第7章 图

杭州電子科技大学
HANGZHOU DIANZI UNIVERSITY

7.1 图的定义和基本术语

7.2 图的存储结构

7.3 图的遍历

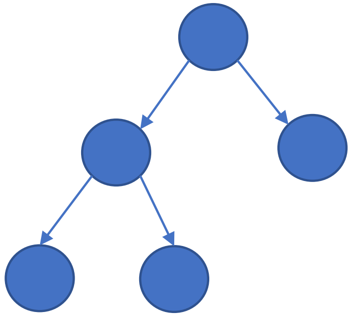
7.4 图的应用

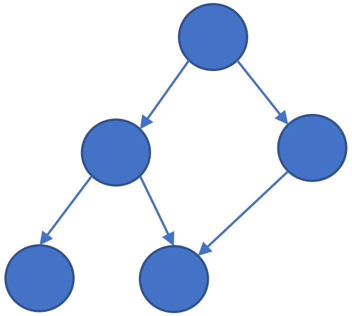
7.4 图的应用

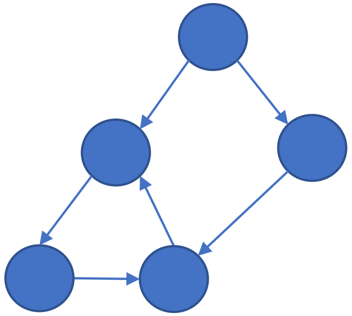
- 最小生成树
- 最短路径
- **拓扑排序**
- 关键路径

有向无环图及其应用

有向无环图：无环的有向图，简称DAG图 (Directed Acycline Graph)


有向树


有向无环图


有向图

- 有向无环图用来描述一个工程或系统的进行过程（通常把计划、施工、生产、程序流程当是一个工程）。
- 一个工程可以分为若干子工程，只要完成了这些子工程（活动），就可以让整个工程完成。

有向无环图及其应用

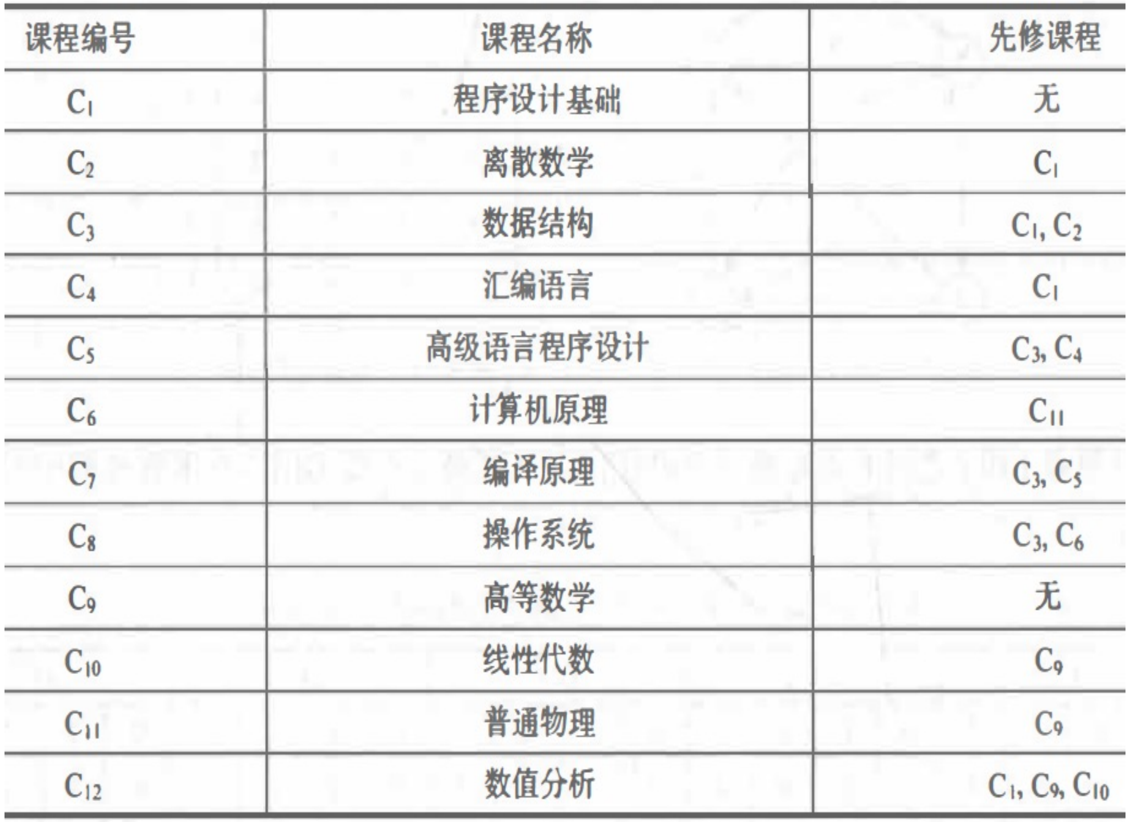
AOV网： **拓扑排序**

用一个有向图表示一个工程的子工程及其相互制约的关系，其中以**顶点**表示**活动**，**弧**表示活动之间的优先制约**关系**，称这种有向图为**顶点表示活动的网**，简称**AOV网**(Activity On Vertex network)

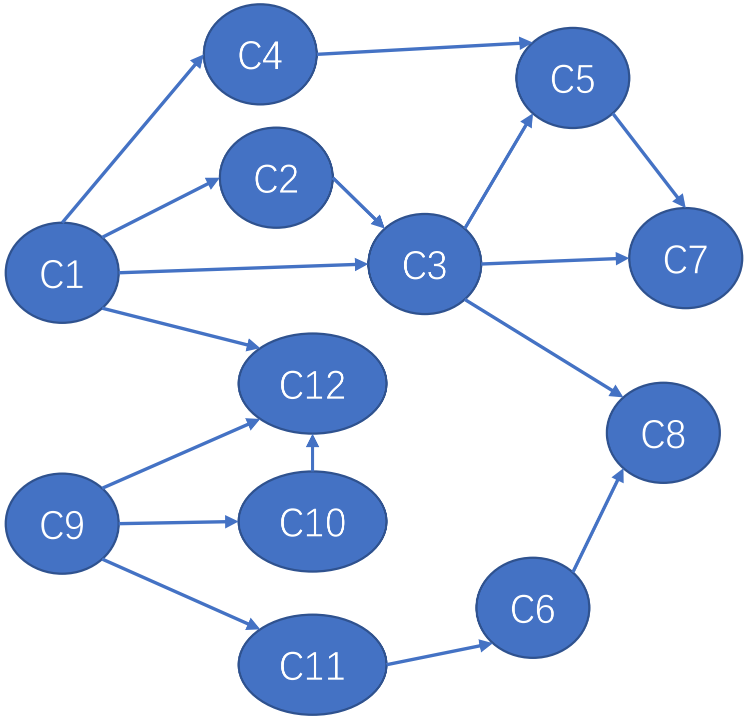
AOE网： **关键路径**

用一个有向图表示一个工程的子工程及其相互制约的关系，**弧表示活动**，以顶点表示活动的开始或结束事件，称这种有向图为**边表示活动的网**，简称**AOE网**(Activity On Edge network)

拓扑排序例：排课表

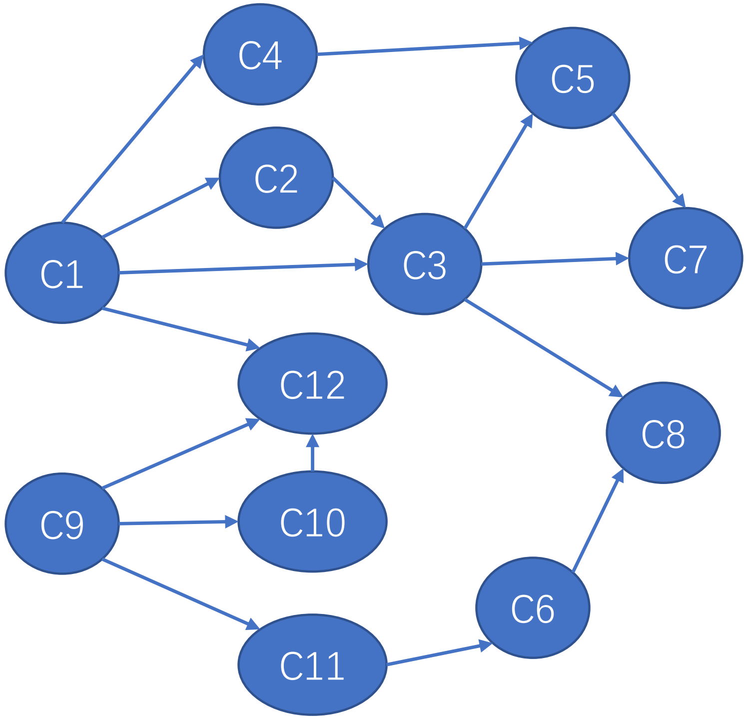


课程编号	课程名称	先修课程
C_1	程序设计基础	无
C_2	离散数学	C_1
C_3	数据结构	C_1, C_2
C_4	汇编语言	C_1
C_5	高级语言程序设计	C_3, C_4
C_6	计算机原理	C_{11}
C_7	编译原理	C_3, C_5
C_8	操作系统	C_3, C_6
C_9	高等数学	无
C_{10}	线性代数	C_9
C_{11}	普通物理	C_9
C_{12}	数值分析	C_1, C_9, C_{10}



AOV网的特点

- 若从 i 到 j 有一条有向路径，则 i 是 j 的前驱；j 是 i 的后继
- 若<i, j>是网中有向边，则 i 是 j 的直接前驱；j 是 i 的直接后继
- AOV网中不允许有回路，因为如果有回路存在，则表明某项活动以自己为先决条件，显然这是荒谬的

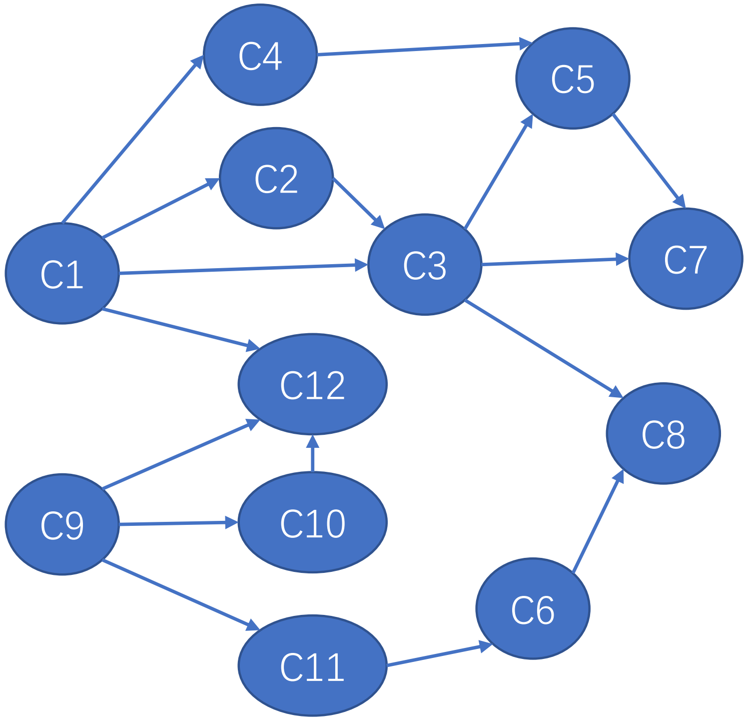


问题：如何判别AOV网中是否存在回路?

