第八章 图

张史梁
slzhang.jdl@pku.edu.cn

内容提要

- 图的基本概念
- 存储表示
- 图的基本运算与周游
- 最小生成树
- 拓扑排序
- 关键路径
- 最短路径

AOV网

- 如果在有向图中，用
- 顶点表示活动，
- 有向边表示活动的先后关系
- 则这样的有向图称为顶点活动网（Activity On Vertex network，简称AOV网）

  - AOV网中的边表示活动之间存在的制约关系；
  - 顶点所表示的事件实际上就是它的入边所表示的活动都已完成，他的出边表示的活动可以开始这样一种状态。

拓扑排序

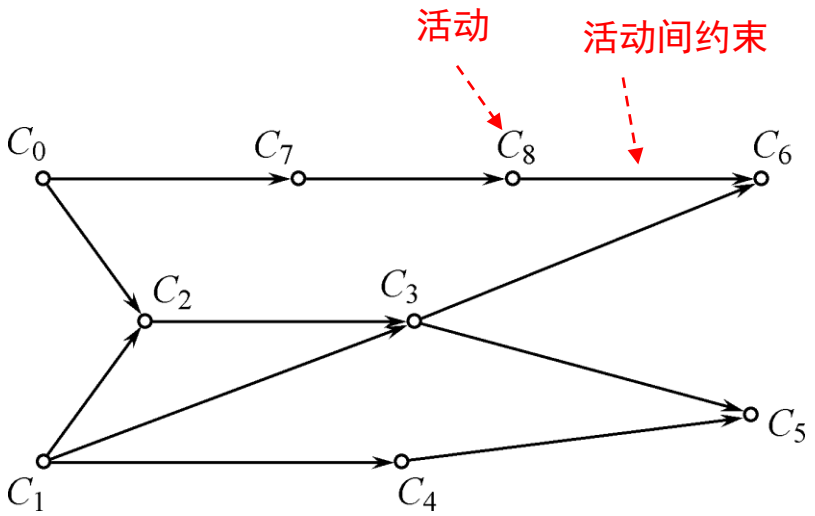
- 对于一个AOV网，
- 如果图中所有顶点（事件）可以排成一个线性序列v_{i1}，v_{i2}，…，v_{in}，并且该线性序列具有以下性质：

  - 如果在AOV网中，从顶点v_i到顶点v_j存在一条路径，则在线性序列中，顶点v_i一定排在顶点v_j之前。
  - 就把这种序列称为拓扑序列，把构造拓扑序列的操作称为拓扑排序。

拓扑排序 – 例子

- 用顶点表示课程，有向边表示课程之间的优先关系，如果课程C_i是课程C_j的先修课，则在AOV网中必定存在一条有向边$<C_i, C_j>$。表中各课程的AOV网如上图所示

课程代码	课程名称	先修课程
C_0	高等数学	
C_1	程序设计语言	
C_2	离散数学	C_0 C_1
C_3	数据结构	C_1 C_2
C_4	算法	C_1
C_5	编译原理	C_3 C_4
C_6	操作系统	C_3 C_8
C_7	普通物理	C_0
C_8	计算机原理	C_7



拓扑排序的结果不唯一，满足局部限制即可

得到一个拓扑序列：C0, C1, C2, C4, C3, C5, C7, C8, C6

拓扑排序的应用

- 在实际中有许多的应用：
  - 课程间的先修关系
  - 术语表中各技术术语定义的依赖关系
  - 课程或书中各主题间的组织
  - 工程的施工图
  - 产品的生产流程图
  - ….

