

[首页](#) [资讯](#) [精华](#) [论坛](#) [问答](#) [博客](#) [专栏](#) [群组](#) [更多](#) ▼
[您还未登录！](#) [登录](#) [注册](#)

知识在于积累


- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

最短路径-Floyd

博客分类：

- [数据结构](#)

最短路径Floyd弗洛伊德数据结构c++

本文章已收录于： [算法与数据结构](#)

之前我们接触学习了Dijkstra算法求解一个顶点到其他各个顶点的最短路径和距离，但如果我们想知道每一对顶点的最短路径和距离时，可以通过以每一个顶点作为源点循环求出每对顶点之间的最小距离。除此之外，我们可以利用本篇博客即将学习的弗洛伊德(Floyd)算法来求两顶点之间的最短距离。

弗洛伊德(Floyd)算法

1) 算法思想原理：

从任意节点i到任意节点j的最短路径不外乎2种可能，1是直接从i到j，2是从i经过若干个节点k到j。所以，我们假设Dis(i, j)为节点u到节点v的最短路径的距离，对于每一个节点k，我们检查Dis(i, k) + Dis(k, j) < Dis(i, j)是否成立，如果成立，证明从i到k再到j的路径比i直接到j的路径短，我们便设置Dis(i, j) = Dis(i, k) + Dis(k, j)，这样一来，当我们遍历完所有节点k，Dis(i, j)中记录的便是i到j的最短路径的距离。

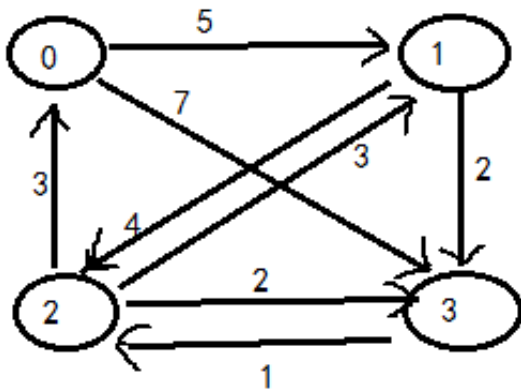
2). 算法描述：

a. 从任意一条单边路径开始。所有两点之间的距离是边的权，如果两点之间没有边相连，则权为无穷大。

b. 对于每一对顶点 u 和 v ，看看是否存在一个顶点 w 使得从 u 到 w 再到 v 比已知的路径更短。如果是更新它。

3) 具体实现步骤

要操作的有向图如图所示：



用邻接矩阵表示为：

Cpp代码  

```

1. #define INF 99999 //表示不可到达
2.
3. #define MAXSIZE 4 //表示图的结点数
4.
5. //邻接矩阵存储图的信息
6. int map[MAXSIZE][MAXSIZE]={
7.     {0, 5, INF, 7},
8.     {INF, 0, 4, 2},
9.     {3, 3, 0, 2},
10.    {INF, INF, 1, 0}
11. };
  
```

定义

$A[\text{MAXSIZE}][\text{MAXSIZE}]$: $A[i][j]$ 表示当前顶点 i 到 j 的最短距离

$\text{path}[\text{MAXSIZE}][\text{MAXSIZE}]$: 保存最短路径

Floyd算法过程矩阵的计算----十字交叉法

先初始化2个数组：

Cpp代码  

```

1. //数据初始化
2.     for(int i=0;i<MAXSIZE;i++)
3.     {
4.         for(int j=0;j<MAXSIZE;j++)
5.         {
6.             A[i][j]=map[i][j];
7.             path[i][j]=-1;//初始化为-1
8.         }
9.     }

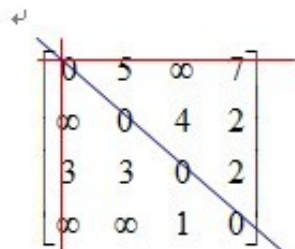
```