拓扑排序

在AOV网没有回路的前提下，我们将全部活动排成一个线性序列，使得若AOV网中有弧<i, j>存在，则这个序列中，i 一定排在 j 的前面，具有这种性质的线性序列成为拓扑有序序列，相应的拓扑有序排序的算法成为拓扑排序。

拓扑排序方法

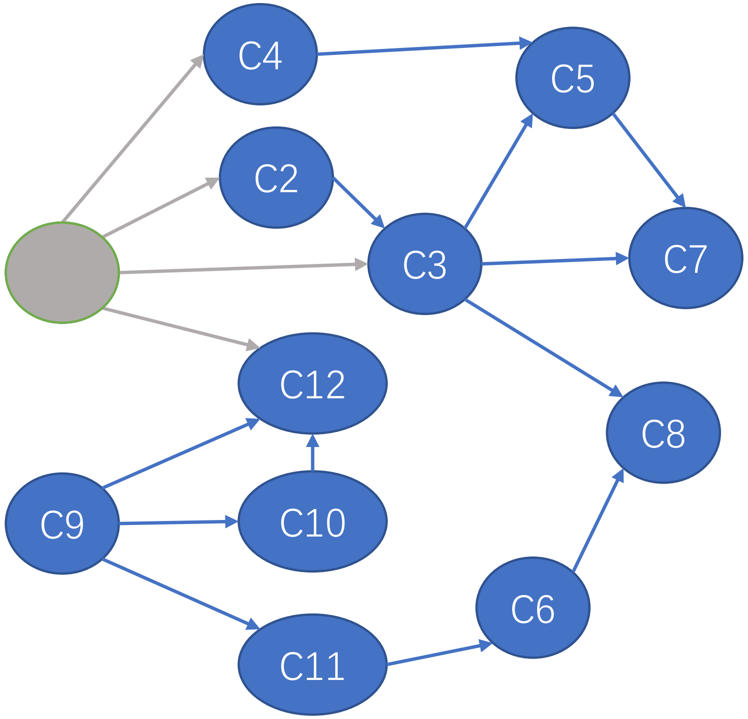


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。

拓扑排序：

C1，

拓扑排序方法

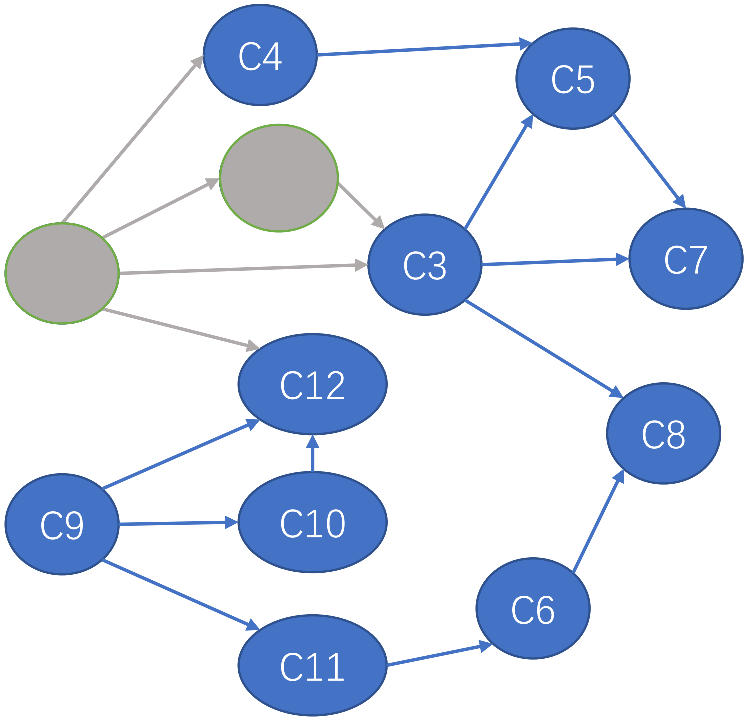


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，

拓扑排序方法



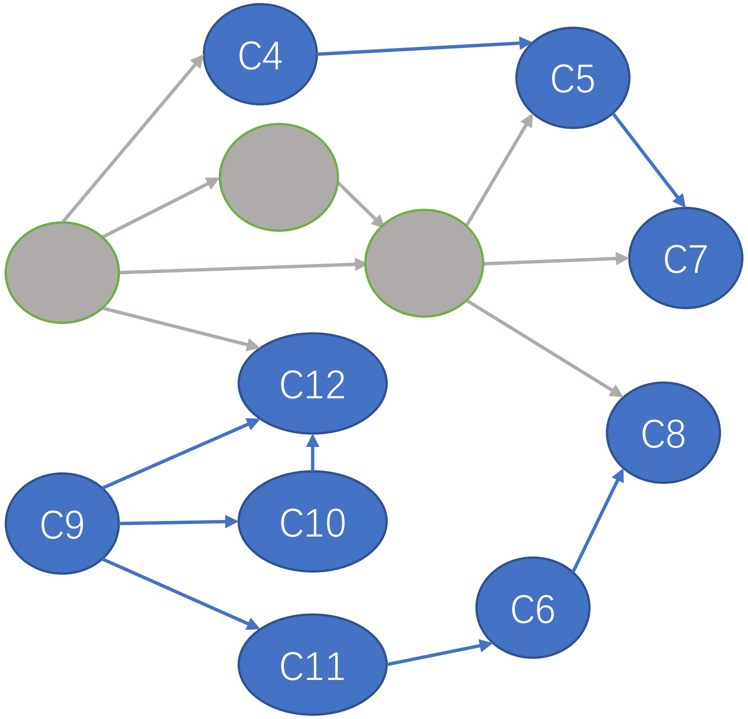
- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当