拓扑排序的基本思想

（1）从AOV网中选择一个入度为0的顶点将其输出。

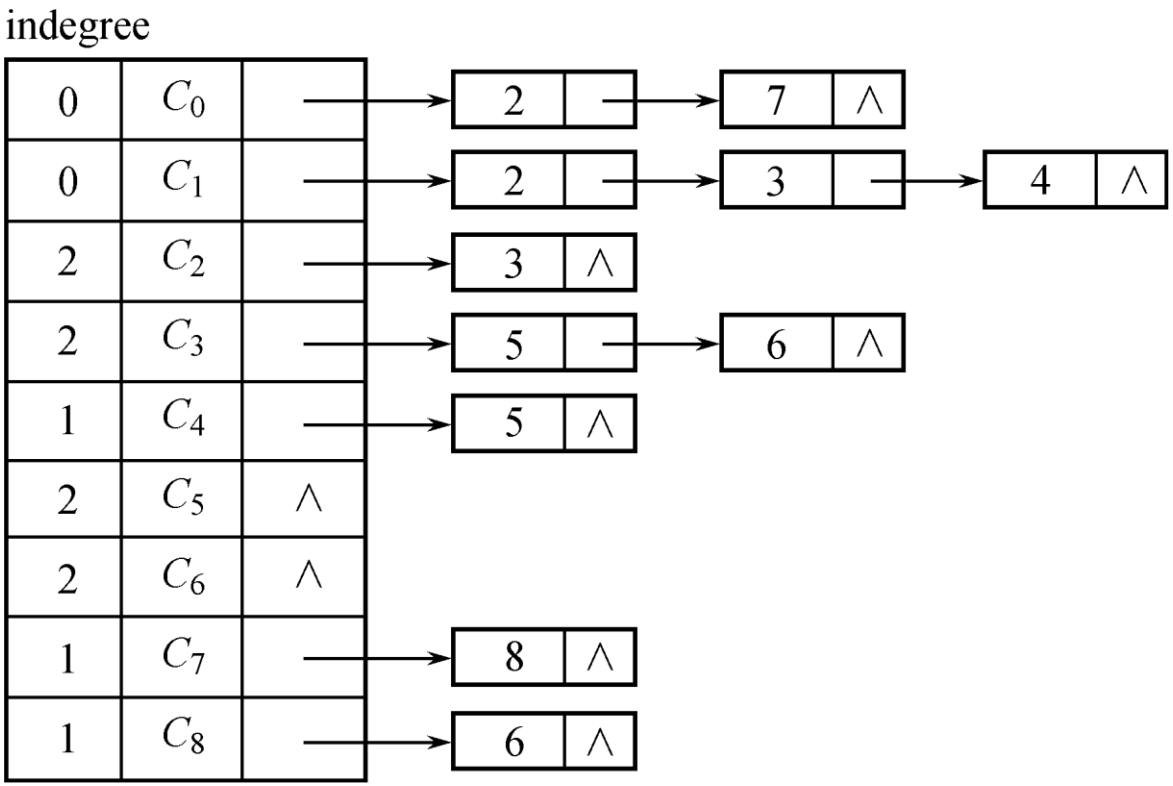
（2）在AOV网中删除此顶点及其所有的出边。

反复执行（1）（2）两步，直到

- 所有顶点都已经输出为止，此时整个拓扑排序完成；
- 或者，剩下的顶点的入度都不为0，此时说明AOV网中存在回路（所有顶点入度不为0），拓扑排序无法再进行。

拓扑排序算法 – 例子

- 图采用邻接出边表表示，增加一个indegree数组，存放各顶点的入度



拓扑排序算法

- int topoSort(GraphList * paov, int * ptopo)
  - 拓扑排序前，先调用findInDegree得到所有结点的入度，然后将所有入度为0的顶点压栈。
  - 从栈顶取出一个顶点V，由它的出边表可以得到以该顶点为起点的出边，将这些边终点的入度减1，即”删除”这些边。
  - 如果某条边终点的入度为0，则将该顶点入栈。
  - 反复进行上述操作，直到栈为空。

- 如果这时输出的顶点个数小于n，则说明该AOV网中存在回路，否则，拓扑排序正常结束。

回顾图的邻接表表示法

```
1.  struct EdgeNode {
2.          int endvex;                          /* 相邻顶点在顶点表中下标 */
3.          AdjType weight;              /* 边的权，非带权图应该省略 */
4.           struct EdgeNode * nextedge;       /* 链字段 */
5.  };                                                    /* 边表中的结点 */
6.  typedef struct {
7.          VexType vertex;              /* 顶点信息 */
8.           struct EdgeNode * edgelist;                  /* 边表头指针 */
9.  } VexNode;

10. typedef struct {
11.         int n;                                   /* 图的顶点个数 */
12.         VexNode *vexs;               /*顶点表 */
13. }GraphList;                                      /* 图的邻接表表示 */
```

拓扑排序算法

```
1.  void findInDegree(GraphList* g, int *indegree)
2.  { /* 求出图中所有顶点的入度 */
3.  int i;
4.  PEdgeNode          p;
5.  for(i=0; i<g->n; i++)          indegree[i] = 0;
6.  for(i=0; i<g->n; i++) {
7.      p = g->vexs[i].edgelist;
8.      while(p) {                          //找到一条以i为起点的边
9.              ++indegree[p->endvex];   //终点的顶点入度+1
10.             p = p->nextedge;  //扫描以i为起点的下一条边
11.             }
12.     }
13. }
```

拓扑排序算法

```
1.  int topoSort(GraphList * paov, int * ptopo)
2.  { /* 拓扑排序 */
3.      EdgeList p; //结果存放在ptopo中
4.      int i, j, k, nodeno=0, top=-1;
5.      int *indegree=(int*)malloc(sizeof(int)*paov->n);
6.      findInDegree(paov, indegree); /*给indegree数组置初值 */
7.
8.      for(i=0; i<paov->n; i++)
9.          if(indegree[i]==0)
10.             { indegree[i]=top; top=i; }  /* 将入度为零的顶点入栈 */
                                    //indegree[top]一定是栈顶元素
                                    //indegree[indegree[top]]是次新进栈的
                                    //栈底元素值一定为-1
```

拓扑排序算法

```
11.  while(top!=-1) {                  /* 栈不为空 */  //还没取到栈底元素值
12.    j=top;  top=indegree[top];   /* 取出当前栈顶元素并退栈 */
13.    ptopo[nodeno++]=j;             /* 将该顶点输出到拓扑序列中 */
14.    p=paov->vexs[j].edgelist;    /* 取边表中的第一个边结点 */
15.   while(p) {                              //找到一条以j为起点的边
16.        k=p->endvex;                       //此边的终点节点为k
17.          indegree[k]--;  /* 删除以该顶点为起点的边 */
18.       if ( indegree[k]==0)
19.            { indegree[k]=top; top=k; }     /* 将新的入度为零的边入栈*/
20.        p=p->nextedge;
21.     }
22.  }
```

