0.写在前面

不论是在现实生活当中，还是软件开发过程中，我们总会遇见一些需要给要做的事情进行先后顺序排序的问题。

比如：今天晚饭我决定吃地三鲜（一种传统的东北菜，主要原料是土豆、茄子和青椒），那么我在为晚餐做准备的过程中，就要做如下考量：

首先，我需要确定晚餐吃地三鲜的的话，我需要做哪些准备工作，例如我要先去买菜，然后洗菜，切菜，准备调料，过油，炒菜，装盘等等……

然后，我们要做的就是确定这些工作的先后顺序。显而易见的，我们必须先把菜买回来，才能够洗菜切菜……

但是在切菜过程中，我是不是可以同时起锅烧油呢？因为毕竟起锅烧油这个事情，并不取决于我有没有把菜切好

于是乎在整个做地三鲜的过程当中，我们还会发现很多步骤之间存在顺序上的问题

所以，这就导引出来一个问题：在整个做菜的过程中，有哪些步骤是可以先行完成的，哪些操作一定要排在后面……

那么，这种对操作步骤进行先后顺序排列的排序操作，我们就称之为拓扑排序

注意：拓扑排序算法有别于传统意义上的排序算法

传统意义上的排序算法指的是将元素按照约定的大小规范进行比较，然后排列出先后顺序，也就是说：传统排序算法是基于比较和交换的

而拓扑排序算法则是基于活动发生的先后顺序的，拓扑排序的结果表示的是我们只有在完成哪些准备工作之后，才能够去做后面的工作，也就是说拓扑排序实际上排的是活动的发生的先后顺序

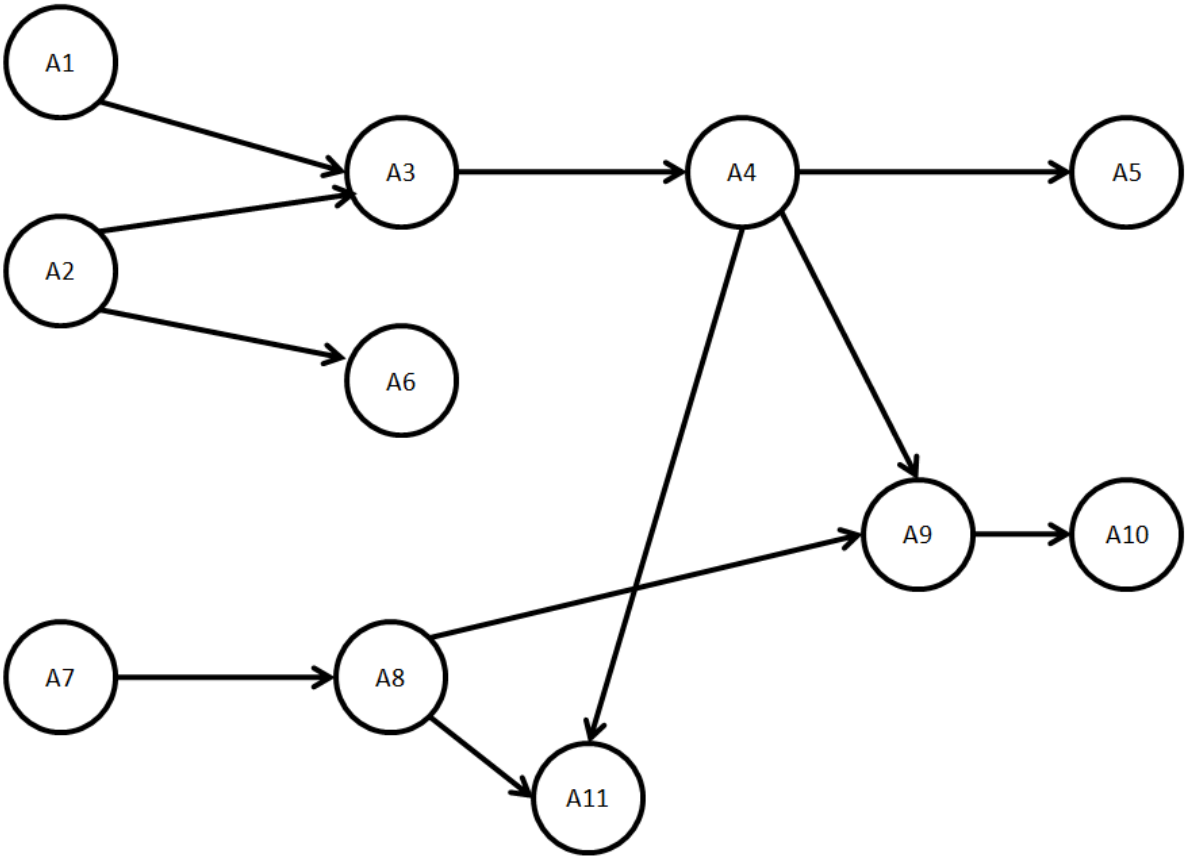
这也是为什么我们说拓扑排序是最不像排序的排序

1.AOV网的相关概念

拓扑排序算法的实现，是基于一种称之为AOV网的图结构实现的

那么在开始学习拓扑排序之前，我们很有必要先来研究一下，什么是AOV网

首先，我们给出一个比较全面的AOV网的示意图：



接下来，我们就以上述图片为例，了解一下AOV网的相关概念

①活动的概念

在AOV网图中，我们将每一个节点称之为一个活动（Activity）

我们可以将活动这个概念想象成为一件要做的事情。例如，在上面做菜的例子当中，买菜、洗菜、切菜等等操作，都可以看做一个活动

活动本身作为一个持续的流程，理论上来讲是有活动的持续时间的，也就是说，你要了解洗菜需要洗多久，切菜需要切多久，下锅过油需要多久（过油时间长了就糊了）……

但是，AOV网主要讨论的并不是一个活动持续的时间长短，而是更加注重讨论活动之间发生的先后顺序问题

所以在AOV网中，我们并不需要记录活动持续时间的长短，这个问题我们将在后续的AOE网和关键路径问题中进行描述

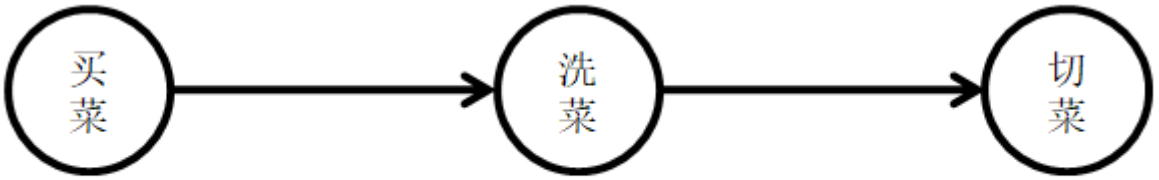
②活动之间的先后顺序

从上图中我们不难发现：AOV网实际上是一个有向无权图

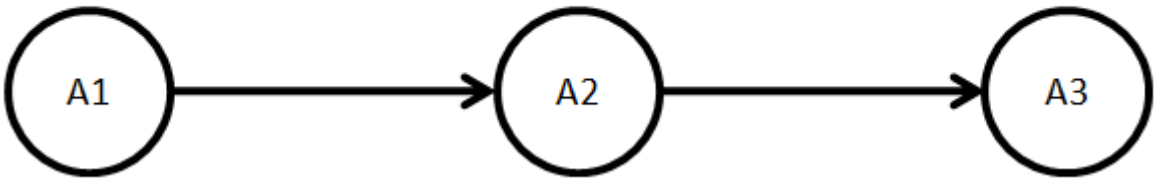
在AOV网中，我们通过节点之间边的指向，来描述事件之间的发生的先后顺序

我们可以做如下理解：处于一条有向边起点上的活动，一定是这条有向边终点活动的“前提条件”