图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，

拓扑排序方法

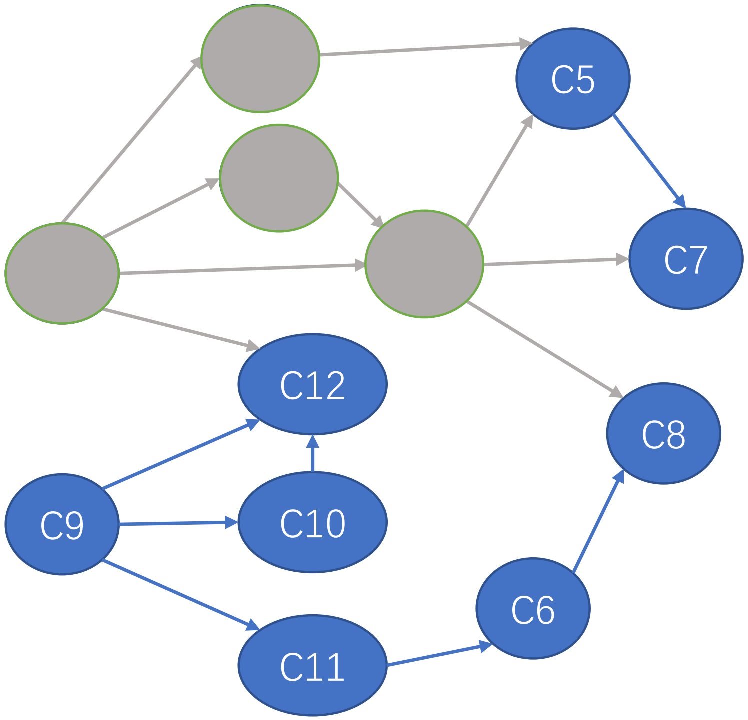


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3， C4，

拓扑排序方法

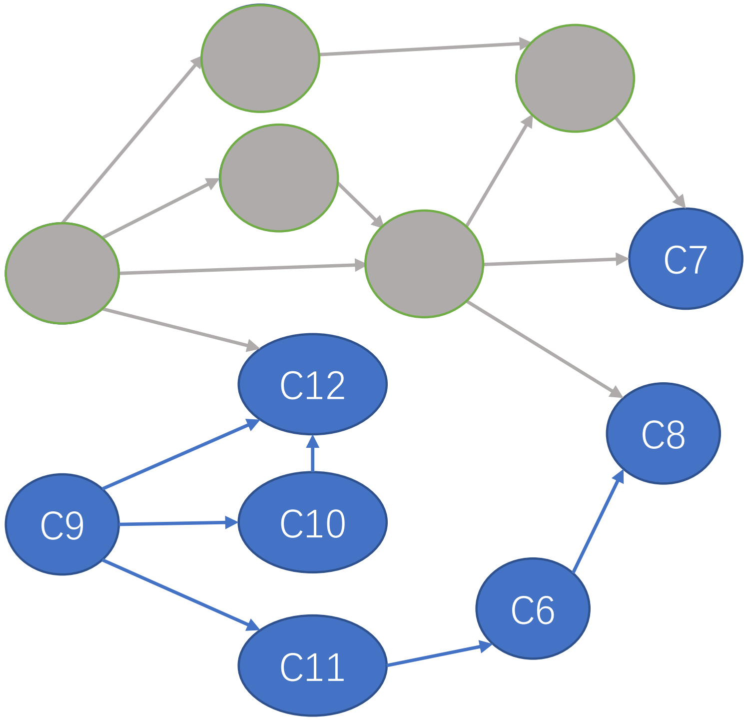


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，

拓扑排序方法

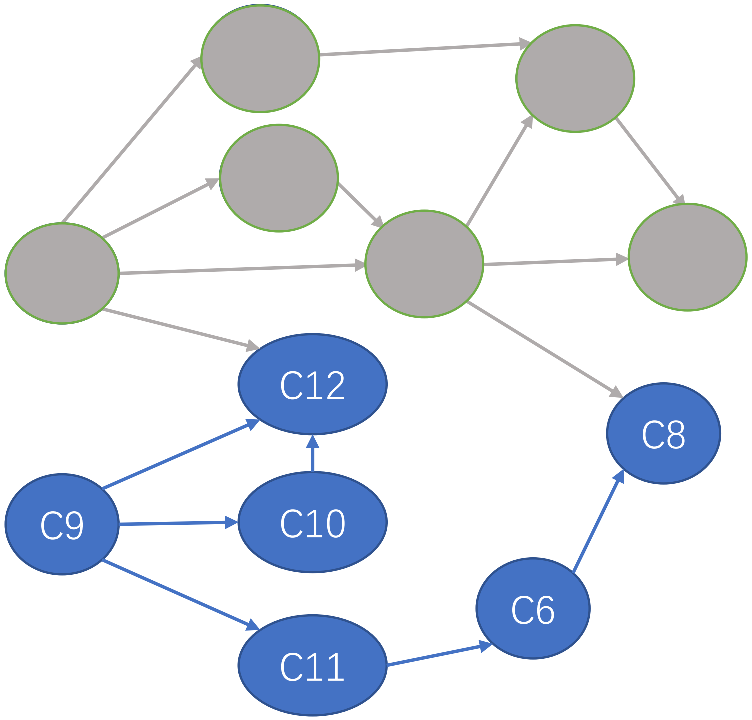


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，C7，

拓扑排序方法

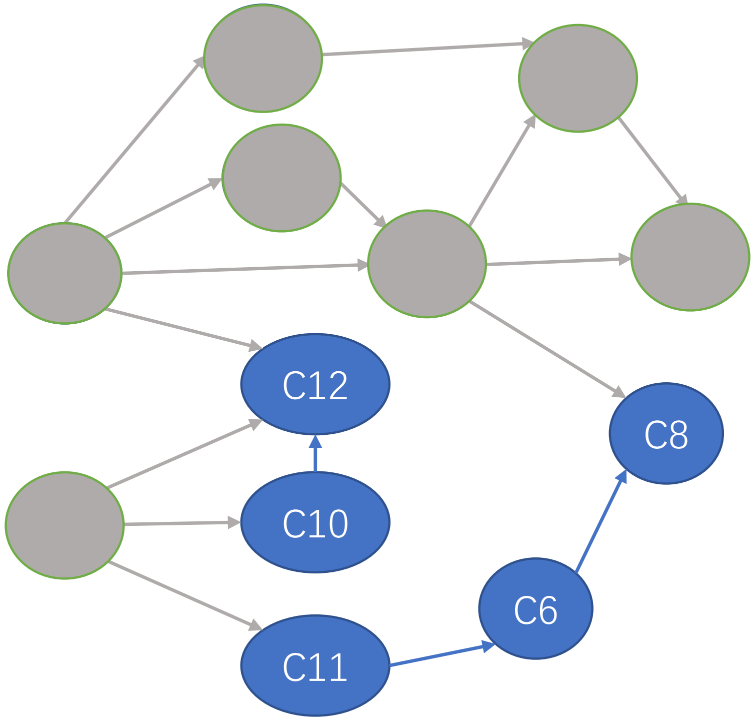


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，C7，C9，

拓扑排序方法

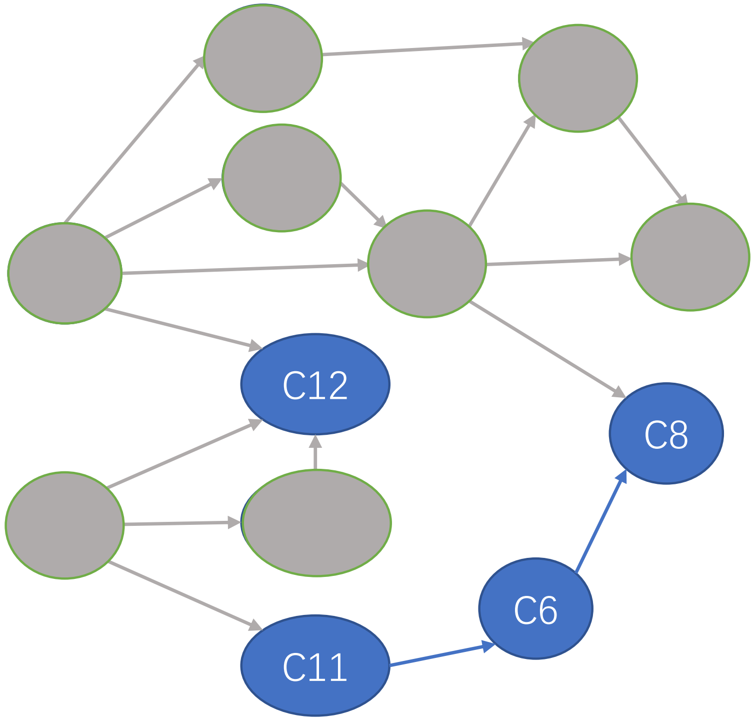


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，C7，C9，C10，

拓扑排序方法

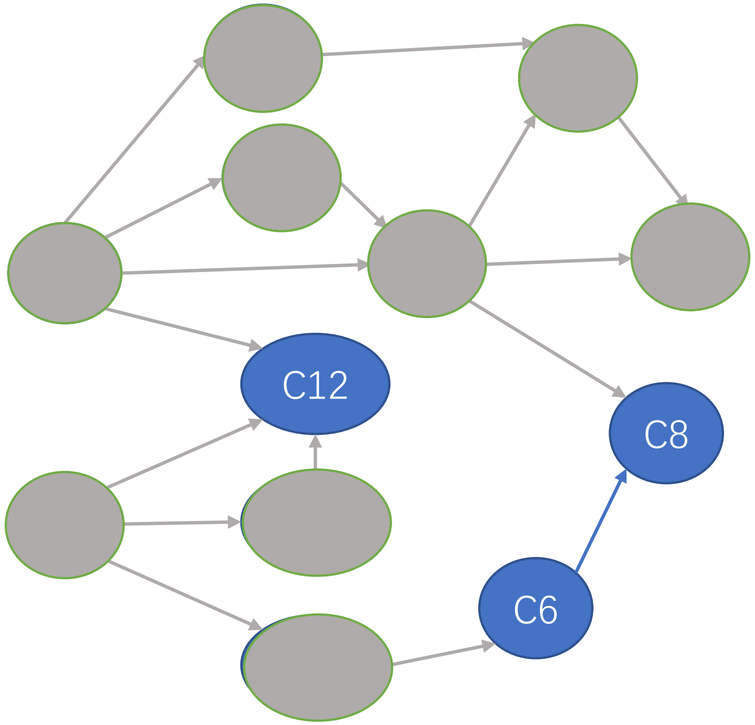


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，C7，C9，C10，C11，

拓扑排序方法

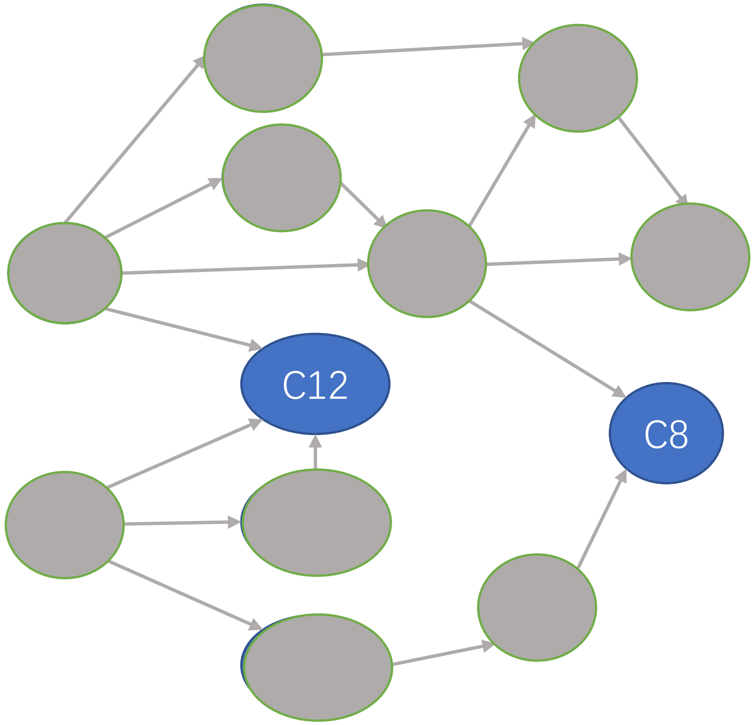


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，C7，C9，C10，C11，C6，

拓扑排序方法

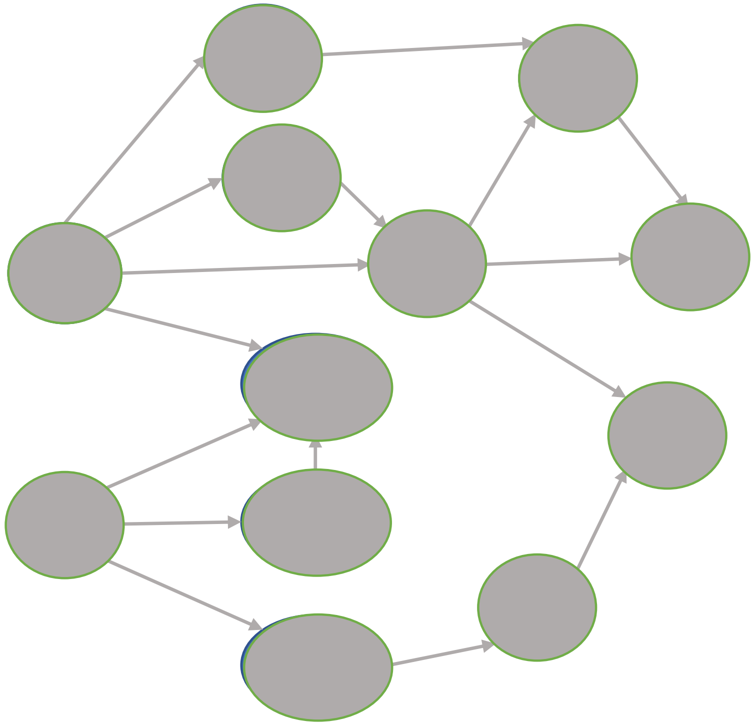


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，C7，C9，C10，C11，C6，C8，

拓扑排序方法

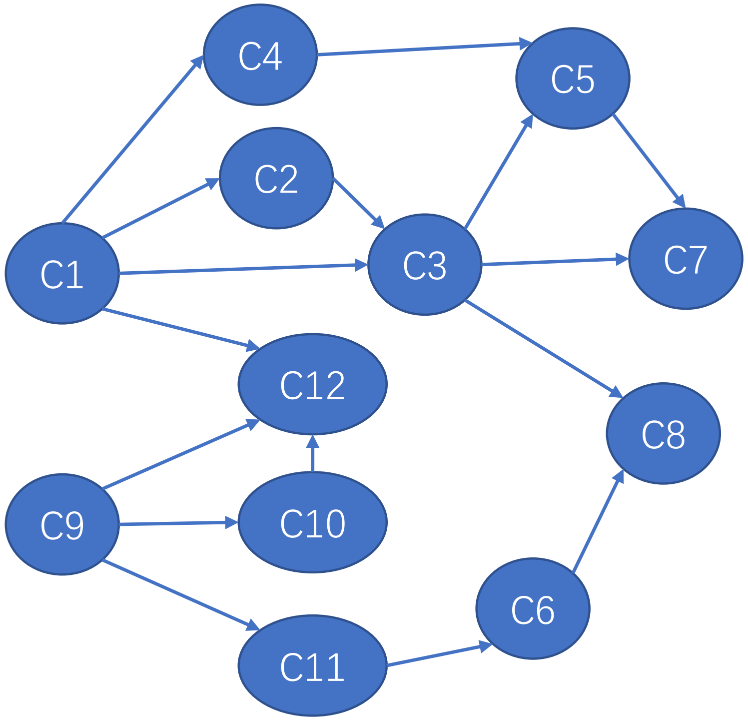


- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

拓扑排序：

C1，C2，C3，C4，C5，C7，C9，C10，C11，C6，C8，C12

拓扑排序方法



- 在有向图中选一个没有前驱的节点且输出之
- 从图中删除该顶点和所有以它为弧尾的弧。
- 重复上述两步，直至全部顶点均已输出，或者当图中不存在无前驱的顶点为止

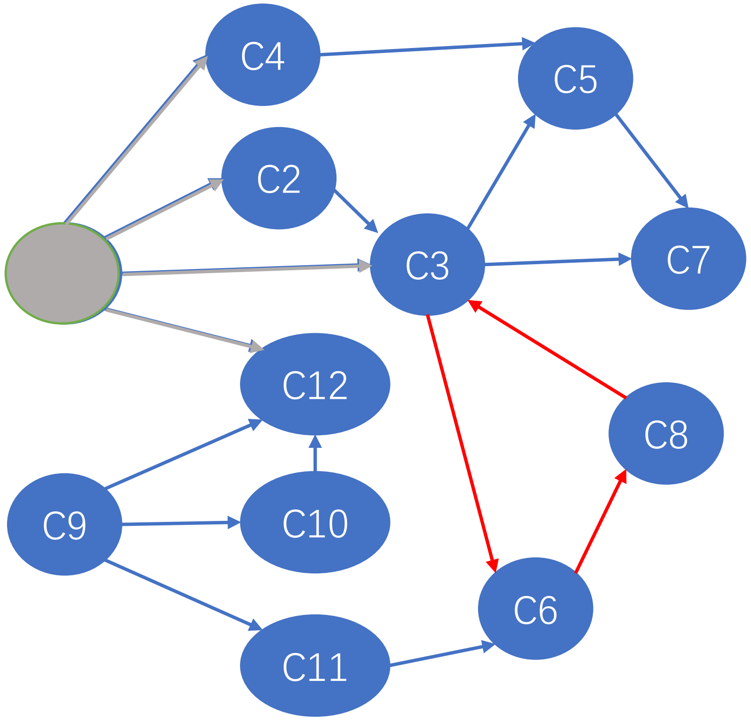
拓扑排序：

一个AOV网的拓扑序列不是唯一的

C1，C2，C3，C4，C5，C7，C9，C10，C11，C6，C8，C12

C9，C10，C11，C6，C1，C12，C4，C2，C3，C5，C7，C8

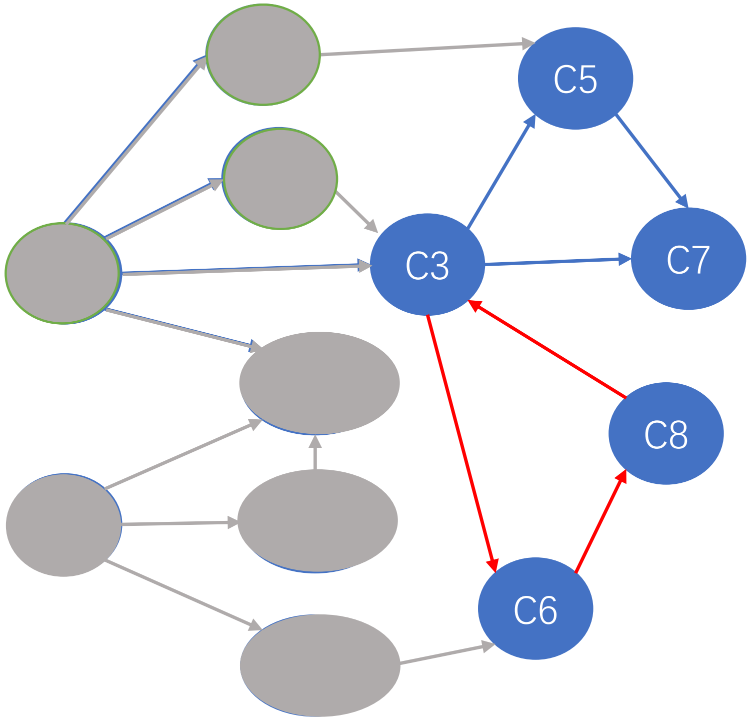
拓扑排序的一个重要应用：



检测AOV网中是否存在环方法：

对有向图构造其顶点的拓扑有序序列，若网中所有顶点都在他的拓扑有序序列中，则该AOV网必定不存在环

