7.5 有向无环图及其应用

问题:

假设以有向图表示一个工程的施工图或程序的数据流图，则图中不允许出现回路。对整个工程和系统，人们关心的是两个方面的问题：一是工程能否顺利进行；二是估算整个工程完成所必需的最短时间。

对应于有向图，即为进行拓扑排序和求关键路径的操作。

7.5.1 拓扑排序

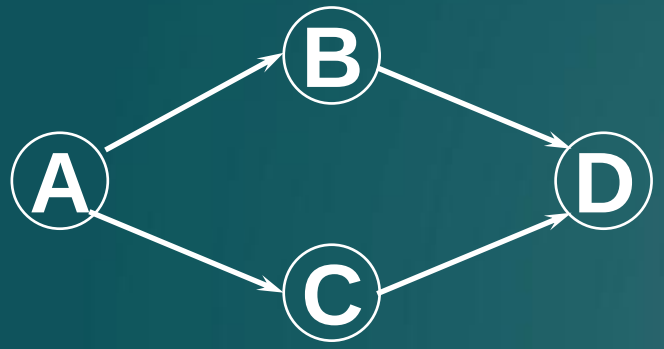
何谓“拓扑排序”？

简单地说，由某个集合上的一个偏序得到该集合上的一个全序，这个操作称之为拓扑排序。

如何得到有向图的一个拓扑序列？

对有向图进行如下操作：按照有向图给出的次序关系，将图中顶点排成一个线性序列，对于有向图中没有限定次序关系的顶点，则可以人为加上任意的次序关系。由此所得顶点的线性序列称之为拓扑有序序列。

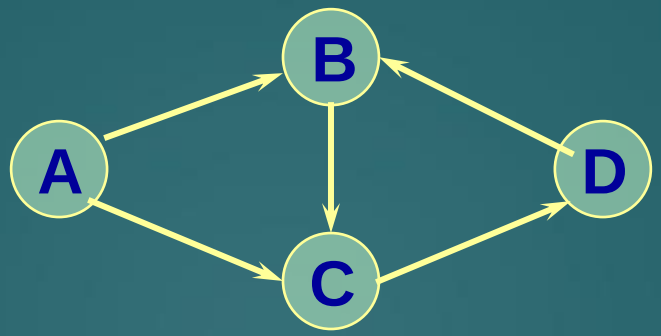
例如：对于下列有向图



可求得拓扑有序序列：

A B C D　或　A C B D

反之：对于下列有向图



不能求得它的拓扑有序序列。

因为图中存在一个回路

{B, C, D}

如何进行拓扑排序？

一、从有向图中选取一个**没有前驱**的顶点，并输出之；

二、从有向图中删去此顶点以及所有以它为尾的弧；

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。后一种情况说明有向图中存在环。

例如，对左图的拓扑排序过程如下所示：

a b h c d g f e

在算法中需要用定量的描述替代定性的概念

没有前驱的顶点 = 入度为零的顶点

删除顶点及以它为尾的弧 = 弧头顶点的入度减1

采用邻接表作有向图的存储结构，且在头结点中增加一个存放顶点入度的数组(indegree)。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。

```
Status Topologicaisort(ALGraph G) { //有向图G采用邻接表
//存储结构。若G无回路，则输出G的顶点的一个拓扑序列并返
//回OK，否则ERROR。
    FindInDegree(G,indegree);   //对各顶点求入度
    InitStack(S);     //建零入度顶点栈S
   for ( i=0; i<G.vexnum; ++i)
     if (!indegree[i])  Push(S, i);       //入度为零的顶点入栈
```

```
    count=0;          //对输出顶点计数
    while (!EmptyStack(S)) {
    Pop(S, v);  ++count;  printf(i,G.vertices[i].data);//输出i号
顶点//并计数。
    for (p= G.vertices[i]. Firstarc; p;  p=p->Nextarc){
        k=p->adjvex;  // 对i号顶点的每个邻接点的入度减1
       if (!(--indegree[k]))  Push(S, k);  //若入度为零，则入栈
     }//for
}//while
if (count<G.vexnum) return ERROR ; //图中无回路
else   return OK;
}//Topologicaisort
```

算法 7.12

7.5.2 关键路径

问题:

