



Data Structures

Ch6



Graphs

2023 年 11 月 16 日

学而不厌 诲人不倦

- ➡ 6.1 引言
- ➡ 6.2 图的逻辑结构
- ➡ 6.3 图的存储结构及实现
- ➡ 6.4 最小生成树
- ➡ 6.5 最短路径
- ➡ **6.6 有向无环图及其应用**
- ➡ 6.7 扩展与提高
- ➡ 6.8 应用实例



6.6 有向无环图及其应用

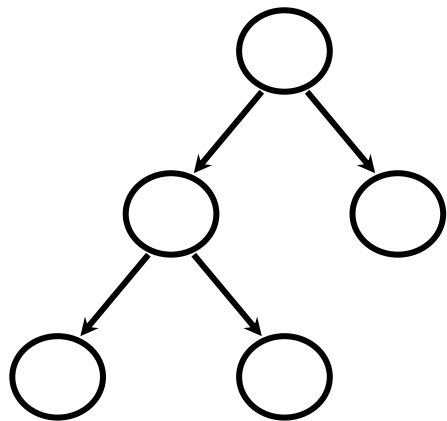
6-6-1 AOV网与拓扑排序



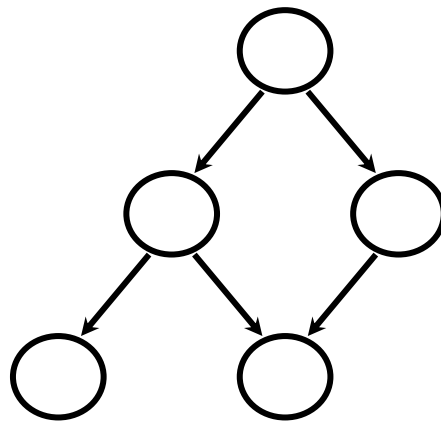
1. 有向无环图

Directed Acyclic Graph

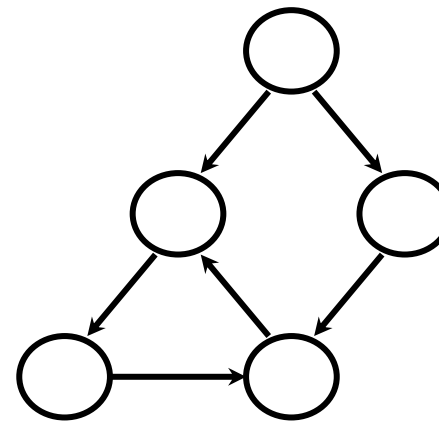
一个无环的有向图称为有向无环图，简称 DAG 图。



有向树



DAG 图



有向图

6.6 有向无环图及其应用

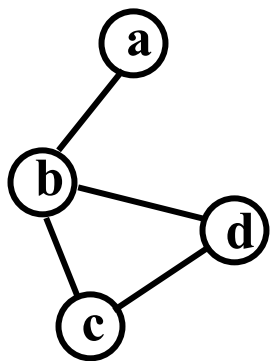
6-6-1 AOV网与拓扑排序



1. 有向无环图

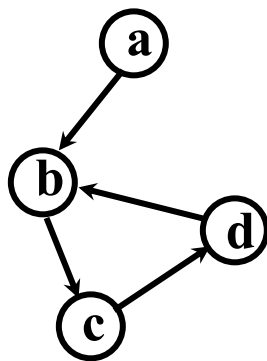
给定一个图，如何判断是否存在环？

利用深度优先搜索算法，若将要指向的顶点已被访问过，则存在环。



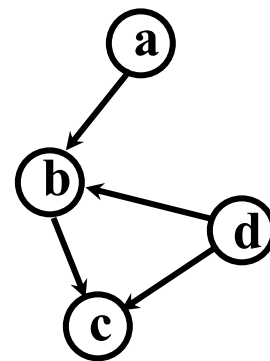
无向图

可正确判定



有向图

可正确判定



特例

不可正确判定

如何判断一个有向图是否为 DAG 图？ **方法1**

在一个连通分量里寻找环。

6.6 有向无环图及其应用

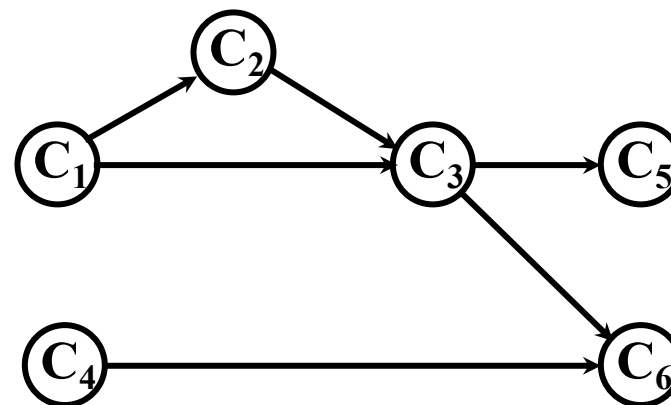
6-6-1 AOV网与拓扑排序



1. 有向无环图

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。

课程编号	课程名称	先决条件
C_1	程序语言基础	
C_2	离散数学	C_1
C_3	数据结构	C_1, C_2
C_4	微机原理	
C_5	编译原理	C_3
C_6	操作系统	C_3, C_4



表示课程之间优先关系的有向图


这种用顶点表示活动，用弧表示活动之间的优先关系的有向图称为**顶点表示活动的网：AOV网**。
在 AOV 网中不应该出现**有向环**，否则将存在某项活动以自己为先决条件。




6.6 有向无环图及其应用

6-6-1 AOV网与拓扑排序

2. AOV网的定义

 什么是工程？工程有什么共性？

几乎所有的工程都可以分为若干个称作**活动**的子工程
某些活动之间通常存在一定的**约束**条件

 **AOV网**（顶点表示活动的网）：在一个表示工程的有向图中，用顶点表示活动，用弧表示活动之间的**优先**关系

AOV网 (activity on vertex network)

 AOV网中出现回路意味着什么？

活动之间的优先关系是矛盾的

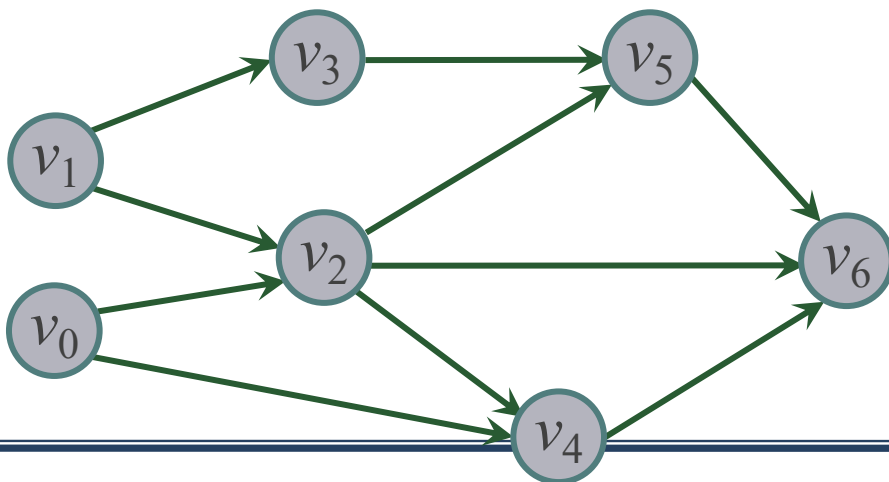


3. 拓扑序列

✚ **拓扑序列**：设有向图 $G=(V, E)$ 具有 n 个顶点，则顶点序列 v_0, v_1, \dots, v_{n-1} 称为一个拓扑序列，当且仅当满足下列条件：若从顶点 v_i 到 v_j 有一条**路径**，则在顶点序列中顶点 v_i 必在顶点 v_j **之前**

使得AOV网中所有应该存在的前驱和后继关系都能得到满足

✚ **拓扑排序**：对一个有向图构造拓扑序列的过程



拓扑序列 1: $v_0 v_1 v_2 v_3 v_4 v_5 v_6$

拓扑序列 2: $v_0 v_1 v_3 v_2 v_4 v_5 v_6$



4. 拓扑排序

性质: 若AOV 网中的所有顶点都在它的**拓扑排序序列**中, 则该 AOV 网中必定不存在**环**; 否则必存在**环**。

拓扑排序算法的思想:

1. 在有向图中选取一个**没有前驱**的顶点并输出;
2. 从图中**删除该顶点**及所有以此顶点为尾的**弧**;
3. 重复上述两步, 直至**全部顶点均已输出**; 或者**当前图中不存在无前驱的顶点**为止。