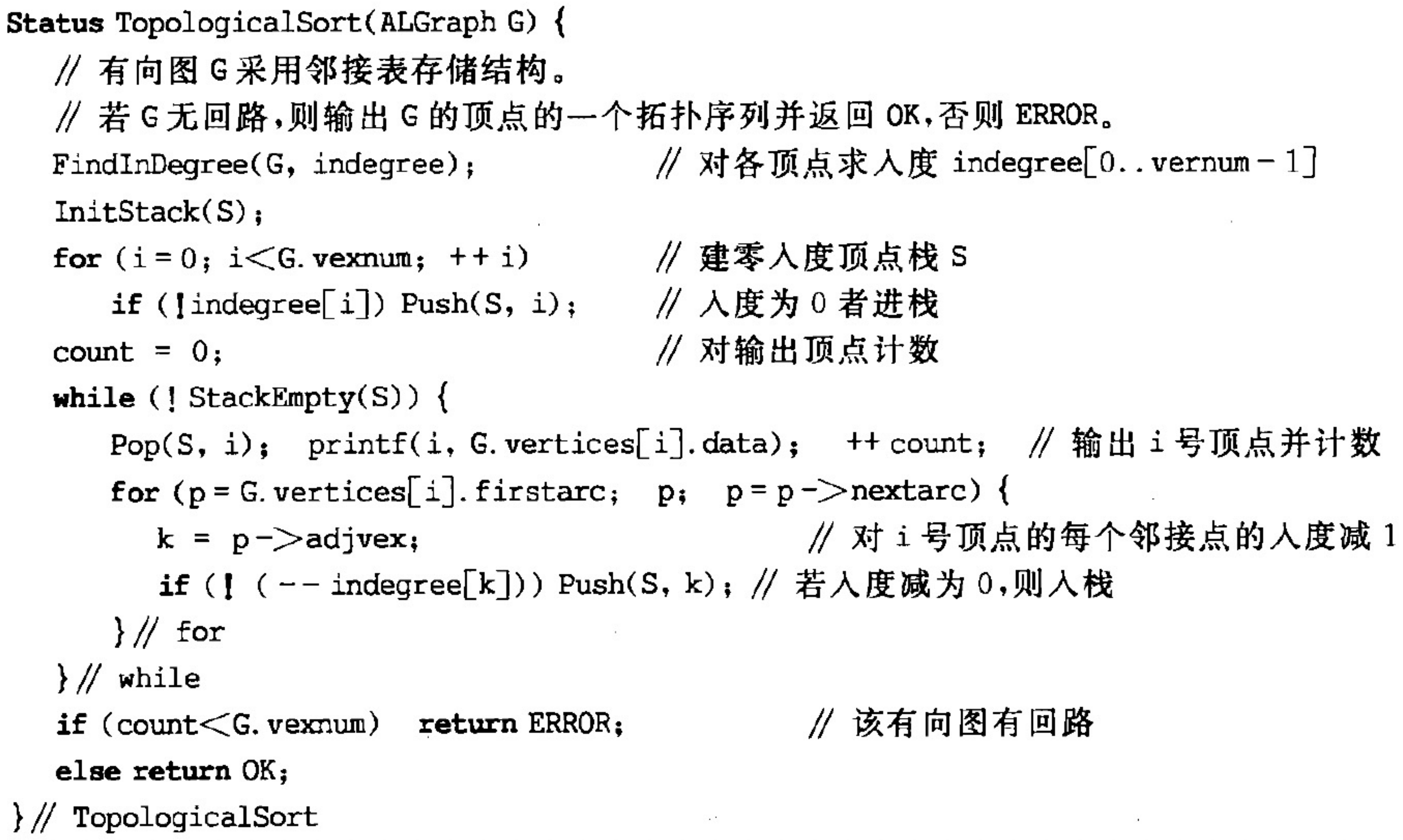
拓扑排序的一个重要应用：



检测AOV网中是否存在环方法：

对有向图构造其顶点的拓扑有序序列，若网中所有顶点都在他的拓扑有序序列中，则该AOV网必定不存在环

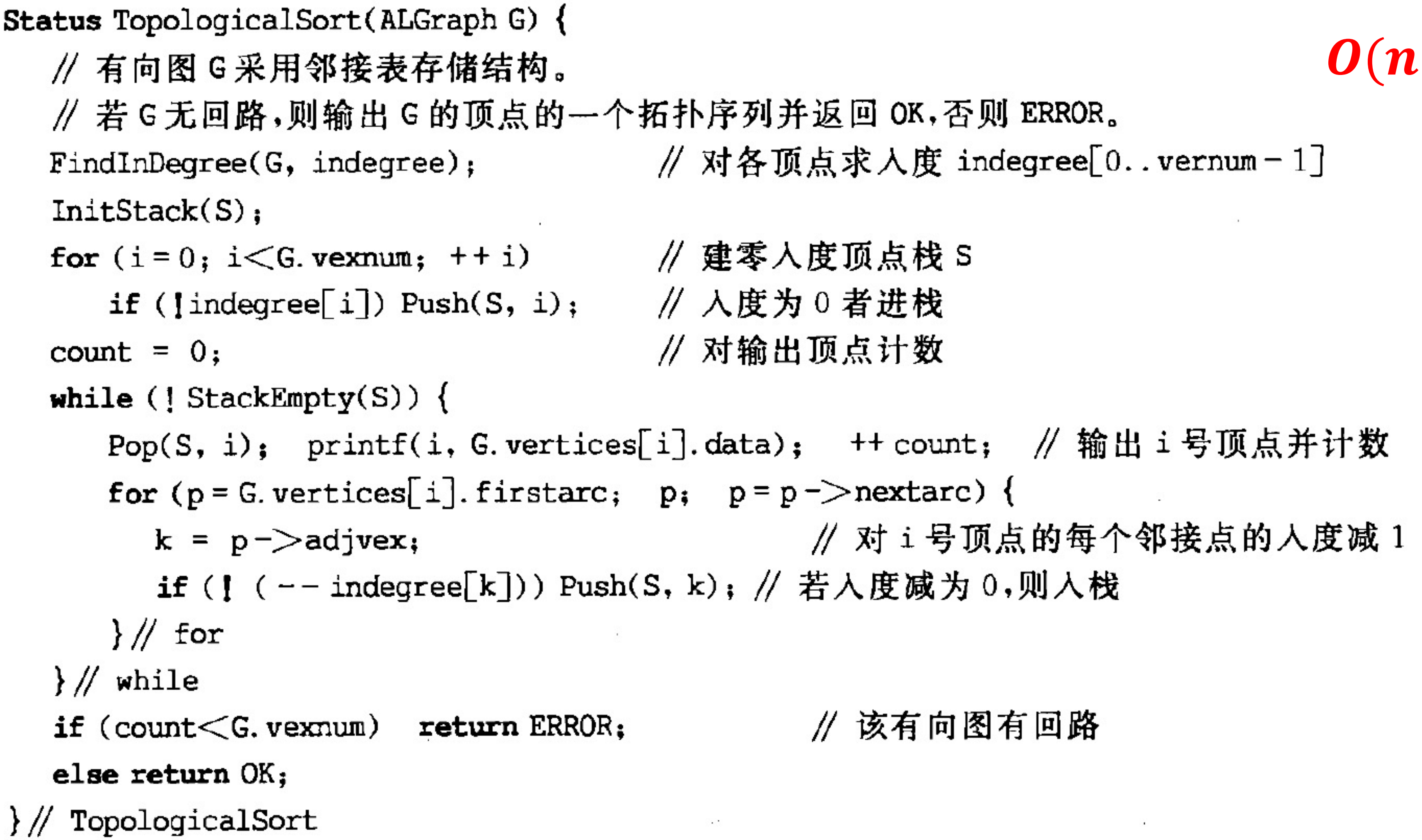
拓扑排序算法



```
Status TopologicalSort(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 indegree[0..vernum－1]
   InitStack(S);
   for (i = 0; i<G.vexnum; ++ i)           // 建零入度顶点栈 S
      if (!indegree[i]) Push(S, i);        // 入度为 0 者进栈
   count = 0;                              // 对输出顶点计数
   while (! StackEmpty(S)) {
      Pop(S, i);  printf(i, G.vertices[i].data);  ++ count;  // 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);  // 若入度减为 0，则入栈
      }// for
   }// while
   if (count<G.vexnum)  return ERROR;             // 该有向图有回路
   else return OK;
}// TopologicalSort
```

拓扑排序算法-时间复杂度

$O(n+e)$



```
Status TopologicalSort(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。
O(e)    FindInDegree(G, indegree);          // 对各顶点求入度 indegree[0..vernum-1]
    InitStack(S);
O(n)    for (i = 0; i<G.vexnum; ++i)        // 建零入度顶点栈S
        if (!indegree[i]) Push(S, i);   // 入度为0者进栈
    count = 0;                          // 对输出顶点计数
O(e)    while (! StackEmpty(S)) {
        Pop(S, i);  printf(i, G.vertices[i].data);  ++count;  // 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);  // 若入度减为0，则入栈
        }// for
    }// while
    if (count<G.vexnum)  return ERROR;          // 该有向图有回路
    else return OK;
}// TopologicalSort
```

7.4 图的应用

最小生成树

最短路径

拓扑排序

关键路径

关键路径

【例1】某项目的任务是对A公司的办公室重新进行装修

如果1月1日前完成装修工程，项目最迟应该何时开始?