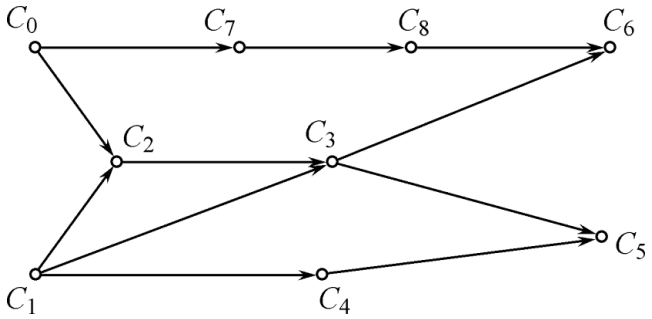
拓扑排序算法

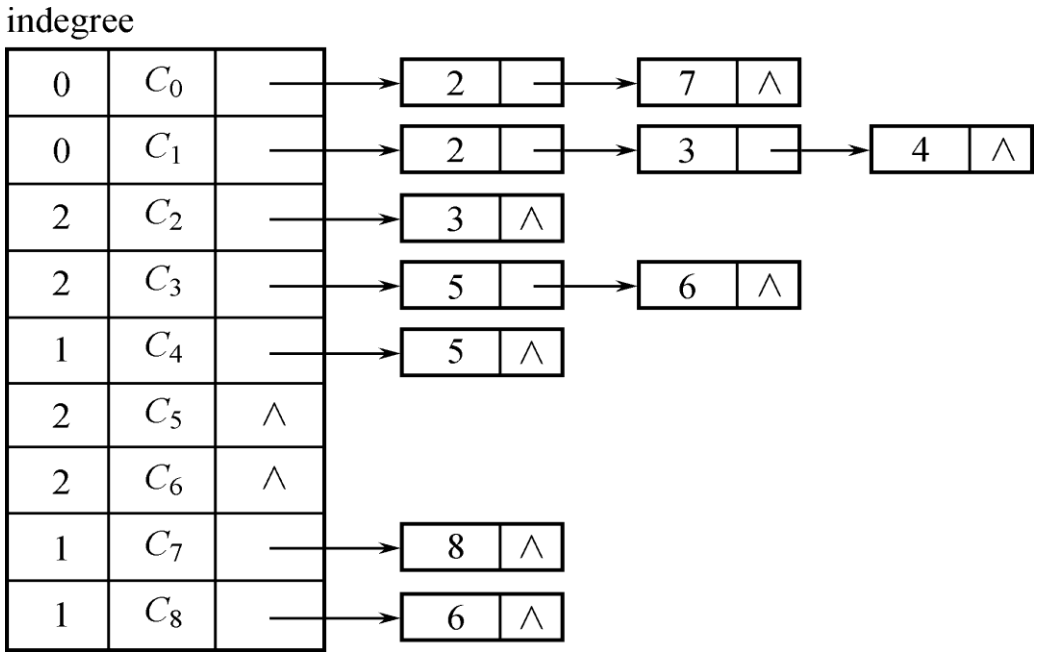
```
23.   free(indegree);
24.   if( nodeno<paov->n)  {        /* AOV网中存在回路 */
25.       printf("The aov network has a cycle\n");
26.       return(0);
27.    }
28. return(1);
29. }
```

如果这时输出的顶点个数小于n，则说明该AOV网中存在回路，否则，拓扑排序正常结束。

例子

- 图采用邻接出边表表示，增加一个indegree数组，存放各顶点的入度





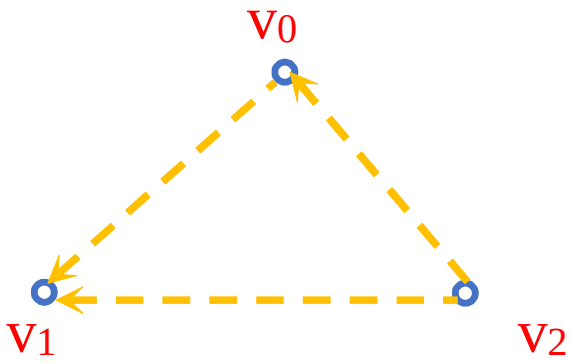
- 拓扑序列为：C1, C4, C0, C7, C8, C2, C3, C6, C5

算法分析

- 设AOV网有n个顶点，e条边，
  - 算法最初首先检查入度为零的顶点，并将这些顶点压栈，花费的时间为O(n+e)。
  - 下面进行拓扑排序时，每个顶点都入栈一次，且每个顶点边表中的边结点都被检查一遍，运行时间为O(n+e)。
  - 拓扑排序算法的时间复杂度为O(n+e)。

注意

- 一个AOV网的拓扑序列不一定是唯一的。
- AOV网某一拓扑序列就是整个工程得以顺利完成的一种可行方案。
- AOV网中一定不能出现回路
  - 设置一个链栈存储入度为0的顶点
  - 栈为空&输出的顶点个数小于n =>该AOV网中存在回路（鸡生蛋-蛋生鸡）



奖金(reward)分配问题

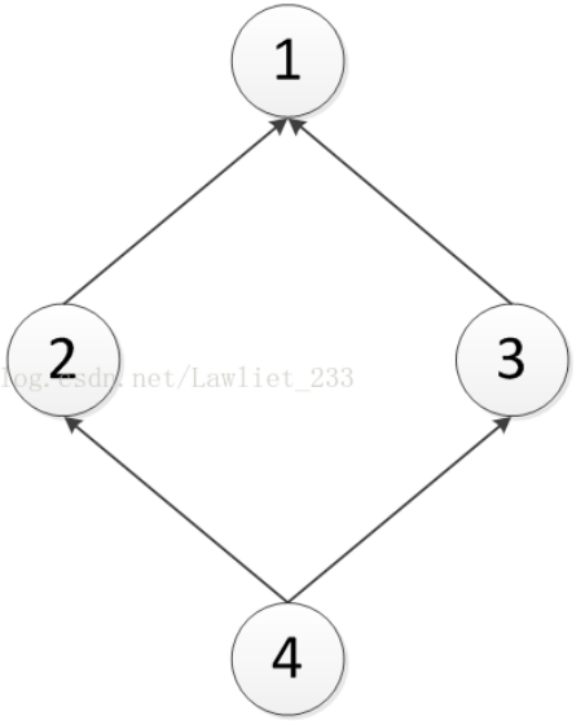
- 由于Yali Company总经理 Mr. Z 决定给n位员工发奖金。公司决定以每个人本年在公司的贡献为标准来计算他们得到奖金的多少。

- 于是Mr. Z下令召开m方会谈。每位参加会谈的代表提出了自己的意见：我认为员工a的奖金应该比b高！

- HR决定要找出一种奖金方案，满足各位代表的意见，且同时使得总奖金数最少。每位员工奖金最少为100元。

思路

- 构建具有n个顶点，m条边的有向图
- 之后进行拓扑排序
  - 首先统计入度为0顶点，这些顶点的奖金就是100元
  - 删除这些顶点相邻边后，入度变为0的顶点的奖金就是101元。
  - 依次类推，直到所有顶点入度都为0
  - 在排序过程中，如果发现成环，无法进行拓扑排序，则退出，无法分配奖金。

