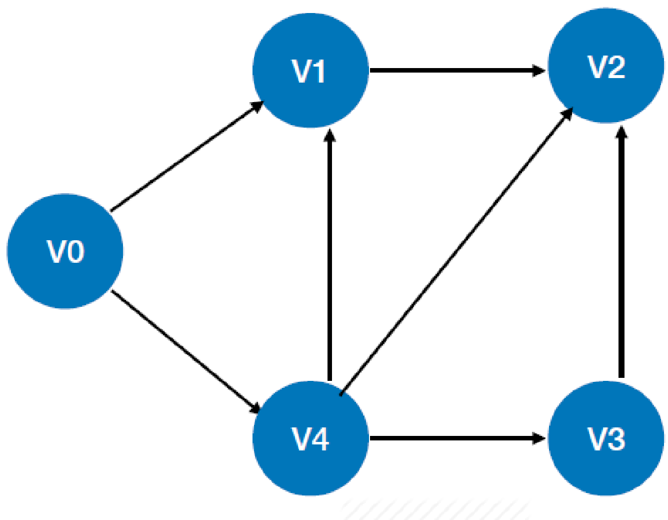
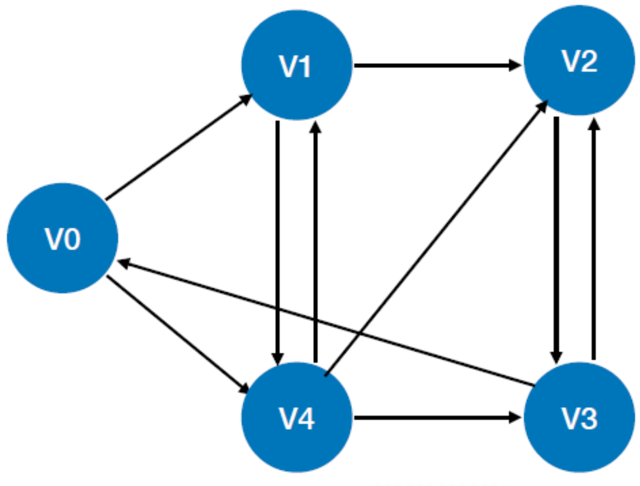
7.5 有向无环图的应用

- **7.5.1 表达式的描述**
- **7.5.2 拓扑排序 (Topological Sort)**
- **7.5.3 关键路径 (Critical Path)**

7.5 有向无环图的应用

- 有向无环图：一个无环的有向图称作有向无环图（directed acycline graph)，简称DAG图。



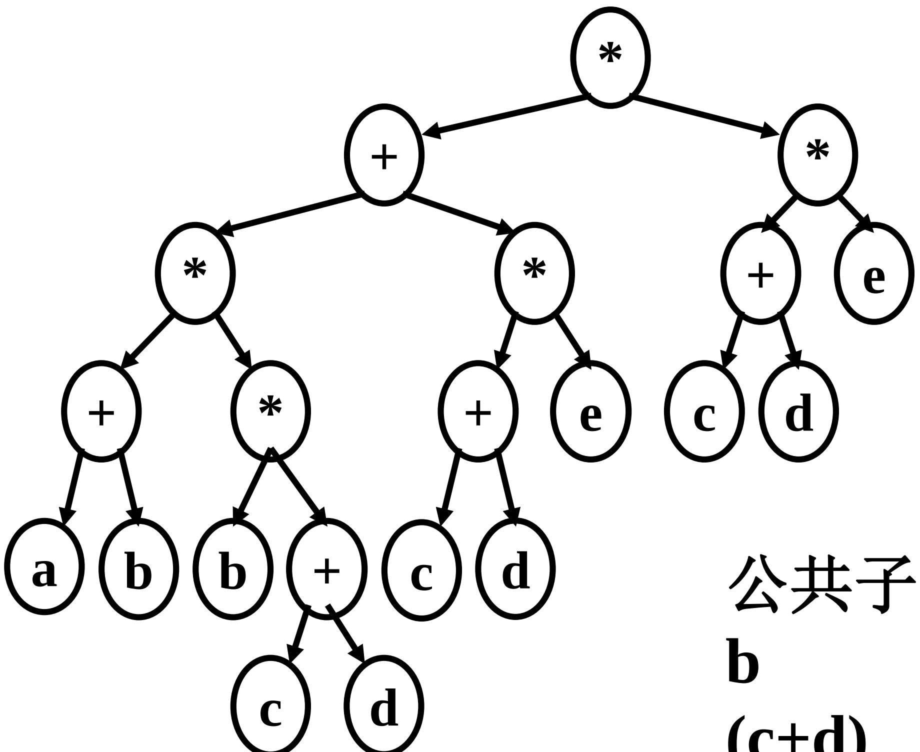


- 注意：无环路的有向图对应的无向图可能存在环路。

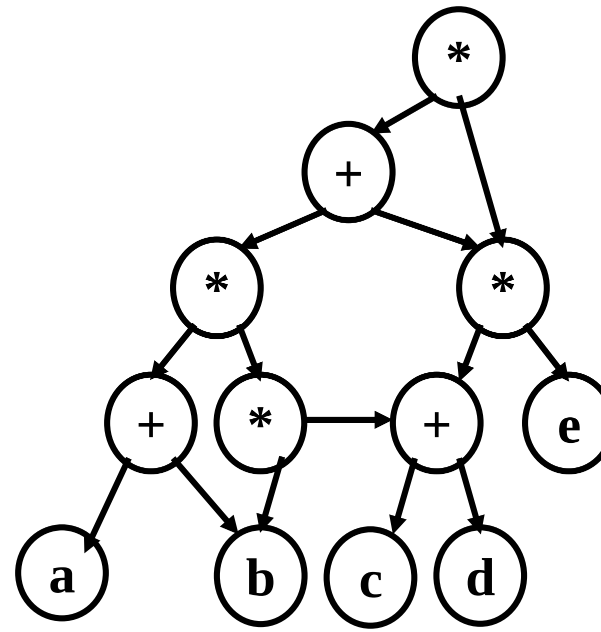
7.5.1 DAG描述表达式

- 无环路的有向图可以描述含有公共子式的表达式。

  ((a+b)*(b*(c+d))+(c+d)*e)*((c+d)*e)

