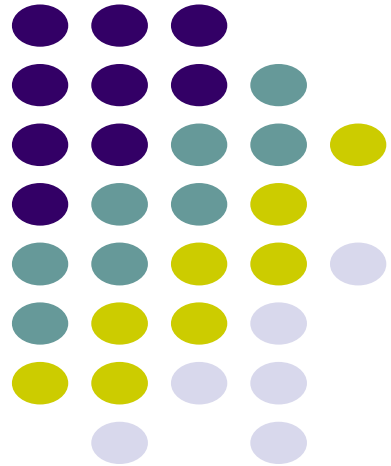
L19：关键路径

吉林大学计算机学院
谷方明
fmgu2002@sina.com

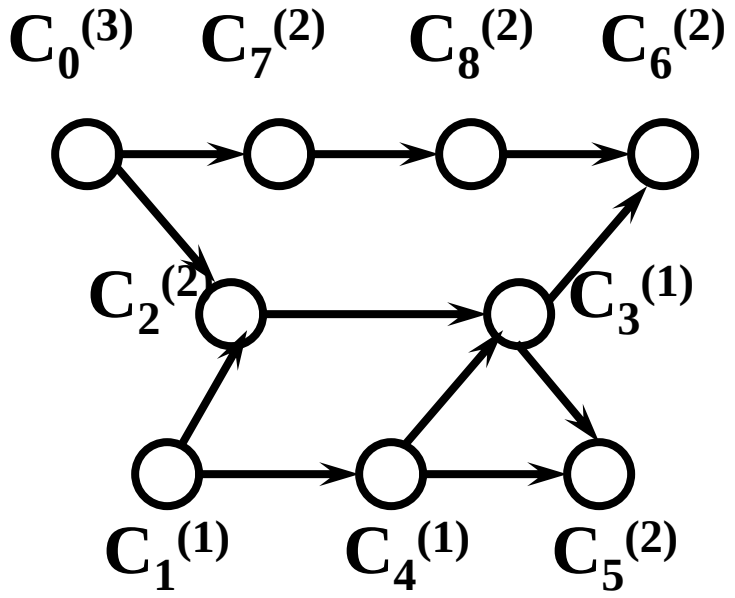


学习目标

- 掌握AOE网、关键路径等概念。
- 掌握关键路径的求解方法、技巧及相关分析。

例：选课（续）

课程代号	课程名称	先修课程	学期数
C_0	高等数学	无	3
C_1	程序设计基础	无	1
C_2	离散数学	C_0,C_1	2
C_3	数据结构	C_2,C_4	1
C_4	程序设计语言	C_1	1
C_5	编译技术	C_3,C_4	2
C_6	操作系统	C_3,C_8	2
C_7	普通物理	C_0	2
C_8	计算机原理	C_7	2

