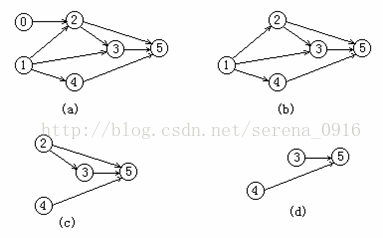
（ 1 ）从有向图中选取一个 没有前驱 ( 入度为 0) 的 顶点，并输出之；

（ 2 ）从有向图中删去此顶点以及所有以它为尾的 弧 ( 弧头顶点的入度减 1 ) ；

重复上述两步，直至图空，或者图不空但找不到 无前驱的顶点为止。

下图：



图a为一个有向图，入度为0的点有两个，任选一个输出，这里输出0，以0为弧尾的顶点入度减一，得图b。继续输出入度为0的点，在此是1，输出1之后，以1为弧尾的顶点入度减一，得图c，以此类推，直到图空为止。

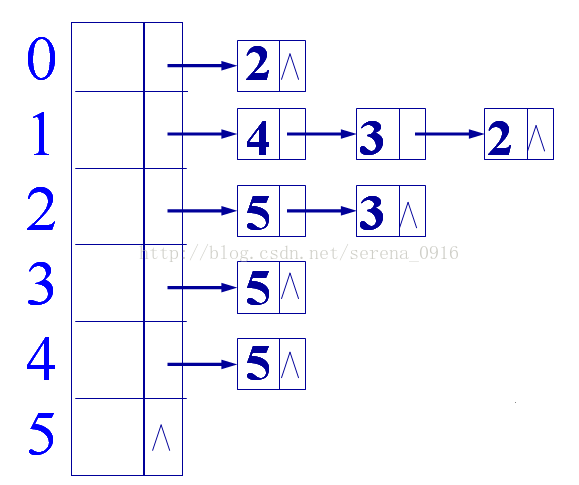
这个理解起来是比较容易的，现在关键的是实现，如何得到入度为0的点？为避免每次都要搜索入度为零的顶点，在算法中设置一个“栈”，以保存“入度为零”的顶点。

在拓扑排序算法中，需要设置一个包含n个元素的一维整形数组，假定用d表示，用它来保存这个有向无环图中每个顶点的入度值。对于上图，得到数组d的初始之为：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 2 | 2 | 1 | 3 |

在进行拓扑排序时，为了把 所有入度为0的顶点都保存起来，而且又便于插入、删除以及节省空间，最好的方法是把它们链接成一个栈。另外，当一个顶点vi的入度为0时，数组d中下标为i的元素d[i]的值为0（这句话的意思是我们没有必要保存入度为0的顶点的入度值，反而我们可以利用这个空间，用它来保存下一个入度为0的顶点的下标（序号），这里很重要，好好理解。这样，就可以把所有入度为0的顶点通过数组d中的对应元素（数组元素的下标对应着顶点编号）静态链接成一个栈。在这个链栈中，栈顶指针top指向第一个入度为0的顶点所对应的数组d中的元素，该元素的值（数组中的值）则指向第二个入度为0的顶点所对应的数组d中的元素，以此类推，最后一个入度为0的顶点所对应的数组d中的元素保存-1，表示为栈底。

例如，根据上图，建立邻接表，如图：



，建立入度为0的初始栈的过程如下：

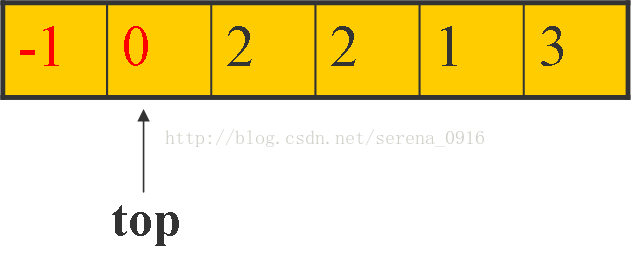
 (1)开始置链栈为空，即给链栈指针top赋初值为-1     top=-1;

 (2)将入度为0的元素d[0]进栈，即： d[0]=top;top=0        /\*因为d[0]存放下一个入度为0的顶点下标，在此，由于它是第一个入度为0的顶点(没有下一个入度为0的顶点），因此d[0]的值为-1,top=0\*/

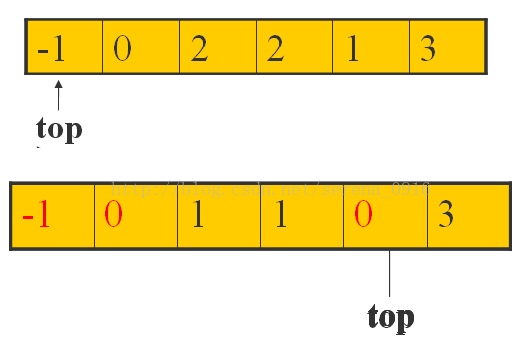
  (3)将入度为0的元素d[1]进栈，此时，d[1]里应该保留下一个入度为0的顶点下标，而此时，下一个入度为0的顶点下标肯定是原top所指向的值，即：

d[1]=top;top=1;

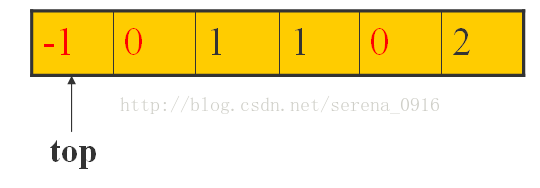
(4)因d[2]至d[5]的值均不为0，所以它们均不进栈。至此，初始栈建立完毕，数组d（多用途哦，即保留了顶点入度不为0的入度值，也作为链栈使用）如下图所示：



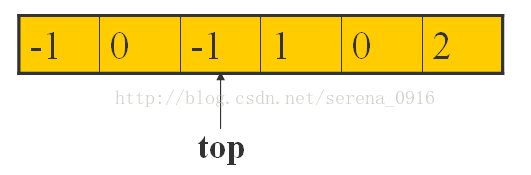
现在开始循环执行拓扑算法中的第一步"选择一个入度为0的顶点并输出之"，利用输出栈顶指针top所代表的顶点序号来实现，所以，输出顶点1（因为top=1).顶点1输出后，修改以顶点1为弧尾的顶点的入度，此例中，顶点4的入度为0，so,d[4]元素入栈，d[4]=0(d[4]=top;top=4，如果理解不了，请模拟一下链栈的出栈和入栈)。



现在顶点4输出，以此为弧尾的顶点入度减一，得图入下：



现在输出顶点0，以此为弧尾的顶点入度减一，此时，顶点2的入度为0，因此入栈。d[2]=-1(d[2]=top;top=2),如图



以此类推，直到top的值为-1，表示栈空，算法执行结束。如果得到的顶点数是n（图的顶点数），则表明该图是有向无环图，否则不是。