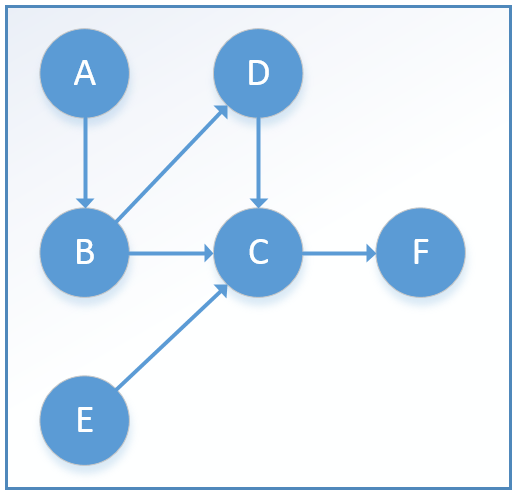
****拓扑排序是针对有向无圈图的顶点的一种排序，使得如果存在一条从A到B的路径，那么在排序中A必定在B的前面。****

AOV网络：就是 Activity On Vertex network的简称，顶点活动网，它是一个有向无环图。有向：就是指有先后顺序，比如技能树中，学习技能有先后顺序。无环：就是图里面没有首尾相接的“圈”，比如技能树中学完高技能后总不能又回到低技能的学习吧？

一个简单的拓扑排序的方案(Kahn算法)是：****先找出任意一个没有入边的顶点，然后打印该顶点，并将它及其边一起从图中剔除，然后对其余的部分继续这个操作****。

这个思路很简单，下面来做具体的实现。针对一个图，如下，来做拓扑排序，使用Java来实现。



以弧表示活动，弧上的权值表示进行该项活动所需的时间，以顶点表示“事件”，称这种有向图为活动网，简称AOE（Activity On Edge）。

# [关键路径：](http://blog.csdn.net/liuquan0071/article/details/50440853" \t "http://blog.csdn.net/liuquan0071/article/details/_blank)

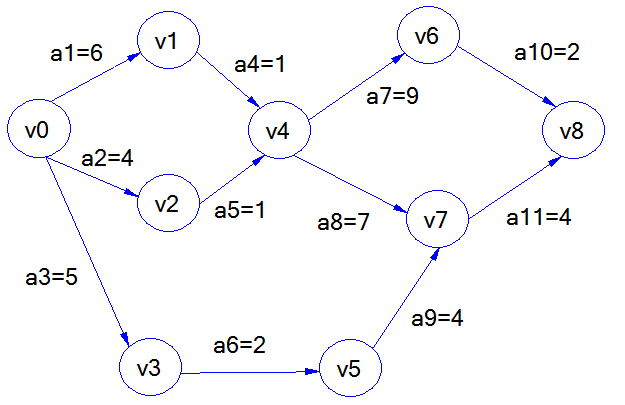
以弧表示活动，弧上的权值表示进行该项活动所需的时间，以顶点表示“事件”，称这种有向图为活动网，简称AOE（Activity On Edge）。

从源点v0出发，令ve0=0，按拓扑排序序列，求其余各顶点的ve(j)=max{ve(i) + |i，j|}，i,j∈T,T是所有以第j个顶点为头的弧的集合

从汇点vn-1出发，令vl（n-1） = ve（n-1），按逆拓扑排序求其余顶点允许的最迟开始时间为：vl（i）= min{vl（j）-|i，j|}，i，j∈S，S是所有以第j个顶点为尾的弧的集合

求每一项活动ai的最早开始时间e（i）=ve（j）和最迟开始时间l（i）=vl（j）-|i，j|。若满足e（i）=l（i），则它是关键路径。

### [测试分析图：](http://blog.csdn.net/liuquan0071/article/details/50440853" \t "http://blog.csdn.net/liuquan0071/article/details/_blank)



| **事件** | **ve** | **vl** |
| --- | --- | --- |
| v0 | 0 | 0 |
| v1 | 6 | 6 |
| v2 | 4 | 6 |
| v3 | 5 | 8 |
| v4 | 7 | 7 |
| v5 | 7 | 10 |
| v6 | 16 | 16 |
| v7 | 14 | 14 |
| v8 | 18 | 18 |

| **活动** | **e** | **l** | **l-e** |
| --- | --- | --- | --- |
| a1 | 0(ve0) | 0(vl1-/v0v1/) | 0 |
| a2 | 0(ve0) | 2(vl2-/v0v2/) | 2 |
| a3 | 0(ve0) | 3(vl3-/v0v3/) | 3 |
| a4 | 6(ve1) | 6(vl4-/v1v4/) | 0 |
| a5 | 4(ve2) | 6(vl4-/v2v4/) | 2 |
| a6 | 5(ve3) | 8(vl5-/v3v5/) | 3 |
| a7 | 7(ve4) | 7(vl6-/v4v6/) | 0 |
| a8 | 7(ve4) | 7(vl7-/v4v7/) | 0 |
| a9 | 7(ve5) | 10(vl7-/v5v7/) | 3 |
| a10 | 16(ve10) | 16(vl8-/v6v8/) | 0 |
| a11 | 14(ve11) | 14(vl8-/v7v8/) | 0 |

针对com.demo.data.struct.criticalpath代码里的两个例子的图：