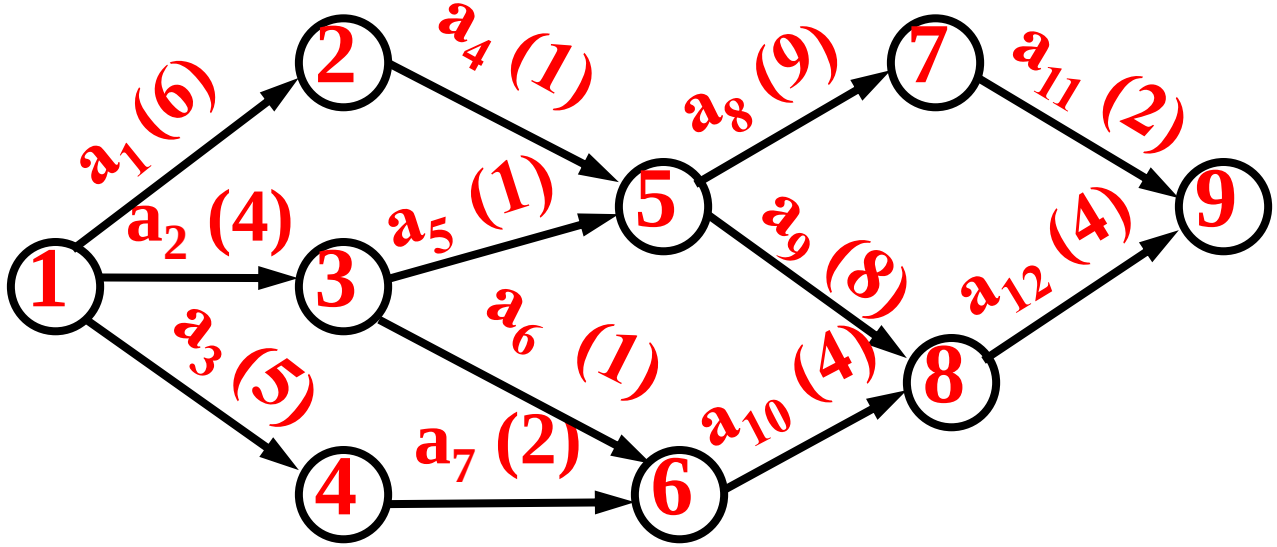


AOE网

- 时间约束

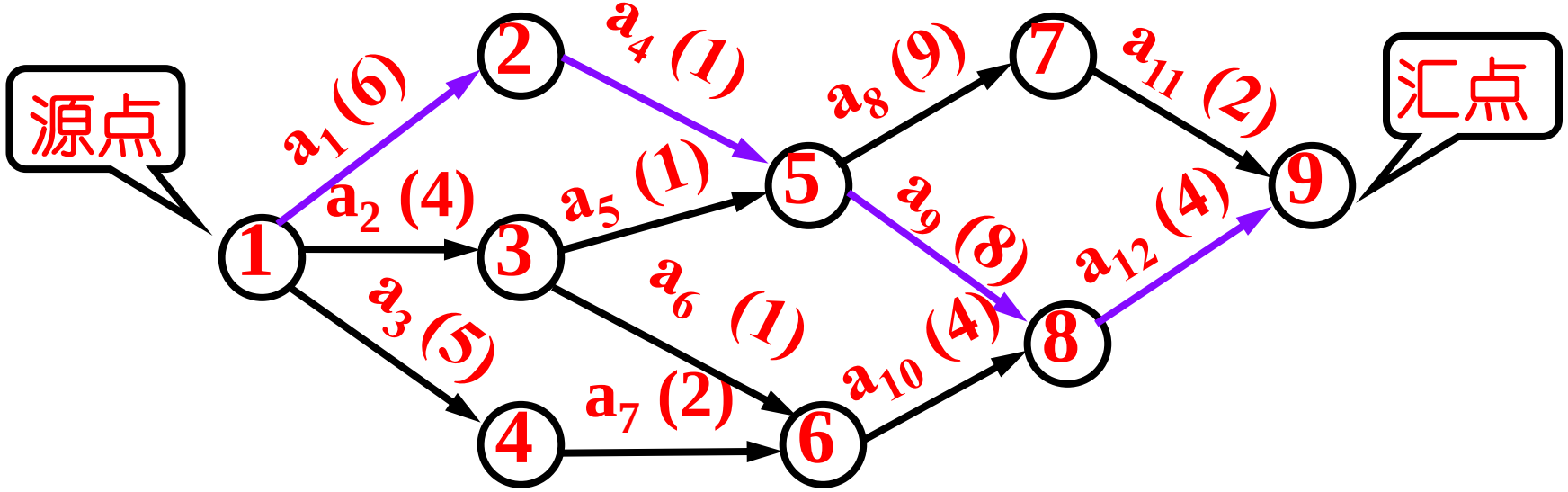
- 边表示活动(Activity)
- 边的权值表示活动的持续时间(Duration)
- 顶点表示入边的活动已完成，出边的活动可以开始的状态，也称为事件(Event)
- 这样的有向无环带权图叫做AOE (Activity On Edges)网。

例:工程



- AOE网中，有些活动可并行进行。只有各条路径上所有活动都完成了，整个工程才算完成。
- 完成整个工程至少需要多少时间？
- 为了缩短工程时间，应加快哪些活动？为了不延误整个工期，哪些活动不得延期，哪些可延期？