得到一个
拓扑排序

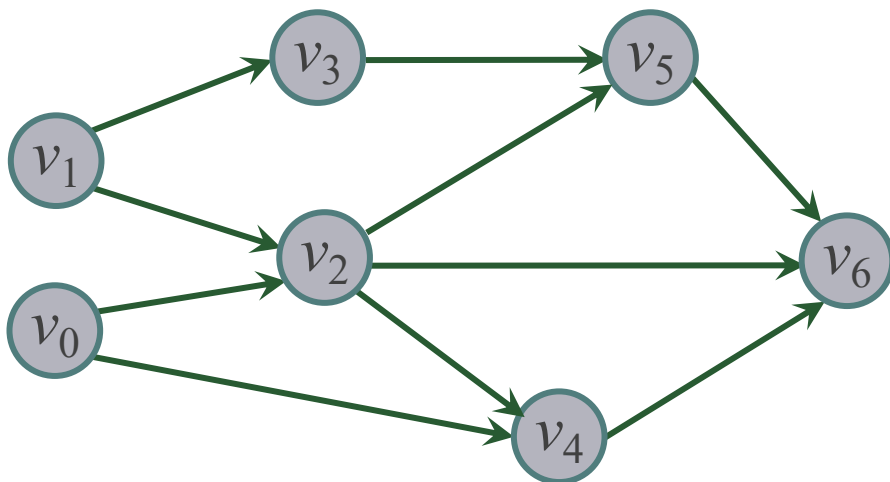
存在环

6.6 有向无环图及其应用

6-6-1 AOV网与拓扑排序



4. 拓扑排序



1. 在有向图中选取一个**没有前驱**的顶点并输出；
2. 从图中**删除该顶点**及所有以此顶点为尾的**弧**；
3. 重复上述两步，直至**全部顶点均已输出**；或者**当前图中不存在无前驱的顶点为止**。

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

拓扑序列： v_0 v_1 v_2 v_3 v_4 v_5 v_6



6.6 有向无环图及其应用

6-6-1 AOV网与拓扑排序

5. 拓扑排序算法-存储结构



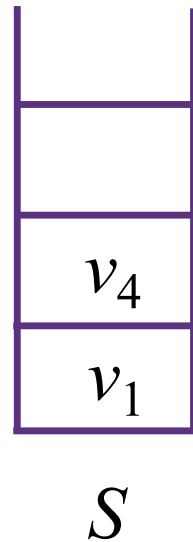
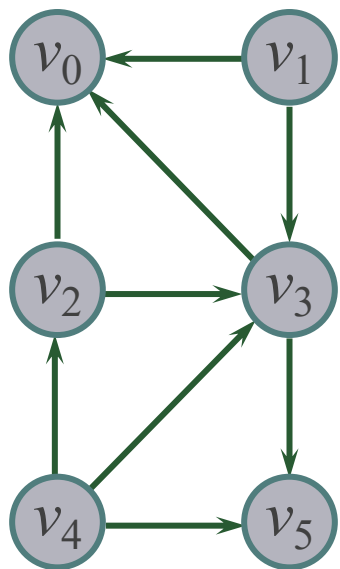
图采用什么存储结构呢？ \Rightarrow 邻接表



在邻接表中，如何求顶点的入度？ \Rightarrow 顶点表中增加入度域



如何查找没有前驱的顶点？ \Rightarrow 设置栈或队列



	in	vertex	firstEdge						
0	3	v_0	Λ						
1	0	v_1	\rightarrow <table><tr><td>0</td><td></td></tr></table> \rightarrow <table><tr><td>3</td><td>Λ</td></tr></table>	0		3	Λ		
0									
3	Λ								
2	1	v_2	\rightarrow <table><tr><td>0</td><td></td></tr></table> \rightarrow <table><tr><td>3</td><td>Λ</td></tr></table>	0		3	Λ		
0									
3	Λ								
3	3	v_3	\rightarrow <table><tr><td>0</td><td></td></tr></table> \rightarrow <table><tr><td>5</td><td>Λ</td></tr></table>	0		5	Λ		
0									
5	Λ								
4	0	v_4	\rightarrow <table><tr><td>2</td><td></td></tr></table> \rightarrow <table><tr><td>3</td><td></td></tr></table> \rightarrow <table><tr><td>5</td><td>Λ</td></tr></table>	2		3		5	Λ
2									
3									
5	Λ								
5	2	v_5	Λ						

6.6 有向无环图及其应用

6-6-1 AOV网与拓扑排序



5. 拓扑排序算法-存储结构

```
struct EdgeNode
{
    int adjvex;
    EdgeNode *next;
};
```

```
template <typename DataType>
struct VertexNode
{
    DataType vertex;
    EdgeNode *firstEdge;
};
```

```
const int MaxSize = 10;
template <typename DataType>
class ALGraph
{
public:
    ALGraph(DataType a[ ], int n, int e);
    ~ALGraph( );
    void DFTraverse(int v);
    void BFTraverse(int v);
private:
    VertexNode<DataType> adjlist[MaxSize];
    int vertexNum, edgeNum;
};
```

6.6 有向无环图及其应用

6-6-1 AOV网与拓扑排序



6. 拓扑排序算法-伪代码



图：带入度的邻接表



栈：入度为 0 的顶点（编号）

算法：TopSort

输入：有向图 $G=(V, E)$

输出：拓扑序列

1. 栈 S 初始化；累加器 $count$ 初始化；
2. 扫描顶点表，将入度为 0 的顶点压栈；
3. 当栈 S 非空时循环
 - 3.1 $j =$ 栈顶元素出栈；输出顶点 j ； $count++$ ；
 - 3.2 对顶点 j 的每一个邻接点 k 执行下述操作：
 - 3.2.1 将顶点 k 的入度减 1；
 - 3.2.2 如果顶点 k 的入度为 0，则将顶点 k 入栈；
4. if ($count < vertexNum$) 输出有回路信息；

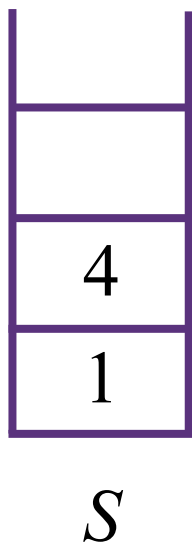
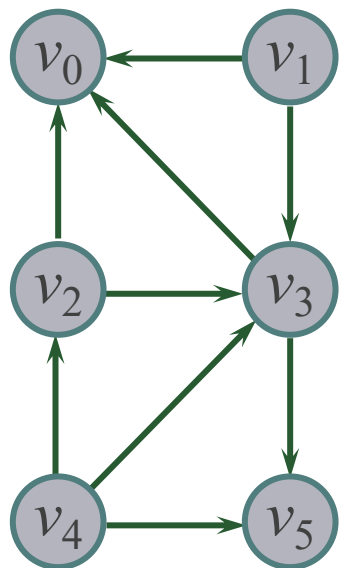
6.6 有向无环图及其应用

6-6-1 AOV网与拓扑排序



6. 拓扑排序算法-实现

```
void TopSort()  
{  
    int i, j, k, count = 0, S[MaxSize], top = -1;  
    for (i = 0; i < vertexNum; i++)  
        if (adjlist[i].in == 0)  
            S[++top] = i;  
}
```



	in	vertex	firstEdge
0	3	v_0	\wedge
1	0	v_1	\rightarrow [0] \rightarrow [3] \wedge
2	1	v_2	\rightarrow [0] \rightarrow [3] \wedge
3	3	v_3	\rightarrow [0] \rightarrow [5] \wedge
4	0	v_4	\rightarrow [2] \rightarrow [3] \rightarrow [5] \wedge
5	2	v_5	\wedge

6.6 有向无环图及其应用

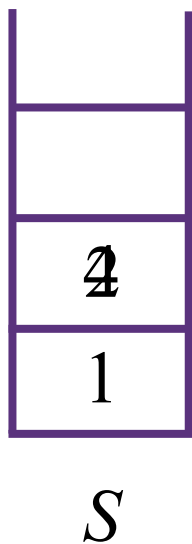
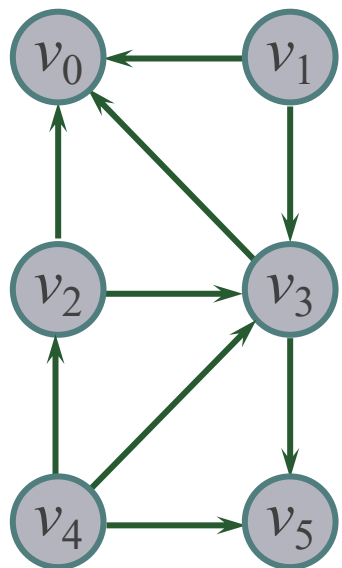
6-6-1 AOV网与拓扑排序