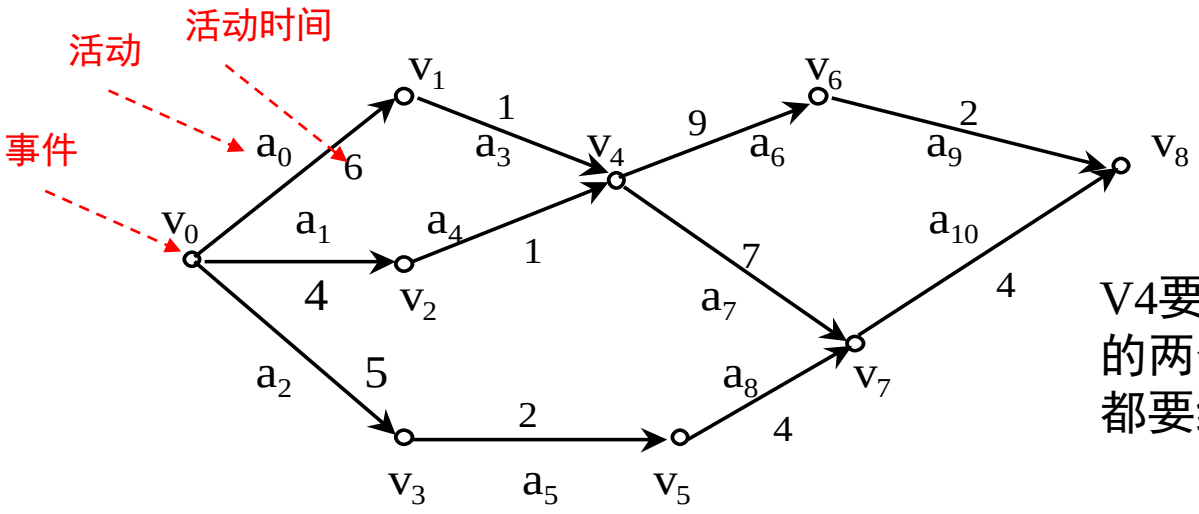


内容提要

- 图的基本概念
- 存储表示
- 图的基本运算与周游
- 最小生成树
- 拓扑排序
- 关键路径
- 最短路径

AOE网

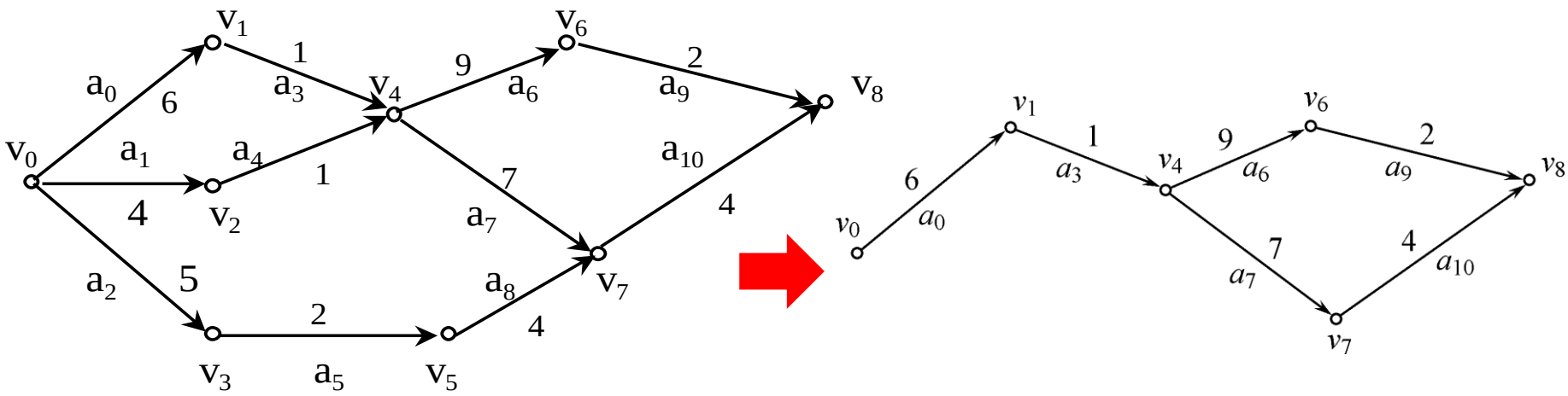
- 如果在带权的有向图中，用顶点表示事件，用有向边表示活动，边上的权值表示活动持续的时间，则此带权的有向图称为边活动网（ Activity On Edge network ， 简称AOE网）。
- 顶点所表示的事件实际上就是它的入边所表示的活动都已完成，它的出边所表示的活动可以开始这样一种状态。



V4要发生，则他对应的两个前期活动a3 a4都要结束才可以

关键路径

- AOE网中有些活动可以并行进行，所以完成整个工程的最短时间是从开始顶点到完成顶点的最长路径长度，路径长度为路径上各边的权值之和。
- 把开始顶点到完成顶点的最长路径称为关键路径。
  - 例如，路径v0, v1, v4, v7, v8是一条关键路径，长度为18，也就是整个工程至少要18天才能完成。
  - AOE网的关键路径可能不止一条