假设以有向网表示一个施工流图，其中，顶点表示事件，弧表示活动，弧上的权值表示活动的持续时间。该网络称之为AOE网络。

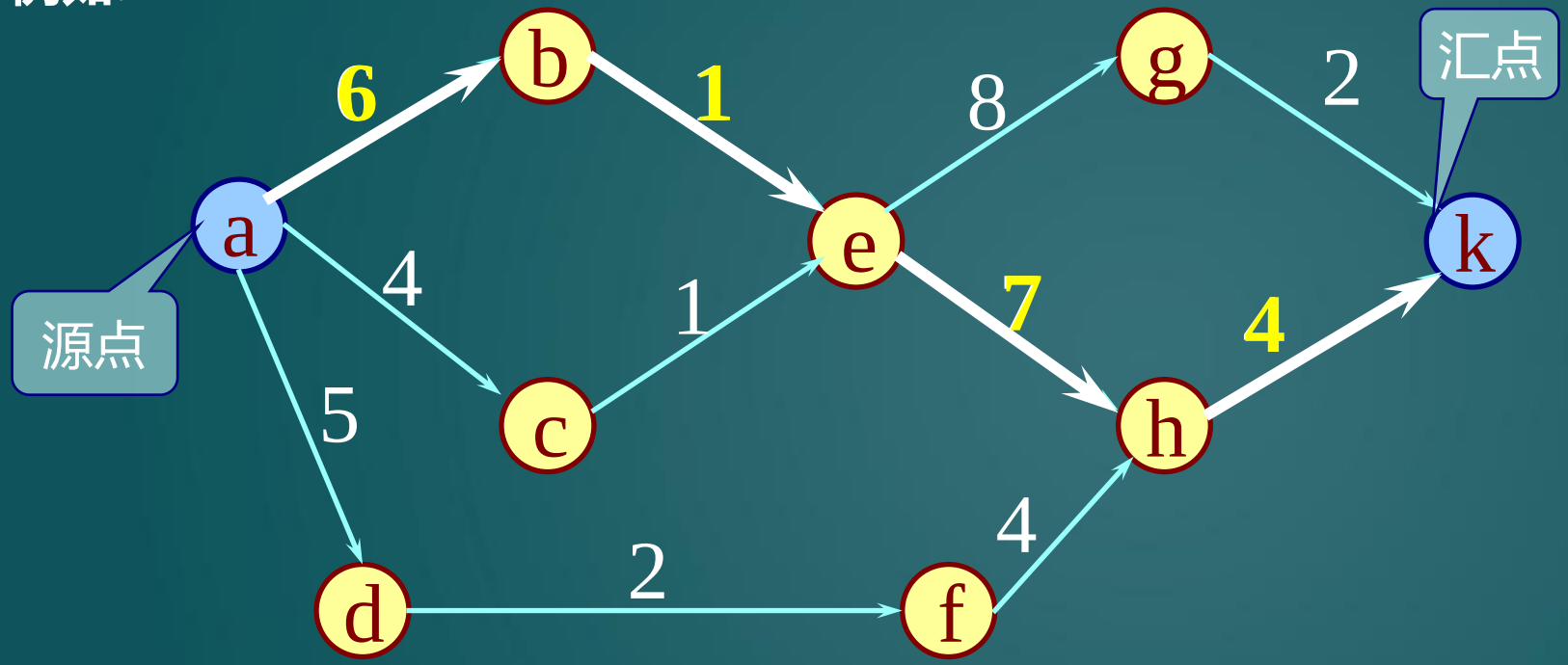
假设以该AOE网络表示一个施工流图，则有待研究的问题是：

（1）完成整项工程至少需要多少时间？

（2）哪些工程是影响整个工程进度的关键？

整个工程完成的时间为：从有向图的**源点**到**汇点**的最长路径。

例如:



关键活动”指的是：该弧上的**权值增加**将使有向图上的**最长路径的长度增加。**

如何求关键活动？

“事件（顶点v_j）” 的 最早发生时间 *ve*(j)

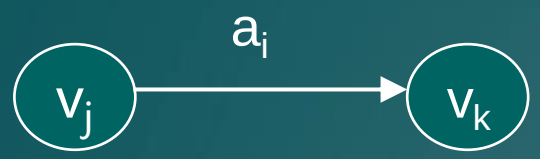
ve(j) = 从源点到顶点V_j的最长路径长度；

这个时间决定了所有以V_j为尾的弧所表示的活动的最早开始时间。

“事件（顶点v_k）” 的 最迟发生时间 vl(k) ，表示在不推迟整个工程完成的前提下，事件最迟发生的时间。

vl(k) = 工程完成时间-从顶点V_k到汇点的最长路径长度。

假设第 i 条弧为 <j, k>，即如下图示：



则 对活动a_i而言，其最早开始时间 e (i)

e (i) = ve (j) ；

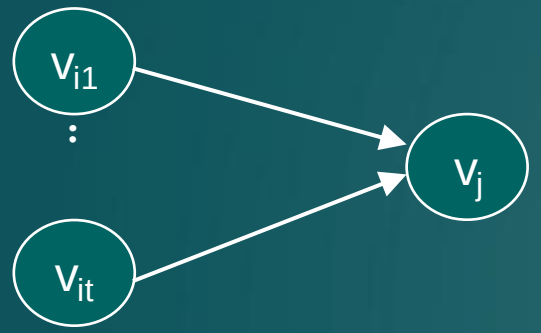
最迟开始时间 l (i)