例如：洗菜的前提是先买菜，切菜的前提是先把菜洗干净，那么这段逻辑使用AOV网进行描述就是：



对应的AOV网结构



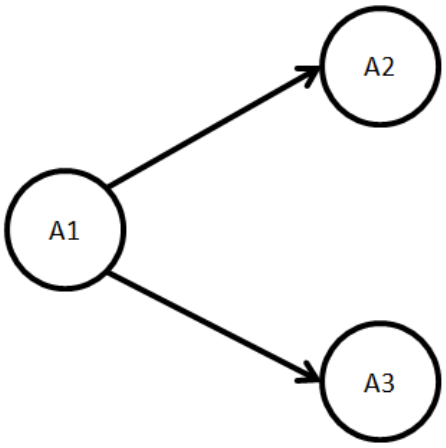
所以，由AOV网中节点（活动）发生的先后顺序，我们引出如下前置条件的概念：

我们认为：在AOV网中的一条有向边上，起点节点（活动）是终点节点（活动）的前提条件

也就是说：只有在前提条件活动发生之后，后面的活动才能够依次发生，这个顺序是不可错乱的

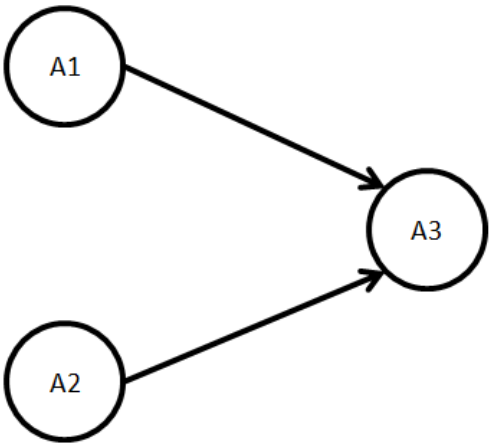
当然，一个前提条件可能引发出很多后续的活动，而一个活动的发生，也可能同时需要很多前提条件的配合才行

一个前置条件引发多个后续活动



(1)

一个活动需要多个前置条件



(2)

我们为了方便后续的计算操作，特别引入出度和入度的概念，对从某一节点指出的边以及指向某一节点的边的数量进行统计

我们将以一个节点（活动）为起点的边的数量，称之为这个节点的出度，也就是说：出度衡量的是从这个节点（活动）指出的边的数量

我们将以一个节点（活动）为终点的边的数量，称之为这个节点的入度，也就是说：入度衡量的是指向这个节点（活动）的边的数量

例如：

上图(1)中：节点A1的入度为0，出度为2，节点A2和节点A3的入度都是1，而出度为0

上图(2)中：节点A1和A2的入度为0，出度都是1，而节点A3的入度为2，出度为0

③AOV网是有向无环图

在了解了AOV网的相关概念之后，我们还需要深刻记住AOV网的一大特征：AOV网是一种有向无环图

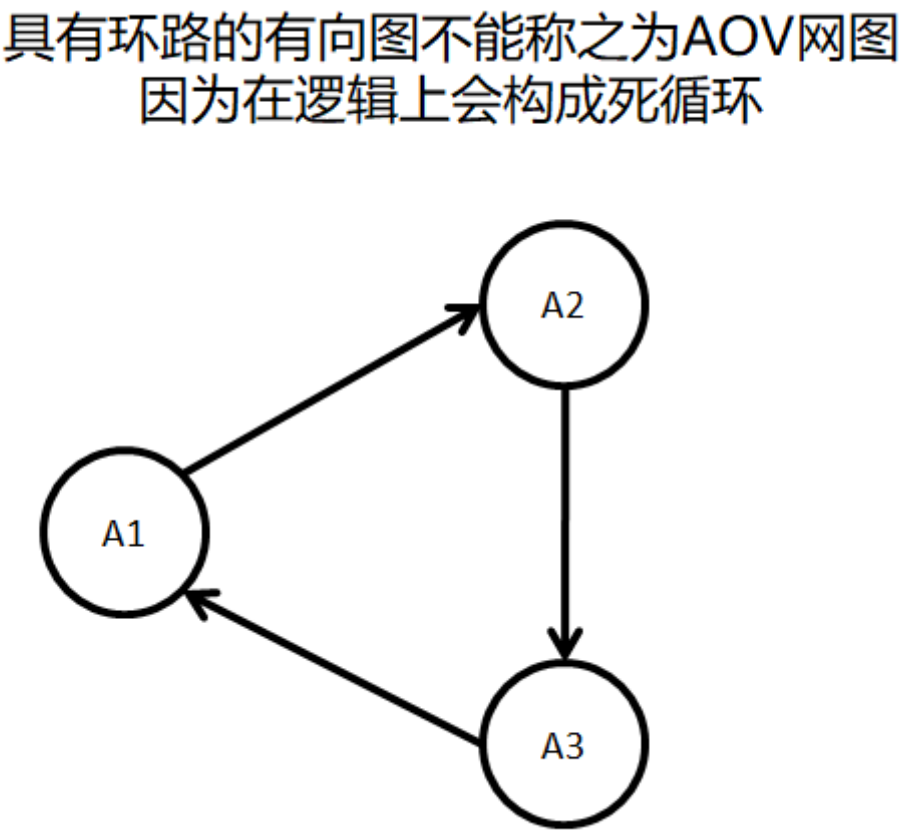
这个特征对于AOV网来说是至关重要的，因为我们之前说过：AOV网描述的是各个活动之间发生的先后顺序