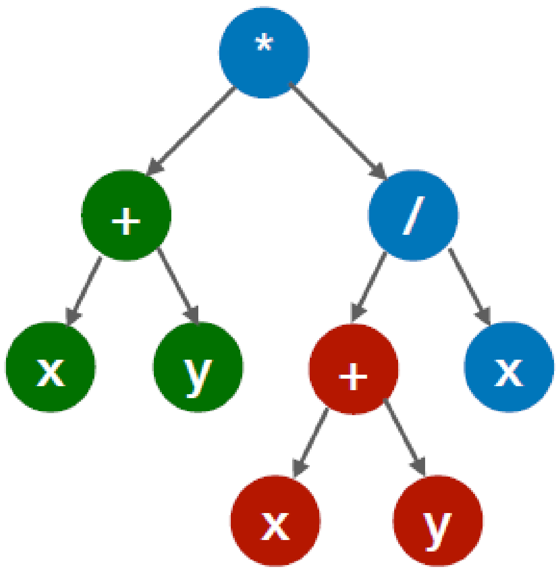


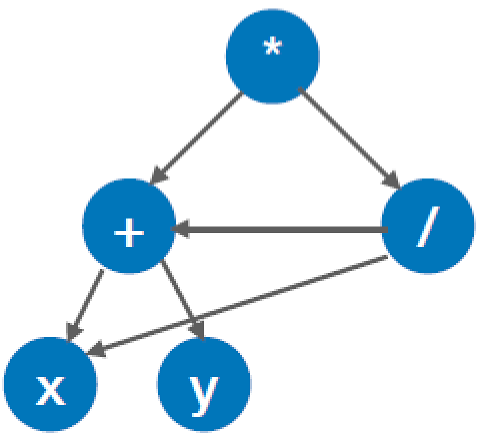
公共子式
b
(c+d)
(c+d)*e



7.5.1 DAG描述表达式

【2019考研题】用DAG描述表达式
（x+y）*（（x+y）/ x）需要的顶点个数至少是?



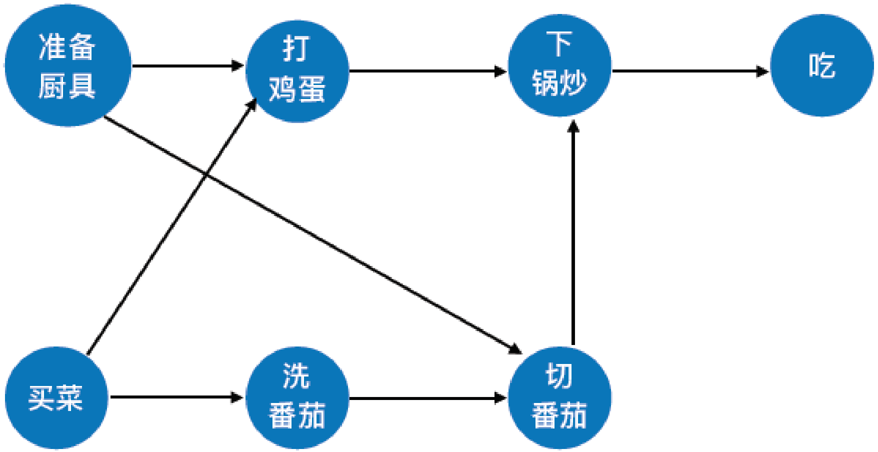


7.5 有向无环图的应用

- 7.5.1 表达式的描述
- 7.5.2 拓扑排序 (Topological Sort)
- 7.5.3 关键路径 (Critical Path)

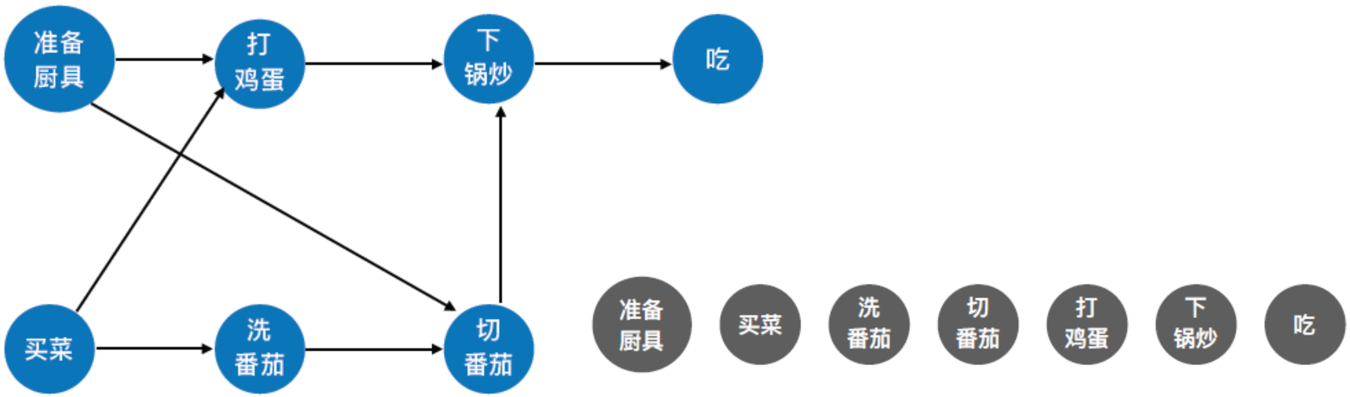
7.5.2 拓扑排序算法

- AOV网：如果用有向图表示一个工程，用顶点表示活动，用弧表示活动之间的优先关系，称这样的有向图为顶点表示活动的网，简称AOV网(Activity On Vertex NetWork)。
  - AOV网中的弧表示活动之间存在的某种制约关系。
  - 在AOV网中，若从顶点 i 到 j 有一条有向路，则称 i 为 j 的前驱，j 为 i 的后继。若（i，j）∈E，则 i 称为 j 的直接前驱，j 称为 i 的直接后继。
  - AOV网中不能出现回路(有向无环图）。



7.5.2 拓扑排序算法

- 拓扑排序：



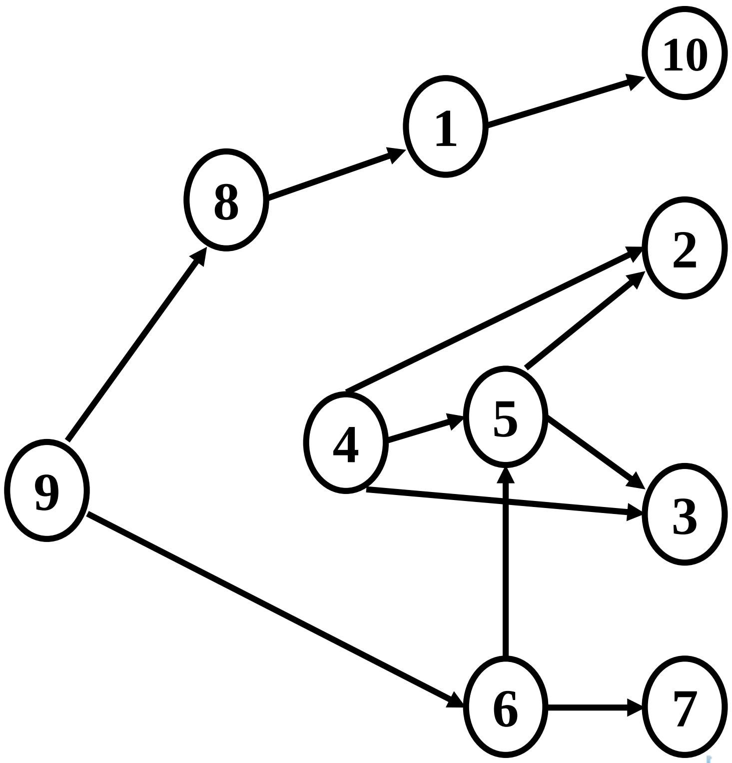
- 在图论中，由一个有向无环图的顶点组成的序列，当且仅当满足下列条件时， 该图的一个拓扑排序：
  - ① 每个顶点出现且只出现一次。
  - ② 若顶点A在序列中排在顶点B的前面，则在图中不存在从顶点B到顶点A的路径
- 或定义为：拓扑排序是对有向无环图的顶点的一种排序，它使得若存在一条从顶点A到顶点B的路径，则在排序中顶点B出现在顶点A的后面。
- 找到做事的先后顺序
- 每个AOV网都有一个或多个拓扑排序序列。