需完成的活动、活动所需时间、及先期需完成工作如图所示

序号	项目	时间(天)	先期完成
1	清空办公室	3	无
2	拆除非承重墙	2	1
3	装修天花板	4	1
4	安装办公家具	5	2
5	重新布线	8	2
6	装修墙壁	3	3
7	装修地板	5	6
8	安装智能系统	10	5
9	清扫办公室	2	4 7 8
合计		42	

关键路径

【例2】准备一个小型家庭宴会，晚6点宴会开始

最迟几点开始准备？压缩哪项活动时间可以使总时间减少

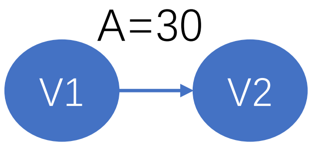
活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

关键路径

把工程计划表示为**边表示活动的网络**，即**AOE网**，用顶点表示事件，弧表示活动，弧的权值表示活动持续时间。

活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

事件表示在它之前的活动已经完成，在它之后的活动可以开始

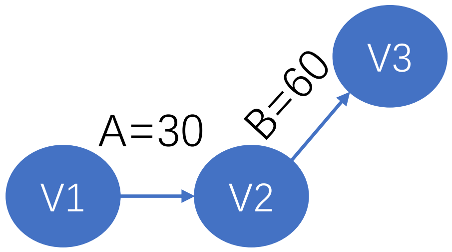


关键路径

把工程计划表示为**边表示活动的网络**，即**AOE网**，用顶点表示事件，弧表示活动，弧的权值表示活动持续时间。

事件表示在它之前的活动已经完成，在它之后的活动可以开始

活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

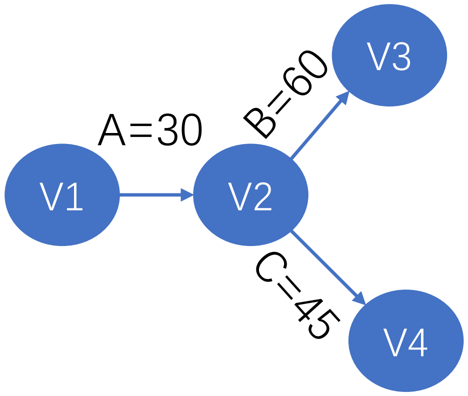


关键路径

把工程计划表示为**边表示活动的网络**，即**AOE网**，用顶点表示事件，弧表示活动，弧的权值表示活动持续时间。

事件表示在它之前的活动已经完成，在它之后的活动可以开始

活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

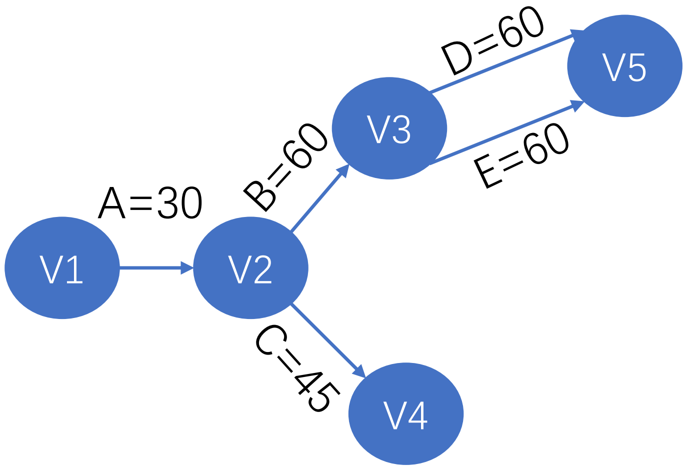


关键路径

把工程计划表示为**边表示活动的网络**，即**AOE网**，用顶点表示事件，弧表示活动，弧的权值表示活动持续时间。

事件表示在它之前的活动已经完成，在它之后的活动可以开始

活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

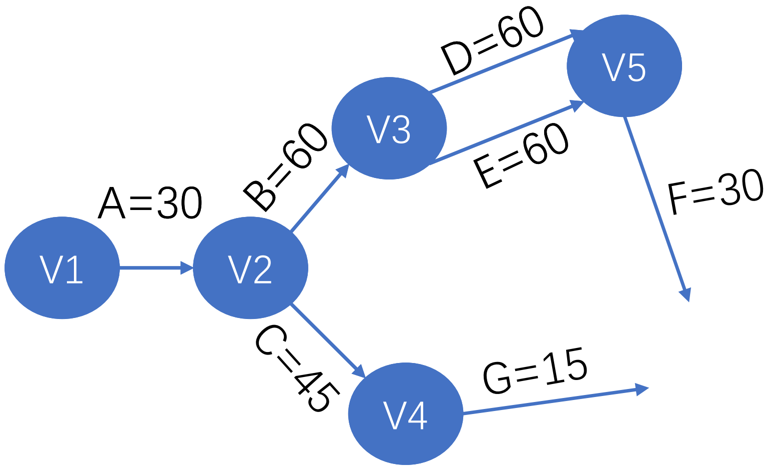


关键路径

把工程计划表示为**边表示活动的网络**，即**AOE网**，用顶点表示事件，弧表示活动，弧的权值表示活动持续时间。

活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

事件表示在它之前的活动已经完成，在它之后的活动可以开始

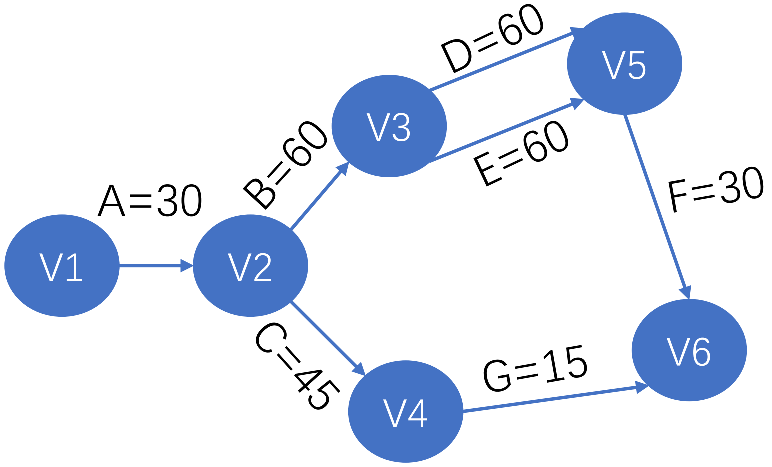


关键路径

把工程计划表示为**边表示活动的网络**，即**AOE网**，用顶点表示事件，弧表示活动，弧的权值表示活动持续时间。

活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

事件表示在它之前的活动已经完成，在它之后的活动可以开始



关键路径

【例2】准备一个小型家庭宴会，晚6点宴会开始

最迟几点开始准备？压缩哪项活动

时间可以使总时间减少

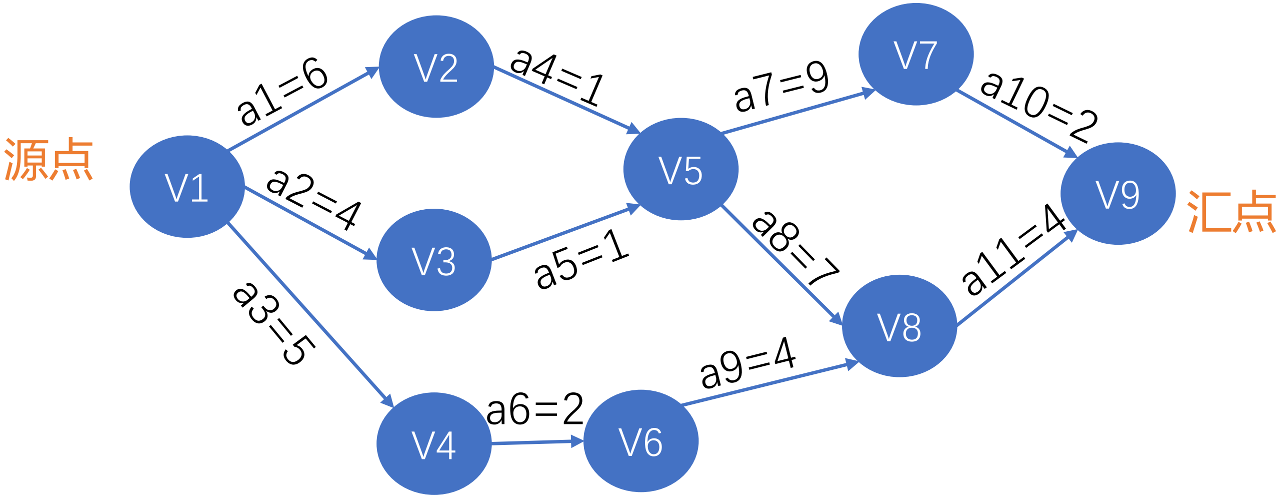
活动代码	活动描述	历时(分钟)	前置任务
A	菜单制定	30	
B	原料采购	60	A
C	餐具准备	45	A
D	甜点准备	60	B
E	原料清洗	60	B
F	烹饪	30	DE
G	桌椅布置	15	C
H	宴会开始	0	FG

关键路径

例：设一个工程有11项活动，9个事件

事件v1——表示整个工程开始（源点：入度为0的顶点）

事件v9——表示整个工程结束（汇点：出度为0的顶点）

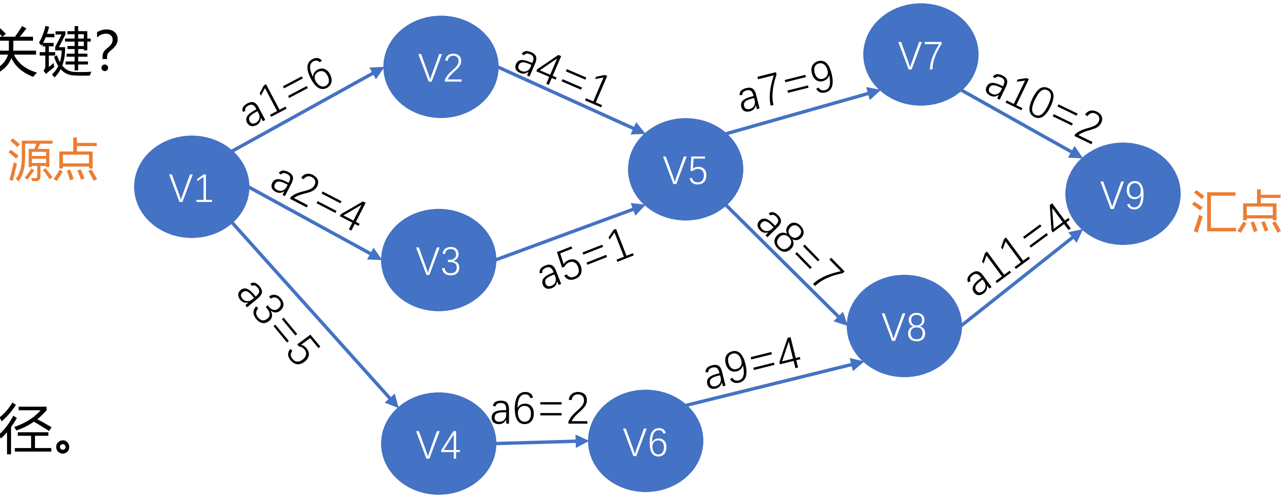


关键路径

对于AOE网，最关心的两个问题：

（1）完成整项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？

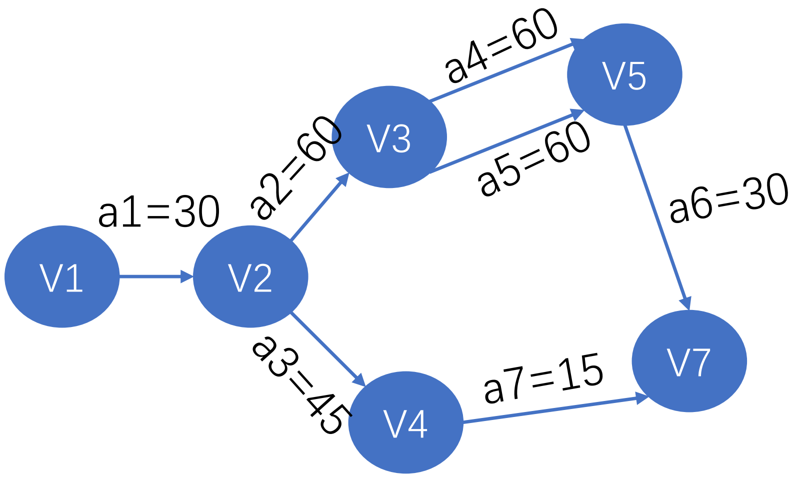


关键路径——路径长度最长的路径。

路径长度——路径上各活动持续时间之和。

关键路径

如何确定关键路径，需要定义4个描述量：



ve(vj) ——表示事件 vj 的最早发生时间

例：ve(v1)=0　　ve(v2)=30

vl(vj) ——表示事件 vj 的最迟发生时间

例：vl(v4)=165　　假设整个工程需要180分钟

e(i) ——表示活动 ai 的最早开始时间

例：e(a3)=30

l(i) ——表示活动 ai 的最迟开始时间

例：l(a3)=120　　假设整个工程需要180分钟

l(i) – e(i) ——表示完成活动 ai 的时间余量　　例：l(a3)-e(a3)=90

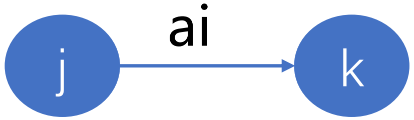
关键路径——关键路径上的活动，即l(i)==e(i)（即l(i)-e(i)==0）的活动

关键路径

- 如何找l(i)==e(i)的关键活动?

