# Floyd算法的证明

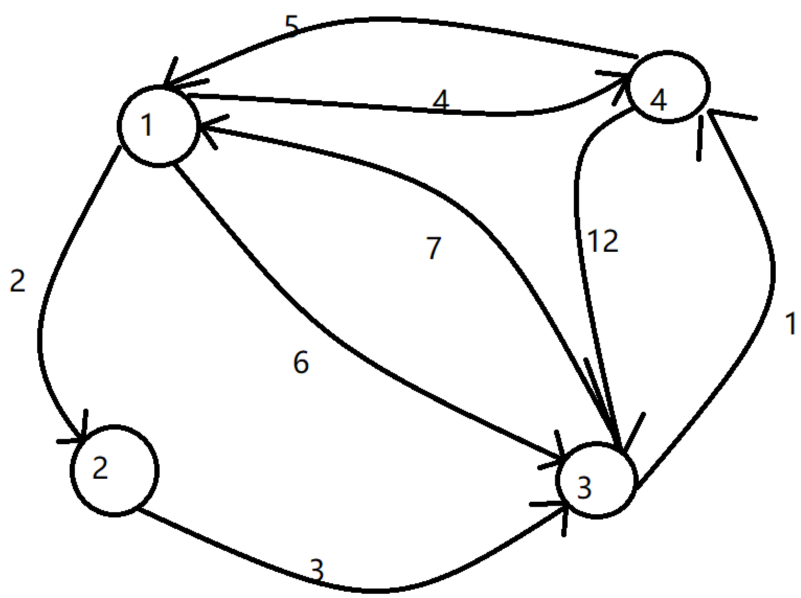
`Floyd`算法是一个动态规划的过程，我们假设有这么一个有向无负环的图，其中有编号为`1 ~ n`个顶点，`Floyd`算法就是求解任意`i`，`j`两个顶点最短路径的算法他的证明过程也就是算法的求解过程

我们令`d(k, i, j)`表示从`i`到`j`经过的顶点编号不超过`k`的最短路径对应的路径长度，也就是从`i`到`j`经过的顶点编号为`1 ~ k`，那么：

当`k = 0`的时候，`d(0, i, j)`是顶点`i`直接到达`j`的路径，我们在初始输入的时候获得`k = 0`的时候的各种值

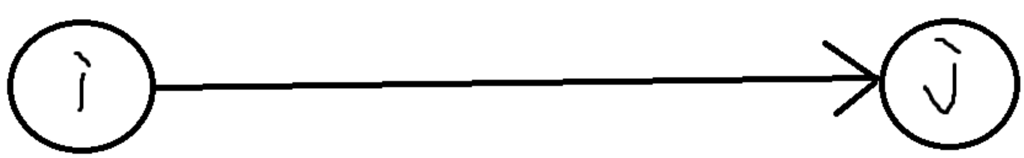
当`k = n`的时候，`d(n, i, j)`就是最后的结果了，也就是顶点`i`经过顶点编号`1 ~ n`到达`j`的最短路径

假设我们有下面这么一张图：

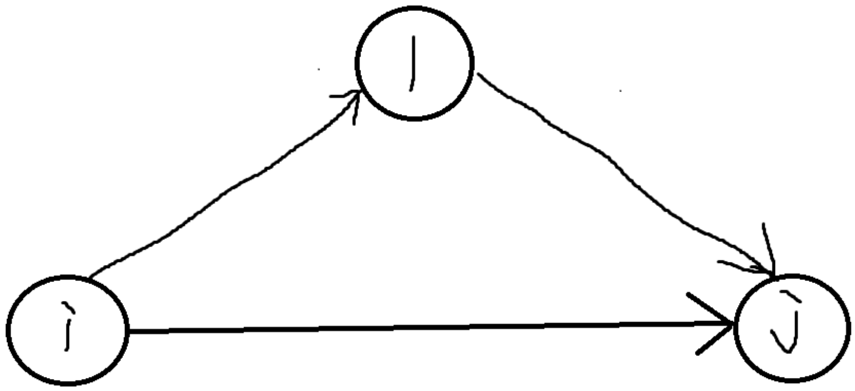



在`k = 0`的时候的时候我们可以得到：`d(0, 1, 2) = 2, d(0, 1, 1) = 0, d(0, 2, 1) = 无穷， d(0, 2, 4) = 无穷`