那么，如果在AOV网当中出现环路的话，也就表示着一件事自己直接或者间接的以自己为前提，这在逻辑上是完全不成立的

就好比说：A向B借钱，B说我没钱，我先向C借一些，C说我也没钱，我要从A手里借到钱才能给B，但是此时A同样没钱借给C（毕竟大家都是穷人……）

这就形成了如下的逻辑关系：A拿到钱的前提是B有钱借给A，B有钱的前提是C把钱借给B，C有钱的前提是A把钱借给C，而A有钱的前提是B把钱借给A……

这样下去，必然会在逻辑上形成一个死循环



通过上述讲解，我们已经认识了AOV网，那么下面就让我们一起通过AOV网，来研究拓扑排序的操作流程

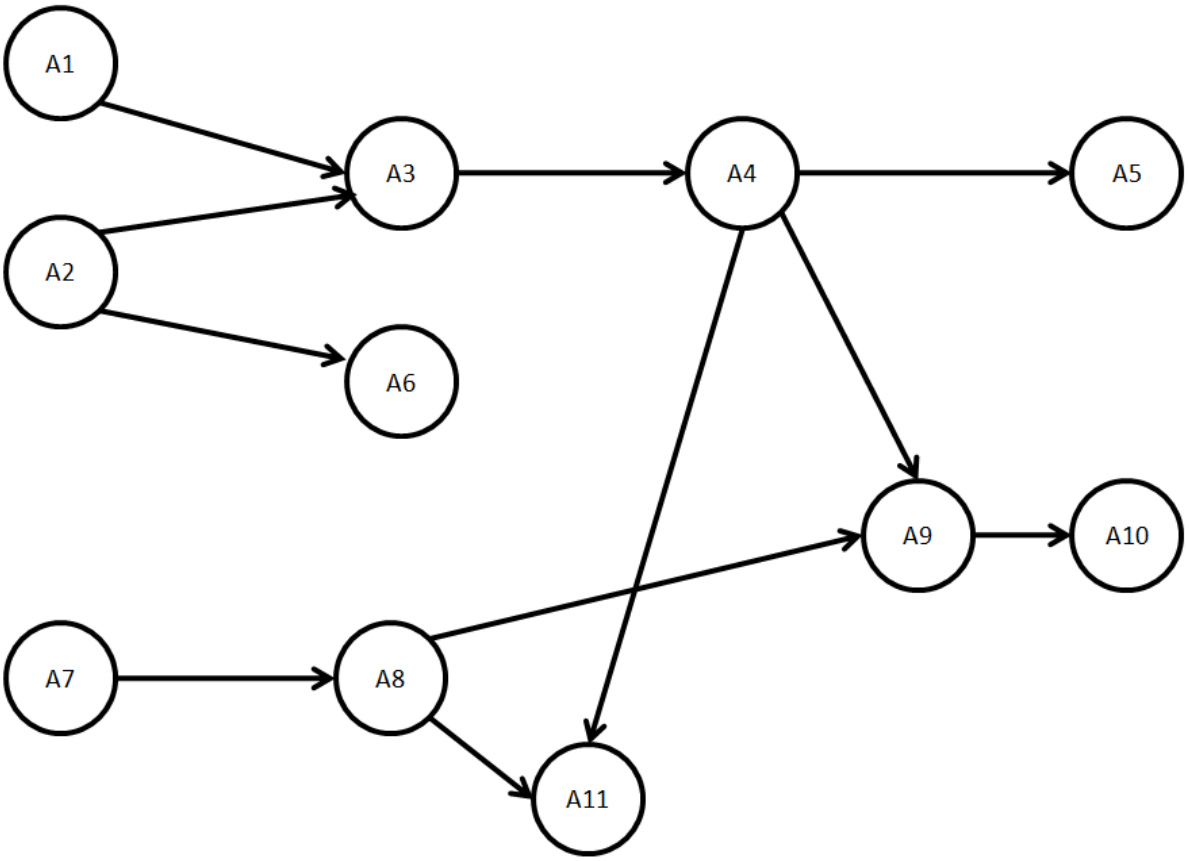
2.拓扑排序问题的解题步骤

①构建问题对应的AOV网

在得到一个问题之后，我们首先需要按照问题中描述的所有活动之间执行的先后顺序，将这些活动化作节点

并且使用有向边将这些节点连接起来，构成AOV网图

下面，我们依然使用上图中给定的AOV网图作为示例，研究拓扑排序的具体过程



②AOV网的存储：邻接表

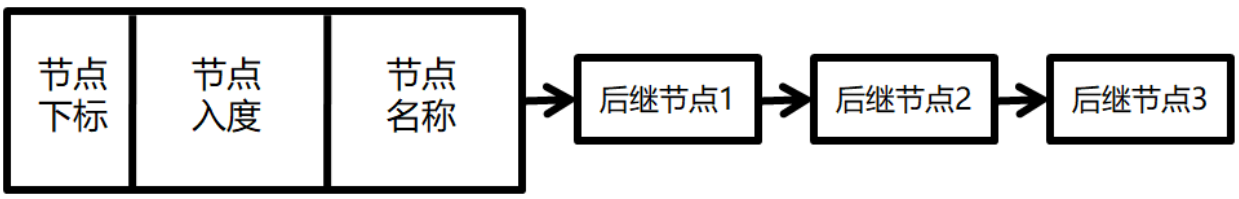
在程序中，为了保存一个AOV网结构，我们需要使用一种特殊的、类似于散列表的结构对AOV网中的节点信息进行存储