设活动ai用弧<j,k>表示，其持续时间记为：$w_{j,k}$

则有：　(1) e(i)==ve(j)　　(2) l(i)==vl(k)- $w_{j,k}$



- 如何求ve(j)和vl(j)?

(1) 从 ve(1)=0 开始向前递推

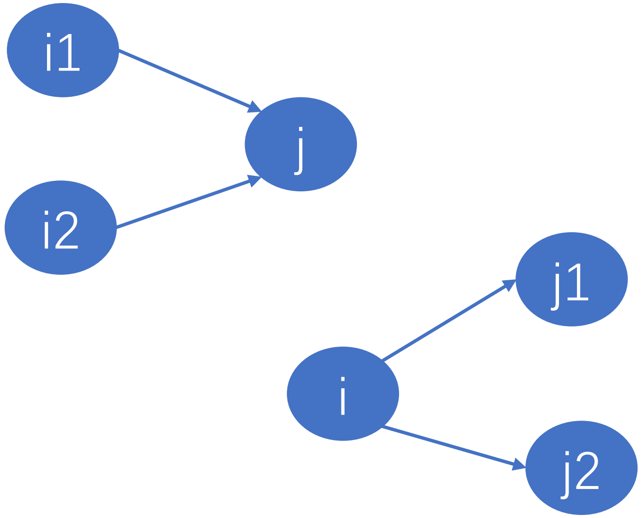
ve(j)=Max{ve(i) + $w_{i,j}$}, $< i,j > \in T, 2 \le j \le n$。

其中T是所有以 j 为头的弧的集合

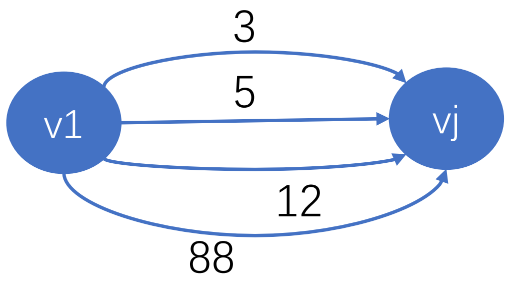
(2) 从 vl(n)=ve(n) 开始向后递推

vl(i)=Min{vl(j) - $w_{i,j}$}, $< i,j > \in S, 1 \le i \le n-1$。

其中S是所有以 i 为尾的弧的集合

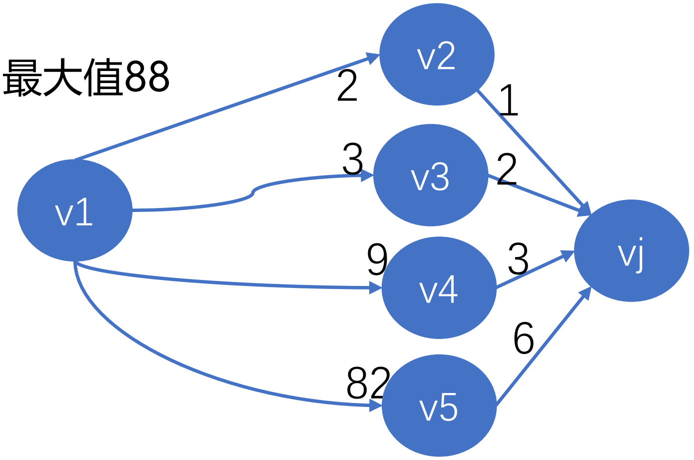


关键路径



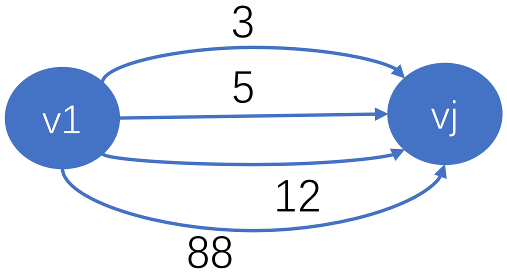
Ve(Vj)=3、5、12、88的最大值88

求Ve(Vj), 由源点至汇点

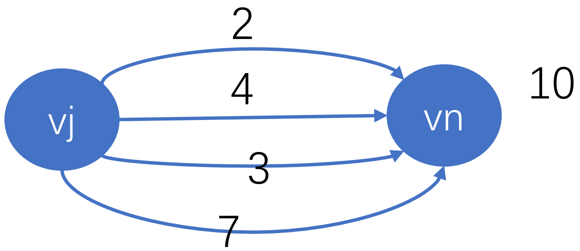


Ve(Vj)=Vj的起始结点的最早发生时间+各边的权值中的和的最大值。所以Ve(Vj)=88

关键路径

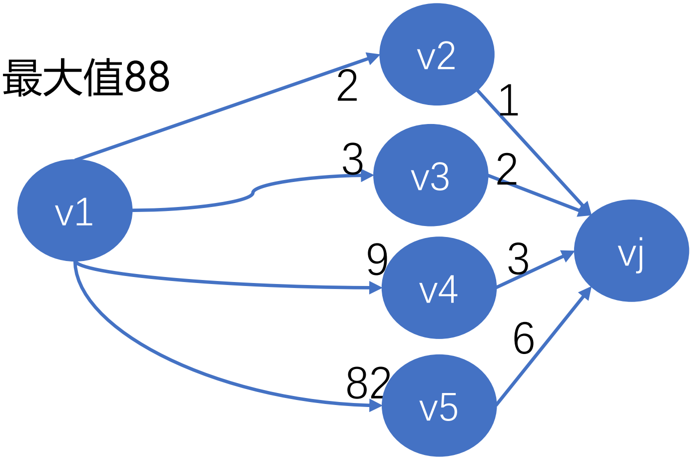


Ve(Vj)=3、5、12、88的最大值88

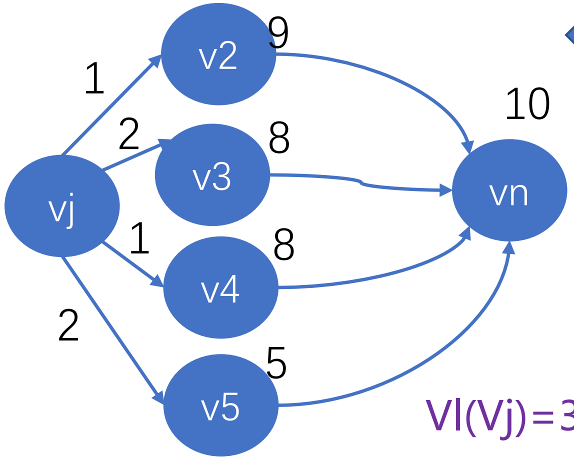


Vl(Vj)取10-2、10-4、10-3、10-7的最小值3

求Ve(Vj), 由源点至汇点



Ve(Vj)=Vj的起始结点的最早发生时间+各边的权值中的和的最大值。所以Ve(Vj)=88



求 Vl(Vj), 由汇点至源点

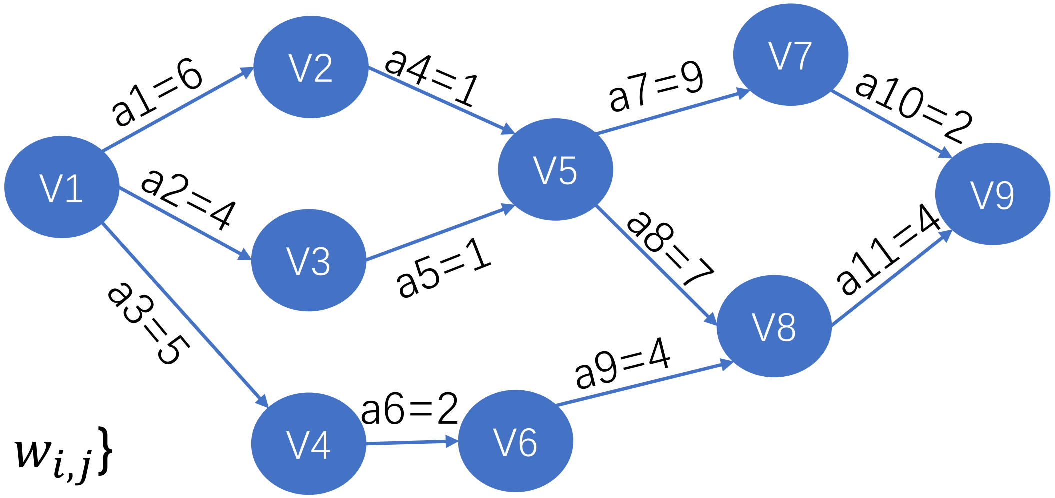
Vl(Vj)=3

关键路径

求关键路径步骤：

1. 求 ve(i)，vl(j)
2. 求 e(i)，l(i)
3. 计算 l(i)-e(i)

$$ve(j)=Max\{ve(i) + w_{i,j}\}$$



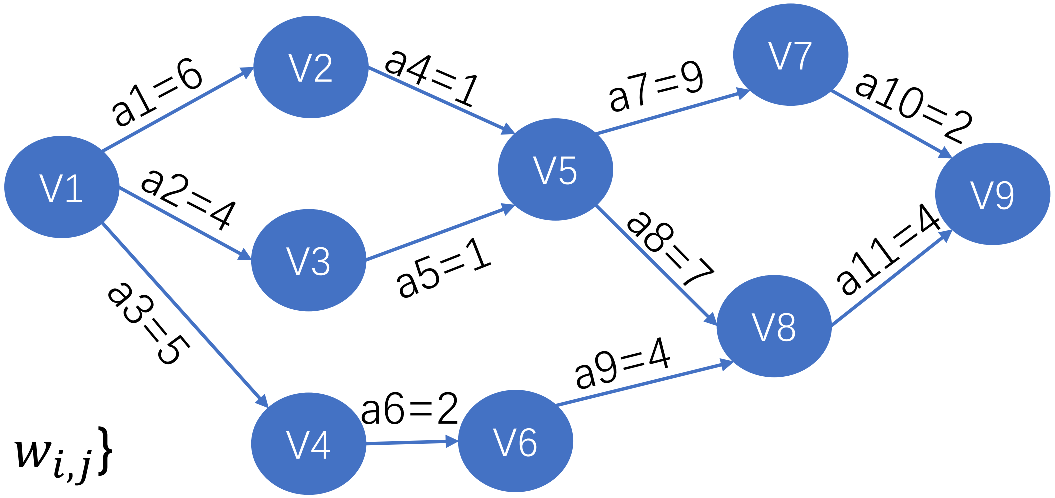
顶点	ve	vl
v1	0	
v2	6	
v3	4	
v4	5	
v5		
v6		
v7		
v8		
v9		