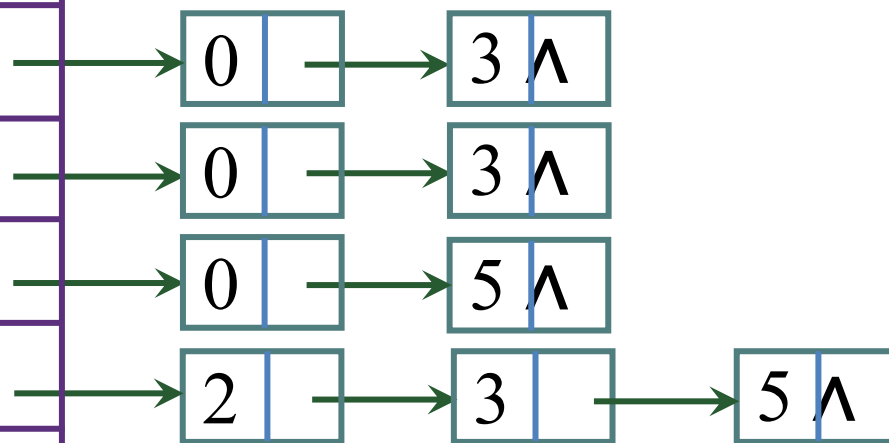
5. 拓扑排序算法-实现

```
while (top != -1 )  
{  
    j = S[top--];  
    cout << adjlist[j].vertex;  
    count++;  
}
```

```
p = adjlist[j].firstEdge;  
while (p != nullptr)  
{  
    k = p->adjvex; adjlist[k].in--;  
    if (adjlist[k].in == 0) S[++top] = k;  
    p = p->next;  
}
```



	in	vertex	firstEdge
0	3	v_0	Λ
1	0	v_1	
2	1 0	v_2	
3	3 2	v_3	
4	0	v_4	
5	2 1	v_5	Λ



6.6 有向无环图及其应用

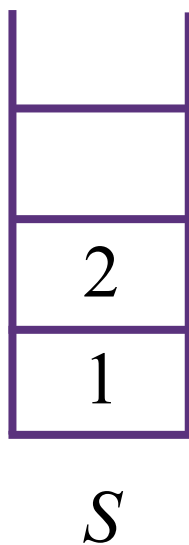
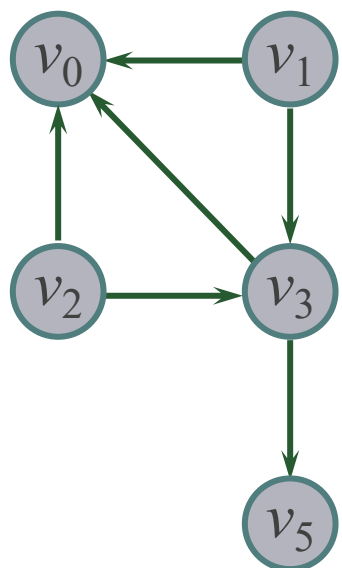
6-6-1 AOV网与拓扑排序



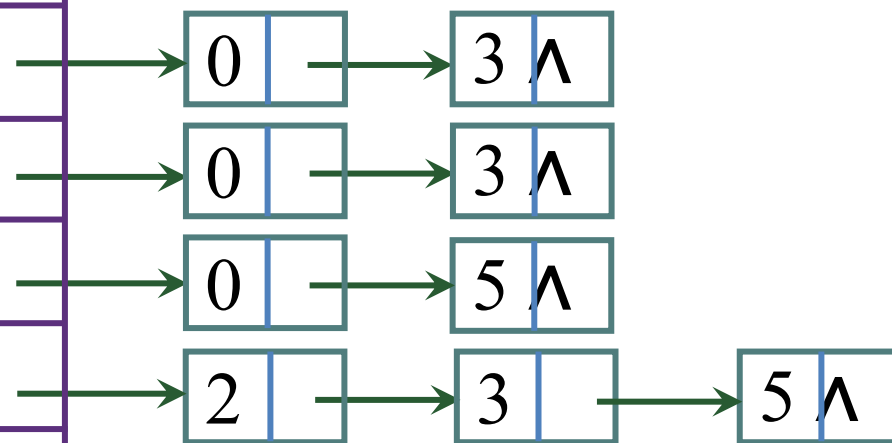
5. 拓扑排序算法-实现

```
while (top != -1 )
{
    j = S[top--];
    cout << adjlist[j].vertex;
    count++;
}
```

```
p = adjlist[j].firstEdge;
while (p != nullptr)
{
    k = p->adjvex; adjlist[k].in--;
    if (adjlist[k].in == 0) S[++top] = k;
    p = p->next;
}
```



	in	vertex	firstEdge
0	3 2	v_0	Λ
1	0	v_1	
2	0	v_2	
3	2 1	v_3	
4	0	v_4	
5	1	v_5	Λ



6.6 有向无环图及其应用

6-6-1 AOV网与拓扑排序



5. 拓扑排序算法-实现

```
void TopSort( )
{
    int i, j, k, count = 0, S[MaxSize], top = -1;
    EdgeNode *p = nullptr;
    for (i = 0; i < vertexNum; i++)
        if (adjlist[i].in == 0) S[++top] = i;
    while (top != -1 )
    {
        j = S[top--]; cout << adjlist[j].vertex; count++;
        p = adjlist[j].first;
        while (p != nullptr)
        {
            k = p->adjvex; adjlist[k].in--;
            if (adjlist[k].in == 0) S[++top] = k;
            p = p->next;
        }
    }
    if (count < vertexNum ) cout << "有回路";
}
```

/*扫描顶点表*/

} $O(n)$

/*当栈中还有入度为0的顶点时*/

count++;

/*描顶点表，找出顶点j的所有出边*/

/*将入度为0的顶点入栈*/

} $O(e)$



时间复杂度?



$O(n+e)$



6.6 有向无环图及其应用

6-6-2 AOE网与关键路径

拓扑排序解决工程的可行性或合理性。

关键路径则关心工程解决的时间问题。

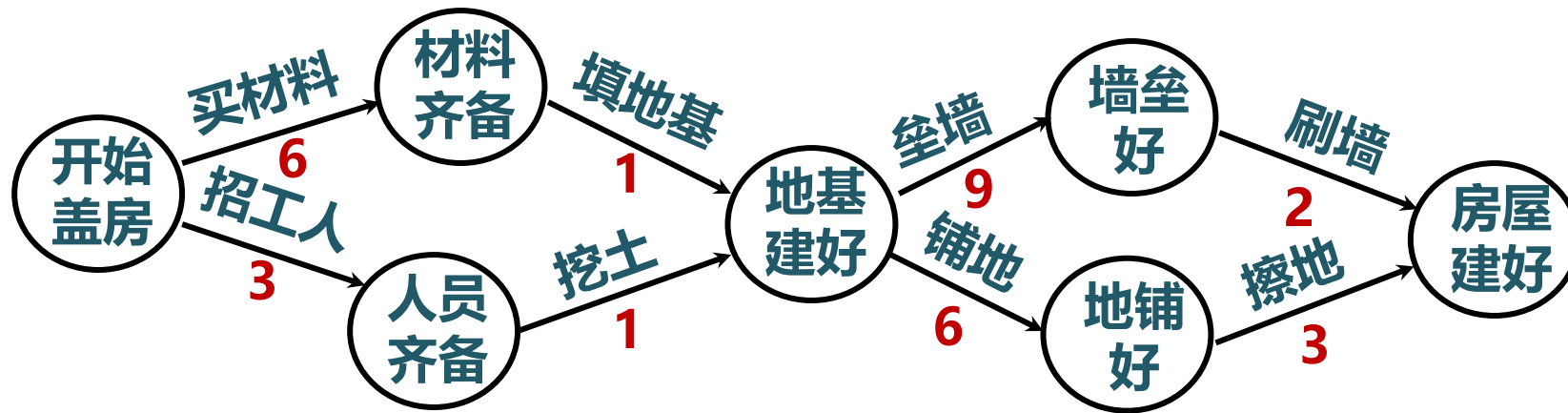
6.6 有向无环图及其应用

6-6-2 关键路径

1. AOE网的定义

AOE网 (Activity on Edge network)

有向无环图也可作为描述工程管理的有效工具。



顶点表示**事件**(状态)