即得到:

$$A_{-1} = \begin{bmatrix} 0 & 5 & \infty & 7 \\ \infty & 0 & 4 & 2 \\ 3 & 3 & 0 & 2 \\ \infty & \infty & 1 & 0 \end{bmatrix}$$

$$Path_{-1} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

1) 使用十字交叉法, 划去第0行和第0列以及左对角线, 即

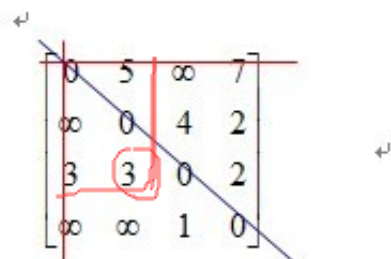


$$\begin{bmatrix} 0 & 5 & \infty & 7 \\ \infty & 0 & 4 & 2 \\ 3 & 3 & 0 & 2 \\ \infty & \infty & 1 & 0 \end{bmatrix}$$

此时不在这三条线上的数据有: $A[1][2]=4; A[1][3]=2; A[2][3]=2; A[2][1]=3$ 等6个数。

此时根据 $A^k(i, j) = \min(A^{k-1}(i, j), A^{k-1}(i, k) + A^{k-1}(k, j))$ 对比看是否要数据更新

例如看 $A[2][1]=3$ 这个数是否要更新。



$$\begin{bmatrix} 0 & 5 & \infty & 7 \\ \infty & 0 & 4 & 2 \\ 3 & 3 & 0 & 2 \\ \infty & \infty & 1 & 0 \end{bmatrix}$$

此时 $A[0][1]+A[2][0]=8 > A[2][1]=3$

所以不用更新, 其他5个数都是这样判断, 会发现都不用更新

1) 使用十字交叉法, 划去第1行和第1列以及左对角线, 即

$$\begin{bmatrix}
 0 & 5 & \infty & 7 \\
 \infty & 0 & 4 & 2 \\
 3 & 3 & 0 & 2 \\
 \infty & \infty & 1 & 0
 \end{bmatrix}$$

此时不在这3条线上的数据依次是： $A[0][2]$, $A[0][3]$, $A[2][0]$, $A[2][3]$, $A[3][0]$, $A[3][2]$

我们来看数据 $A[0][2]$ 。

$$\begin{bmatrix}
 0 & 5 & \infty & 7 \\
 \infty & 0 & 4 & 2 \\
 3 & 3 & 0 & 2 \\
 \infty & \infty & 1 & 0
 \end{bmatrix}$$

发现

$A[0][2] > A[0][1] + A[1][2] = 5 + 4 = 9$ (图中画出矩形顶点的和, 即 $A[0][2]$ 附近2个顶点的和, 不是对角线那个顶点);

此时修改 $A[0][2] = 9$, $path[0][2] = 1$ (即划去的行号和列号, 也是3条线交点的坐标)


按照此方法检查其他剩下的5个数, 最后得到

$$\begin{bmatrix}
 0 & 5 & \infty(9) & 7 \\
 \infty & 0 & 4 & 2 \\
 3 & 3 & 0 & 2 \\
 \infty & \infty & 1 & 0
 \end{bmatrix}
 \text{ 所以有 } A_1 = \begin{bmatrix}
 0 & 5 & 9 & 7 \\
 \infty & 0 & 4 & 2 \\
 3 & 3 & 0 & 2 \\
 \infty & \infty & 1 & 0
 \end{bmatrix}
 \text{ } Path_1 = \begin{bmatrix}
 -1 & -1 & 1 & -1 \\
 -1 & -1 & -1 & -1 \\
 -1 & -1 & -1 & -1 \\
 -1 & -1 & -1 & -1
 \end{bmatrix}$$

以此类推。最后得到:

$$\begin{bmatrix}
 0 & 5 & 9(8) & 7 \\
 7(6) & 0 & 4(3) & 2 \\
 3 & 3 & 0 & 2 \\
 4 & 4 & 1 & 0
 \end{bmatrix}
 \text{ 所以有 } A_3 = \begin{bmatrix}
 0 & 5 & 8 & 7 \\
 6 & 0 & 3 & 2 \\
 3 & 3 & 0 & 2 \\
 4 & 4 & 1 & 0
 \end{bmatrix}
 \text{ } Path_3 = \begin{bmatrix}
 -1 & -1 & 3 & -1 \\
 3 & -1 & 3 & -1 \\
 -1 & -1 & -1 & -1 \\
 2 & 2 & -1 & -1
 \end{bmatrix}$$