我们将这种结构称之为邻接表

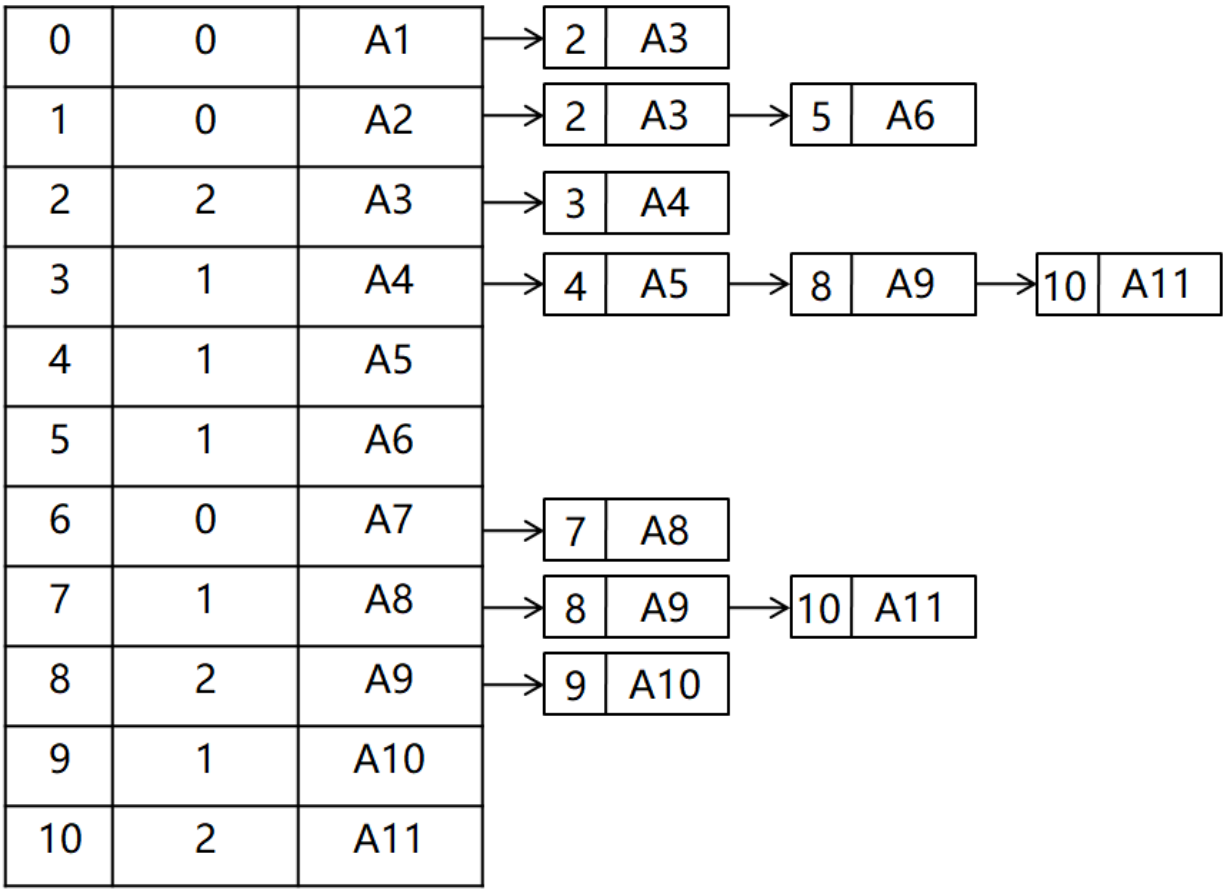
邻接表作为之前我们认识过的邻接矩阵的远房表亲，能够记录AOV网中节点更多的信息

邻接表的每一行，主要记录如下信息：节点下标 + 节点名称 + 节点入度 + 当前节点的后续节点构成的链表

一行邻接表的示意图：



那么如果将上图中的AOV网整体使用邻接表进行表示的话，我们将得到如下结构：



在上图中，表格中的部分我们可以看做是一个数组，而这个数组的每一个元素，都是一个链表的头结点

这个头结点中不仅记录了当前活动的下标、名称和入度，更重要的是能够向后以链表的方式记录其后续节点有哪些

这种结构为后面的拓扑排序步骤提供了极大的方便

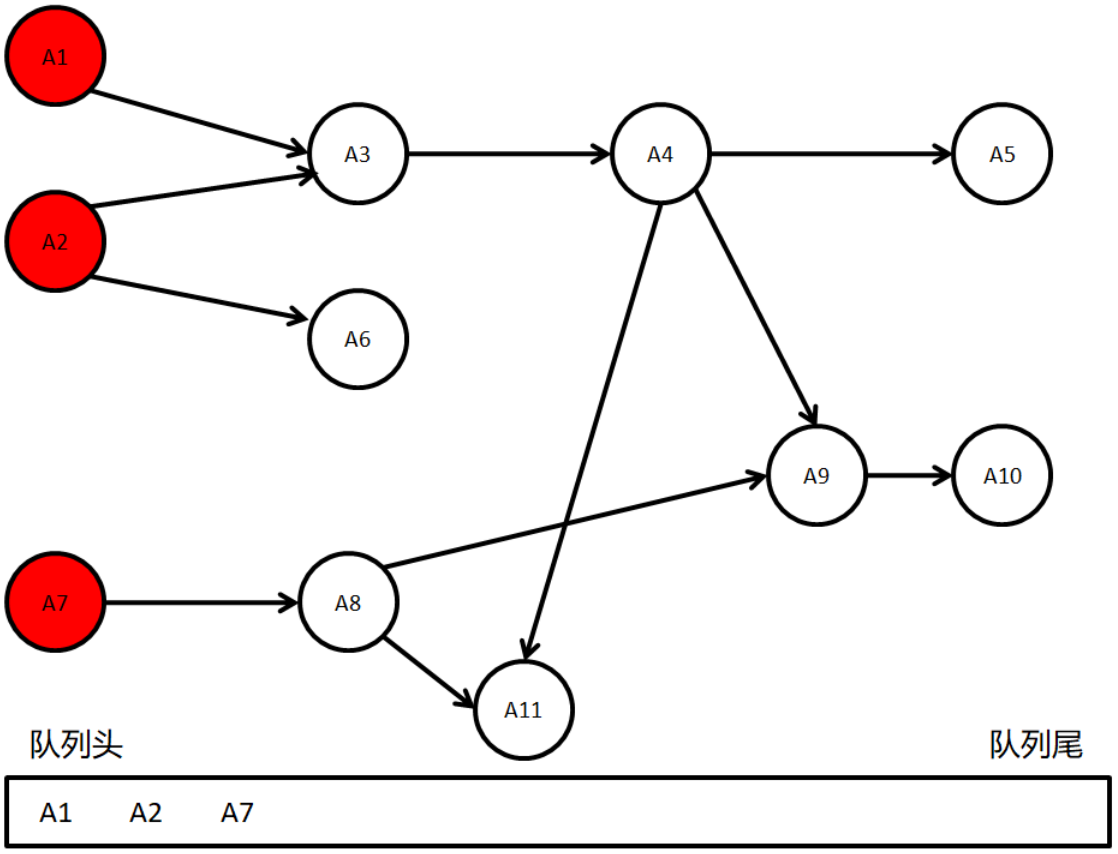
③AOV网节点的拓扑排序执行流程

在对AOV网图中的节点进行拓扑排序的时候，我们需要借助一个额外的结构：队列（有的教程中使用的是栈结构，也是可行的），来辅助完成

至于这个队列结构的作用，我们会在下面的步骤中详细指出