l (i) = vl (k) – dut (<j,k>) ；

其中： dut (<j,k>) 为活动a_i的权（持续时间）。

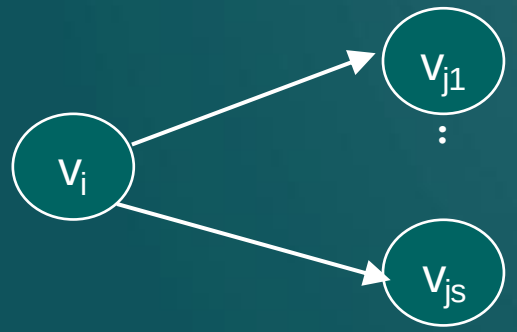
事件发生时间的计算公式：

事件最早发生时间的计算公式：

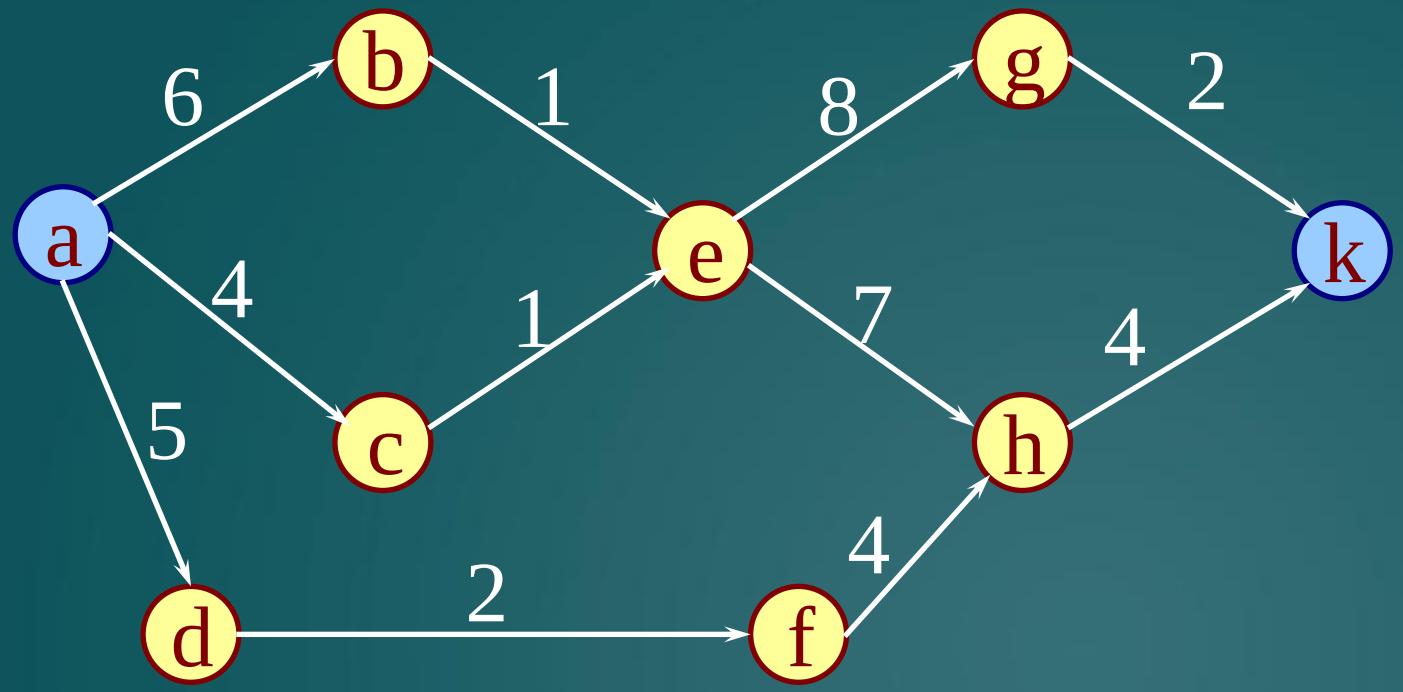


$$\begin{cases} ve(\text{源点}) = 0; \\ ve(j) = \max\{ve(ik) + dut(<ik, j>)\} \end{cases}$$

事件最晚发生时间的计算公式：



$$\begin{cases} Vl(\text{汇点}) = ve(\text{汇点}); \\ vl(i) = \min\{vl(jl) - dut(<i, jl>)\} \end{cases}$$