这也就是一开始读入每条边的权值的时候，我们在计算从`k`从`1 ~ n`的过程中可以看作慢慢加点的过程，比如`k = 0`的时候就代表任意两点之间没有点：




此时的`d(0, i, j)`就是读入的过程，当`k = 1`的时候就加入了顶点为`1`这个点：




那么很显然，从`i`到`j`就有两种的可能路线，经过`1`和不经过`1`，经过`1`的路线有两段，从`i`经过其他的点到`1`，加上从`1`经过其他的点到`j`，在此处其他的点没有，也就是`0`，即`d(1, i, j) = min[d(0, i, j), d(0, i, 1)+d(0, 1, j)]`，在这里我们双重循环遍历`i, j`也就求出来了`k = 1`的时候`d(1, i, j)`的值

同理当`k = 2`的时候，我们加入第二个点，也就是编号为`2`的这个点，就有经过`2`和不经过`2`，不经过`2`的时候也就是`i`经过其他的点到达`j`的路线，这里就是经过不超过`1`的路线即：`d(1, i, j)`，经过`2`的时候分为两段，`i -> 2`和`2 -> j`，这两段路线的中间都没有经过`2`，所以综合起来就有:`d(2, i, j) = min[d(1, i, j) , d(1, i, 2) + d(1, 2, j)]`，然后再双重循环得到`k = 2`的时候的各种值

于是我们就得到了一个递推公式：`d(k, i, j) = min[d(k - 1, i, j) , d(k - 1, i, k) + d(k - 1, k, j)]`

这个就是`Floyd`的递推公式，我们注意到当`j = k`的时候有：`d(k, i, k) = min[d(k - 1, i, k) , d(k - 1, i, k) + d(k - 1, k, k)] = d(k - 1, i, k)`，所以`d(k - 1, i, k)`的值与层数`k`是没有关系的，可以将第一维去掉，同理当`i = k`的时候可以得到`d(k - 1, k, j)`的值也是与第一维没有关系的，于是我们可以将上面的递推公式优化为：

`d(k, i, j) = min[d(k - 1, i, j) , d(i, k) + d(k, j)]`

在代码中表示就是：

```
for(int k = 1; k <= n; k ++)
    for(int i = 1; i <= n; i ++)
        for(int j = 1; j <= n; j ++)
            d[i, j] = min(d[i, j], d[i, k] + d[k, j]);
```

上述代码中的`d[i, j] = min(d[i, j], d[i, k] + d[k, j])`，其中`min`函数中的`d[i, j]`其实就代表上一层的`d(k - 1, i, j)`

另外分析上面的加点过程中我们需要加深理解的是，在上述的证明中`k`其实代表的是加入了多少个点，并不一定代表加入了顶点编号是`1 ~ k`这么个点，`d(k, i, j)`表示的意义是`i`经过不超过这`k`个点到达`j`的最短路径，这条路径分为两种，一种是经过了第`k`个加入的点，另一种是经过了其他的点，在加入`n`个点的过程中点的编号排列是完全等价的，我们第一次可以加入编号为`1`这个点，也可以加入编号为`4`这个点，所以上述`Floyd`算法的等价写法是：

```
//num是点的编号，nodes[n]是一个数组，是点的编号的随意排列
for(auto num : nodes)
    for(int i = 1; i <= n; i ++)
        for(int j = 1; j <= n; j ++)
            d[i, j] = min(d[i, j], d[num, k] + d[num, j]);
```