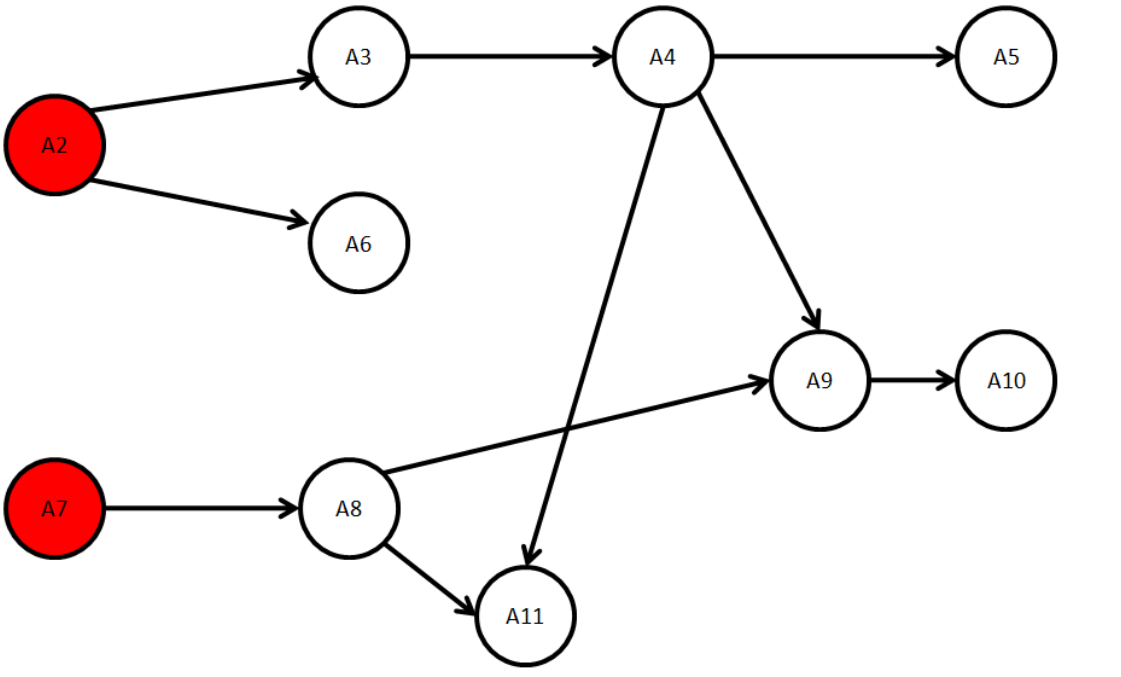
下面我们开始讨论拓扑排序的步骤

步骤1：找出AOV网图中，入度为0的节点，加入队列中



拓扑排序序列：

步骤2：将队列中处于队列头的元素出队列，加入拓扑排序序列中，并将这个节点从AOV网图中删除



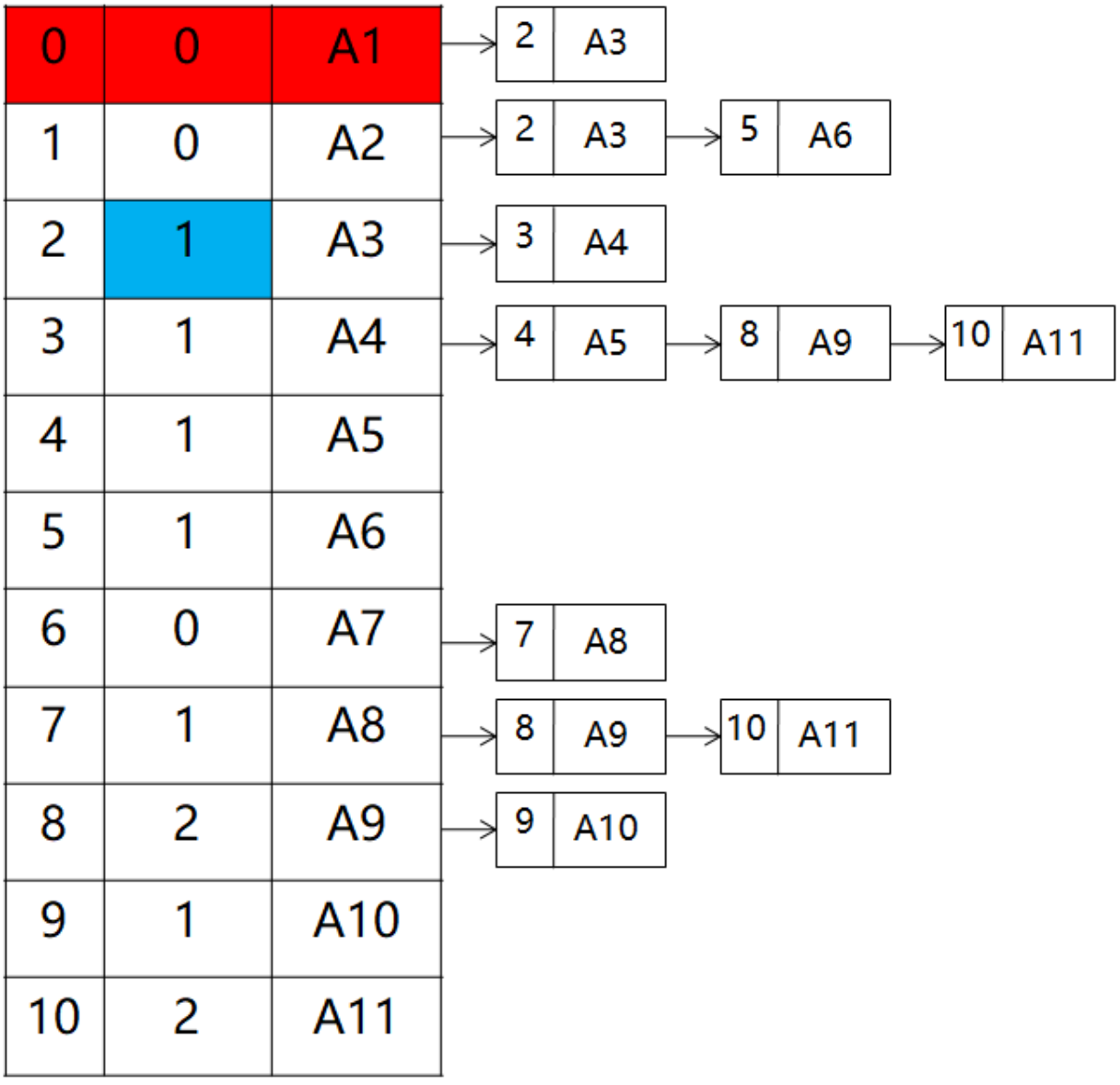
拓扑排序序列：A1

步骤3：被删除节点的所有后续节点的入度减1。在入度减1之后，如果有后续节点的入度取值为0，那么直接将这个后续节点加入队列中

此时我们就能看出来使用邻接表法表示AOV网图的优点了：在选中节点从AOV网中删除之后，只有删除节点的后续节点才会执行入度减1操作

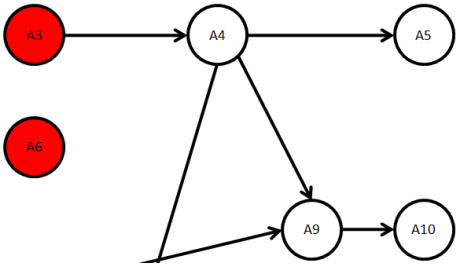
也就是说：只有被删除节点的后续节点，才有可能成为新的入度为0的节点

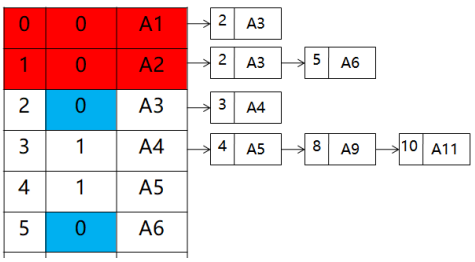
所以此时我们在寻找新的入度为0的节点的时候，不需要重新遍历整个AOV网图，我们只要从删除节点出发，按照链表遍历其后续节点即可