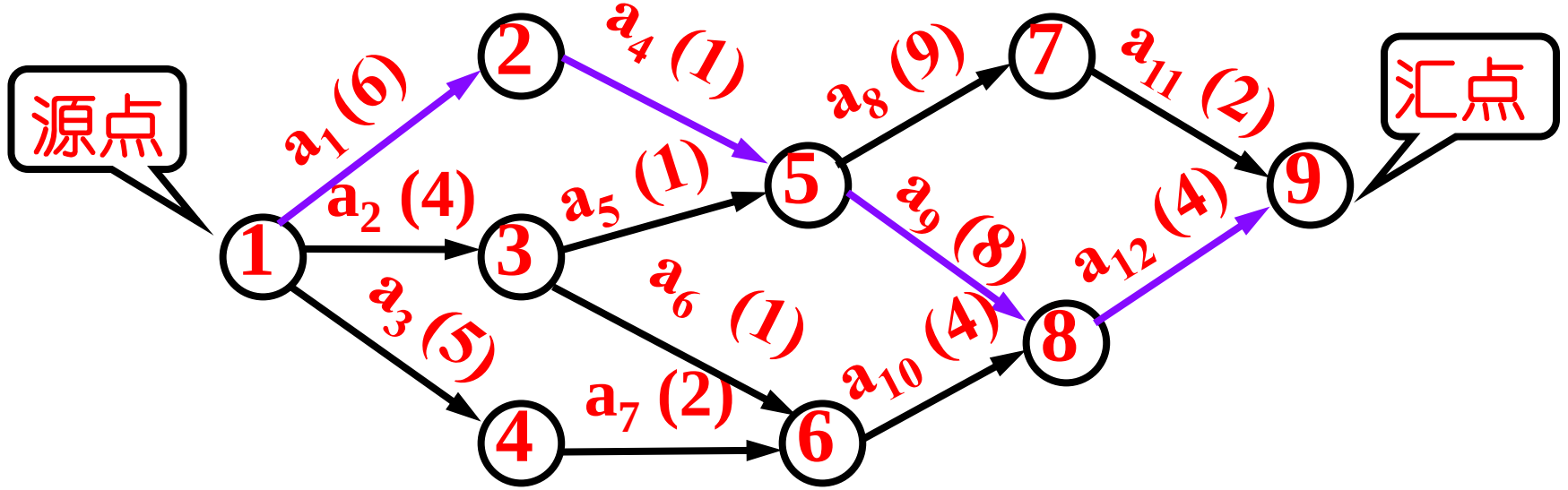
源点和汇点



- 源点：表示整个工程的开始（入度为零）
- 汇点：表示整个工程的结束（出度为零）
- 完成整个工程所需的时间：取决于从源点到汇点的最长路径长度。

关键路径和关键活动

- 关键路径： AOE网中具有最大长度的路径。
- 关键活动：不得延期的活动。

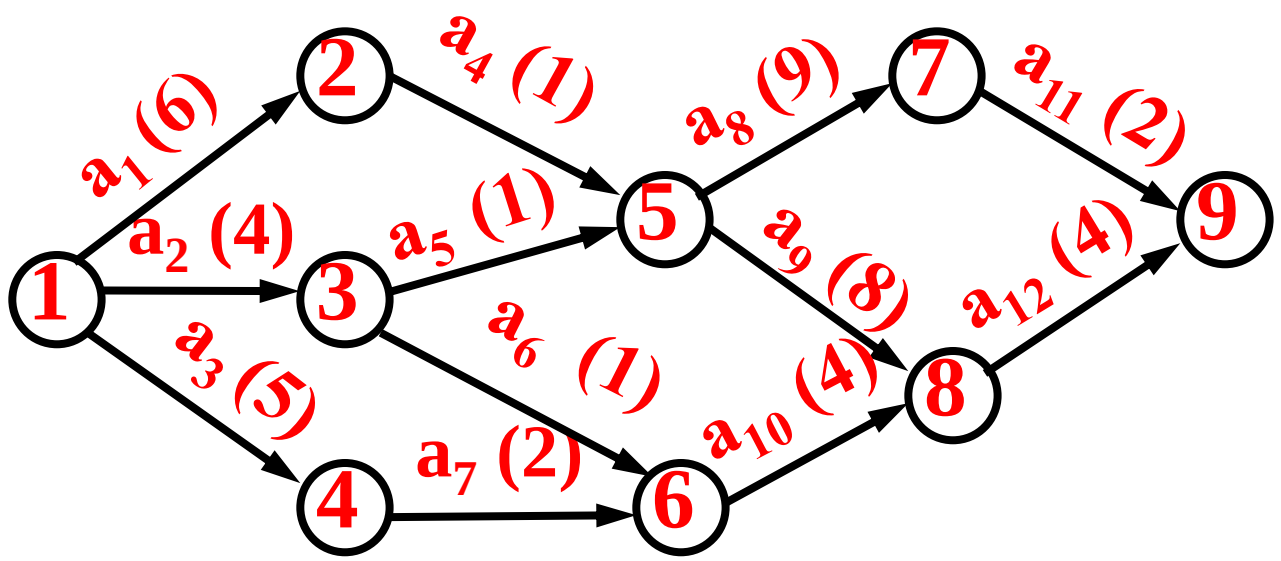


- 关系：关键活动是关键路径上的活动；关键路径是从源点到汇点由关键活动构成的路径。

1. 求关键路径

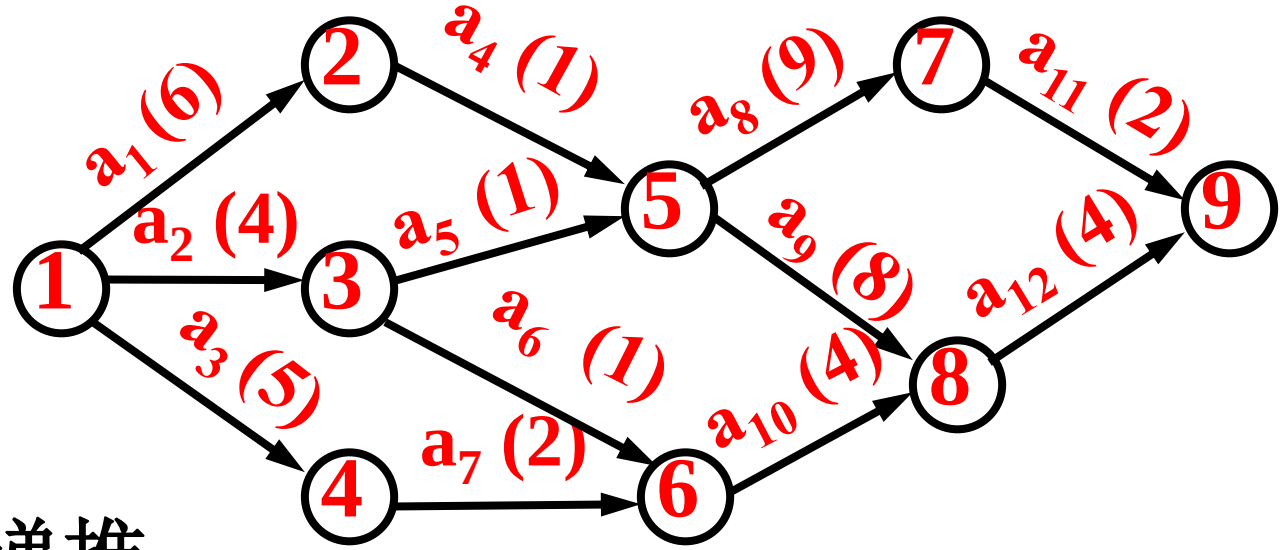
①定义从源点v_0到v_i的最长路径长度

为事件v_i的最早开始时间 ve(i)



ve(1)=0
ve(2)= 6
ve(3)= 4
ve(4)= 5
ve(5)= 7
ve(6)= 7
ve(7)= 16
ve(8)= 15
ve(9)= 19