弧表示**活动**

每个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**

为弧加**权**，通常表示活动所需要的时间

这种**边表示活动**的带权的有向无环图称为 **AOE 网**

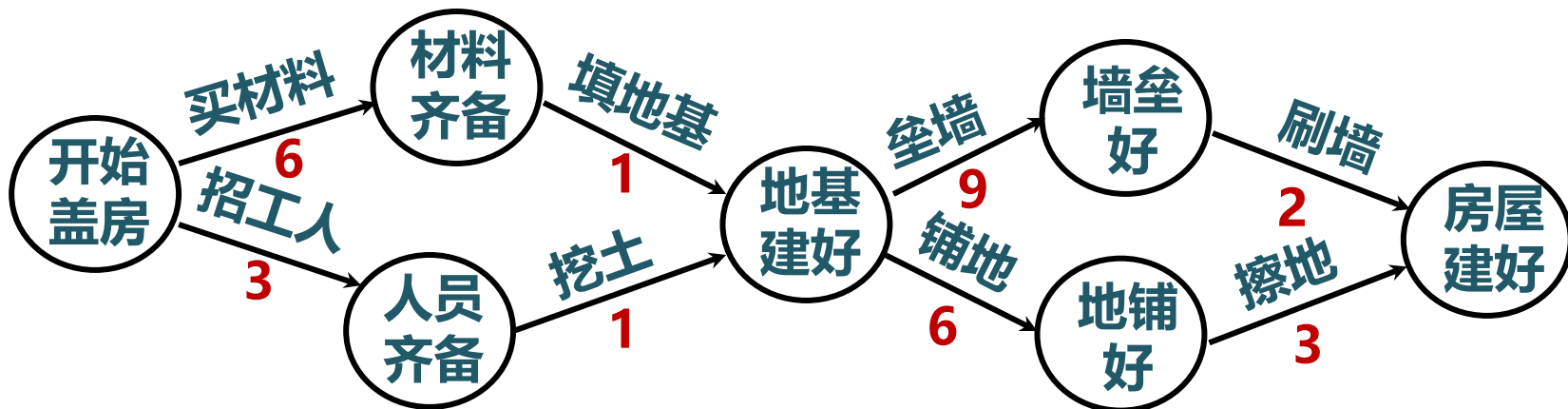
6.6 有向无环图及其应用

6-6-2 关键路径



2. 关键路径

如何求“关键活动”和“关键路径”？



通常，AOE 网中只有一个入度为 0 的顶点(源点)，一个出度为 0 的顶点(终点)。

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？

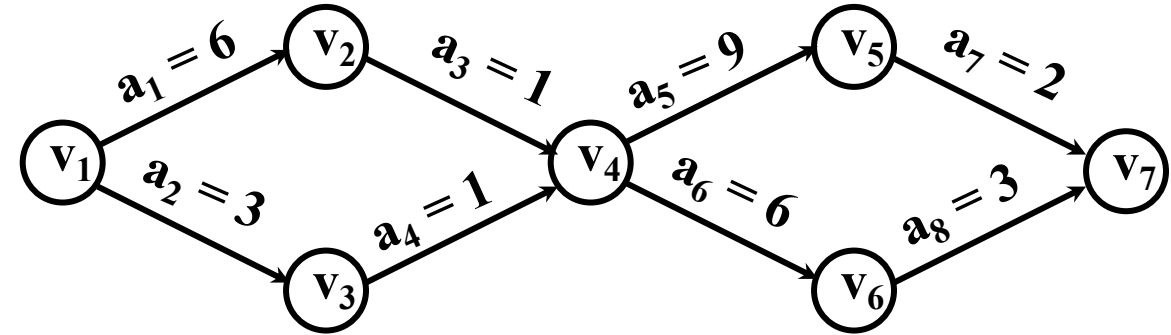
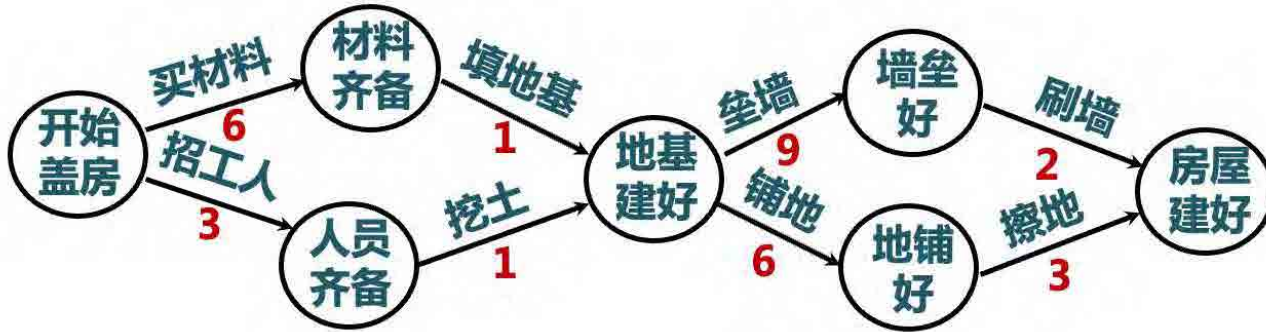
整个工程完成的时间为：从有向网的源点到汇点（终点）的最长路径（权和）。
这里的最长路径叫做**关键路径**。

“关键活动” 指的是：该弧上的权值增加将使得有向网上的最长路径的长度增加。

6.6 有向无环图及其应用

6-6-2 关键路径

2. 关键路径



“事件（顶点）”的最早发生时间 $ve(j)$

$ve(j)$ = 从源点到顶点 j 的最长路径长度；

“事件（顶点）”的最迟发生时间 $vl(k)$

不推迟整个工程完成的前提下，事件 k 最迟发生的时间。需要考虑顶点 k 到终点的最长路径。

弧表示一个子工程，

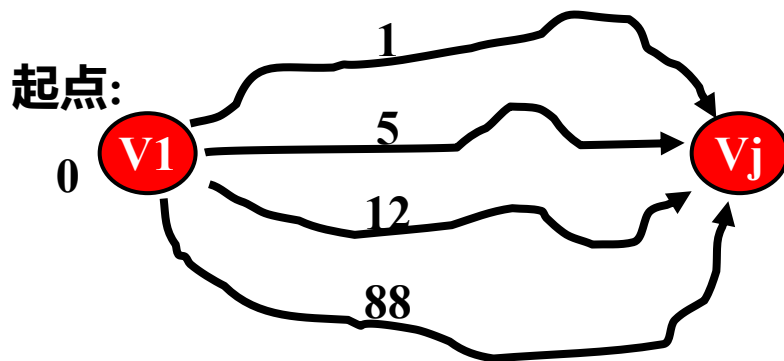
弧上的权值表示完成该项子工程所需时间。

从源点到汇点有多条不同的路径，我走捷径，工程岂不是可以最早完工？



2. 关键路径

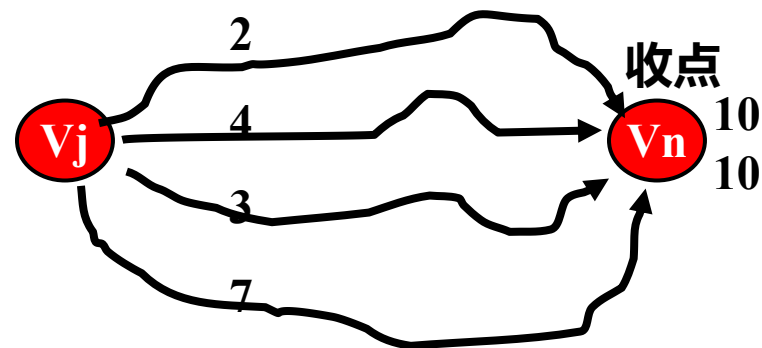
最早发生时间 $ve(j)$



$$ve(V_j) = 88$$

取 1、5、12、88 的最大值 88

最迟发生时间 $vl(k)$



$vl(V_j) =$ 取 $10-2$ 、 $10-4$ 、 $10-3$ 、 $10-7$ 的最小值 3;

或 10 - 最长路径 7

有向网中的每一条弧都是工程中必不可少的一项子工程，并行的各项子工程，以持续时间最长的子工程为关键工程，决定了整个工程完工的最早时间，而持续时间较短的子工程则有一定的自由度。