理解清楚步骤后, 写出Floyd算法代码为:

Cpp代码  

```

1. //弗洛伊德算法
2. void Floyd()
3. {
4.     int path[MAXSIZE][MAXSIZE]; //保存最短路径
5.
6.     int A[MAXSIZE][MAXSIZE]; //a[i][j]表示当前顶点i到j的最短距离
7.
8.     //数据初始化
9.     for(int i=0; i<MAXSIZE; i++)
10.    {
11.        for(int j=0; j<MAXSIZE; j++)
12.        {
13.            A[i][j]=map[i][j];
14.            path[i][j]=-1; //初始化为-1
15.        }
16.    }
17.
18.    for(int diagonal=0; diagonal<MAXSIZE; diagonal++) //左对角线
19.    {
20.        for(int k=0; k<MAXSIZE; k++) //行
21.        {
22.            if(k!=diagonal) //除去此行所有的点
23.                for(int j=0; j<MAXSIZE; j++) //列
24.                {
25.                    if(j!=diagonal) //除去此列所有的点
26.                    {
27.                        if(k!=j) //除去对角线的点
28.                        {
29.                            if(A[k][j]>A[diagonal][j]+A[k]
30.                                [diagonal]) //满足条件
31.                                {
32.                                    A[k][j]=A[diagonal]
33.                                        [j]+A[k][diagonal];
34.                                    path[k]
35.                                        [j]=diagonal;
36.                                }
37.                            }
38.                        }
39.                    }
40.                }
41.            }
42.        }
43.    }
44.    }
45.    }
46.    }
47.    }
48.    }
49.    }
50.    }
51.    }
52.    }
53.    }
54.    }
55.    }
56.    }
57.    }
58.    }
59.    }
60.    }
61.    }
62.    }
63.    }
64.    }
65.    }
66.    }
67.    }
68.    }
69.    }
70.    }
71.    }
72.    }
73.    }
74.    }
75.    }
76.    }
77.    }
78.    }
79.    }
80.    }
81.    }
82.    }
83.    }
84.    }
85.    }
86.    }
87.    }
88.    }
89.    }
90.    }
91.    }
92.    }
93.    }
94.    }
95.    }
96.    }
97.    }
98.    }
99.    }
100.   }
```

得到A[MAXSIZE][MAXSIZE]和path[]数组后。

A[i][j]: 表示从顶点i到顶点j的最短距离。

而最短路径还要通过path[]数组计算得来。计算方法如下:

$$Path_3 = \begin{bmatrix} -1 & -1 & 3 & -1 \\ 3 & -1 & 3 & -1 \\ -1 & -1 & -1 & -1 \\ 2 & 2 & -1 & -1 \end{bmatrix}$$

例如我们求解顶点3到顶点1的最小距离和路径：

最小距离：A[3][1]=4

最短路径：

path[3][1]=2;

path[2][1]=-1(一旦值为-1，停止计算)

所以顶点1前面经过的是顶点2，

即最后路径为：

3->2->1;

j结果显示代码为：

Cpp代码  

```

1. //结果输出：
2.     for(int i=0;i<MAXSIZE;i++)
3.     {
4.         for(int j=0;j<MAXSIZE;j++)
5.         {
6.             if(A[i][j]==INF)
7.                 cout<<"从顶点"<<i<<"到顶点"<<j<<"不存在路径"<<endl;
8.             else
9.             {
10.                 cout<<"从顶点"<<i<<"到顶点"<<j<<"最短距离为： "<<A[i][j]
11.                 <<" 其路径为： ";
12.                 vector<int>temp;
13.                 temp.insert(temp.begin(),j);//把终点插入
14.                 int ok1=i,ok2=j;
15.                 while(true)
16.                 {
17.                     ok1=path[ok1][ok2];
18.                     if(ok1==-1)
19.                         break;
20.                     temp.insert(temp.begin(),ok1);
21.                 }
22.                 temp.insert(temp.begin(),i);//把起点插入
23.                 for(int z=0;z<temp.size();z++)
24.                     cout<<temp[z]<<" ";
25.                 cout<<endl;
26.             }
27.         }
28.     }
29. }
30. 
```