7.5.2 拓扑排序算法

AOV网示例：

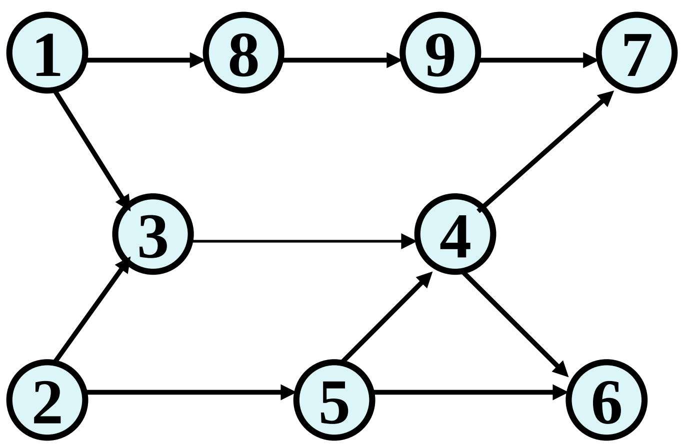
- 课程及课程间的先修关系可以用AOV网表示

课程代号	课程名称	先修课代号
1	计算机原理	8
2	编译原理	4,5
3	操作系统	4,5
4	程序设计	无
5	数据结构	4,6
6	离散数学	9
7	形式语言	6
8	电路基础	9
9	高等数学	无
10	计算机网络	1



7.5.2 拓扑排序算法

- 利用AOV网进行拓扑排序的基本思想：
  - (1) 从AOV网中选择一个没有前驱的顶点并且输出它；
  - (2) 从AOV网中删去该顶点和所有以该顶点为尾的弧；
  - (3) 重复上述两步，直到全部顶点都被输出，或AOV网中不存在没有前驱的顶点。



1 2 3 5 4

6 8 9 7

任何无环路的AOV网，其顶点都可以排成一个拓扑序列，并且其拓扑序列不一定是唯一的。

7.5.2 拓扑排序算法

拓扑排序算法

- 输入：有向图的邻接表，输出：所有顶点组成的拓扑序列
- 算法实现步骤：（使用队列）

1.建立空队列
2.扫描顶点表，将入度为0的顶点入队；
3.while（队列不空）{
    输出队头结点vi；
    记下输出结点的数目；
    删去与vi关联的出边（检查vi的出边表,将每条出边<vi,vj>的终点vj的入度减1）；
    若有入度为0的结点，入队
}
7.若输出结点个数小于n，则输出有环路；否则拓扑排序正常结束。

☞图中还有未输出的顶点，但已跳出循环处理。说明图中还剩下一些顶点，它们都有直接前驱。这时网络中必存在有向环；或

☞全部顶点均已输出，拓扑有序序列形成，拓扑排序完成。

7.5.2 拓扑排序算法

拓扑排序算法

```
void   Topologicalsort( AdjGraph G )
{   Queue  Q ; nodes = 0 ;
    MakeNull( Q ) ;
    for( v=1; v<=G.n ; ++v )
        if ( indegree[v] ==0 )  EnQueue( v, Q ) ;
    while ( !Empty( Q ) ) {
        v = Front(Q) ;
        DeQueue( Q ) ;
        cout << v ;   nodes ++ ;
        for( 邻接于 v 的每个顶点 w ){
            w入度减一；if (w入度为0) w入队;}
    }
    if ( nodes < n )  cout<<“图中有环路”  ;
}
```

```
for(p=G.verlist[v].firstedge； p； p=p->next)
{  w=p->adjvex;
   if( !(--indegree[w]))
    EnQueue(w,Q) ;
}
```

7.5.2 拓扑排序算法

- 关于广度优先拓扑排序的几点说明
  - 与先广搜索的差别：
    - 搜索起点是入度为0的顶点；
    - 需删除邻接于 v 的边（引入数组indegree[ ]或在顶点表中增加一个属性域indegree）。
    - 需对访问并输出的顶点计数（引入计数器nodes），判断是否有环路；
  - 采用栈数据结构进行广度优先拓扑排序?

7.5.2 拓扑排序算法

- 利用栈结构进行拓扑排序
  - 输入：有向图的邻接表，输出：所有顶点组成的拓扑序列
  - 算法实现步骤：（使用栈）

1. 建立空栈
2. 扫描顶点表，将入度为0的顶点入栈；
3. while（栈不空）{
   输出栈顶结点；
   记下输出结点的数目；
   删去与之关联的出边；
   若有入度为0的结点，入栈
}
7. 若输出结点个数小于n，则输出有环路；否则拓扑排序正常结束。

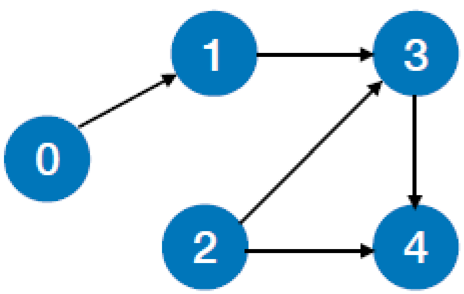
7.5.2 拓扑排序算法

- 利用栈结构进行拓扑排序

```
void   Topologicalsort( AdjGraph  G )
{    MakeNull( S ) ; count = 0 ;
     for( v=0; v<n ; ++v )
          if ( !indegree[v] )  Push( v, S ) ;
     while ( !Empty( S ) ) {
          v = Pop ( S ) ; printf( v );  ++count ;
          for( 邻接于 v 的每个顶点 w ) {
               if( !(--indegree[w]))
                    Push(S, w) ;
          }
     }
     if ( count < n )  cout<<"图中有环路" ;
}
```