6.6 有向无环图及其应用

6-6-2 关键路径

2. 关键路径

$ve(v_i)$ —— 事件 v_i 的最早发生时间

$vl(v_i)$ —— 事件 v_i 的最迟发生时间

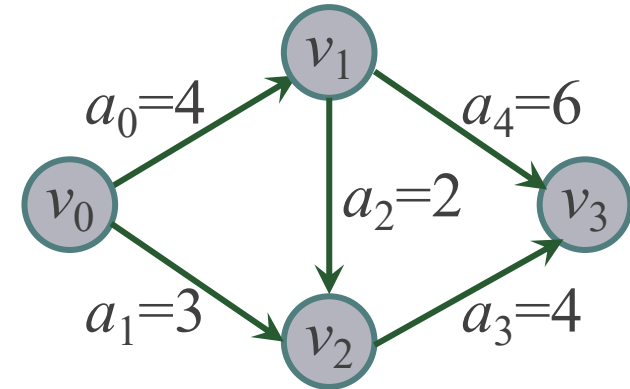
$ee(a_i)$ —— 活动 a_i 的最早开始时间

$el(a_i)$ —— 活动 a_i 的最迟开始时间

$$dut(j, k) = \text{len}\langle v_j, v_k \rangle$$

$dut(j, k)$ —— 活动 a_i 的持续时间, 满足 a_i 关联事件 v_j 、 v_k

$el(a_i) - ee(a_i)$ 意味着完成活动 a_i 的时间余量



事件——发生 → 最早
活动——开始 → 最迟

$el(a_i) = ee(a_i)$ 的活动叫做**关键活动**; $vl(v_i) = ve(v_i)$ 的事件叫做**关键事件**;