关键活动

- 哪些活动将影响工程的进度呢？
  - 关键路径上的活动如果延期，都将导致整项工程不能按时完成
  - 要提前完成工程，需要在某些关键活动上投入更多的人力物力

- 关键路径上的活动为关键活动
- 分析关键路径的目的是确定关键活动

几个概念

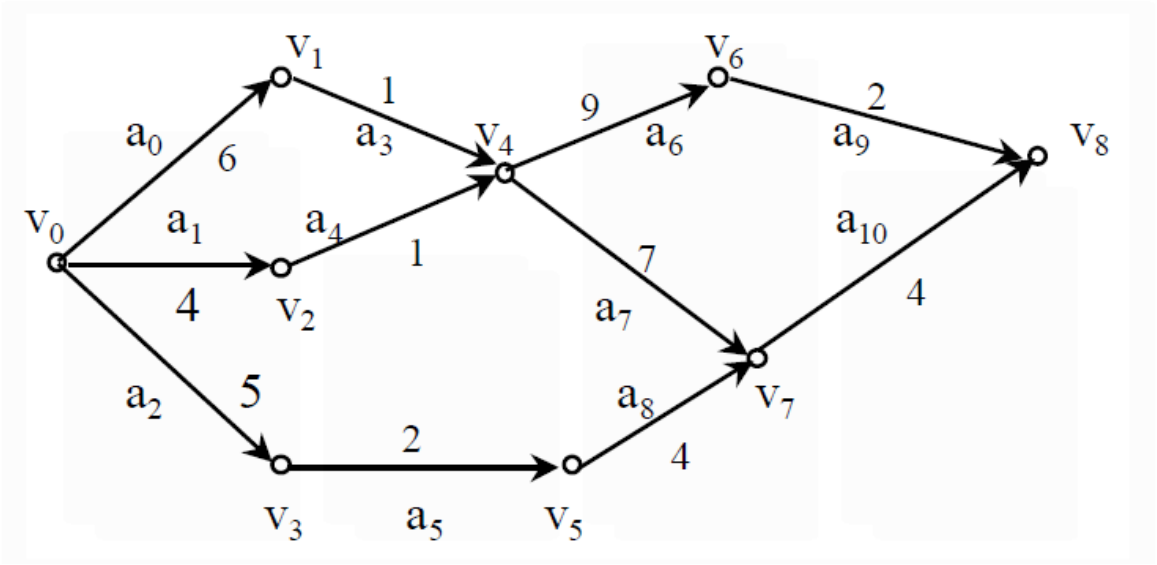
- 关键活动
  - 事件v_j可能的最早发生时间ee(j)
  - 事件v_i允许的最迟发生时间le(i)
  - 活动a_k=< v_i,v_j>的最早开始时间e(k)
  - 活动a_k=< v_i,v_j>的最晚开始时间l(k)

- e(k)=l(k)的那些活动均为关键活动

- l(k)-e(k)表示完成活动a_k的时间余量，是在不延误工期的前提下，活动a_k可以延迟的时间。

关键路径的求解示例

- 按上述公式分别求出事件的最早发生时间和最迟发生时间、活动的最早开始时间和最晚开始时间



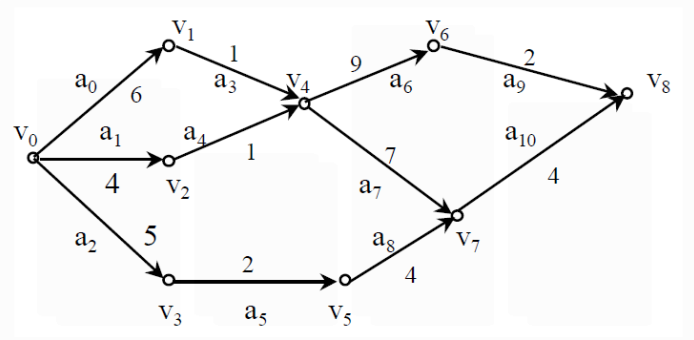
某事件可能的最早发生时间

- 从开始顶点到顶点v_j（事件v_j）的最长路径长度
  - 所有进入v_j的活动$< v_i, v_j>$都结束时，v_j所代表的事件才可立即发生（所以是最早发生事件），即

$$ee(0) = 0$$

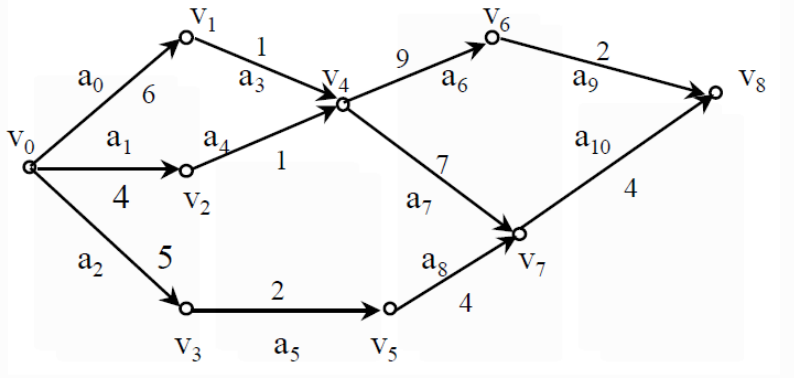
$$ee(j) = \max\{ee(i) + weight(< v_i, v_j >)\}$$

$$< v_i, v_j > \in T, 1 \le j \le n-1$$



其中T为所有以v_j为终点的入边的集合

关键路径的求解示例



ee(0)=0

ee(1)=ee(0)+weight(<v_0,v_1>)=0+6=6

ee(2)=ee(0)+ weight(<v_0,v_2>)=0+4=4

ee(3)=ee(0)+ weight(<v_0,v_3>)=0+5=5

ee(4)=max{ee(1)+ weight(<v_1,v_4>), ee(2)+ weight(<v_2,v_4>)

=max{6+1, 4+1}=7

ee(5)=ee(3)+ weight(<v_3,v_5>)=5+2=7

ee(6)=ee(4)+ weight(<v_4,v_6>)=7+9=16

ee(7)=max{ee(4)+ weight(<v_4,v_7>), ee(5)+ weight(<v_5,v_7>)

=max{7+7, 7+4}=14

ee(8)=max{ee(6)+ weight(<v_6,v_8>), ee(7)+ weight(<v_7,v_8>)

=max{16+2, 14+4}=18

某事件允许的最晚发生时间

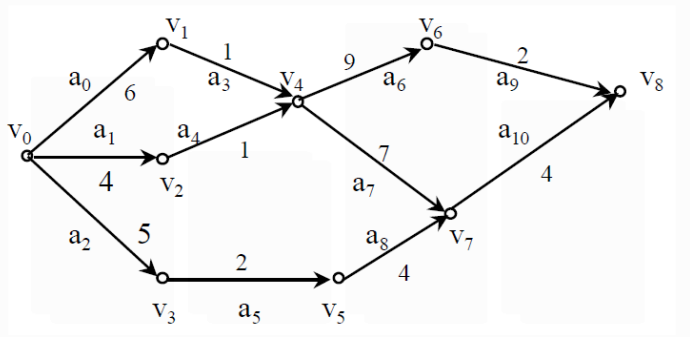
- 在不推延整个工期的前提下，事件v_i允许的最晚发生时间
  - 不得迟于其后继事件v_j的最晚发生时间减去活动$< v_i, v_j >$的持续时间（不延误后续的活动前提下的最晚开始时间）