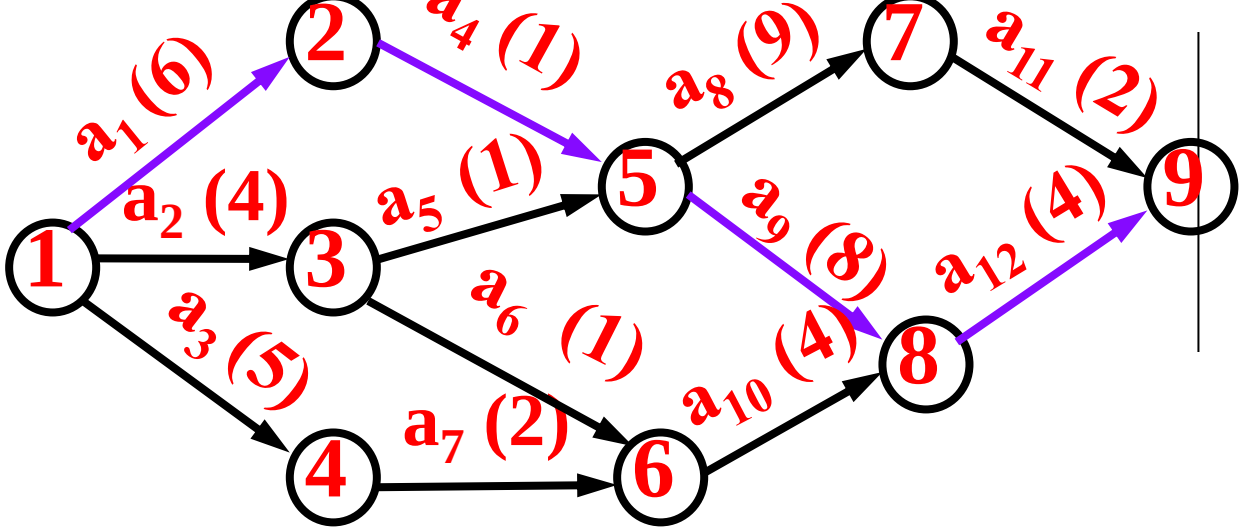
ve(i) 的计算



按拓扑序递推

$$ve(1)=0 \qquad k=1$$

$$ve(k)\begin{cases} \max\{ve(j)+weight(<j,k>)\} \\ <v_j, v_k> \in E(G),\ k=2, 3, \ldots, n \end{cases}$$



ve(1)=0

ve(2)= ve(1)+weight(<1,2>)=6

ve(3)= ve(1)+weight(<1,3>)=4

ve(4)= ve(1)+weight(<1,4>)=5

ve(5)= max{ve(2)+ weight(<2 ,5>),
ve(3)+ weight(<3 ,5>)}=max{6+1,4+1}=7

ve(6)= max{ve(3)+ weight(<3 ,6>),
ve(4)+ weight(<4 ,6>)}=max{4+1,5+3}=7

ve(7)= ve(5)+weight(<5,7>)=7+9=16

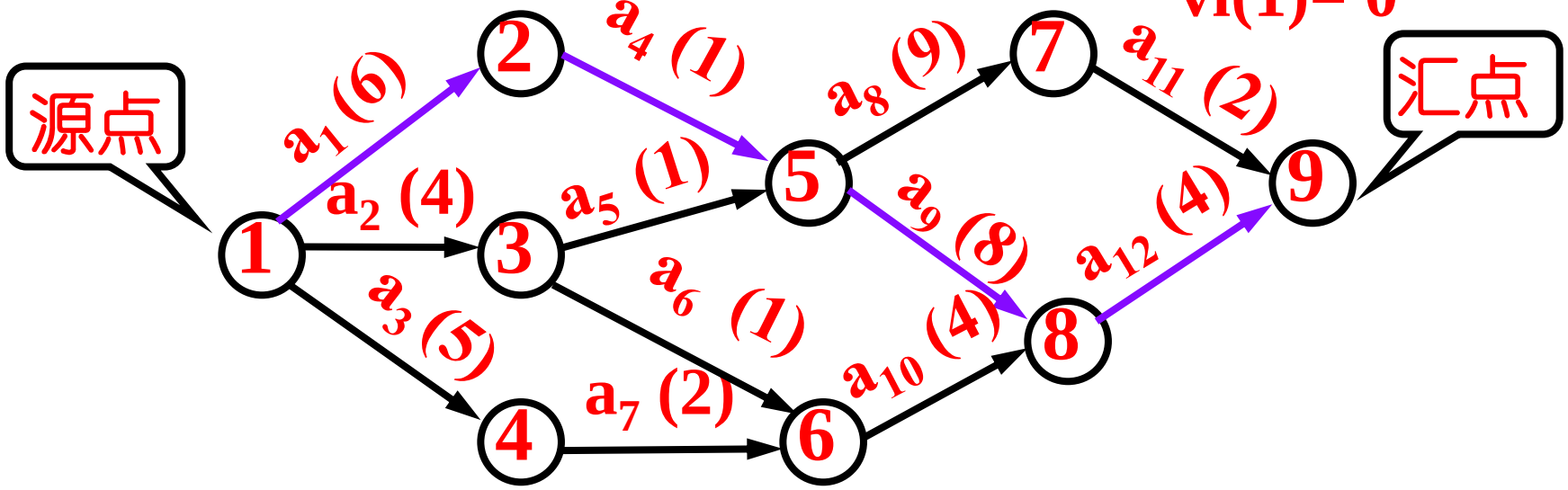
ve(8)= max{ve(5)+ weight(<5,8>),
ve(6)+ weight(<6,8>)}=max{7+8,7+4}=15