7.5.2 拓扑排序算法

- 逆拓扑排序：
  - (1) 从AOV网中选择一个没有后继的顶点并且输出它；
  - (2) 从AOV网中删去该顶点和所有以该顶点为弧头的弧；
  - (3) 重复上述两步，直到全部顶点都被输出，或AOV网中不存在没有前驱的顶点。



7.5.2 拓扑排序算法

- 算法分析
- 对e条弧求各顶点的入度的时间复杂度是O(n+e)（邻接表）或O(n^2)（邻接矩阵）
- 初始建立入度为0 的顶点栈（队列），要检查所有顶点一次，执行时间为O(n)；
- 排序中，若AOV网无回路，则每个顶点入、出栈（队列）各一次，每个边被检查一次，执行时间为O(n+e)（邻接表）或O(n^2)（邻接矩阵）；
- 拓扑排序算法的时间复杂度为O(n+e)或O(n^2)。

思考题:

1、图的路径问题

(1)无向图两点之间是否有路径存在?

(2)有向图两点之间是否有路径存在?

2、图的环路问题

(1)无向图是否存在环路?

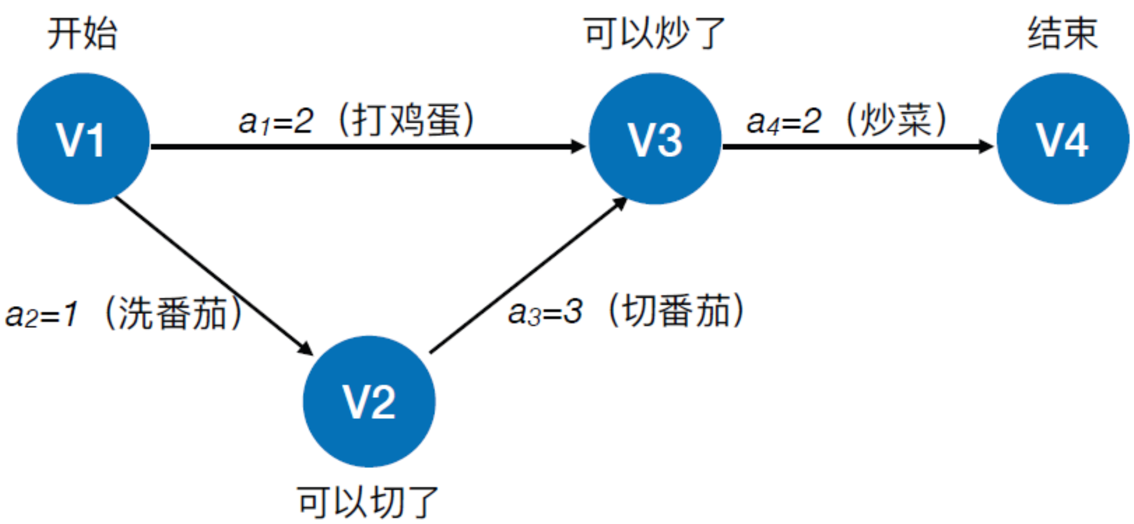
(2)有向图是否存在环路?

7.5 有向无环图的应用

- 7.5.1 表达式的描述
- 7.5.2 拓扑排序 (Topological Sort)
- 7.5.3 关键路径 (Critical Path)

7.5.3 关键路径算法

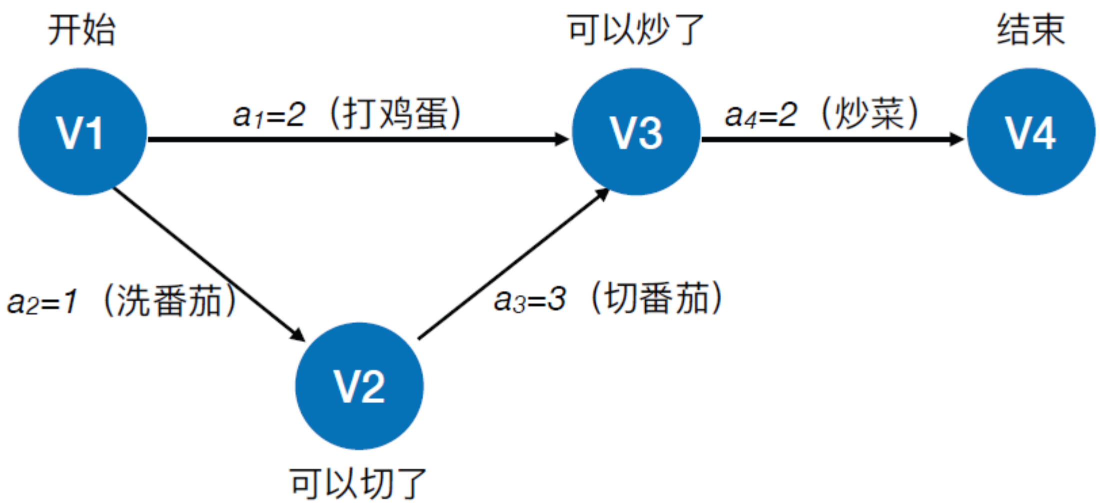
- **AOE网（Activity On Edge Network）**
  - 在带权的有向图中，用顶点表示事件，用边表示活动，边上权表示活动的开销（如持续时间），则称此有向图为边表示活动的网络，简称**AOE网**。



7.5.3 关键路径算法

AOE网的性质

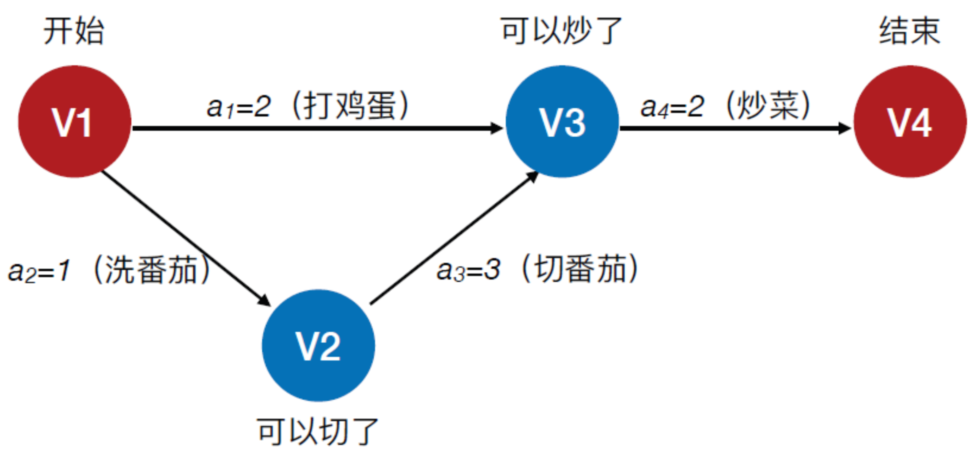
- 只有在某个顶点所代表的事件发生后，从该顶点出发的各有向边代表的活动才能开始；
- 只有在进入某一顶点的各有向边代表的活动已经结束，该顶点所代表的事件才能发生；
- 有些活动可以并行