不能延误工期，所以最后一个事件的最晚发生时间==最早发生时间

$$le(n-1) = ee(n-1)$$

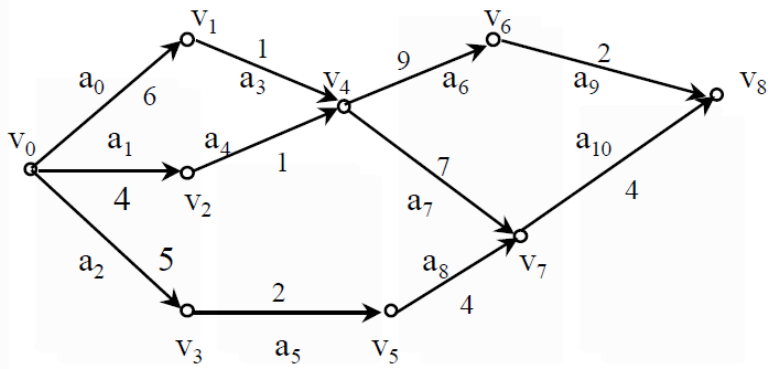
$$le(i) = \min\{le(j) - weight(< v_i, v_j >)\}$$

$$< v_i, v_j > \in S, 0 \le i \le n-2$$



其中S为所有以v_i为始点的出边的集合

关键路径的求解示例



le(8)=ee(8)=18

le(7)=ee(8)－weight(<v_7,v_8>)=18－4=14

le(6)=ee(8)－weight(<v_6,v_8>)=18－2=16

le(5)=ee(7)－weight(<v_5,v_7>)=14－4=10

le(4)=min{le(7)—weight(<v_4,v_7>), le(6)—weight(<v_4,v_6>)}

=min{14－7, 16－9 }=7

le(3)=le(5)－weight(<v_3,v_5>)=10－2=8

le(2)=le(4)－weight(<v_2,v_4>)=7－1=6

le(1)=le(4)－weight(<v_1,v_4>)=7－1=6

le(0)=min{le(1)—weight(<v_0,v_1>), le(2)—weight(<v_0,v_2>),

le(3)—weight(<v_0,v_3>)}

=min{6－6, 6－4, 8－5 }=0

某活动的最早/晚开始事件

- 活动$a_k = < v_i, v_j >$的最早开始时间e(k)
  - 只有事件v_i发生了，活动a_k才能开始，也即它的最早开始时间为事件v_i的最早发生时间

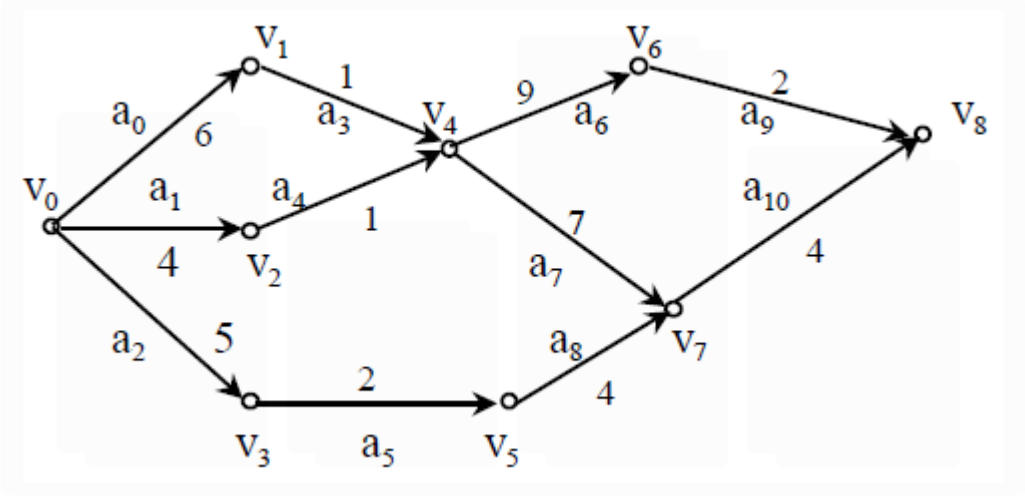
$$e(k) = ee(i)$$

- 活动$a_k = < v_i, v_j >$的最晚开始时间l(k)
  - 活动a_k的最晚开始时间为事件v_j的最晚完成时间减去a_k的持续时间

$$l(k) = le(j) - weight(< v_i, v_j >)$$

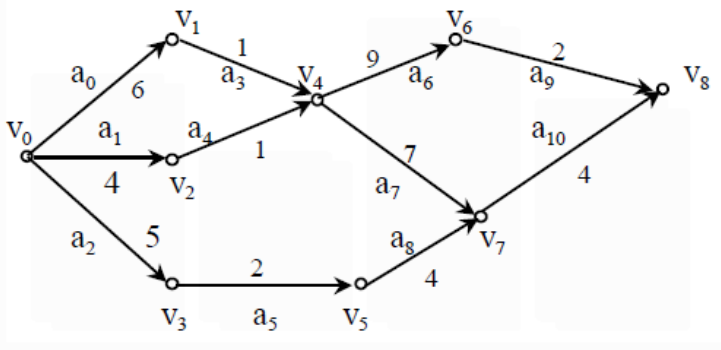
关键路径求解示例

e(0)=ee(0)=0
e(1)=ee(0)=0
e(2)=ee(0)=0
e(3)=ee(1)=6
e(4)=ee(2)=4
e(5)=ee(3)=5
e(6)=ee(4)=7
e(7)=ee(4)=7
e(8)=ee(5)=7
e(9)=ee(6)=16
e(10)=ee(7)=14