关键路径上的活动都是关键活动。关键路径上的事件都是关键事件。



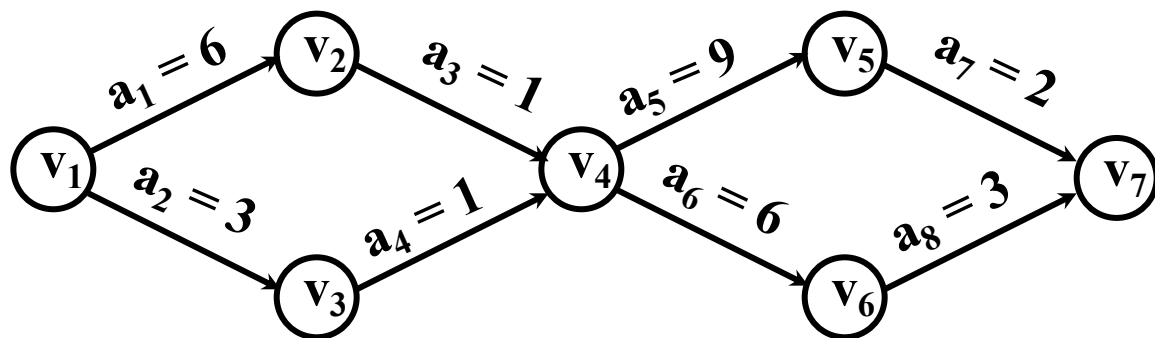
6.6 有向无环图及其应用

6-6-2 关键路径

2. 关键路径

辨别关键活动就是要找 $el(a_i) = ee(a_i)$ 的活动

因此首先必须求出 AOE 网中的所有活动的 $ee(a_i)$ 和 $el(a_i)$ 。



例 a_4

$$ee(a_4) = ve(v_3)$$

$$el(a_4) = vl(v_4) - dut(3, 4)$$

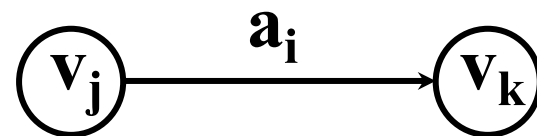
事件与活动之间的关系

设活动 a_i 关联的前后事件分别为 v_j 、 v_k

则有

$$ee(a_i) = ve(v_j)$$

$$el(a_i) = vl(v_k) - dut(j, k)$$



6.6 有向无环图及其应用

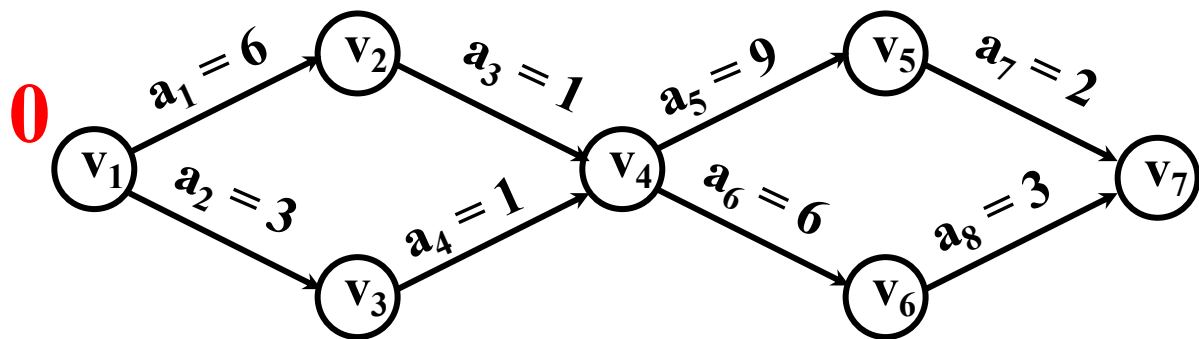
6-6-2 关键路径



2. 关键路径

问题转化为求各事件的 $ve(v_i)$ 和 $vl(v_i)$ 。

注意：源点和汇点一定是关键事件，其最早和最迟发生时间相等



$$ve(v_j) = \text{Max}\{ ve(v_i) + dut(i, j) \}$$

例，最早发生时间

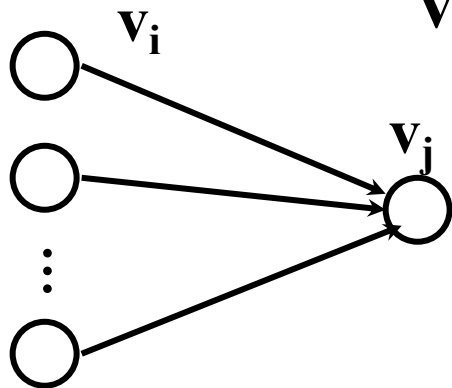
$$ve(v_2) = ve(v_1) + 6$$

$$ve(v_3) = ve(v_1) + 3$$

$$ve(v_4) = ve(v_2) + 1 ?$$

$$ve(v_4) = ve(v_3) + 1 ?$$

v_i 是 v_j 的前驱事件



6.6 有向无环图及其应用

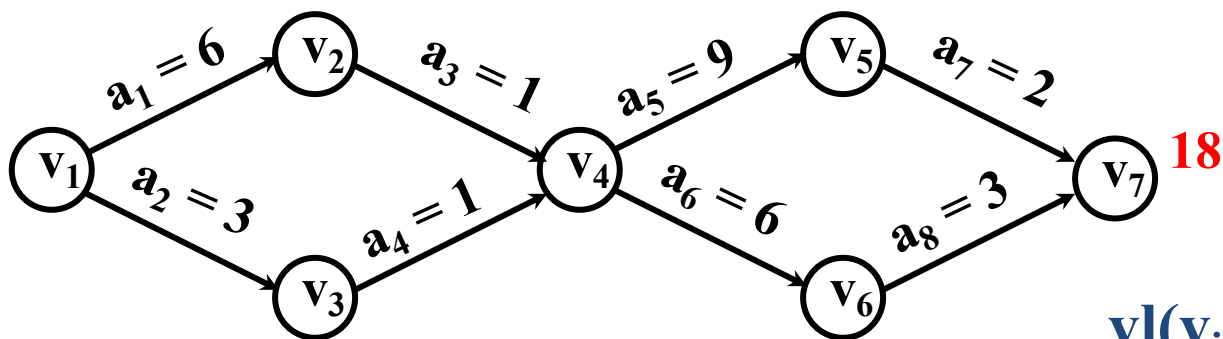
6-6-2 关键路径



2. 关键路径

问题转化为求各事件的 $ve(v_i)$ 和 $vl(v_i)$ 。

注意：源点和汇点一定是关键事件，其最早和最迟发生时间相等



$$vl(v_i) = \text{Min}\{ vl(v_j) - \text{dut}(i, j) \}$$

例，最迟发生时间

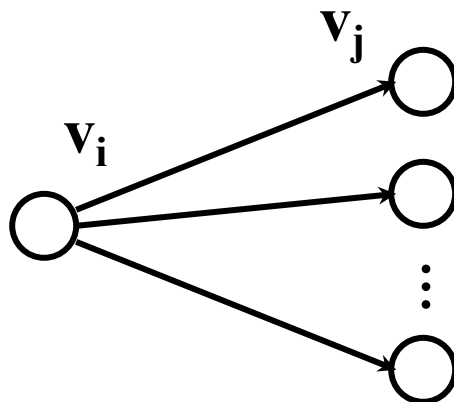
$$\text{例， } vl(v_5) = vl(v_7) - 2$$

$$vl(v_6) = vl(v_7) - 3$$

$$vl(v_4) = vl(v_5) - 9 ?$$

$$vl(v_6) - 6 ?$$

v_j 是 v_i 的后继事件





6.6 有向无环图及其应用

6-6-2 关键路径

3. 关键路径算法描述

1. 从源点 v_0 出发, 令 $ve(v_0) = 0$, 按**拓扑有序**求其余各事件的最早发生时间 $ve(v_i)$ 。

2. 从终点 v_n 出发, 令 $vl(v_n) = ve(v_n)$, 按**逆拓扑有序**求其余各事件的最迟发生时间 $vl(v_i)$ 。

3. 根据各事件的 $ve(v_i)$ 和 $vl(v_i)$, 求各活动的最早开始时间 $ee(a_j)$ 和最迟开始时间 $el(a_j)$ 。

4. $el(a_j) = ee(a_j)$ 的活动为**关键活动**。

$$ve(v_j) = \text{Max}\{ ve(v_i) + dut(i, j) \}$$

v_i 是 v_j 的前驱事件

$$vl(v_i) = \text{Min}\{ vl(v_j) - dut(i, j) \}$$

v_j 是 v_i 的后继事件

设活动 a_i 关联的前后事件分别为 v_j 、 v_k

$$ee(a_i) = ve(v_j)$$

$$el(a_i) = vl(v_k) - dut(j, k)$$

$$dut(j, k) = \text{len}\langle v_j, v_k \rangle$$



3. 关键路径算法实现

算法：关键路径算法

输入：带权有向图 $G=(V, E)$

输出：关键活动

1. 计算各个活动的最早开始时间和最晚开始时间
2. 计算各个活动的时间余量，时间余量为 0 即为关键活动

设带权有向图 $G=(V, E)$ 含有 n 个顶点 e 条边，设置 4 个一维数组：

- (1) 事件的**最早**发生时间 $ve[n]$
- (2) 事件的**最迟**发生时间 $vl[n]$ ：
- (3) 活动的**最早**开始时间 $ee[e]$
- (4) 活动的**最晚**开始时间 $el[e]$



6.6 有向无环图及其应用

6-6-2 关键路径

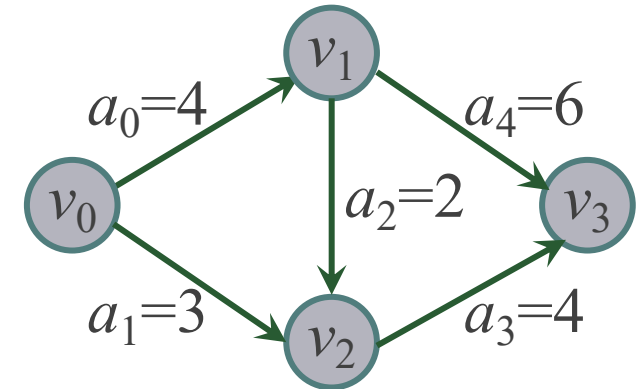
3. 关键路径算法实现

(1) 事件的**最早**发生时间 $ve[k]$

$$\begin{cases} ve[0] = 0 \\ ve[k] = \max \{ve[j] + \text{len}\langle v_j, v_k \rangle\} (\langle v_j, v_k \rangle \in p[k]) \end{cases} \quad p[k]: \text{所有到达 } v_k \text{ 的有向边}$$

 事件 v_2 的最早发生时间是多少？

$$ve[2] = \max \{ve[0] + a_1, ve[1] + a_2\} = \{0 + 3, 4 + 2\} = 6$$



AOE网的性质：只有进入 v_k 的所有活动 $\langle v_j, v_k \rangle$ 都结束， v_k 代表的事件才能发生



6.6 有向无环图及其应用

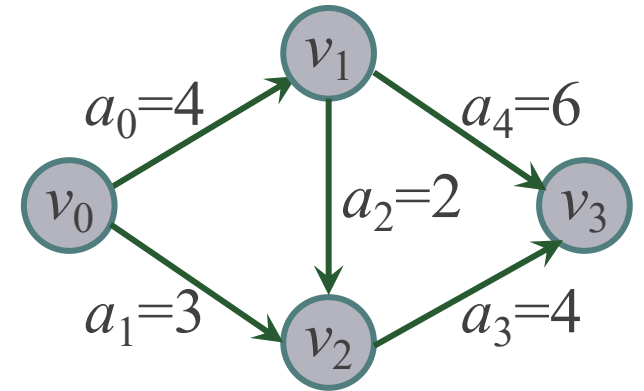
6-6-2 关键路径

3. 关键路径算法实现

(1) 事件的**最早**发生时间 $ve[k]$

$$\begin{cases} ve[0] = 0 \\ ve[k] = \max \{ve[j] + \text{len}\langle v_j, v_k \rangle\} \quad (\langle v_j, v_k \rangle \in p[k]) \end{cases} \quad p[k]: \text{所有到达 } v_k \text{ 的有向边}$$

	v_0	v_1	v_2	v_3
$ve[k]$	0	4	6	10



6.6 有向无环图及其应用

6-6-2 关键路径

3. 关键路径算法实现

(2) 事件的最迟发生时间 $vl[k]$

$$\begin{cases} vl[n-1] = ve[n-1] \\ vl[k] = \min\{vl[j] - \text{len}\langle v_k, v_j \rangle\} \quad (\langle v_k, v_j \rangle \in s[k]) \end{cases} \quad s[k]: \text{所有从 } v_k \text{ 发出的有向边}$$

 事件 v_3 的最迟发生时间是多少？

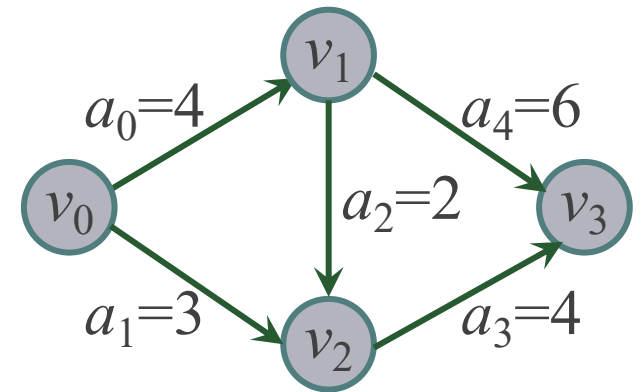
$$vl[3] = ve[3] = 10$$

 事件 v_2 的最迟发生时间是多少？

$$vl[2] = vl[3] - a_3 = 6$$

 事件 v_0 的最迟发生时间是多少？

$$ve[0] = \min\{ve[1] - a_0, ve[2] - a_1\} = \{4 - 4, 6 - 3\} = 0$$



6.6 有向无环图及其应用

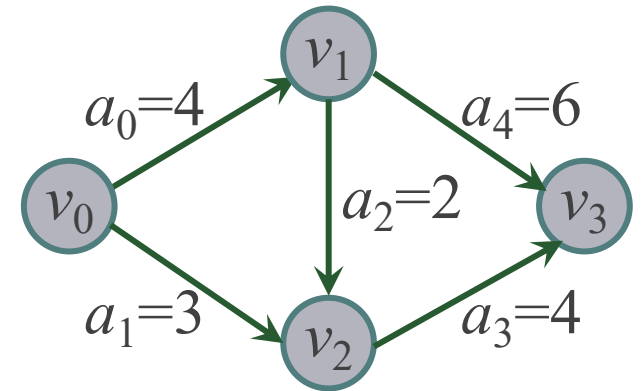
6-6-2 关键路径

3. 关键路径算法实现

(2) 事件的最迟发生时间 $vl[k]$

$$\begin{cases} vl[n-1] = ve[n-1] \\ vl[k] = \min \{ vl[j] - len\langle v_k, v_j \rangle \mid \langle v_k, v_j \rangle \in s[k] \} \end{cases} \quad s[k] : \text{所有从 } v_k \text{ 发出的有向边}$$

	v_0	v_1	v_2	v_3
$ve[k]$	0	4	6	10
$vl[k]$	0	4	6	10





3. 关键路径算法实现

(3) 活动的**最早**开始时间 $ee[i]$

(4) 活动的**最晚**开始时间 $el[i]$

若活动 a_i 由有向边 $\langle v_k, v_j \rangle$ 表示, 则

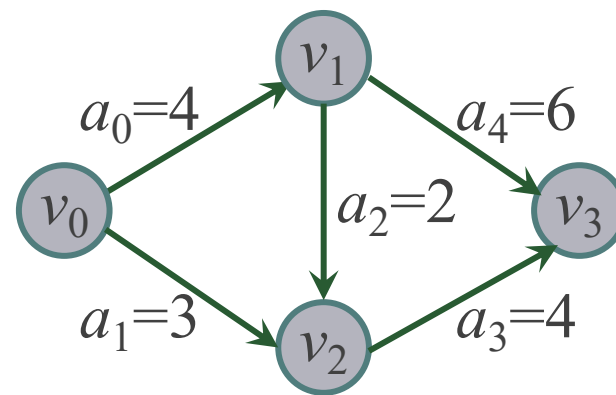
$$\begin{cases} ee[i] = ve[k] \\ el[i] = vl[j] - len\langle v_k, v_j \rangle \end{cases}$$