ve(9)= max{ve(7)+ weight(<7,9>),
ve(8)+ weight(<8,9>)}=max{16+2,15+4}=19

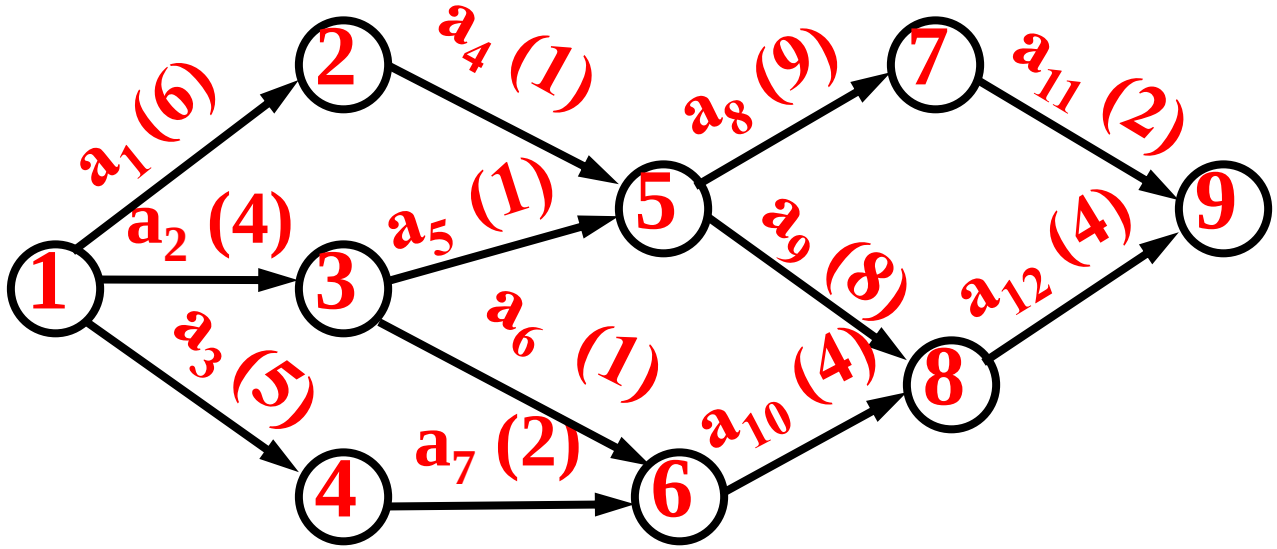
2.求关键活动

vl(9)= 19
vl(8)= 15
vl(7)= 17
vl(6)= 11
vl(5)= 7
vl(4)= 9
vl(3)= 6
vl(2)= 6
vl(1)= 0

②保证工期不推迟的前提下，事件v_j允许的最迟开始时间 vl (j):等于ve(n)减去从v_j到v_n最长路径长度。

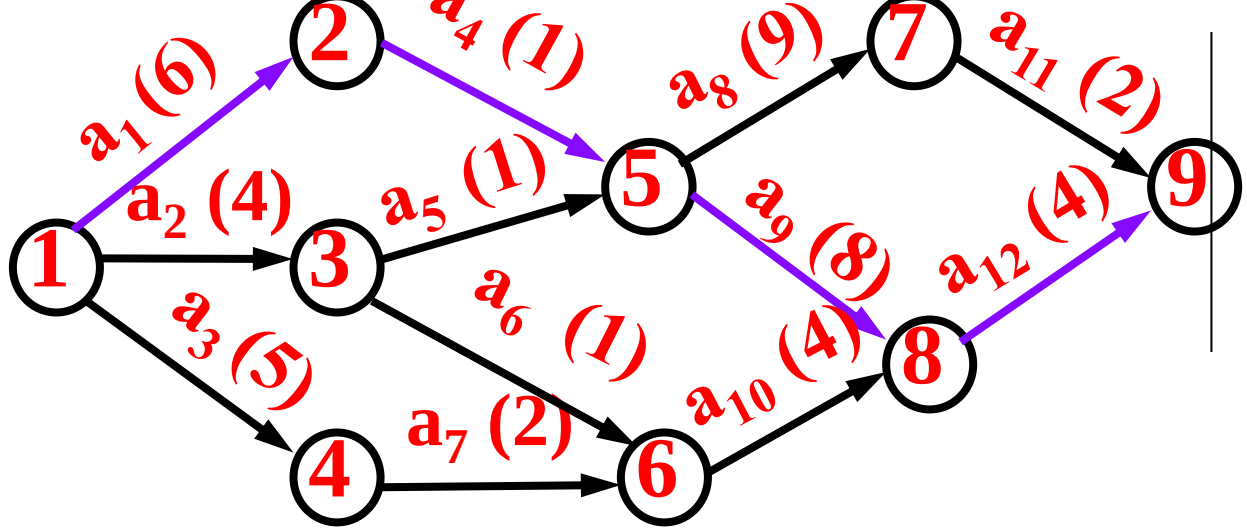


递推求vl(k)



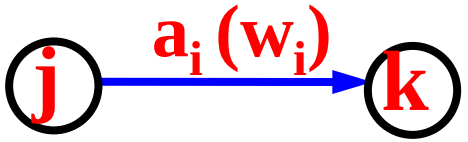
按逆拓扑序递推:

$$vl(j)\begin{cases} ve(n) & j=n \\ \min\{vl(k)-weight(<j,k>)\} & <v_j, v_k>\in E(G),\ j=n-1, n-2,\ldots,1 \end{cases}$$



vl(9)= ve(9)=19
vl(8)= vl(9)-weight(<8,9>)=15
vl(7)= vl(9)-weight(<7,9>)=17
vl(6)= vl(8)-weight(<6,8>)=11
vl(5)= min{vl(8)- weight(<5 ,8>),
vl(7)- weight(<5 ,7>)} =min{15-8,16-9}=7
vl(4)= vl(6)-weight(<4,6>)=11-2=9
vl(3)= min{vl(6)- weight(<3 ,6>),
vl(5)- weight(<3 ,5>)} =min{11-1,7-1}=6
vl(2)= vl(5)-weight(<2,5>)=7-1=6
vl(1)= min{vl(2)- weight(<1 ,2>),
vl(3)- weight(<1 ,3>),
vl(4)- weight(<1 ,4>)} = min{6-6,6-4,9-5}=0

关键活动有关的量



③ 活动a_i的最早开始时间e(i)：

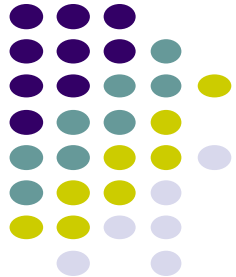
设活动 a_i 为有向边$<v_j, v_k>$，则 e(i) = ve(j)。