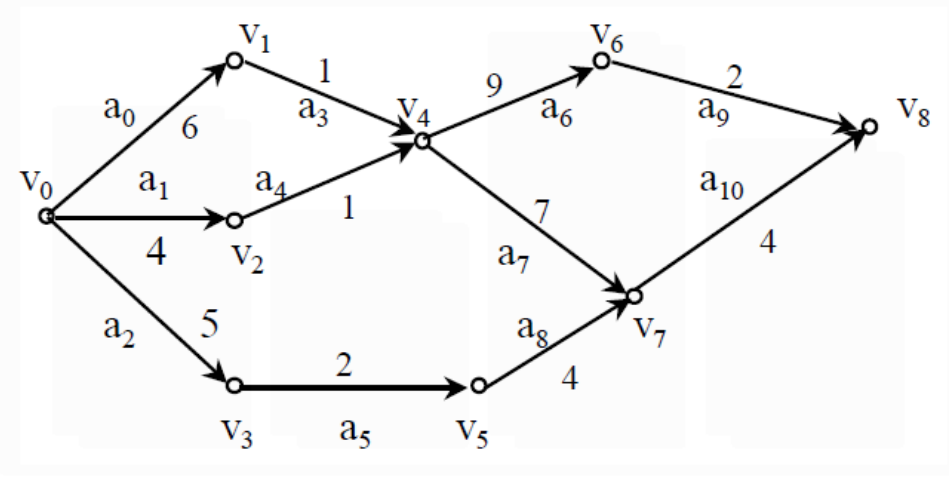
关键路径求解示例

$l(0)=le(1)-weight(<v_0,v_1>)=6-6=0$
$l(1)=le(2)-weight(<v_0,v_2>)=6-4=2$
$l(2)=le(3)-weight(<v_0,v_3>)=8-5=3$
$l(3)=le(4)-weight(<v_1,v_4>)=7-1=6$
$l(4)=le(4)-weight(<v_2,v_4>)=7-1=6$
$l(5)=le(5)-weight(<v_3,v_5>)=10-2=8$
$l(6)=le(6)-weight(<v_4,v_6>)=16-9=7$
$l(7)=le(7)-weight(<v_4,v_7>)=14-7=7$
$l(8)=le(7)-weight(<v_5,v_7>)=14-4=10$
$l(9)=le(8)-weight(<v_6,v_8>)=18-2=16$
$l(10)=le(8)-weight(<v_7,v_8>)=18-4=14$



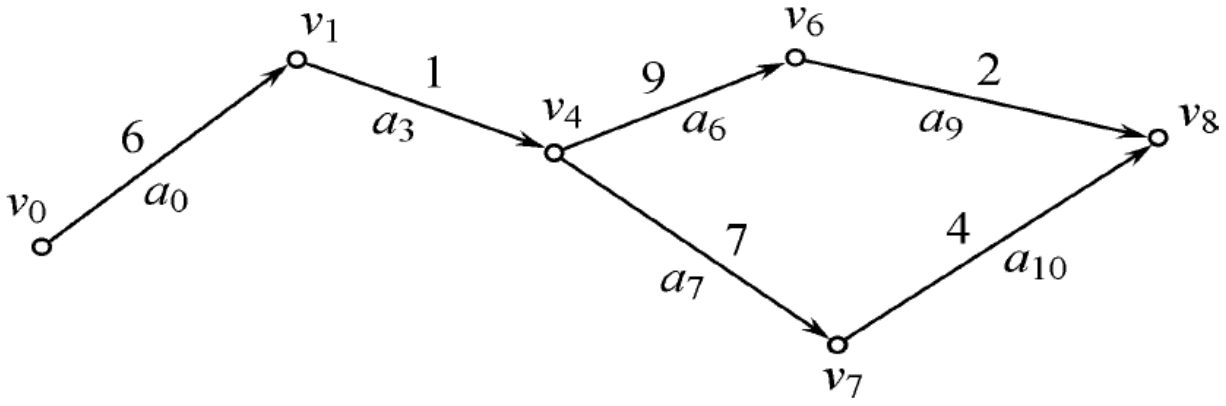
关键路径求解示例

l(0)－e(0)=0－0=0
l(1)－e(1)=2－0=2
l(2)－e(2)=3－0=3
l(3)－e(3)=6－6=0
l(4)－e(4)=6－4=2
l(5)－e(5)=8－5=3
l(6)－e(6)=7－7=0
l(7)－e(7)=7－7=0
l(8)－e(8)=10－7=3
l(9)－e(9)=16－16=0
l(10)－e(10)=14－14=0



关键路径求解示例

- 结果：活动a_0, a_3, a_6, a_7, a_9, a_{10}为关键活动



关键路径算法

- 在没有环存在的情况下，按照上面的思路，逐步计算，并找出e(k)=l(k)的关键活动
  - 计算ee(j)必须在顶点v_j所有前驱顶点的最早发生时间都已经求出的前提下进行；
  - 计算le(i)必须在顶点v_i所有后继顶点的最迟发生时间都已经求出的前提下进行；
  - 故顶点序列须是一个拓扑序列

```
int criticalPath(GraphList * paoe)
```

```
int criticalPath(GraphList * paoe) {
    int i, j, k; EdgeList p; AdjType l, e; AdjType ee[VN], le[VN]; int topo[VN];
    if(topoSort(paoe, topo)==FALSE)  return(FALSE);   /*图中有环*/
    for(i=0; i<paoe->n; i++)  ee[i]=0;   /*求事件可能的最早发生时间ee*/
    for(k=0; k<paoe->n; k++)   {
        i=topo[k];  p=paoe->vexs[i].edgelist;
        while (p!=NULL) {
            j=p->endvex;
            if(ee[i]+p->weight>ee[j])  ee[j]=ee[i]+p->weight;
            p=p->nextedge;
        } }
    for(i=0;i<paoe->n; i++)  le[i]=ee[topo[paoe->n-1]]; /*求事件允许的最迟发生时间le*/
    for(k=paoe->n-2; k>=0; k--) {
        i=topo[k];  p=paoe->vexs[i].edgelist;
        while(p!=NULL){
            j=p->endvex;
            if((le[j]-p->weight)<le[i])  le[i]=le[j]-p->weight;
            p=p->nextedge;
        } }
    for(i=0; i<paoe->n; i++){          /*求每个活动a的最早开始时间e及最晚开始时间l */
        p=paoe->vexs[i].edgelist;
        while(p!=NULL) {
            j=p->endvex;  e=ee[i];  l=le[j]-p->weight;
            if(e==l)  printf("<v%2d,v%2d>, ",i,j);   /*输出关键活动*/
            p=p->nextedge;
        } }
   printf("\n"); return(TRUE);
}
```