7.5.3 关键路径算法

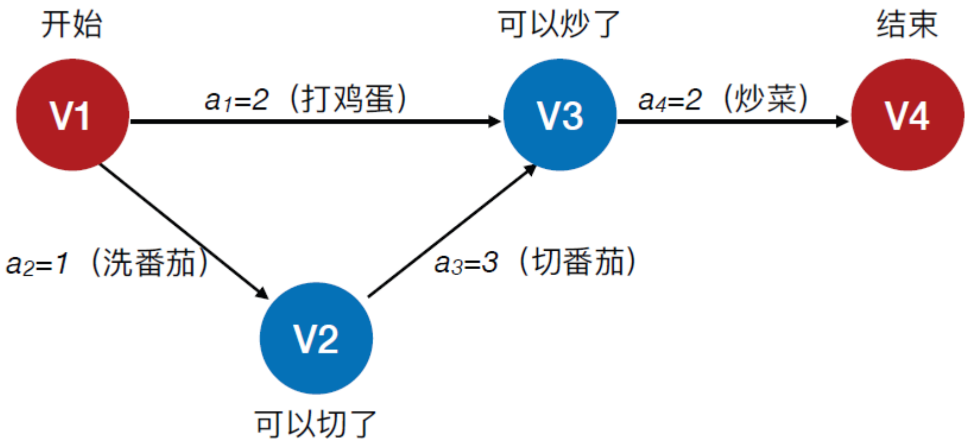
AOE网的性质

- 表示实际工程计划的AOE网应该是无环的，并且存在唯一的入度为0的开始顶点（源点），标志活动开始；和唯一的出度为0的结束点（汇点），标志活动结束。



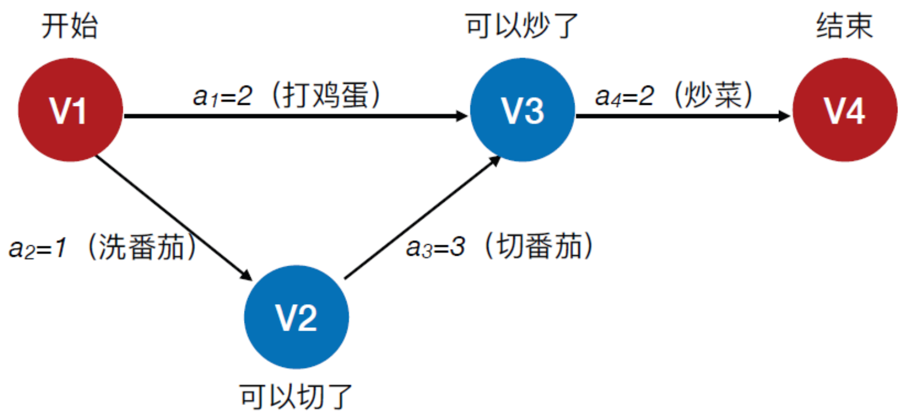
7.5.3 关键路径算法

- **AOE网研究的主要问题：**
  - 如果用AOE 网表示一项工程，那么仅仅考虑各个子工程之间的优先关系还不够，更多地是关心整个工程完成的最短时间是多少，哪些活动的延迟将影响整个工程进度，而加速这些活动能否提高整个工程的效率，因此AOE网有待研究的问题是:
    - (1)完成整个工程至少需要多少时间？
    - (2)哪些活动是影响工程进度的关键活动？



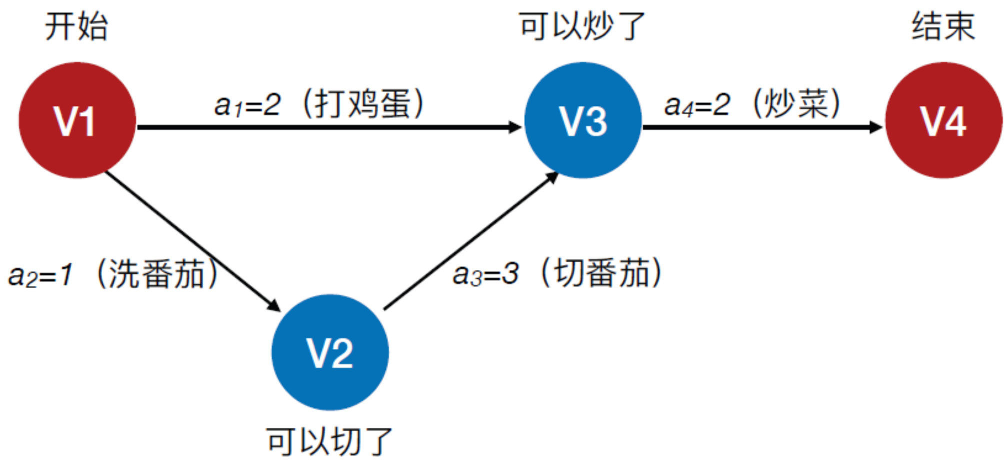
7.5.3 关键路径算法

- 路径长度、关键路径、关键活动：
  - 路径长度：是指从源点到汇点一条路径上所有活动的持续时间之和。
  - 关键路径：在AOE网中，由于有些活动可以并行，所以完成工程的最短时间是从源点到汇点的最大路径长度。因此，把从源点到汇点具有最大长度的路径称为关键路径。
  - 关键活动：关键路径上的活动称为关键活动。关键活动若不能按时完成，则整个工程的时间会延长
  - 一个AOE中，关键路径可能不只一条。