- ✓ ve(j)是从源点v_0到v_j的最长路径长度，决定了所有从v_j开始的活动的最早开始时间。

④ 活动a_i的最迟开始时间 l(i)：

设活动 a_i 为有向边$<v_j, v_k>$，则 l(i) = vl(k)-weight(<j, k>)。

- ✓ l(i) 是在不会引起工期延误的前提下，该活动允许的最迟开始时间。

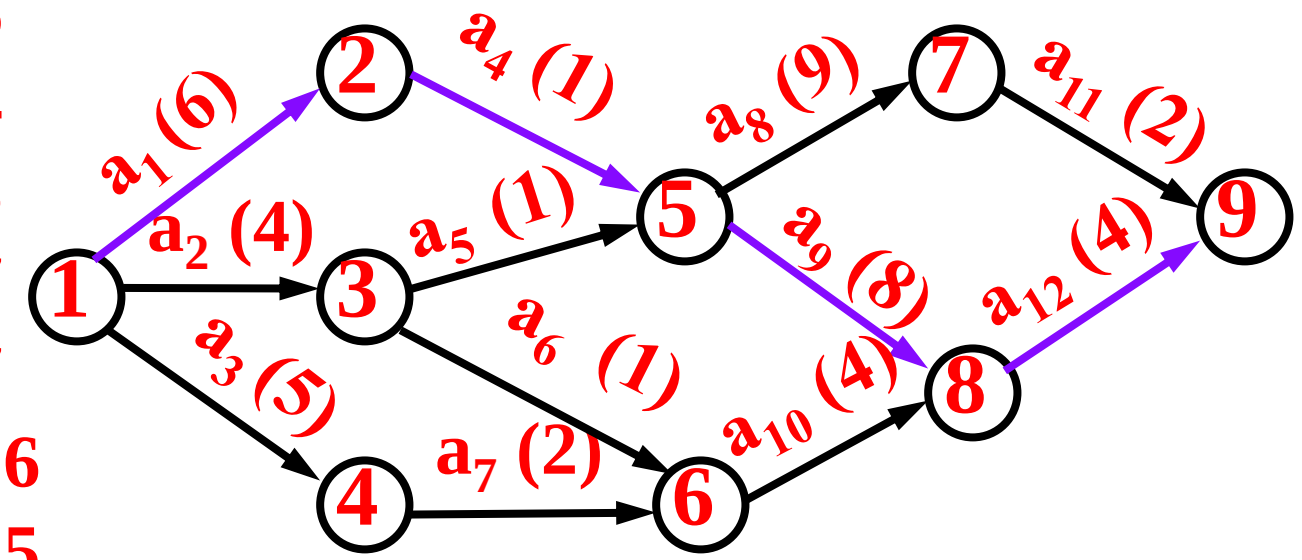


- 关键活动： l(i)＝ e(i)，表示活动a_i 是没有时间余量的关键活动。

- 为找出关键活动, 需要求各个活动的 **e(i)** 与 **l(i)**；为求得**e(i)** 与 **l(i)**，需要先求得各个顶点V_j 的 **ve(j)** 和 **vl(j)**。

[例] 求关键活动

ve(1)=0
ve(2)= 6
ve(3)= 4
ve(4)= 5
ve(5)= 7
ve(6)= 7
ve(7)= 16
ve(8)= 15
ve(9)= 19

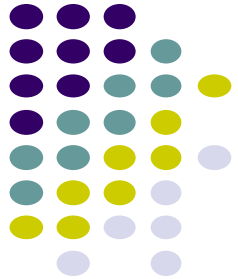


vl(9)= 19
vl(8)= 15
vl(7)= 17
vl(6)= 11
vl(5)= 7
vl(4)= 9
vl(3)= 6
vl(2)= 6
vl(1)= 0

e(i)=ve(j), l(i)=vl(k)-weight(<j,k>)

a_i	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
e(i)	0	0	0	6	4	4	5	7	7	7	16	15
l(i)	0	2	4	6	6	10	9	8	7	11	17	15
l_i-e_i	0	2	2	0	2	5	3	1	0	4	0	0

求关键路径和关键活动的算法



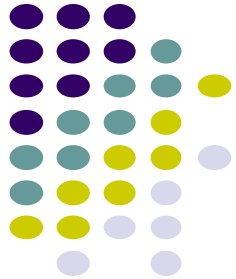
① 对AOE网拓扑排序; 若网中有回路，则终止算法;

② 按拓扑次序求出各顶点事件的最早发生时间ve;

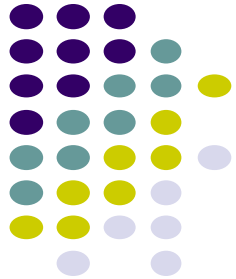
③ 按拓扑序列的逆序求各顶点事件的最迟发生时间vl;

④ 根据ve和vl的值，求各活动的最早开始时间e(i)与最迟开始时间l(i)，若e(i)=l(i)，则i是关键活动。

图的关键路径核心算法

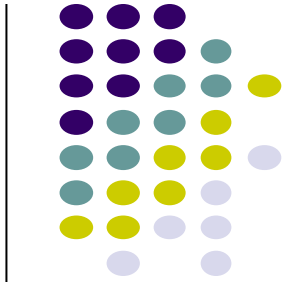


```
算法CriticalPath ( )
/* 关键路径核心算法,假定源点1、汇点n、拓扑序1~n */
CPath2[计算事件的最早发生时间]
    for ( i  = 1 ; i <= n ; i ++ ) ve [i] = 0;
    for ( i  = 1 ; i <= n-1 ; i ++ ) /*按拓扑序递推求ve[i]*/
        for ( p = Head [i] . adjacent ; p ; p = p-> link) {
            k = p -> VerAdj ;
            if (ve[i] + p -> cost  > ve[k] )
                  ve[k] =ve[i] + p -> cost ;
      } //不必存逆图；前点处理时更新后点，下标1..n-1
```



