjacency list graph:邻接表
```

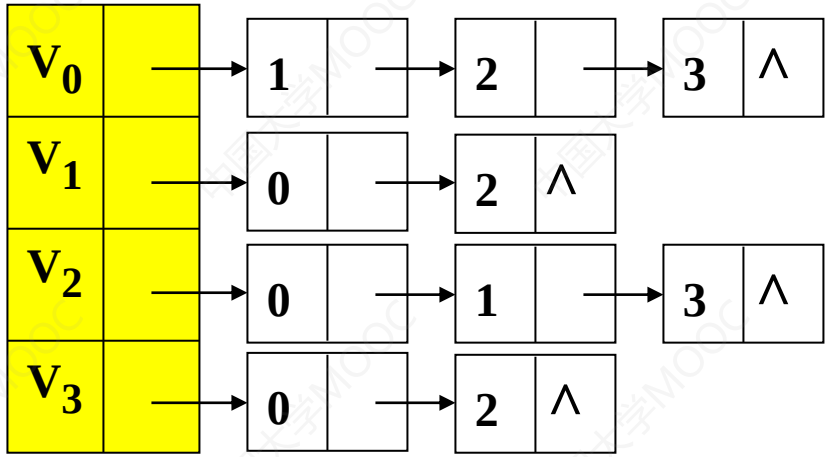
Dijkstra 算法

Dijkstra Algorithm

图的邻接表存储结构



邻 接 表



边上信息

邻接点域

adjvextex	info	next

指针域

Dijkstra 算法

Dijkstra Algorithm

在Dijkstra.cpp文件里，构建邻接表：

```
/*****************************************************************
函数名称：CreateGraph
函数功能：创建邻接表
输入：顶点数vertexNum，边数edgeNum
输出：指向已创建好的邻接表的指针
*****************************************************************/
ALGraph* CreateGraph(int vertexNum, int edgeNum) { ... }
```

Dijkstra 算法

Dijkstra Algorithm

在Dijkstra.cpp文件里，建立顶点表：

```
//建立顶点表
for (k = 0; k < G->vertexNum; k++)
{
    G->adjlist[k].boolval = 0; /*boolval值判断该结点到源结点的距离是否是最短距离，
                                 是1表示已达最短距离，是0表示还没有达最短距离*/
    G->adjlist[k].firstedge = NULL;
}
```

Dijkstra 算法

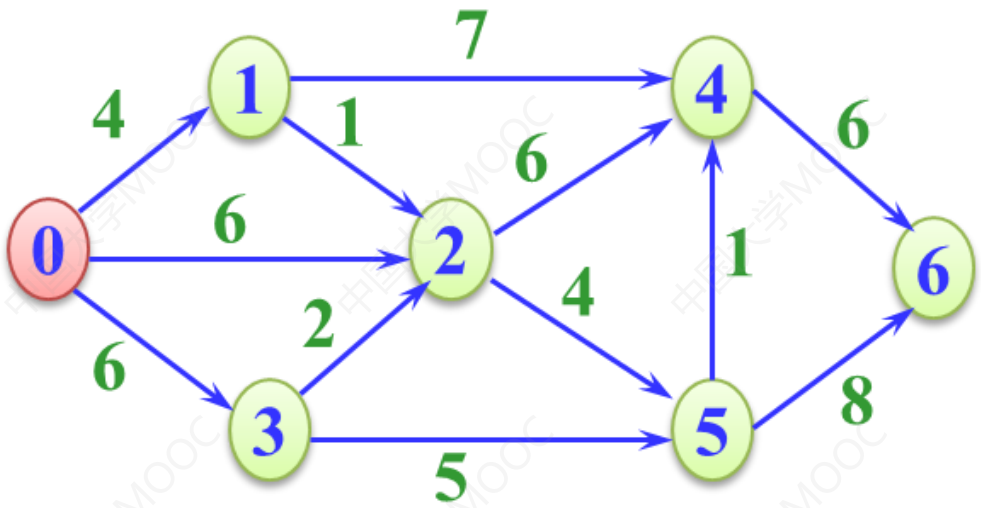
Dijkstra Algorithm

在Dijkstra.cpp文件里，建立边表：

```
//建立边表
printf("请输入顶点、其邻接顶点和权值信息：\n");
for (k = 0; k < G->edgeNum; k++) {
    int i, j;
    infoType info;