7.5.3 关键路径算法

E(1)=0
E(2)=0
E(3)=1
E(4)=4



最快多久开始做？
最快多久开始切？
最快多久开始炒？
最快多久开始吃？

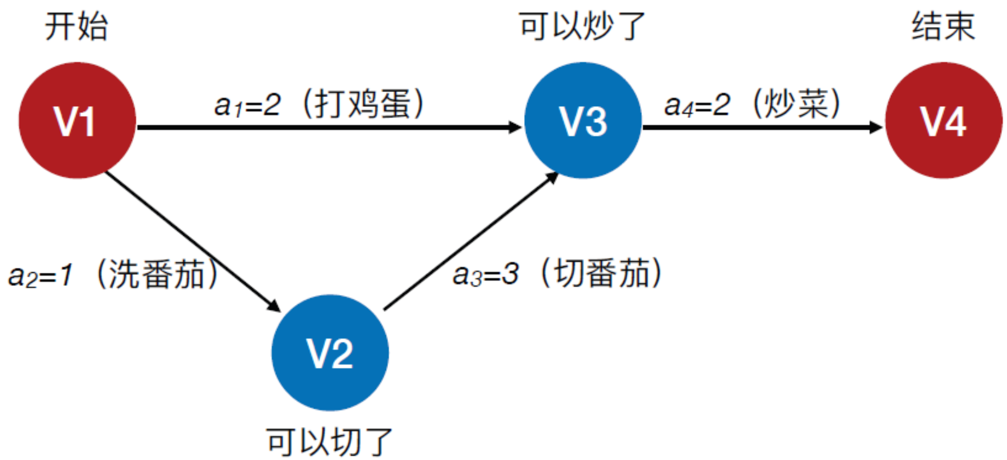
现在，马上！
1分钟
4分钟
6分钟

事件V_j 的最早可能发生时间VE(j)，决定了所有从V_j开始的活动开始的最早时间

活动a_i 的最早可能开始时间 E(i)，该弧起点事件发生的最早时间

7.5.3 关键路径算法

L(1)=2
L(2)=0
L(3)=1
L(4)=4



好饿好饿好饿
我真的好饿

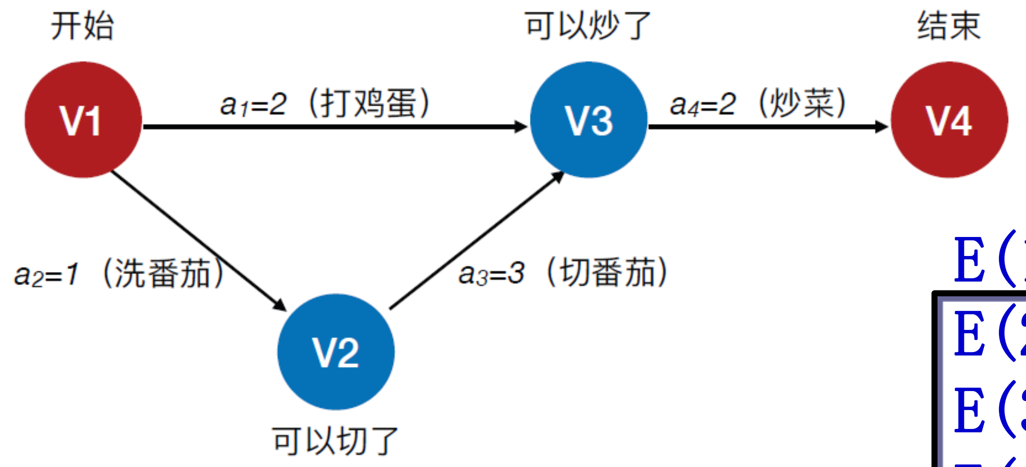
6分钟吃不上我可能会死

6分钟要结束
4分钟要开始炒
1分钟要开始切
现在要立刻开始

事件Vk的最迟发生时间VL(k)

活动ai 的最迟允许开始时间 L(i) ，该弧终点事件发生的最迟时间于活动所需时间差

7.5.3 关键路径算法



E(1)=0　L(1)=2

E(2)=0	L(2)=0
E(3)=1	L(3)=1
E(4)=4	L(4)=4

活动a_i 的最早可能开始时间 E(i)
活动ai 的最迟允许开始时间 L(i)

L(i)-E(i)意味着完成活动ai的时间余量。

时间余量表示在不增加完成整个工程所需总时间的情况下，活动a可以拖延的时间，若一个活动的时间余量为零，就是关键活动

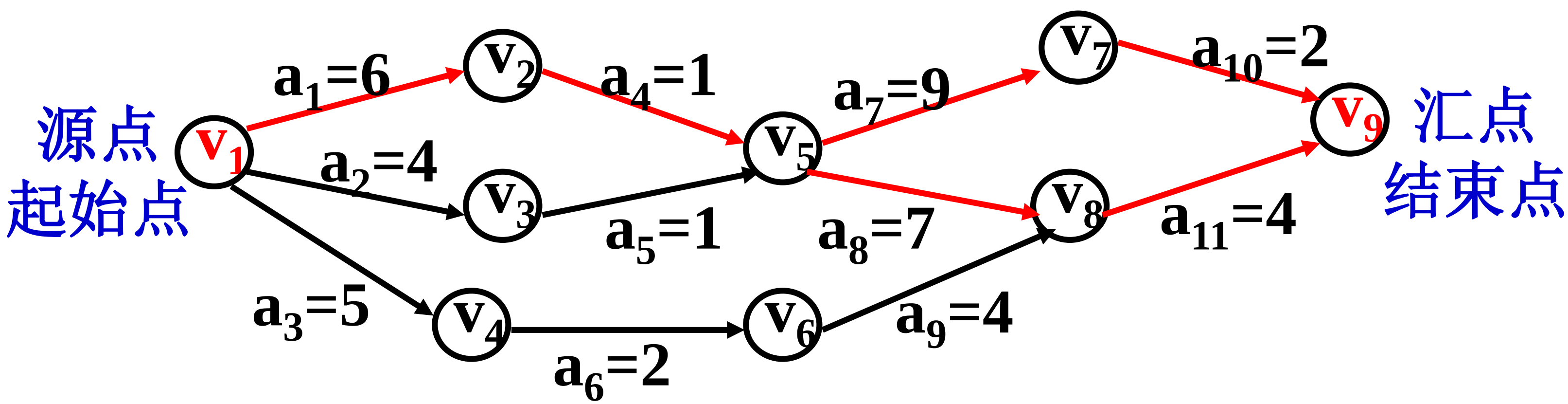
关键活动： L(i)=E(i)的活动。

由关键活动组成的路径就是关键路径

7.5.3关键路径算法

- 关键路径算法
- ①求所有事件V_j 的最早可能发生时间VE(j)
- ②求所有事件V_k的最迟发生时间VL(k)
- ③求活动a_i 的最早可能开始时间 E(i)
- ④求活动a_i的最迟允许开始时间 L(i)
- ⑤求所有活动的时间余量