CPath3[计算事件的最迟发生时间]

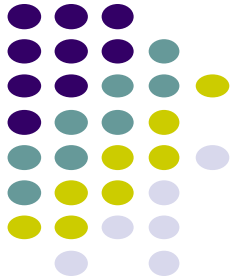
```
for ( i  = 1 ; i <= n ; i ++ ) vl [i] = ve [n];

for ( i  = n-1 ; i >= 1 ; i -- ) /*按逆拓扑序求vl*/
   for ( p = Head [i] . adjacent ; p ; p = p-> link) {
      k = p -> VerAdj ;
      if (vl[k] –  p-> cost  < vl[i] )
            vl[i] = vl[k] –  p-> cost ;
```



CPath4[求诸活动的最早开始时间和最迟开始时间]

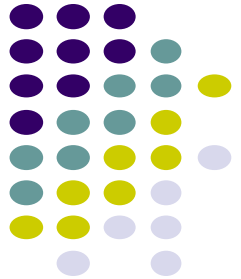
```
for ( i  = 1 ; i <= n ; i ++ )
    for ( p = Head [i] . adjacent ; p ; p = p-> link) {
        k = p -> VerAdj ;
        e =   ve[i] ;
        l  =   vl[k] – cost(p) ;
        if ( l = = e )
                cout << "< i, k> is Critical Activity! "
    } ▌
```

CriticalPath算法的说明



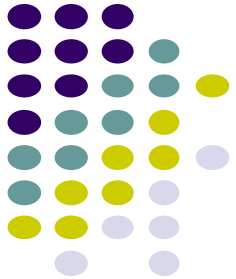
- 增加拓扑排序
  - ✓ CPath1：调用算法TopoOrder( )，若返回false，则有环，CPath算法结束；否则，拓扑序列存放在torder[ ].
  - ✓ 更改CPath算法的2、3两步。即在第2、3计算循环中引入变量 j，使 j=torder[i];然后用 j 代替 i 进行计算。
- 如果只求关键路径
  - ✓ 方法一： 只需CPath1 和 CPath2两步
  - ✓ 方法二： 进行拓扑排序时直接求

CriticalPath算法分析



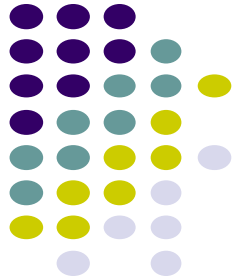
- 时间复杂性
  - ✓ 进行拓扑排序的时间复杂性为O(n+e),
  - ✓ 以拓扑序求ve[i]和以逆拓扑序求vl[i], 所需时间均为O(e),
  - ✓ 求各个活动的e[k]和l[k]的时间复杂度为O(e),
  - ✓ 整个算法的时间复杂性是O(n+e)

算法CriticalPath的补充



- **CriticalPath算法正确性**
  - ✓ 关键路径的定义和ve的递推公式
  - ✓ 关键活动的定义（也可从定理6.3的推论看出）
- **定理6.3**:任意的非空AOE网至少存在一条关键路径。**请忽略该定理。**
- **定理6.3（新）： 关键路径上的顶点满足ve(i)= vl(i) 。**

引理：假设AOE网的事件集为{T1 ,T2 , …,Tn }，n ≥1. 对任意i有vl(i) ≥ ve(i).



- 证明：由vl(i)和ve(i)的定义知，对任意出度为零的顶点上式显然成立。假设顶点Ti的所有后继顶点也满足上式，根据vl 的定义知存在顶点Tj和边<Ti, Tj>, 且

  vl(i)=vl(j)–weight(<Ti, Tj>)

  又由ve的定义知：

  ve(j) ≥ve(i)+weight(<Ti, Tj>)

  ve(j)–weight(<Ti, Tj>) ≥ ve(i)

  由归纳假设知vl(j) ≥ ve(j)， 从而

  vl(i) = vl(j)–weight(<Ti, Tj>)

  ≥ve(j)–weight(<Ti, Tj>)≥ ve(i)

定理6.3（新）证明

假定路径 $Ti_1, Ti_2, \ldots, Ti_k$是所有从入度为零的顶点到出度为零的顶点的具有最大路径长度的路径。由于该路径具有最大路径长度，因此

$ve(i_1)=0$, $ve(i_2)=ve(i_1)+weight(< Ti_1, Ti_2 >)$,…,

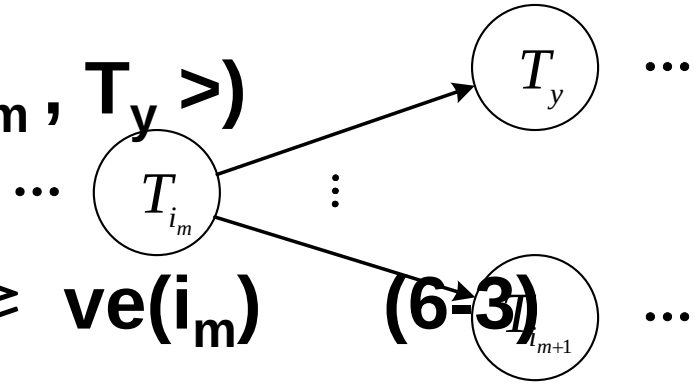
$ve(i_j)=ve(i_{j-1})+weight(< Ti_{j-1}, Ti_j >)$,… (6-2)

当$j=k$时，显然有$ve(i_k)=vl(i_k)$

假定当$j>m$时有$ve(i_j)=vl(i_j)$，并假定Ti_m 发出的边至少有两条（只有一条，显然得证）。不妨设是$< Ti_m , Ti_{m+1}>$ 和$< Ti_m , T_y>$ ，如图所示.