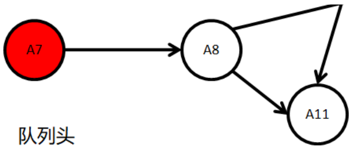


步骤4：重复步骤1-3，直到所有节点全部从AOV网图中删除为止

下面让我们按照上述流程，完成上面示例AOV网图中的所有节点的拓扑排序操作





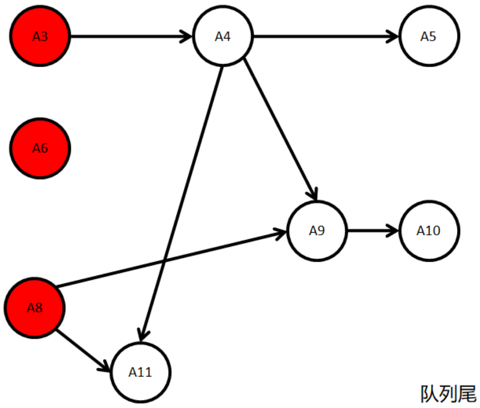


队列头　　　　　　　　　　　　　　　　　　　　队列尾

队列
A7　A3　A6

6	0	A7	→ 7 A8
7	1	A8	→ 8 A9 → 10 A11
8	2	A9	→ 9 A10
9	1	A10	
10	2	A11	

拓扑排序序列：A1 A2

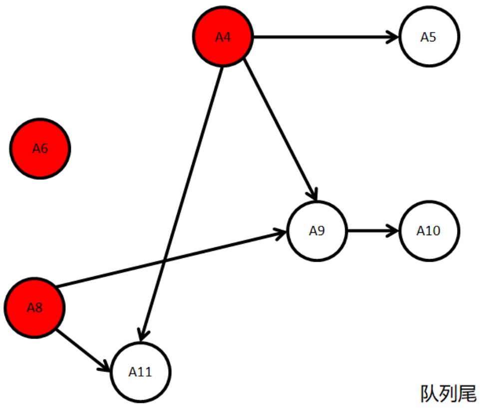


队列头　　　　　　　　　　　　　　　　　　　　队列尾

队列
A3　A6　A8

0	0	A1	→ 2 A3
1	0	A2	→ 2 A3 → 5 A6
2	0	A3	→ 3 A4
3	1	A4	→ 4 A5 → 8 A9 → 10 A11
4	1	A5	
5	0	A6	
6	0	A7	→ 7 A8
7	0	A8	→ 8 A9 → 10 A11
8	2	A9	→ 9 A10
9	1	A10	
10	2	A11	

拓扑排序序列：A1 A2 A7

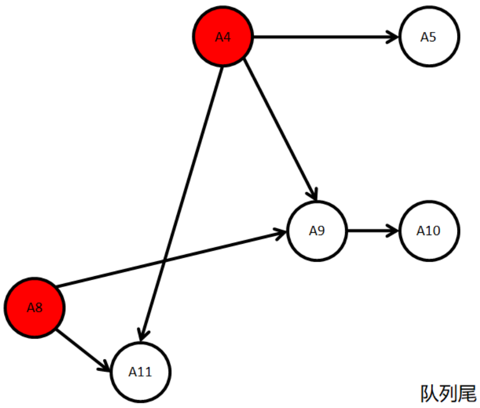


队列头　　　　　　　　　　　　　　　　　　　　队列尾

队列
A6　A8　A4

0	0	A1	→ 2 A3
1	0	A2	→ 2 A3 → 5 A6
2	0	A3	→ 3 A4
3	0	A4	→ 4 A5 → 8 A9 → 10 A11
4	1	A5	
5	0	A6	
6	0	A7	→ 7 A8
7	0	A8	→ 8 A9 → 10 A11
8	2	A9	→ 9 A10
9	1	A10	
10	2	A11	

拓扑排序序列：A1 A2 A7 A3

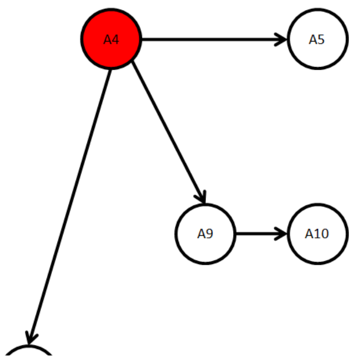


队列头　　　　　　　　　　　　　　　　　　　　队列尾

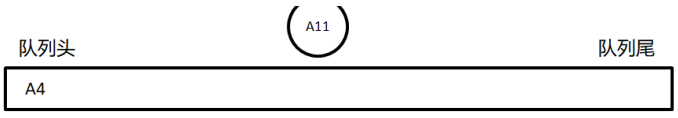
队列
A8　A4

0	0	A1	→ 2 A3
1	0	A2	→ 2 A3 → 5 A6
2	0	A3	→ 3 A4
3	0	A4	→ 4 A5 → 8 A9 → 10 A11
4	1	A5	
5	0	A6	
6	0	A7	→ 7 A8
7	0	A8	→ 8 A9 → 10 A11
8	2	A9	→ 9 A10
9	1	A10	
10	2	A11	

拓扑排序序列：A1 A2 A7 A3 A6



0	0	A1	→ 2 A3
1	0	A2	→ 2 A3 → 5 A6
2	0	A3	→ 3 A4
3	0	A4	→ 4 A5 → 8 A9 → 10 A11
4	1	A5	
5	0	A6	
6	0	A7	→ 7 A8
7	0	A8	→ 8 A9 → 10 A11
8	1	A9	