复杂度分析

- 在n个顶点，e条边的AOE网中
- 求事件可能的最早发生时间及允许的最迟发生时间，以及活动的最早开始时间和最晚开始时间时，都要对图中所有顶点及每个顶点边表中所有的边结点进行检查，时间花费为O(n+e)
- 故关键路径算法的时间复杂度为O(n+e)

AOV网

- 如果在有向图中，用
- 顶点表示活动，
- 有向边表示活动的先后关系
- 则这样的有向图称为顶点活动网（Activity On Vertex network，简称AOV网）
  - AOV网中的边表示活动之间存在的制约关系；
  - 顶点所表示的事件实际上就是它的入边所表示的活动都已完成，他的出边表示的活动可以开始这样一种状态。

拓扑排序的基本思想

（1）从AOV网中选择一个入度为0的顶点将其输出。

（2）在AOV网中删除此顶点及其所有的出边。

反复执行（1）（2）两步，直到

- 所有顶点都已经输出为止，此时整个拓扑排序完成；
- 或者，剩下的顶点的入度都不为0，此时说明AOV网中存在回路（所有顶点入度不为0），拓扑排序无法再进行。

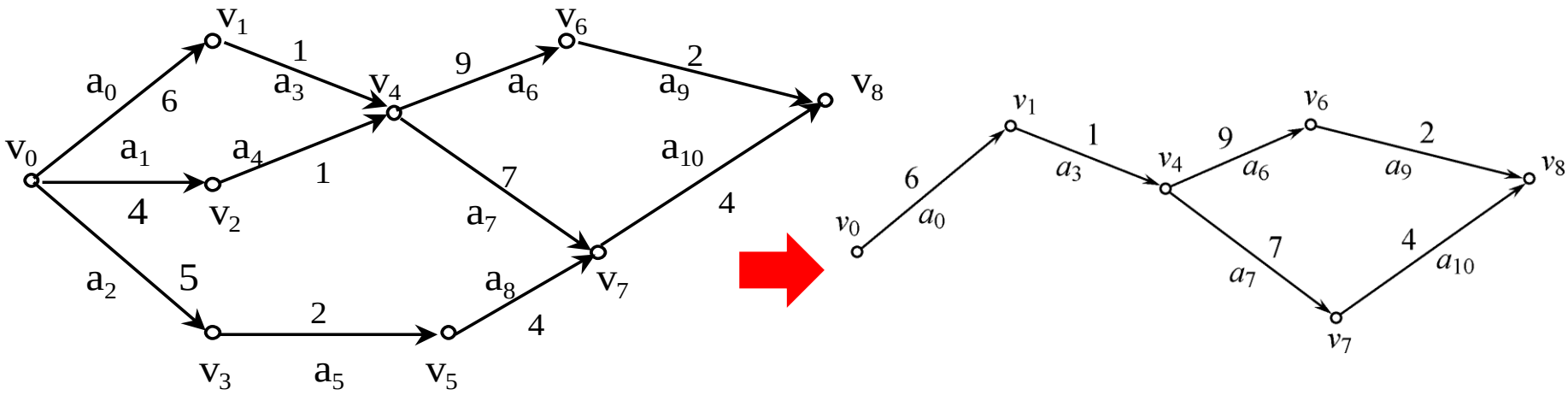
拓扑排序算法

- int topoSort(GraphList * paov, int * ptopo)
  - 拓扑排序前，先调用findInDegree得到所有结点的入度，然后将所有入度为0的顶点压栈。
  - 从栈顶取出一个顶点V，由它的出边表可以得到以该顶点为起点的出边，将这些边终点的入度减1，即”删除”这些边。
  - 如果某条边终点的入度为0，则将该顶点入栈。
  - 反复进行上述操作，直到栈为空。

- 如果这时输出的顶点个数小于n，则说明该AOV网中存在回路，否则，拓扑排序正常结束。

关键路径

- AOE网中有些活动可以并行进行，所以完成整个工程的最短时间是从开始顶点到完成顶点的最长路径长度，路径长度为路径上各边的权值之和。
- 把开始顶点到完成顶点的最长路径称为关键路径。
  - 例如，路径v0, v1, v4, v7, v8是一条关键路径，长度为18，也就是整个工程至少要18天才能完成。
  - AOE网的关键路径可能不止一条



几个概念

- 关键活动
  - 事件v_j可能的最早发生时间ee(j)
  - 事件v_i允许的最迟发生时间le(i)
  - 活动a_k=< v_i,v_j>的最早开始时间e(k)
  - 活动a_k=< v_i,v_j>的最晚开始时间l(k)

$$ee(0) = 0$$

$$ee(j) = \max\{ee(i) + weight(< v_i, v_j >)\}$$

$$< v_i, v_j > \in T, 1 \le j \le n-1$$

$$le(n-1) = ee(n-1)$$

$$le(i) = \min\{le(j) - weight(< v_i, v_j >)\}$$

$$< v_i, v_j > \in S, 0 \le i \le n-2$$

$$e(k) = ee(i) \qquad l(k) = le(j) - weight(< v_i, v_j >)$$

- e(k)=l(k)的那些活动均为关键活动

- l(k)-e(k)表示完成活动a_k的时间余量，是在不延误工期的前提下，活动a_k可以延迟的时间。