由ve的定义知

$ve(y) \geqslant ve(i_m) + weight(< Ti_m , T_y >)$



由引理,得

$vl(y) - weight(< Ti_m , T_y >) \geqslant ve(i_m)$ (6-3)

又由(6-2)式得

$ve(i_m) = ve(i_{m+1}) - weight(< Ti_m , Ti_{m+1} >)$

再由归纳假设有

$ve(i_m) = vl(i_{m+1}) - weight(< Ti_m , Ti_{m+1} >)$

从而结合(6-3)式得 ：

$vl(i_{m+1}) - weight(< Ti_m , Ti_{m+1} >) \leqslant vl(y) - weight(< Ti_m , T_y >)$

由y的任意性知:

$vl(i_m) = vl(i_{m+1}) - weight(< Ti_m , Ti_{m+1} >) = ve(i_m)$

定理6.3（新）拓展

- 推论 1：假设边<Ti, Tj>属于AOE网，则有

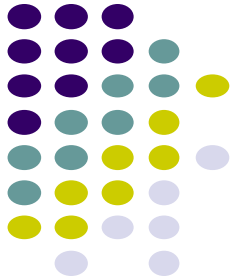
  vl(j) - ve(i) ≥ weight(< Ti , Tj >)

  且如果<Ti, Tj>属于某一条关键路径，则

  vl(j) - ve(i) = weight(< Ti , Tj >)

- 由推论可知，活动<T_i, T_j>的最大可利用时间是***vl(j)-ve(i)***. 如果最大可利用时间等于权值***Weight(<T_i, T_j>)***，则该活动是关键活动，该活动延期将导致整个工程延期。如果最大可利用时间大于***Weight(<T_i, T_j>)***，则该活动不是关键活动，其持续时间只要不超过最大可利用时间，就不会影响整个工程的进度。

定理6.3（新）的逆定理不成立

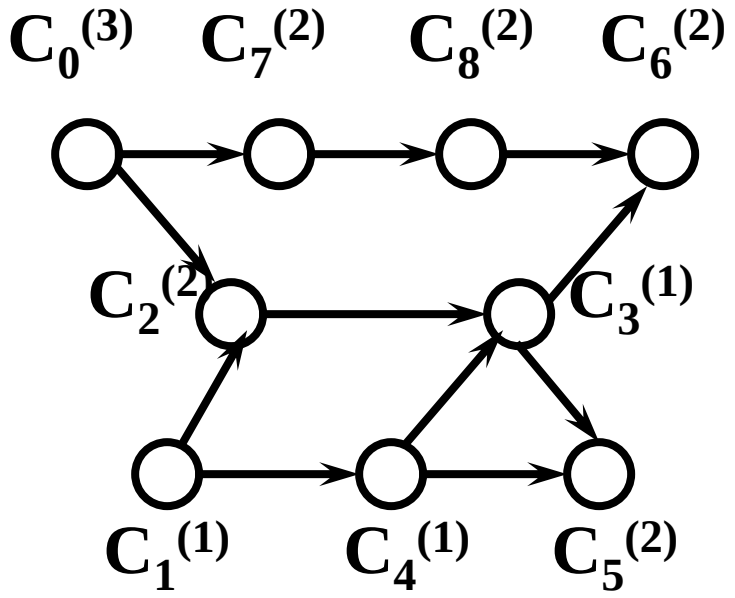


- 一条路径上的所有顶点都满足ve(i)= vl(i)，则为关键路径；（错误）

- 因为，边<Ti, Tj>两端点都满足该等式，不能保证是边<Ti, Tj>关键活动。

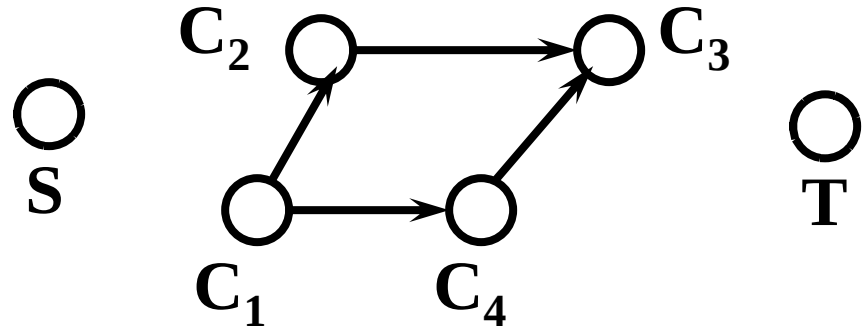
例：选课（续）

课程代号	课程名称	先修课程	学期数
C_0	高等数学	无	3
C_1	程序设计基础	无	1
C_2	离散数学	C_0,C_1	2
C_3	数据结构	C_2,C_4	1
C_4	程序设计语言	C_1	1
C_5	编译技术	C_3,C_4	2
C_6	操作系统	C_3,C_8	2
C_7	普通物理	C_0	2
C_8	计算机原理	C_7	2



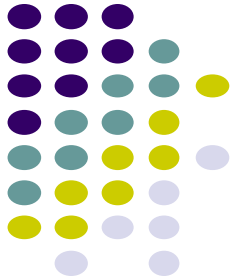
点权图关键路径处理方法

- 引入虚源虚汇（常用技巧）
  - ✓ 适用多源多汇



- 点权处理处理方法
  - ✓ 按点计算路径
  - ✓ 点权转换为边权
    - 每个点v引入镜像点v’，点权放到新增边<v,v’>上；原来的边<u,v>转换为<u’,v>。
    - 点权向后推到边上

关键路径小结



- 关键路径的求解方式是DAG上的递推。

- 思考
  - ✓ 如果有多个源点或多个汇点如何处理？
  - ✓ 如何求出一条关键路径（如序号排列最小）？
  - ✓ 如何计算关键路径的条数？