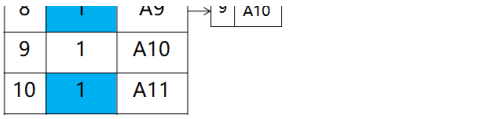
拓扑排序序列：A1 A2 A7 A3 A6 A8

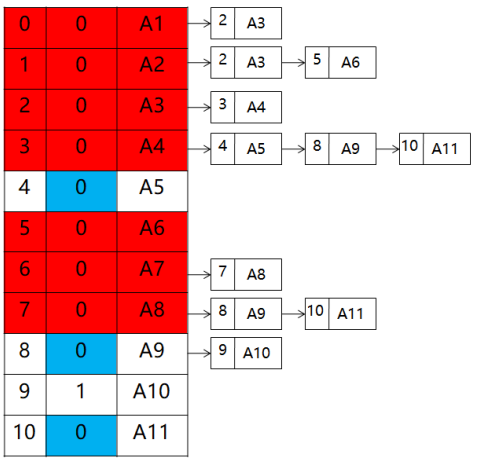
拓扑排序序列：A1 A2 A7 A3 A6 A8 A4

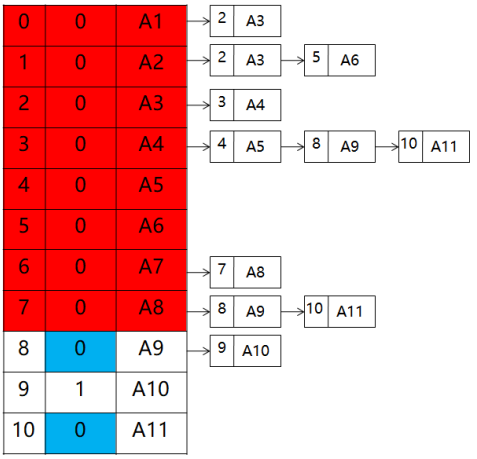
拓扑排序序列：A1 A2 A7 A3 A6 A8 A4 A5

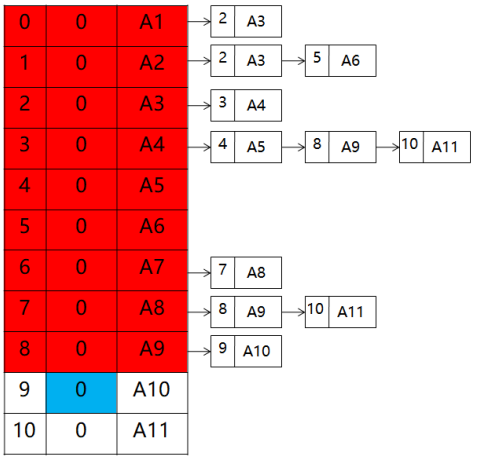
拓扑排序序列：A1 A2 A7 A3 A6 A8 A4 A5 A9

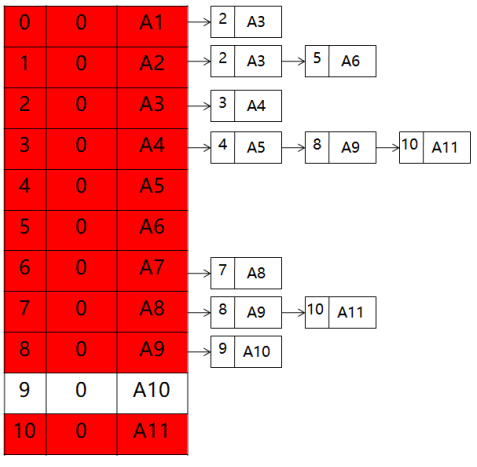
拓扑排序序列：A1 A2 A7 A3 A6 A8 A4 A5 A9 A11

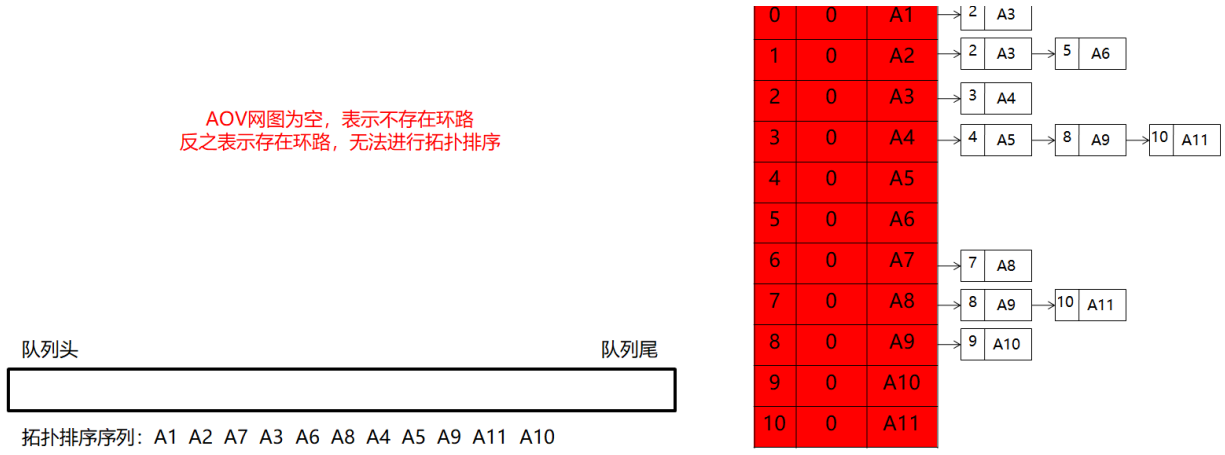












上图的最终拓扑排序序列为：A1 A2 A7 A3 A6 A8 A4 A5 A9 A11 A10

需要注意的是：在执行拓扑排序的过程中，队列结构在拓扑排序完成前始终是不为空的

我们在使用这个队列结构维护节点（活动）之间的拓扑顺序的同时，也可以使用这个队列结构来判断拓扑排序是否完成

这就是这个辅助的队列结构在拓扑排序中的作用

并且，如果当队列为空后，邻接表中还存在没有遍历过的节点，那么说明这些节点之间一定构成了环路

我们也可以反过来进行理解：如果AOV网中存在环路，那么环路中的节点之间，入度永远不可能清零

此时表示：当前AOV网图不能够进行有效的拓扑排序操作

④案例：一道拓扑排序的面试题

下面我们通过一道面试题来看一下，拓扑排序在现实生活（也可能不是现实生活……）中的应用

题目：火星文字典（你告诉我这是现实生活？？？）
题干：
已知一种新的火星文的单词由英文字母组成（全宇宙都在说英文……），但是此火星文中的字母先后顺序未知。
给出一组非空的火星文单词，且此组单词已经按照火星文字典序进行好了排序
请推断出次火星文中的字母先后顺序
输入：
一行文本。为一组按照火星文字典序排好的单词（单词两端无引号）
单词之间通过空格隔开
输出：按火星文字母顺序输出出现过的字母，字母之间无其他符号

如果无法确定顺序或者无合理的字母排序可能，请输出"invalid"（哈哈！看来火星人也有说不明白话的时候……）
输入样例1：
wrt wrf er ett rftt
输出样例1：
werft
输入样例2：
z x
输出样例2：
zx
输入样例3：
zxz
输出样例3：
invalid

下面我们一起来分析一下这道面试题：

解题思路：

首先，这是一道很典型的使用拓扑排序进行运算的题目，因为题目中暗示我们，我们需要通过几个采样的单词，来确定一组文字之间的字典顺序

并且通过输入输出样例3我们可以看出来：如果出现类似ABA这样结构的单词，那么就无法正确确定AB之间的字典顺序

这种情况正好是符合拓扑排序中，AOV网图出现环路的情况的

所以根据上述推断，我们总结出解题的3个步骤：

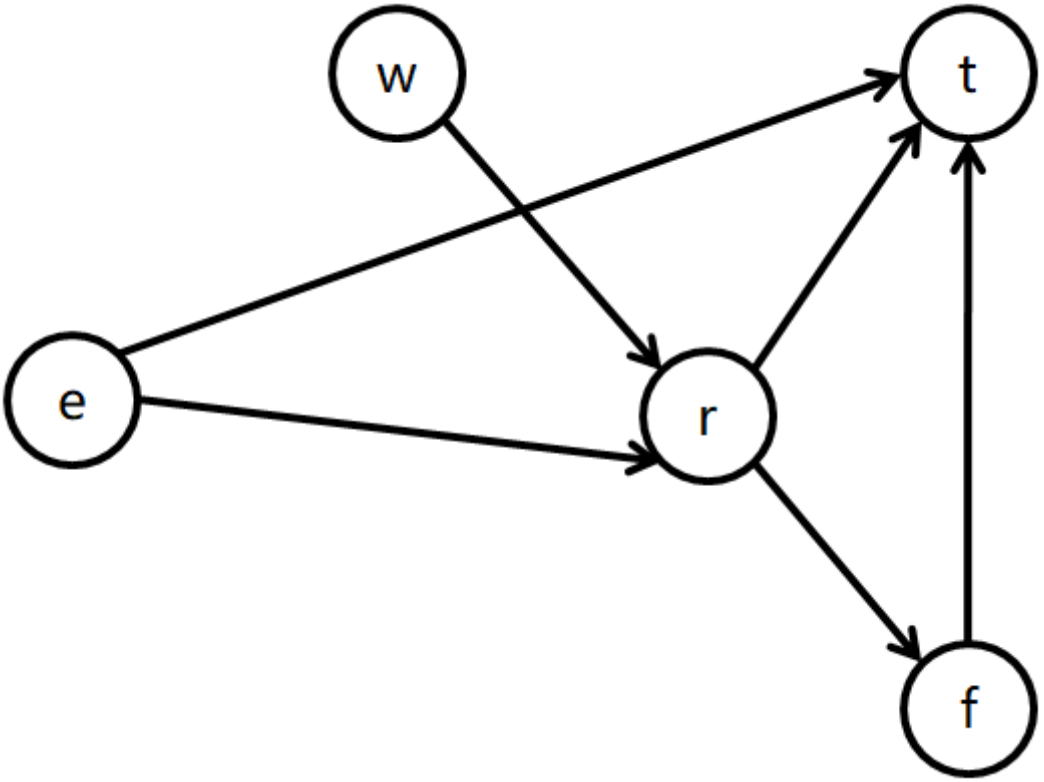
步骤1：根据已有的单词样例（输入输出样例1）来构建字符之间先后顺序的AOV网图和邻接表