    //<i,j>表现的是边的关系，有多少对<i,j>就有多少边，所以for循环次数为G->edgeNum
    scanf("%d,%d,%d", &i, &j, &info);
    if (i != j) {
        p = (EdgeNode *)malloc(sizeof(EdgeNode));
        p->next = G->adjlist[i].firstedge;
        G->adjlist[i].firstedge = p;
        p->adjvertex = j;
        p->info = info;
    }
```

Dijkstra 算法

Dijkstra Algorithm



起始顶点——邻接顶点——边的权值

① 0, 1, 4	⑦ 2, 5, 4
② 0, 2, 6	⑧ 3, 2, 2
③ 0, 3, 6	⑨ 3, 5, 5
④ 1, 2, 1	⑩ 4, 6, 6
⑤ 1, 4, 7	⑪ 5, 4, 1
⑥ 2, 4, 6	⑫ 5, 6, 8

0 ×7 ⟶ ×12

Dijkstra 算法

Dijkstra Algorithm

在Dijkstra.cpp文件里，构建dijkstra函数：

```
/*************************************************************
函数名称：dijkstra（迪杰斯特拉/迪斯奎诺）
函数功能：实现迪杰斯特拉算法，找出每个顶点到源定点u的最短距离
输入：邻接表指针G，源顶点u，记录每个顶点到源顶点的最短距离的数组d[],
输出：记录每个顶点到源顶点的最短距离的数组d[]，到源顶点的最短路径上的
*************************************************************/
void dijkstra(ALGraph *G, int u, int d[], int p[]) { ... }
```

Dijkstra 算法

Dijkstra Algorithm

dijkstra函数——初始化

（1）S只包含源点即S={v}，v的最短路径为0。U包含除v外的其他顶点，U中顶点i距离为边上的权值（若v与i有边<v，i>）或∞（若i不是v的出边邻接点）。

```
//初始化参数
for (i = 0; i < G->vertexNum; i++)
{
    //G->adjlist[i].boolval = 0;
    d[i] = MAX_FLOAT_NUM;
    p[i] = -1;
}
```

Dijkstra 算法

Dijkstra Algorithm

dijkstra函数——更新最短路径

（2）从U中选取一个距离v最小的顶点u，把u加入S中（该选定的距离就是v ⇨ u的最短路径长度）。

```
//更新源顶点直接子结点到源结点的最短距离
if (!(pnode = G->adjlist[u].firstedge))
{
    return;
}

while (pnode)
{
    d[pnode->adjvertex] = pnode->info;
    p[pnode->adjvertex] = u;
    pnode = pnode->next;
}
```

```
//更新所有除源结点外的结点到源结点的最短距离
for (i = 1; i < G->vertexNum; i ++) {
    int min = MAX_FLOAT_NUM;
    t = u;
```

Dijkstra 算法

Dijkstra Algorithm

dijkstra函数——更新源点

（3）以u为新考虑的中间点，修改U中各顶点j的最短路径长度：若从源点v到顶点j的最短路径长度（经过顶点u）比原来最短路径长度（不经过顶点u）短，则修改顶点j的最短路径长度。

```
//在所有结点中找出一个距离源结点距离最小的一个结点
for (j = 0; j < G->vertexNum; j++)
{
    if (G->adjlist[j].boolval != 1 && min > d[j])
    {
        t = j;
        min = d[j];
    }
}
```

Dijkstra 算法

Dijkstra Algorithm

dijkstra函数——循环步骤

（4）重复步骤（2）和（3）直到所有顶点都包含在S中。

```
/*
    找到一个距离源结点距离最小的结点时，将该结点看成是一个源结点，更新
    它的所有直接子结点到源结点u的最短距离d[i]，然后再去找一个距离源结点
    距离最小的结点，如此反复的更新所有结点到源结点的最短距离。
*/
pnode = G->adjlist[t].firstedge;

while (pnode) {
    if ((G->adjlist[pnode->adjvertex].boolval != 1) && (d[pnode->adjvertex] > (d[t] + pnode->info)))
    {
        d[pnode->adjvertex] = d[t] + pnode->info;
        p[pnode->adjvertex] = t;
    }
    pnode = pnode->next;
```

Dijkstra 算法

Dijkstra Algorithm

main函数——输入

```
printf("请输入顶点个数和边个数：\n");
scanf("%d,%d", &vertexNum, &edgeNum);
printf("\n");
```