关键路径

求关键路径步骤：

1. 求 ve(i)，vl(j)
2. 求 e(i)，l(i)
3. 计算 l(i)-e(i)

$$ve(j)=Max\{ve(i) + w_{i,j}\}$$



顶点	ve	vl
v1	0	
v2	6	
v3	4	
v4	5	
v5	7	
v6	7	
v7	16	
v8	14	
v9	18	

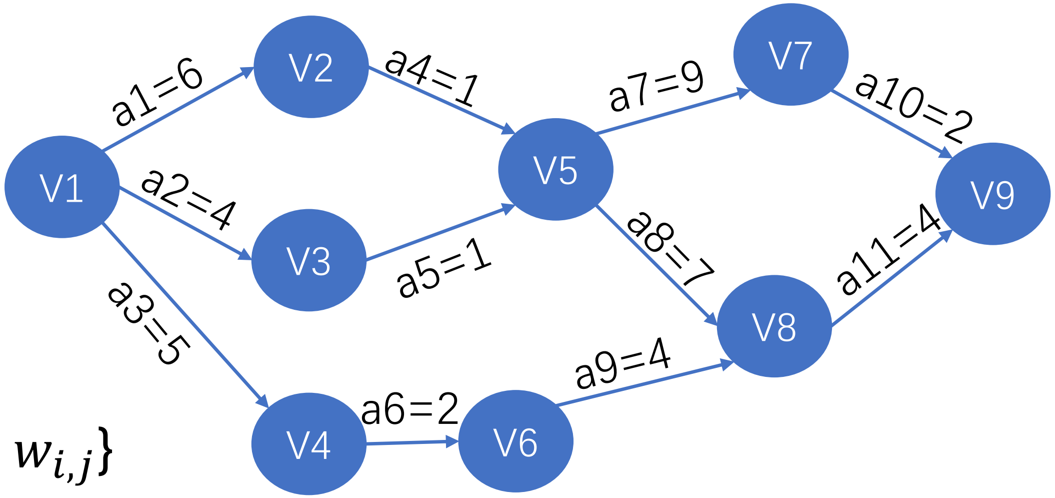
关键路径

求关键路径步骤：

1. 求 ve(i)，vl(j)
2. 求 e(i)，l(i)
3. 计算 l(i)-e(i)

$$ve(j)=Max\{ve(i) + w_{i,j}\}$$

$$vl(i)=Min\{vl(j) - w_{i,j}\}$$



顶点	ve	vl
v1	0	
v2	6	
v3	4	
v4	5	
v5	7	7
v6	7	10
v7	16	16
v8	14	14
v9	18	18

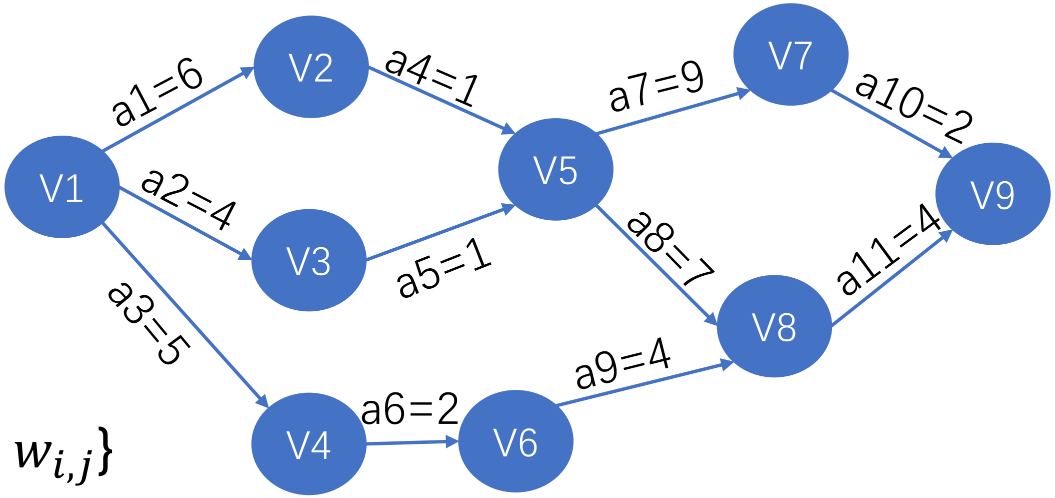
关键路径

求关键路径步骤：

1. 求 ve(i)，vl(j)
2. 求 e(i)，l(i)
3. 计算 l(i)-e(i)

$$ve(j)=Max\{ve(i) + w_{i,j}\}$$

$$vl\ (i)=Min\{vl(j) - w_{i,j}\}$$

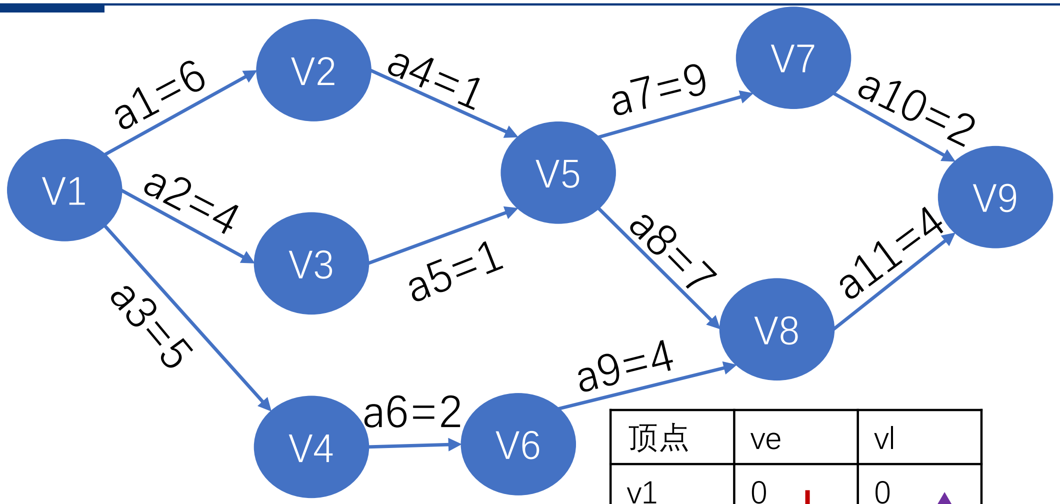


顶点	ve	vl
v1	0	0
v2	6	6
v3	4	6
v4	5	8
v5	7	7
v6	7	10
v7	16	16
v8	14	14
v9	18	18

关键路径

求关键路径步骤：

1. 求 ve(i)，vl(j)
2. 求 e(i)，l(i)
3. 计算 l(i)-e(i)

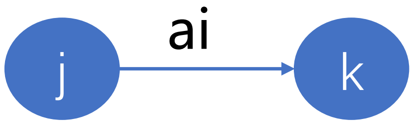


$$ve(j)=\text{Max}\{ve(i) + w_{i,j}\}$$

$$vl\ (i)=\text{Min}\{vl(j) - w_{i,j}\}$$

(1) e(i) = ve(j)

(2) l(i) = vl(k)- $w_{j,k}$



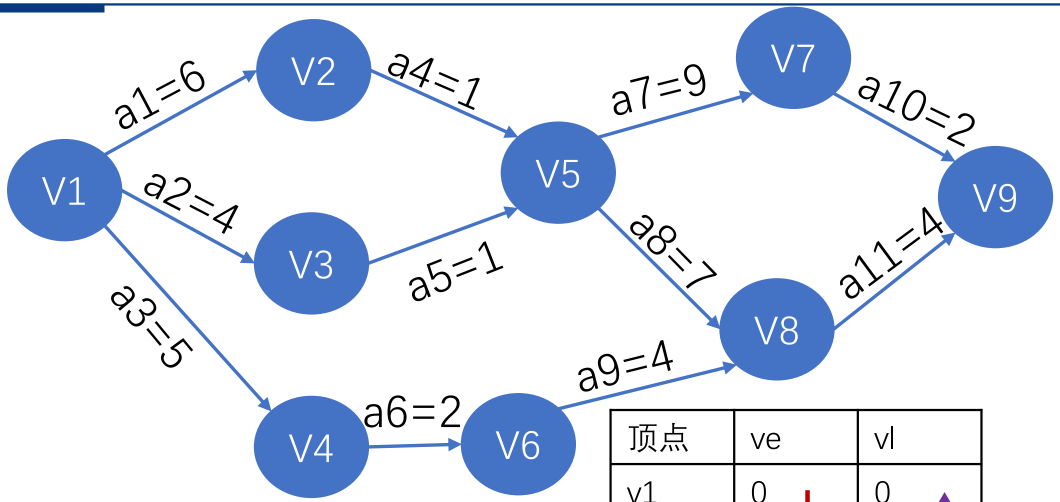
顶点	ve	vl
v1	0	0
v2	6	6
v3	4	6
v4	5	8
v5	7	7
v6	7	10
v7	16	16
v8	14	14
v9	18	18

活动	e	l	l-e
a1	0		
a2	0		
a3	0		
a4	6		
a5	4		
a6	5		
a7	7		
a8	7		
a9	7		
a10	16		
a11	14		

关键路径

求关键路径步骤：

1. 求 ve(i)，vl(j)
2. 求 e(i)，l(i)
3. 计算 l(i)-e(i)



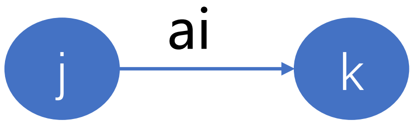
活动	e	l	l-e
a1	0	0	
a2	0	2	
a3	0	3	
a4	6	6	
a5	4	6	
a6	5	8	
a7	7	7	
a8	7	7	
a9	7	10	
a10	16	16	
a11	14	14	

$$ve(j)=Max\{ve(i) + w_{i,j}\}$$

$$vl\ (i)=Min\{vl(j) - w_{i,j}\}$$

(1) e(i) = ve(j)

(2) l(i) = vl(k)- $w_{j,k}$

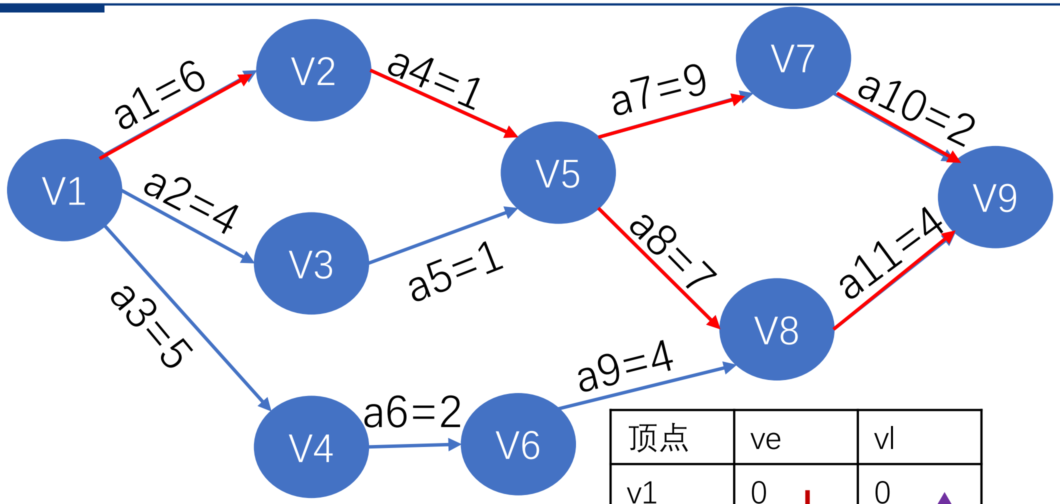


顶点	ve	vl
v1	0	0
v2	6	6
v3	4	6
v4	5	8
v5	7	7
v6	7	10
v7	16	16
v8	14	14
v9	18	18

关键路径

求关键路径步骤：

1. 求 ve(i)，vl(j)
2. 求 e(i)，l(i)
3. 计算 l(i)-e(i)

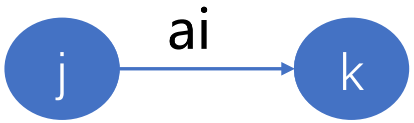


$ve(j)=Max\{ve(i) + w_{i,j}\}$

$vl\ (i)=Min\{vl(j) - w_{i,j}\}$

(1) e(i) = ve(j)