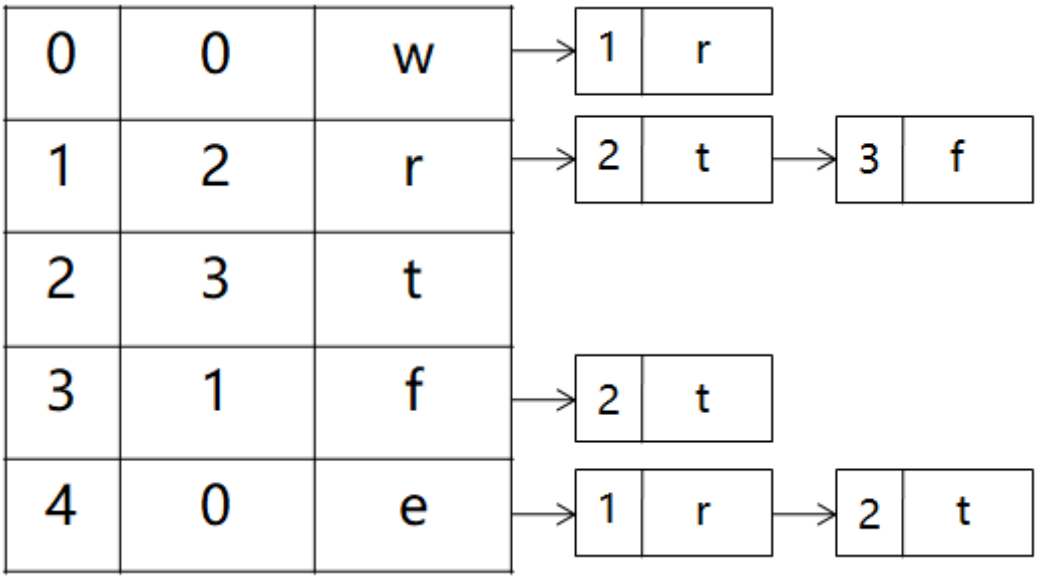
步骤2：对字符构成的AOV网图进行拓扑排序

步骤3：拓扑排序完成后，对AOV网图对应的邻接表进行遍历，确定其中是否还有未使用的字符，也就是判断AOV网图是否存在环路的过程

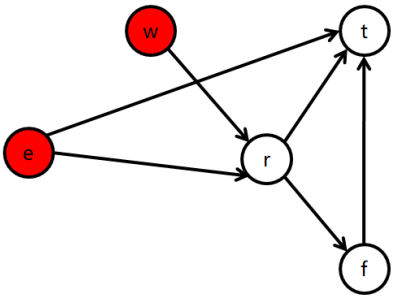
解题演示：

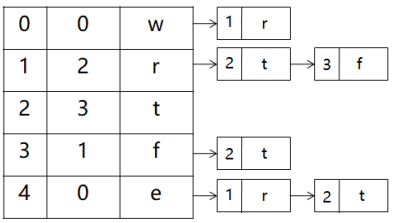
步骤1：根据单词组wrt wrf er ett rftt构建字符之间的AOV网图，并给出相应的邻接表





步骤2：对上述AOV网图和邻接表执行拓扑排序操作

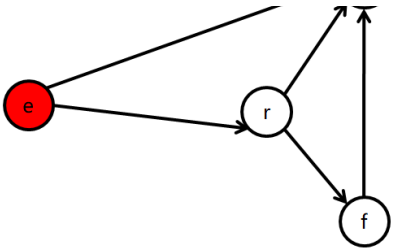




队列头　　队列尾

w　e

拓扑排序序列：

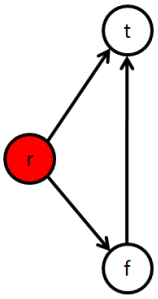


0	0	w	→	1	r			
1	1	r	→	2	t	→	3	f
2	3	t						
3	1	f	→	2	t			
4	0	e	→	1	r	→	2	t

队列头 队列尾

e

拓扑排序序列：w



0	0	w	→	1	r			
1	0	r	→	2	t	→	3	f
2	2	t						
3	1	f	→	2	t			
4	0	e	→	1	r	→	2	t

队列头 队列尾

r

拓扑排序序列：w e

0	0	w	→	1	r			
1	0	r	→	2	t	→	3	f
2	1	t						
3	0	f	→	2	t			
4	0	e	→	1	r	→	2	t

队列头 队列尾

f

拓扑排序序列：w e r

0	0	w	→	1	r			
1	0	r	→	2	t	→	3	f
2	0	t						
3	0	f	→	2	t			
4	0	e	→	1	r	→	2	t

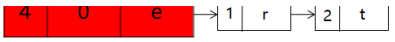
队列头 队列尾

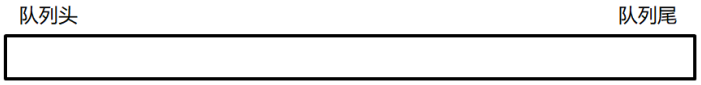
t

拓扑排序序列：w e r f

拓扑排序完成，全部节点被遍历
不存在环路

0	0	w	→	1	r			
1	0	r	→	2	t	→	3	f
2	0	t						
3	0	f	→	2	t			





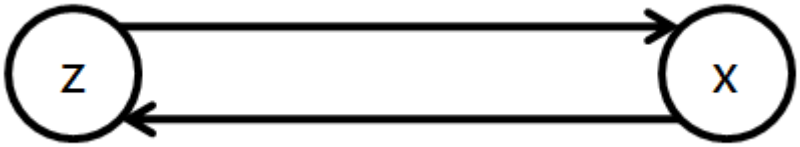
拓扑排序序列：w e r f t

步骤3：验证AOV网中是否存在环路

这一步骤比较简单：当队列结构为空之后，我们只要再对邻接表进行一次遍历即可

如果邻接表中记录的节点中，尚且存在没有进行过遍历的节点，那么说明此事的AOV网中存在环路

例如：按照输入样例3给定的z x z结构中，对应的AOV网图就是这样的：



此图中明显存在环路，并且节点z和节点x的入度永远不为0，那么就表示这两个节点永远会留在邻接表当中

并且，此时的队列结构也是无法被成功清空的，所以无法确定这两个字符之间正确的相对顺序

到此为止，我们就确定了这个题目中，给定正确单词之中火星文字符的字典顺序为：werft