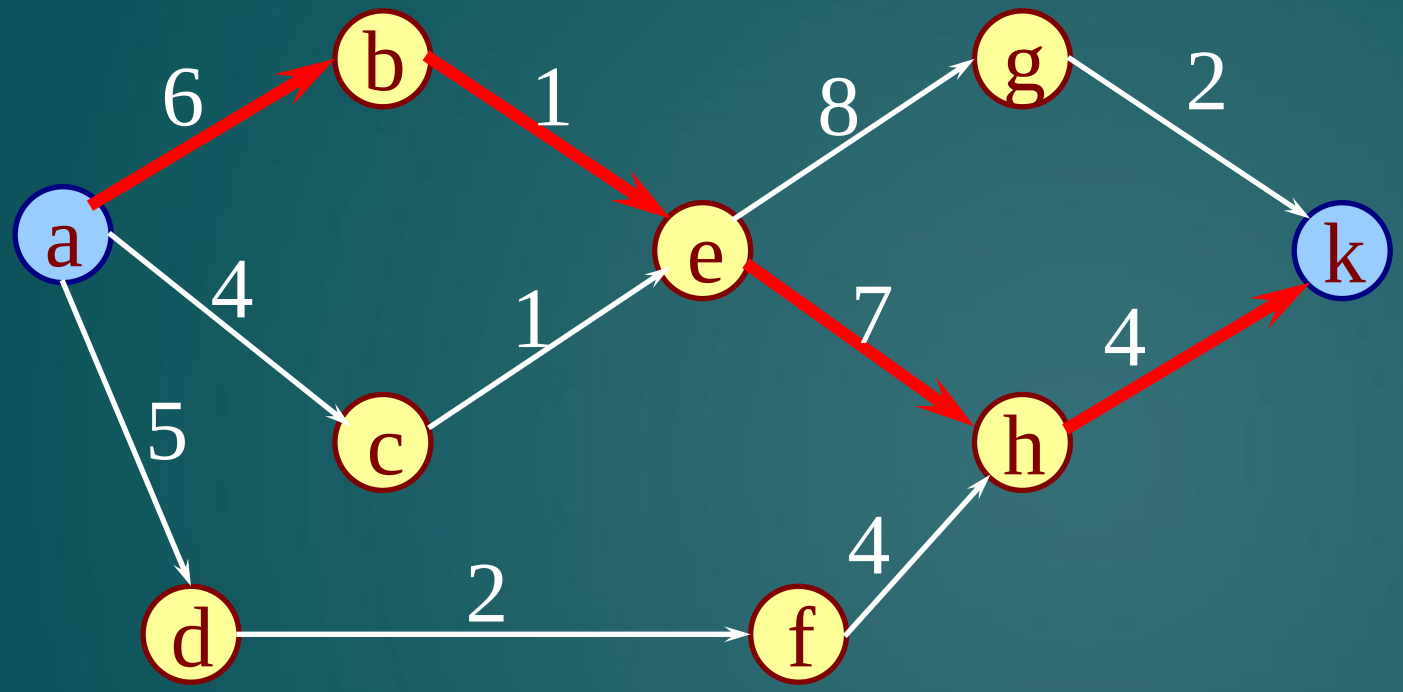


	a	b	c	d	e	f	g	h	k
ve	0	6	4	5	7	7	15	14	18
vl	0	6	6	8	7	10	16	14	18

拓扑有序序列: a - d - f - c - b - e - h - g - k

	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

	ab	ac	ad	be	ce	df	eg	eh	fh	gk	hk
权	6	4	5	1	1	2	8	7	4	2	4
e	0	0	0	6	4	5	7	7	7	15	14
l	0	2	3	6	6	8	8	7	10	16	14
	✓			✓				✓			✓



红色路径表示关键路径，其中的活动代表关键活动。

算法的实现要点:

显然，求ve的顺序应该是按**拓扑有序**的次序；

而　　求vl的顺序应该是按拓扑逆序的次序；

所谓拓扑逆序序列即为拓扑有序序列的逆序列，

因此，应该在拓扑排序的过程中，另设一个“**栈**”记下拓扑有序序列。

求关键路径的算法:

1、输入e条弧<j,k>，建立AOE网的存储结构；

2、从源点v0出发，令ve[0]=0，按拓扑有序求其余各顶点的最早发生时间ve[i]。若得到的拓扑序列中的顶点个数小于网中顶点数，则说明网中存在环，不能求关键路径，算法终止；否则执行步骤(3）；

3、从汇点vn出发，令vl[n-1]= ve[n-1] ，按逆拓扑有序求其余各顶点的最迟发生时间vl[i]；

4、根据个顶点的ve和 vl值 ，求每条弧的最早开始时间e(s)和最迟开始时间l(s)。若某弧满足条件e(s)=l(s)，则为关键活动。

求关键路径的具体算法：讲义P