 活动 a_2 的最早开始时间是多少?

$$ee[2] = ve[1] = 4$$

 活动 a_2 的最晚开始时间是多少?

$$el[2] = vl[2] - 2 = 4$$





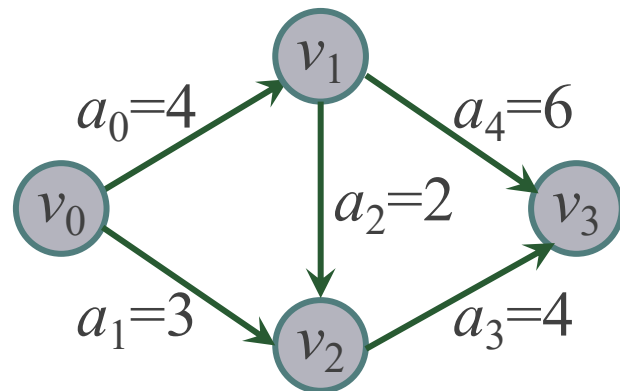
3. 关键路径算法实现

(3) 活动的**最早**开始时间 $ee[i]$

若活动 a_i 由有向边 $\langle v_k, v_j \rangle$ 表示, 则

$$\begin{cases} ee[i] = ve[k] \\ el[i] = vl[j] - len\langle v_k, v_j \rangle \end{cases}$$

(4) 活动的**最晚**开始时间 $el[i]$



	v_0	v_1	v_2	v_3
$ve[k]$	0	4	6	10
$vl[k]$	0	4	6	10

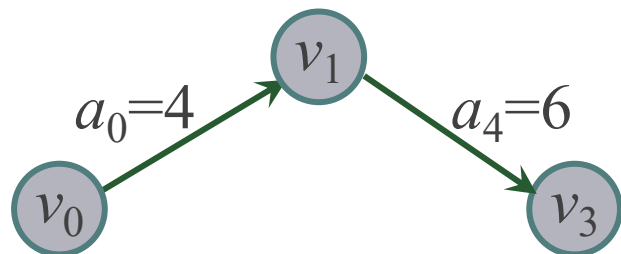
	a_0	a_1	a_2	a_3	a_4
$ee[i]$	0	0	4	6	4
$el[i]$	0	3	4	6	4