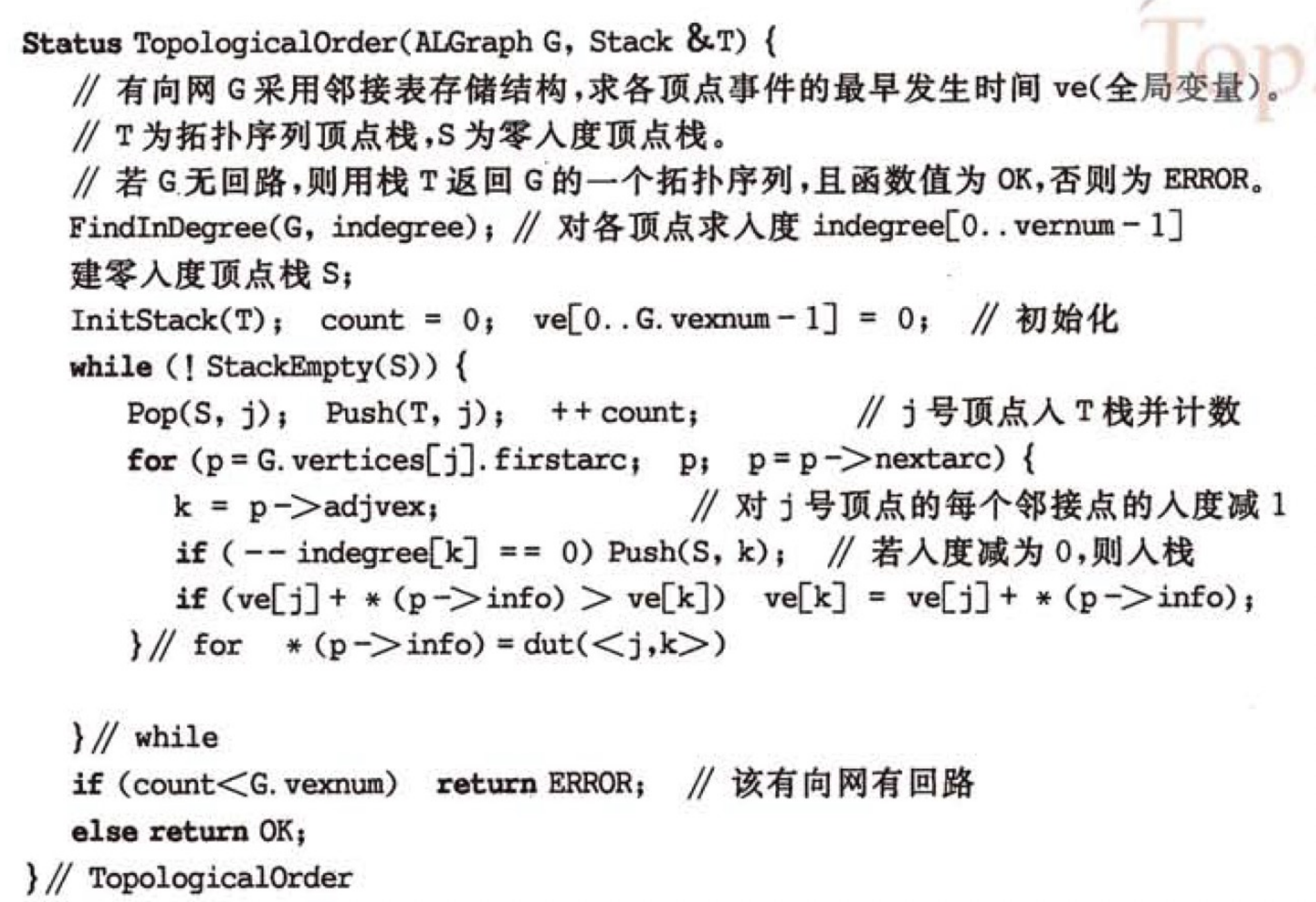
(2) l(i) = vl(k)- $w_{j,k}$



顶点	ve	vl
v1	0	0
v2	6	6
v3	4	6
v4	5	8
v5	7	7
v6	7	10
v7	16	16
v8	14	14
v9	18	18

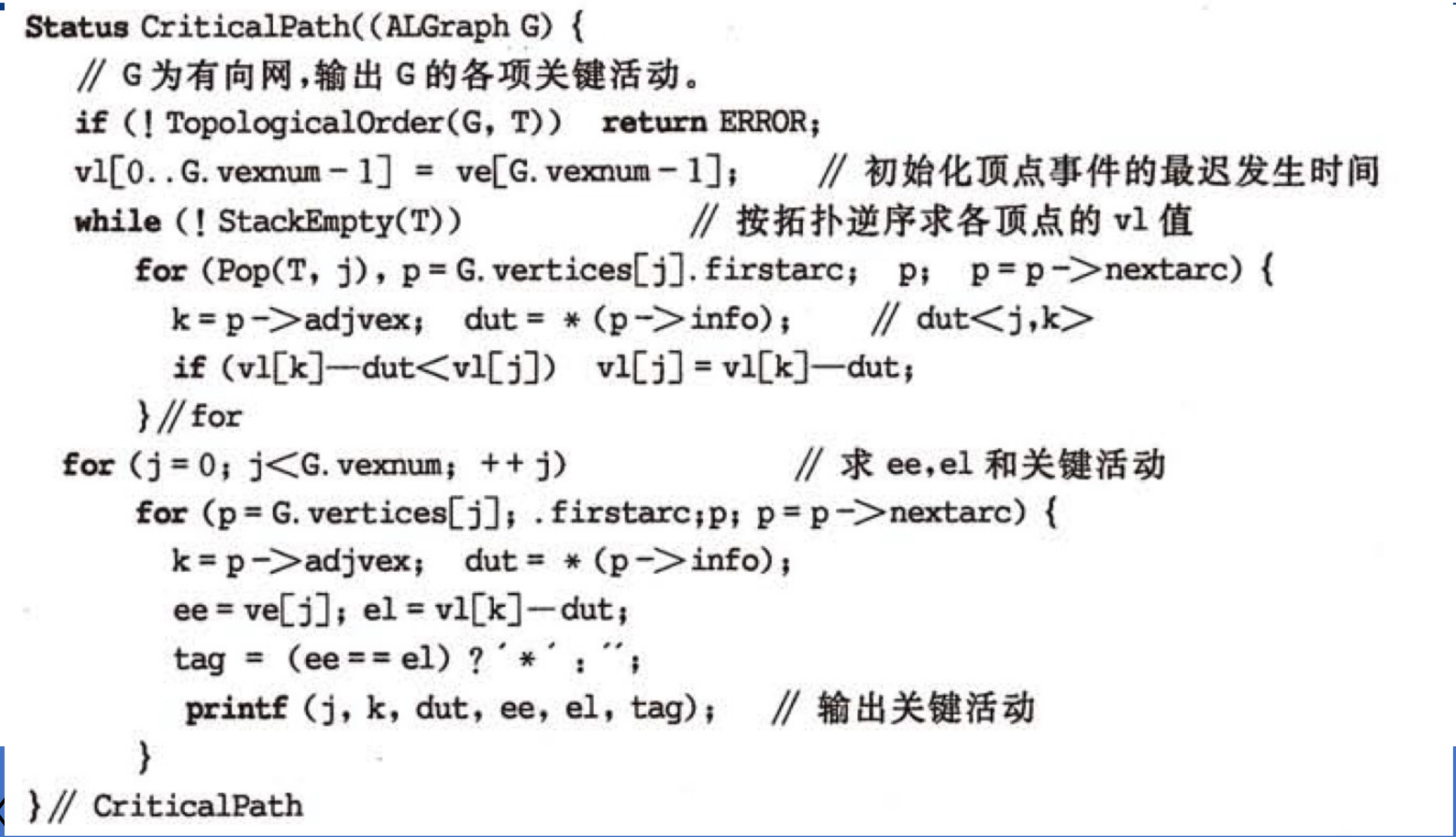
活动	e	l	l-e
a1	0	0	0 ✓
a2	0	2	2
a3	0	3	3
a4	6	6	0 ✓
a5	4	6	2
a6	5	8	3
a7	7	7	0 ✓
a8	7	7	0 ✓
a9	7	10	3
a10	16	16	0 ✓
a11	14	14	0 ✓

关键



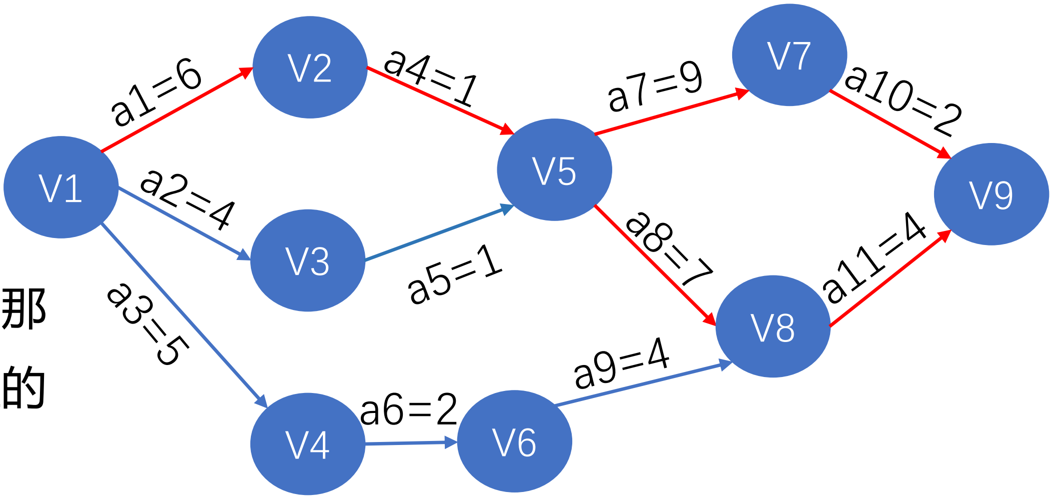
```
Status TopologicalOrder(ALGraph G, Stack &T) {
    // 有向网G采用邻接表存储结构,求各顶点事件的最早发生时间ve(全局变量)。
    // T为拓扑序列顶点栈,S为零入度顶点栈。
    // 若G无回路,则用栈T返回G的一个拓扑序列,且函数值为OK,否则为ERROR。
    FindInDegree(G, indegree); // 对各顶点求入度 indegree[0..vernum-1]
    建零入度顶点栈 S;
    InitStack(T);   count = 0;   ve[0..G.vexnum-1] = 0;   // 初始化
    while (! StackEmpty(S)) {
        Pop(S, j);   Push(T, j);   ++count;            // j号顶点入T栈并计数
        for (p=G.vertices[j].firstarc;  p;  p=p->nextarc) {
            k = p->adjvex;                     // 对j号顶点的每个邻接点的入度减1
            if (--indegree[k] == 0) Push(S, k);   // 若入度减为0,则入栈
            if (ve[j]+*(p->info) > ve[k])  ve[k] = ve[j]+*(p->info);
        }// for   *(p->info) = dut(<j,k>)

    }// while
    if (count<G.vexnum)  return ERROR;   // 该有向网有回路
    else return OK;
}// TopologicalOrder
```

关键路径算法



```
Status CriticalPath((ALGraph G) {
   // G 为有向网,输出 G 的各项关键活动。
   if (! TopologicalOrder(G, T))  return ERROR;
   vl[0..G.vexnum - 1] = ve[G.vexnum - 1];     // 初始化顶点事件的最迟发生时间
   while (! StackEmpty(T))                      // 按拓扑逆序求各顶点的 vl 值
       for (Pop(T, j), p = G.vertices[j].firstarc;  p;  p = p->nextarc) {
         k = p->adjvex;   dut = *(p->info);      // dut<j,k>
         if (vl[k]—dut<vl[j])  vl[j] = vl[k]—dut;
       }// for
   for (j = 0; j<G.vexnum;  ++j)                 // 求 ee,el 和关键活动
       for (p = G.vertices[j]; .firstarc;p; p = p->nextarc) {
         k = p->adjvex;   dut = *(p->info);
         ee = ve[j]; el = vl[k]—dut;
         tag = (ee == el) ? '*' : '';
          printf (j, k, dut, ee, el, tag);   // 输出关键活动
       }
}// CriticalPath
```

关键路径



1、若网中有几条关键路径，则需加快同时在几条关键路径上的关键活动。

如：a11、a10、a8、a7。

2、如果一个活动处于所有的关键路径上，那么提高这个活动的速度，就能缩短整个工程的完成时间。

如：a1、a4

3、处于所有的关键路径上的活动完成时间不能缩短太多，否则会是原来的关键路径变成不是关键路径。这时，必须重新寻找关键路径。

如：a1由6天变成3天，就会改变关键路径