```
printf("请输入源结点：\n");
scanf("%d", &u);
printf("\n");
```

Dijkstra 算法

Dijkstra Algorithm

main函数——输出

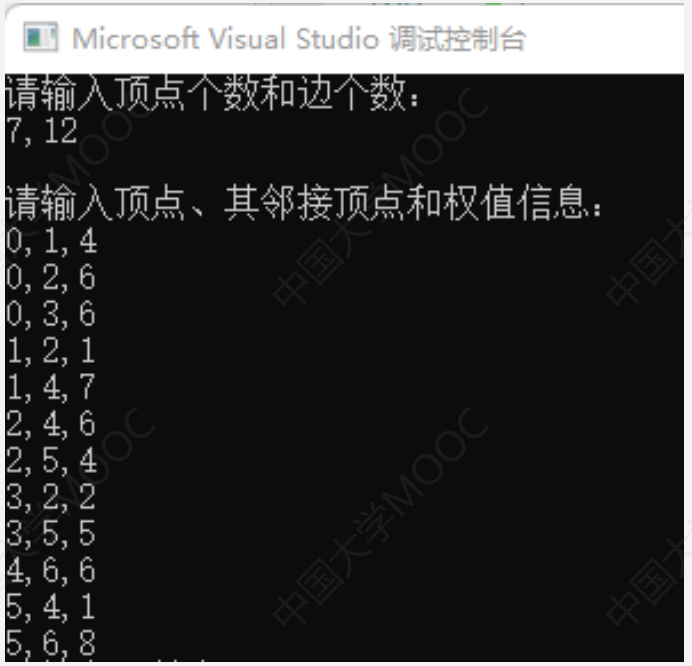
```
printf("各点到源顶点%d的距离：\n", u);
for (i = 0; i < vertexNum; i++)
{
    printf("顶点%d距离源顶点%d的距离: %d\t", i, u, d[i]);
    printf("\n");

    printf("所走最短路径为：\t");
    j = 0;
    tmp[j++] = i;
    t = p[i];
    while (t != -1) {
        tmp[j++] = t;
        t = p[t];
    }

    for (k = --j; k >= 0; k--) {
        printf("%d\t", tmp[k]);
    }

    printf("\n\n");
}
```

Dijkstra 算法

Dijkstra Algorithm

main函数——调用dijkstra函数

```
//调用迪杰斯特拉算法函数
dijkstra(G, u, d, p);
```



Dijkstra 算法

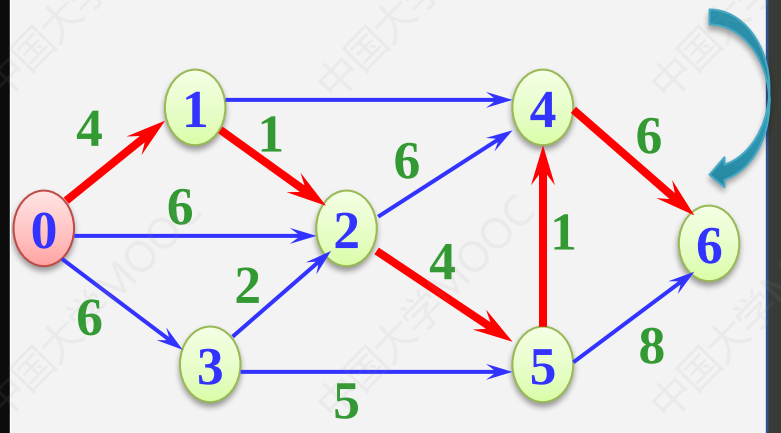
Dijkstra Algorithm

输出结果

```
Microsoft Visual Studio 调试控制台
请输入源结点：
0

各点到源顶点0的距离：
顶点0距离源顶点0的距离：0
所走最短路径为：        0

顶点1距离源顶点0的距离：4
所走最短路径为：        0       1

顶点2距离源顶点0的距离：5
所走最短路径为：        0       1       2

顶点3距离源顶点0的距离：6
所走最短路径为：        0       3

顶点4距离源顶点0的距离：10
所走最短路径为：        0       1       2       5       4

顶点5距离源顶点0的距离：9
所走最短路径为：        0       1       2       5

顶点6距离源顶点0的距离  16
所走最短路径为：        0       1       2       5       4       6


D:\Code\Djistra\Debug\Dijkstra.exe (进程 33388)已退出，返回代码为：0。
若要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

最短路径为：

0→1 →2 →5 →4→6



谢 谢 观 看