时间余量为0活动就是关键活动，由关键活动可得关键路径

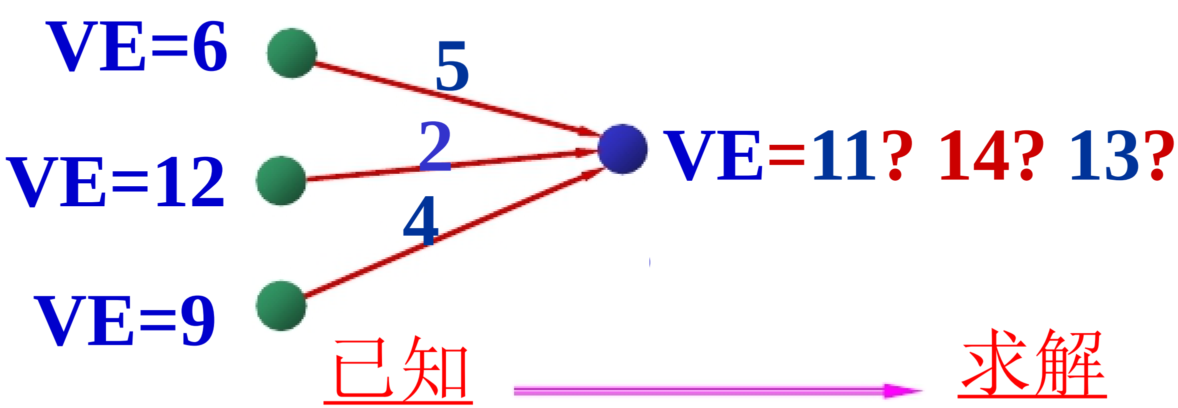


7.5.3 关键路径算法

- 利用拓扑排序算法求关键路径和关键活动
  - （1）前进阶段：计算VE[j]。从源点V_1出发，令VE(1) = 0，按拓扑序列次序求出其余各顶点事件的最早发生时间：

$$VE(j) = \max_{j \in T}\{ VE(i)+ACT[i][j] \}$$

- 其中T是以顶点V_j为尾的所有边的弧头顶点的集合(2≤j≤n)
- 如果网中有回路，不能求出关键路径则算法中止；否则转（2）

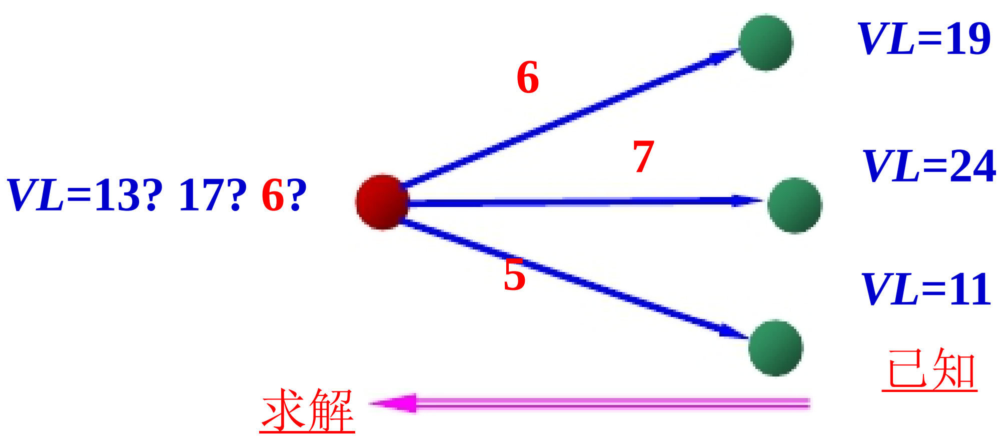


7.5.3 关键路径算法

- 利用拓扑排序算法求关键路径和关键活动
  - （2）回退阶段：计算VL[j]从汇点V_n出发，令VL(n) = VE(n)，按逆拓扑有序求其余各顶点的最晚发生时间：

$$VL(j) = \min_{k \in S}\{ VL(i)-ACT[j][i] \}$$

- 其中S是以顶点V_j为头的所有边的弧尾顶点的集合(2≤j≤n-1)



■ （3） 计算E(i) 和L(i)

求每一项活动a_i的最早开始时间:

$$E(i) = VE(j)$$

求每一项活动a_i的最晚开始时间:

$$L(i) = VL(k) - ACT[j][k]$$

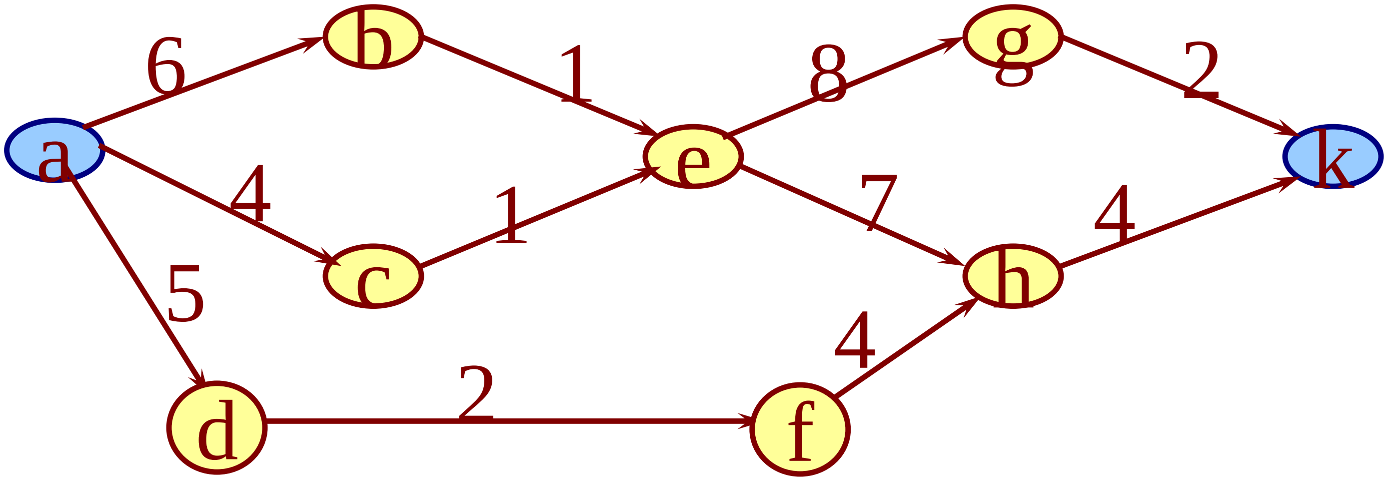
■ （4） 若某条边满足E(i) = L(i)，则它是关键活动。

◆为了简化算法，可以在求关键路径之前已经对各顶点实现拓扑排序，并按拓扑有序的顺序对各顶点重新进行了编号。

7.5.3 关键路径算法

- **AOE**网可以用邻接矩阵或邻接表表示；矩阵中行下标**i**表示起点，列下标**j**表示终点，**ACT[i][j]>0**表示时间，**ACT[i][j]=-1**表示无边。

