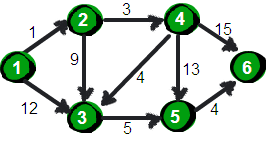
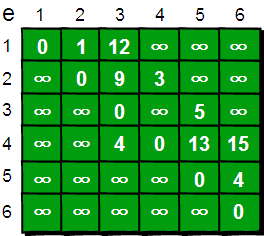
【坐在马桶上看算法】算法7：Dijkstra最短路算法 推荐

[原创](javascript:;)[ahalei](http://blog.51cto.com/ahalei)[2014-04-01 09:17:52](javascript:;)[评论(6)](http://blog.51cto.com/ahalei/1387799#comment)[19873人阅读](javascript:;)

       上周我们介绍了神奇的只有五行的Floyd最短路算法，它可以方便的求得任意两点的最短路径，这称为“多源最短路”。本周来来介绍指定一个点（源点）到其余各个顶点的最短路径，也叫做“单源最短路径”。例如求下图中的1号顶点到2、3、4、5、6号顶点的最短路径。



       与Floyd-Warshall算法一样这里仍然使用二维数组e来存储顶点之间边的关系，初始值如下。



       我们还需要用一个一维数组dis来存储1号顶点到其余各个顶点的初始路程，如下。

090657ofidcactthcig33i.png

       我们将此时dis数组中的值称为最短路的“估计值”。

       既然是求1号顶点到其余各个顶点的最短路程，那就先找一个离1号顶点最近的顶点。通过数组dis可知当前离1号顶点最近是2号顶点。当选择了2号顶点后，dis[2]的值就已经从“估计值”变为了“确定值”，即1号顶点到2号顶点的最短路程就是当前dis[2]值。为什么呢？你想啊，目前离1号顶点最近的是2号顶点，并且这个图所有的边都是正数，那么肯定不可能通过第三个顶点中转，使得1号顶点到2号顶点的路程进一步缩短了。因为1号顶点到其它顶点的路程肯定没有1号到2号顶点短，对吧O(∩\_∩)O~

       既然选了2号顶点，接下来再来看2号顶点有哪些出边呢。有2->3和2->4这两条边。先讨论通过2->3这条边能否让1号顶点到3号顶点的路程变短。也就是说现在来比较dis[3]和dis[2]+e[2][3]的大小。其中dis[3]表示1号顶点到3号顶点的路程。dis[2]+e[2][3]中dis[2]表示1号顶点到2号顶点的路程，e[2][3]表示2->3这条边。所以dis[2]+e[2][3]就表示从1号顶点先到2号顶点，再通过2->3这条边，到达3号顶点的路程。

       我们发现dis[3]=12，dis[2]+e[2][3]=1+9=10，dis[3]>dis[2]+e[2][3]，因此dis[3]要更新为10。这个过程有个专业术语叫做“松弛”。即1号顶点到3号顶点的路程即dis[3]，通过2->3这条边松弛成功。这便是Dijkstra算法的主要思想：通过“边”来松弛1号顶点到其余各个顶点的路程。

       同理通过2->4（e[2][4]），可以将dis[4]的值从∞松弛为4（dis[4]初始为∞，dis[2]+e[2][4]=1+3=4，dis[4]>dis[2]+e[2][4]，因此dis[4]要更新为4）。

       刚才我们对2号顶点所有的出边进行了松弛。松弛完毕之后dis数组为：

090706vmjy7l2ee2lyalia.png

       接下来，继续在剩下的3、4、5和6号顶点中，选出离1号顶点最近的顶点。通过上面更新过dis数组，当前离1号顶点最近是4号顶点。此时，dis[4]的值已经从“估计值”变为了“确定值”。下面继续对4号顶点的所有出边（4->3，4->5和4->6）用刚才的方法进行松弛。松弛完毕之后dis数组为：

090714f2p1wppynngj2pep.png

       继续在剩下的3、5和6号顶点中，选出离1号顶点最近的顶点，这次选择3号顶点。此时，dis[3]的值已经从“估计值”变为了“确定值”。对3号顶点的所有出边（3->5）进行松弛。松弛完毕之后dis数组为：

090722ywunackk35i8cni5.png

       继续在剩下的5和6号顶点中，选出离1号顶点最近的顶点，这次选择5号顶点。此时，dis[5]的值已经从“估计值”变为了“确定值”。对5号顶点的所有出边（5->4）进行松弛。松弛完毕之后dis数组为：

090730eq6oqzyq7laqha9y.png

       最后对6号顶点所有点出边进行松弛。因为这个例子中6号顶点没有出边，因此不用处理。到此，dis数组中所有的值都已经从“估计值”变为了“确定值”。

       最终dis数组如下，这便是1号顶点到其余各个顶点的最短路径。

090738azt5clcozl899ekt.png

       OK，现在来总结一下刚才的算法。算法的基本思想是：每次找到离源点（上面例子的源点就是1号顶点）最近的一个顶点，然后以该顶点为中心进行扩展，最终得到源点到其余所有点的最短路径。基本步骤如下：

* 将所有的顶点分为两部分：已知最短路程的顶点集合P和未知最短路径的顶点集合Q。最开始，已知最短路径的顶点集合P中只有源点一个顶点。我们这里用一个book[ i ]数组来记录哪些点在集合P中。例如对于某个顶点i，如果book[ i ]为1则表示这个顶点在集合P中，如果book[ i ]为0则表示这个顶点在集合Q中。
* 设置源点s到自己的最短路径为0即dis=0。若存在源点有能直接到达的顶点i，则把dis[ i ]设为e[s][ i ]。同时把所有其它（源点不能直接到达的）顶点的最短路径为设为∞。
* 在集合Q的所有顶点中选择一个离源点s最近的顶点u（即dis[u]最小）加入到集合P。并考察所有以点u为起点的边，对每一条边进行松弛操作。例如存在一条从u到v的边，那么可以通过将边u->v添加到尾部来拓展一条从s到v的路径，这条路径的长度是dis[u]+e[u][v]。如果这个值比目前已知的dis[v]的值要小，我们可以用新值来替代当前dis[v]中的值。
* 重复第3步，如果集合Q为空，算法结束。最终dis数组中的值就是源点到所有顶点的最短路径。

       完整的Dijkstra算法代码如下：

       可以输入以下数据进行验证。第一行两个整数n  m。n表示顶点个数（顶点编号为1~n），m表示边的条数。接下来m行表示，每行有3个数x y z。表示顶点x到顶点y边的权值为z。

6 9

1 2 1

1 3 12

2 3 9

2 4 3

3 5 5

4 3 4

4 5 13

4 6 15

5 6 4

运行结果是

0 1 8 4 13 17

   通过上面的代码我们可以看出，这个算法的时间复杂度是O(N2)。其中每次找到离1号顶点最近的顶点的时间复杂度是O(N)，这里我们可以用“堆”（以后再说）来优化，使得这一部分的时间复杂度降低到O(logN)。另外对于边数M少于N2的稀疏图来说（我们把M远小于N2的图称为稀疏图，而M相对较大的图称为稠密图），我们可以用邻接表（这是个神马东西？不要着急，下周再仔细讲解）来代替邻接矩阵，使得整个时间复杂度优化到O( (M+N)logN )。请注意！在最坏的情况下M就是N2，这样的话MlogN要比N2还要大。但是大多数情况下并不会有那么多边，因此(M+N)logN要比N2小很多。

欢迎转载，码字不容易啊，转载麻烦注明出处  
【啊哈！算法】系列7：Dijkstra最短路算法  
<http://ahalei.blog.51cto.com/4767671/1387799>