6.6 有向无环图及其应用

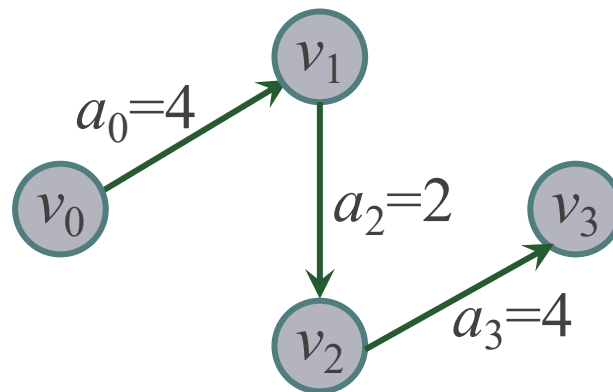
6-6-2 关键路径



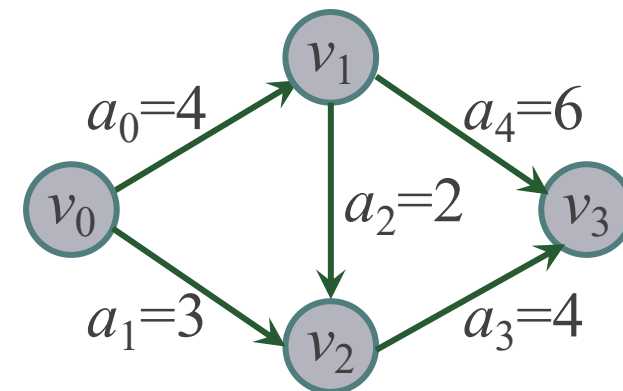
3. 关键路径算法实现



	v_0	v_1	v_2	v_3
ve[k]	0	4	6	10
vl[k]	0	4	6	10



	a_0	a_1	a_2	a_3	a_4
ee[i]	0	0	4	6	4
el[i]	0	3	4	6	4

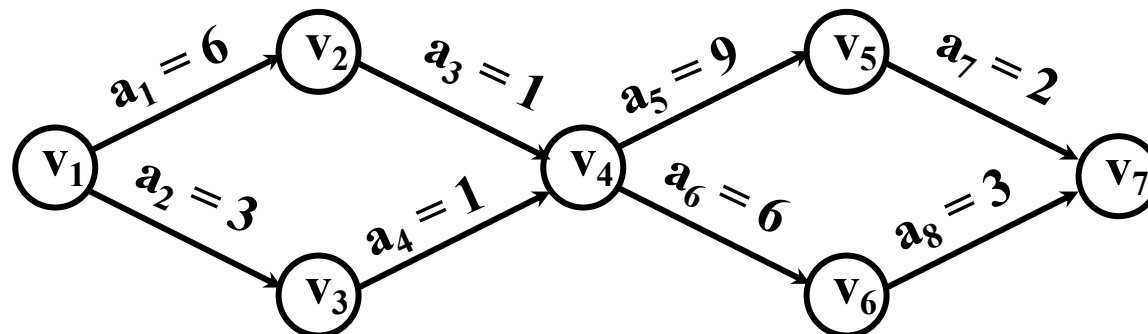


6.6 有向无环图及其应用



6-6-2 关键路径

例题



事件——发生 → 最早
活动——开始 → 最迟

关键路径 **a₁ a₃ a₅ a₇**

$$ve(v_j) = \text{Max}\{ ve(v_i) + \text{dut}(i, j) \}$$

v_i 是 v_j 的前驱事件

$$vl(v_i) = \text{Min}\{ vl(v_j) - \text{dut}(i, j) \}$$

v_j 是 v_i 的后继事件

设活动 **a_i** 关联的前后事件分别为 **v_j**、**v_k**

$$ee(a_i) = ve(v_j)$$

$$el(a_i) = vl(v_k) - \text{dut}(j, k)$$

拓扑有序

事件	ve	vl
v ₁	0	0
v ₂	6	6
v ₃	3	6
v ₄	7	7
v ₅	16	16
v ₆	13	15
v ₇	18	18

逆拓扑有序

活动	ee	el
a ₁	0	0
a ₂	0	3
a ₃	6	6
a ₄	3	6
a ₅	7	7
a ₆	7	9
a ₇	16	16
a ₈	13	15

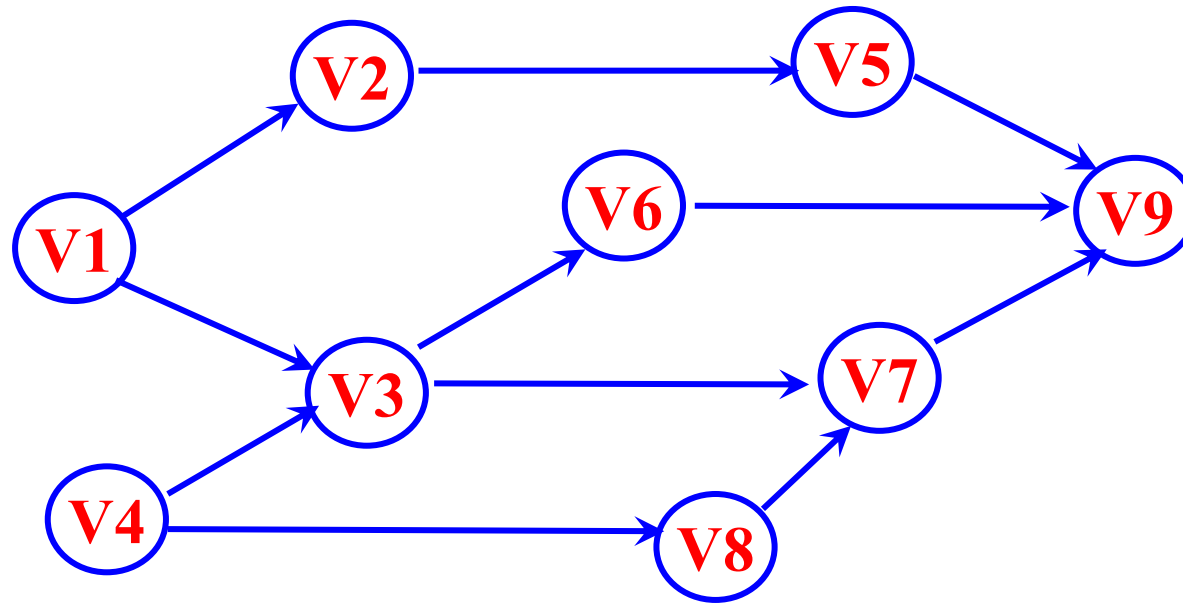


小结

1. 掌握拓扑排序算法及实现
2. 理解AOV网的定义及性质
3. 理解关键路径算法及实现方法

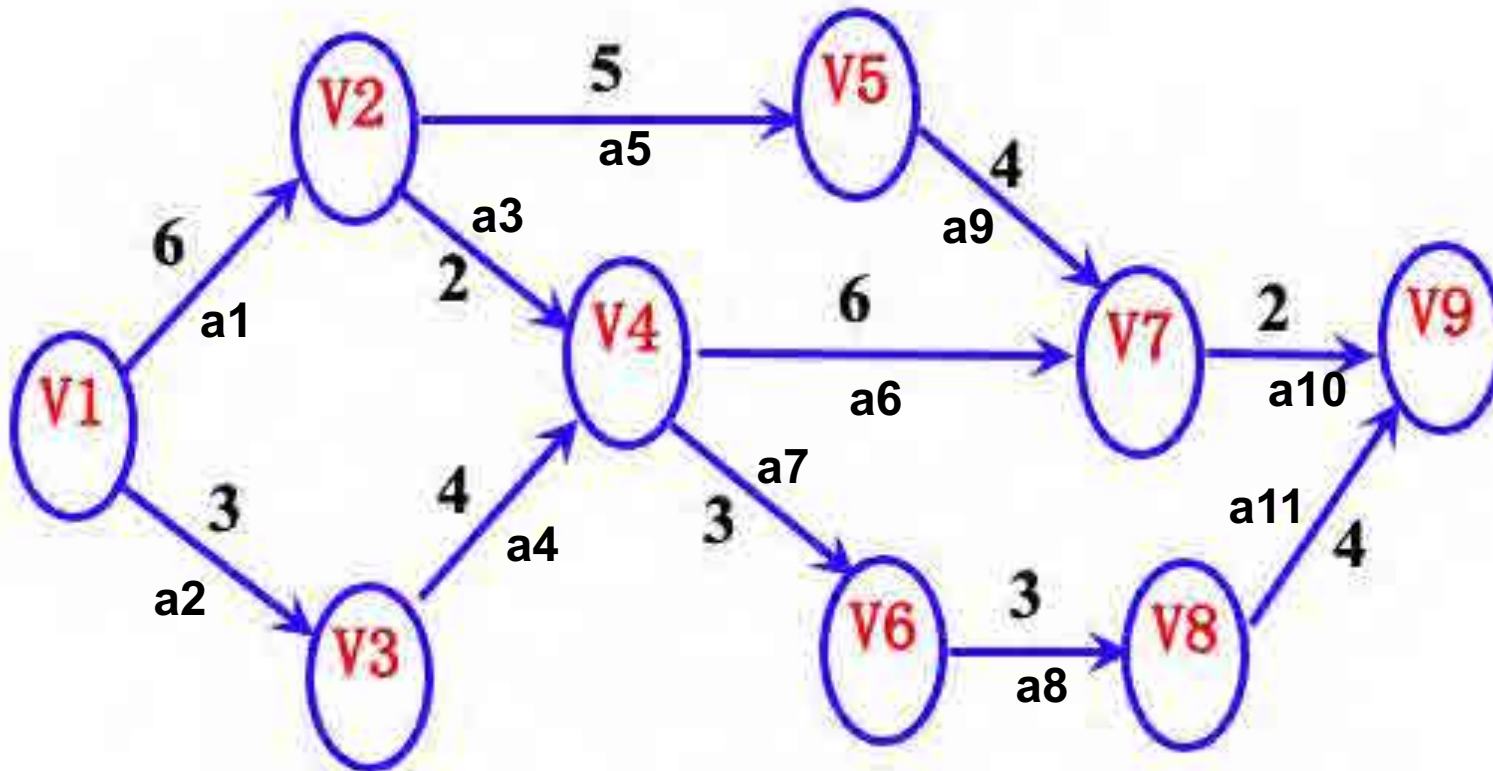
作业

有向图如下图所示，请写出它的一个拓扑有序序列。



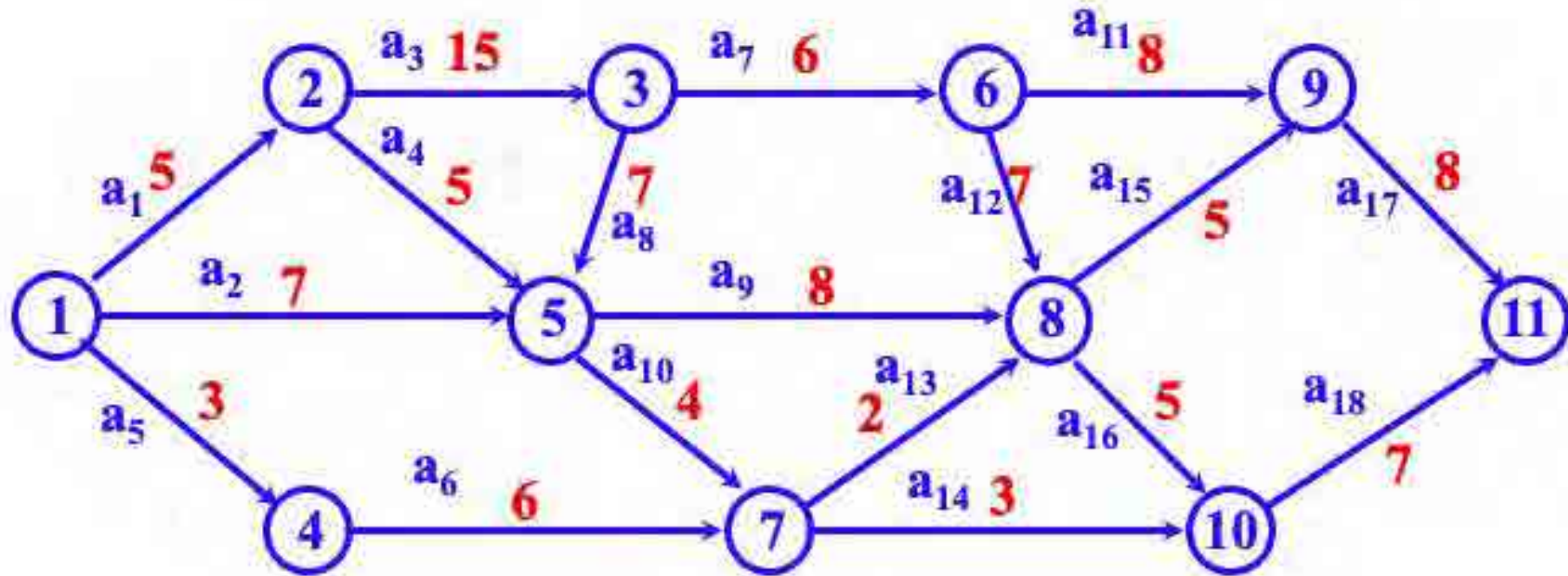
作业

求 AOE 网关键路径 要求：构造事件表和活动表



作业

求 AOE 网关键路径 要求：构造事件表和活动表





本章总结

- 1、熟悉图的各个基本概念
- 2、掌握图的各种存储结构，能写出其存储表示
- 3、掌握图遍历的递归算法（DFS&BFS）
- 4、掌握图的连通性问题，能求出最小生成树：Prim, Kruskal
- 5、掌握最短路径的算法：Dijkstra、Floyd
- 6、掌握有向无环图，能进行拓扑排序，会求关键路径



Thank You !

Q & A

作业

有向图如下图所示，请写出它的一个拓扑有序序列。