关键路径和关键活动分析计算示例：



	a	b	c	d	e	f	g	h	k
ve	0	6	4	5	7	7	15	14	18
vl	0	6	6	8	7	10	16	14	18

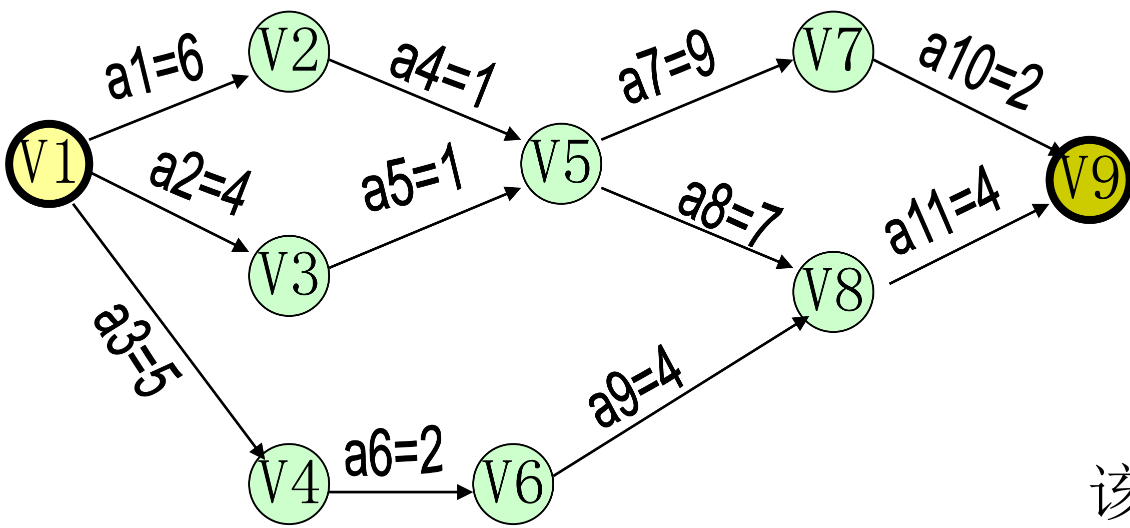
关键路径算法步骤：

（1）从开始点V1出发，令VE(1)=0,按拓朴排序序列求其它各顶点的最早发生时间

VE(k)=max{VE(j)+act(<j,k>)}

(vj为以顶点vk为弧头的所有弧的弧尾对应的顶点集合)

顶点	VE(i)	VL(i)
v_1	0	
v_2	6	
v_3	4	
v_4	5	
v_5	7，5	
v_6	7	
v_7	16	
v_8	14，11	
v_9	18，18	



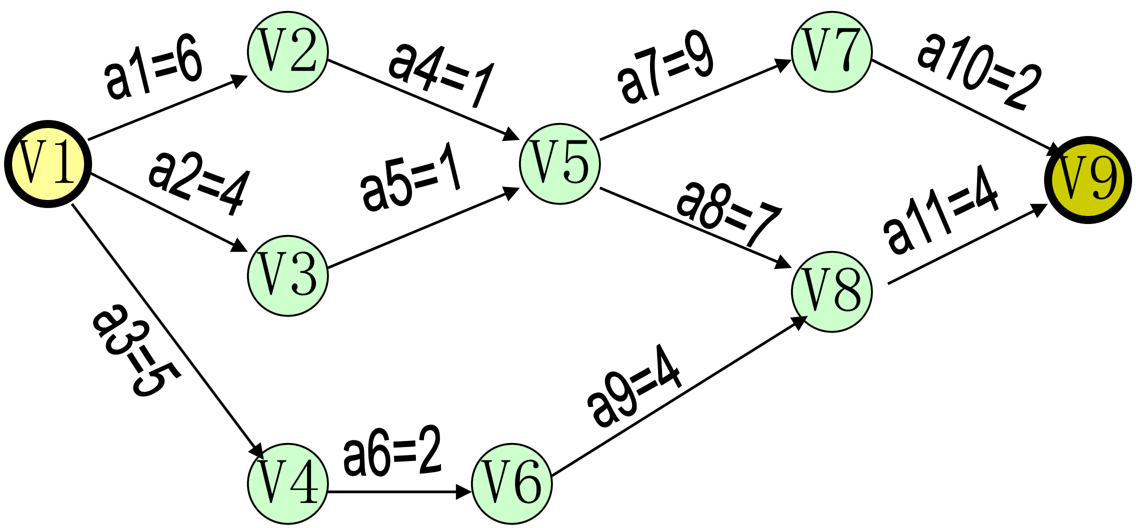
该表次序为一拓扑排序序列

关键路径算法步骤：

（2）从完成点v_n出发，令VL(n)=VE(n),按逆拓朴排序序列求其它各顶点的最迟发生时间

VL(j)=min{VL(k)-ACT(<j,k>)}

(vk为以顶点vj为弧尾的所有弧的弧头对应的顶点集合)

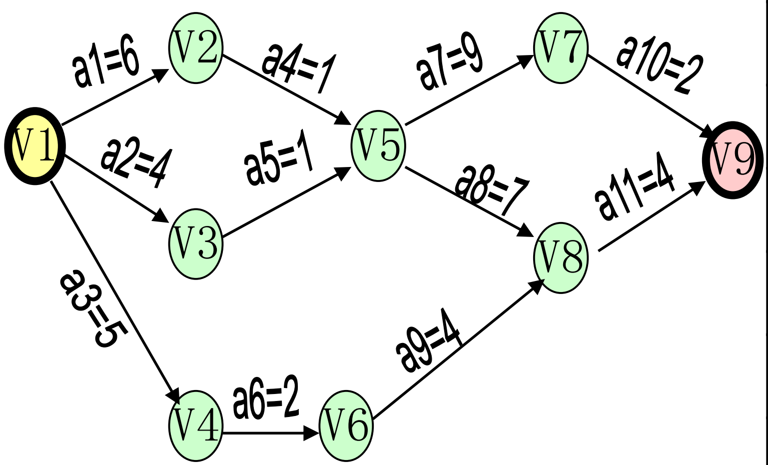


顶点	VE(i)	VL(i)
v_1	0	0，2，3
v_2	6	6
v_3	4	6
v_4	5	8
v_5	7	7，7
v_6	7	10
v_7	16	16
v_8	14	14
v_9	18	18

关键路径算法步骤：

（3）求每一项活动ai(vj,vk)：

E(i)=VE(j)　　L(i)=VL(k)-ACT(ai)



顶点	VE(i)	VL(i)
v_1	0	0
v_2	6	6
v_3	4	6
v_4	5	8
v_5	7	7
v_6	7	10
v_7	16	16
v_8	14	14
v_9	18	18

活动	E(i)	L(i)	L(i)-E(i)
a_1	0	0	0
a_2	0	2	2
a_3	0	3	3
a_4	6	6	0
a_5	4	6	2
a_6	5	8	3
a_7	7	7	0
a_8	7	7	0
a_9	7	10	3
a_{10}	16	16	0
a_{11}	14	14	0

关键活动：选取E(i)=L(i)的活动。

关键路径：

（1） v1→ v2 → v5 → v7 → v9

（2） v1→ v2 → v5 → v8 → v9