最终程序结果：



```

E:\VS2012\Projects\Floyd\Debug\Floyd.exe
从顶点0到顶点0最短距离为: 0 其路径为: 0 0
从顶点0到顶点1最短距离为: 5 其路径为: 0 1
从顶点0到顶点2最短距离为: 8 其路径为: 0 3 2
从顶点0到顶点3最短距离为: 7 其路径为: 0 3
从顶点1到顶点0最短距离为: 6 其路径为: 1 2 3 0
从顶点1到顶点1最短距离为: 0 其路径为: 1 1
从顶点1到顶点2最短距离为: 3 其路径为: 1 3 2
从顶点1到顶点3最短距离为: 2 其路径为: 1 3
从顶点2到顶点0最短距离为: 3 其路径为: 2 0
从顶点2到顶点1最短距离为: 3 其路径为: 2 1
从顶点2到顶点2最短距离为: 0 其路径为: 2 2
从顶点2到顶点3最短距离为: 2 其路径为: 2 3
从顶点3到顶点0最短距离为: 4 其路径为: 3 2 0
从顶点3到顶点1最短距离为: 4 其路径为: 3 2 1
从顶点3到顶点2最短距离为: 1 其路径为: 3 2
从顶点3到顶点3最短距离为: 0 其路径为: 3 3
  
```

附上源码地址: <https://github.com/longpo/algorithm/tree/master/Floyd>

- [查看图片附件](#)

3
顶
1
踩

分享到:  

[查找算法--顺序表查找](#) | [最短路径-Dijkstra](#)

- 2015-03-28 16:41
- 浏览 1334
- [评论\(0\)](#)
- 分类: [编程语言](#)
- [相关推荐](#)

相关知识库:  [算法与数据结构知识库](#)

评论



发表评论



[您还没有登录, 请您登录后再发表评论](#)



hm4123660

- 浏览：70326 次
- 性别：
- 来自：广州
-  我现在离线

最近访客

[更多访客>>](#)



[lu87689](#)



[shiyingzhan](#)

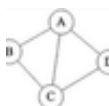


[saulyp](#)



[liugan5371375](#)

博客专栏



[数据结构](#)

浏览量：17875

文章分类

- [全部博客 \(99\)](#)
- [Java \(26\)](#)
- [Android \(11\)](#)
- [数据结构 \(17\)](#)

- [设计模式 \(6\)](#)
- [JNI \(5\)](#)
- [XML \(3\)](#)
- [Jboss \(3\)](#)
- [Extjs4 \(3\)](#)
- [JS \(6\)](#)
- [数据库 \(5\)](#)
- [div+css \(3\)](#)
- [J2EE \(5\)](#)
- [Nodejs \(5\)](#)
- [生活杂谈 \(0\)](#)
- [cocos2d \(1\)](#)

社区版块

- [我的资讯 \(0\)](#)
- [我的论坛 \(1\)](#)
- [我的问答 \(0\)](#)

存档分类

- [2015-08 \(1\)](#)
- [2015-06 \(6\)](#)
- [2015-05 \(11\)](#)
- [更多存档...](#)

评论排行榜

- [据说一半以上的java程序员会出错的题](#)
- [装饰者设计模式](#)
- [java内部类](#)
- [A星寻路算法](#)
- [java的类加载器ClassLoader](#)

最新评论

- [comsci:](#) 拓扑分析算法.....寻径与导 ...
[A星寻路算法](#)
- [manxisuo:](#) 感谢博主，好文章。
[java的类加载器ClassLoader](#)
- [User_Java:](#) 类的静态变量初始化顺序与其声明的顺序有关。自增操

作都执行后保存 ...

[据说一半以上的java程序员会出错的题](#)

- [flashsnow](#): 在公司写这样的代码是要遭雷劈的But, 为了理解ClassLoa ...

[据说一半以上的java程序员会出错的题](#)

- [hm4123660](#): 求求你帮帮我 写道为什么初始化count1没有赋予初始值, 却把 ...

[据说一半以上的java程序员会出错的题](#)

声明: ITeye文章版权属于作者, 受法律保护。没有作者书面许可不得转载。若作者同意转载, 必须以超链接形式标明文章原始出处和作者。

© 2003-2016 ITeye.com. All rights reserved. [京ICP证110151号 京公网安备110105010620]