数据结构

深圳技术大学
大数据与互联网学院

第七章 图

7.1 图的定义和术语
7.2 图的存储结构
7.3 图的遍历
7.4 图的连通
7.5 有向无环图及其应用
7.6 最短路径

上节复习

- 最短路径:求两个顶点之间距离最短的路径
- 迪杰斯特拉(Dijkstra)算法
  - 求v0到其他顶点的所有最短路径
  - 算法流程

1、初始化，将v0加入已搜索顶点集合

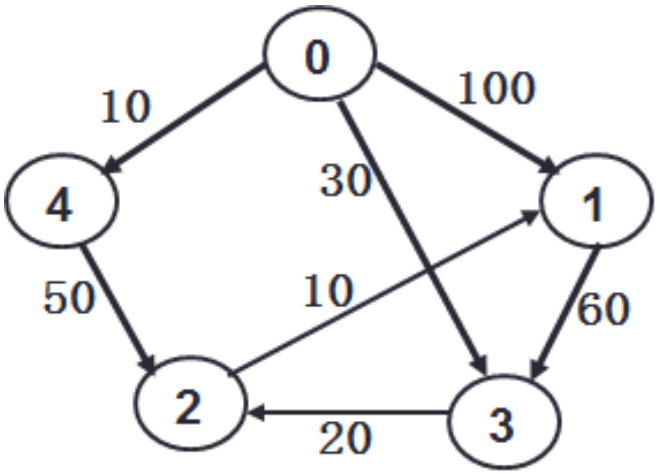
2、找出离v0最近的顶点v，加入集合

3、比较v0到其他顶点、v0到v再到其他顶点的路径，从而更新其他顶点的最短路径

4、重复步骤2

练习

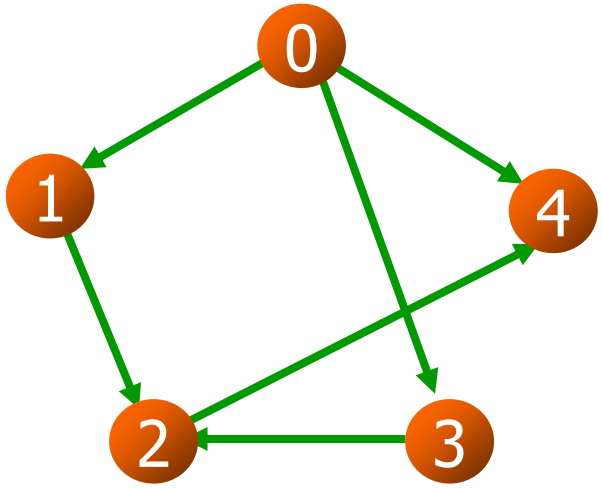
- 已知带权有向图如下，请求出顶点0到其他顶点的最短路径和长度，要求使用迪杰斯特拉算法写出求解过程

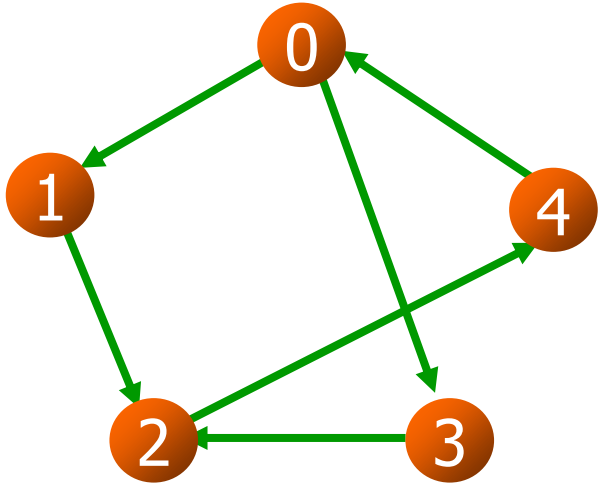


7.5 有向无环图及其应用

一. 有向无环图

- 有向无环图(DAG:Directed Acycline Graph)是图中无环的有向图

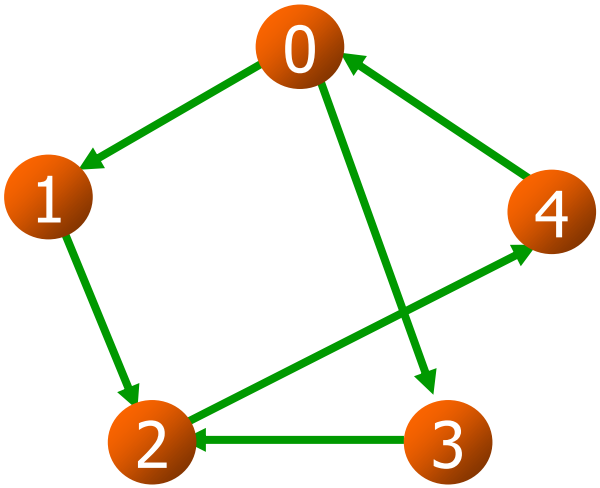




7.5 有向无环图及其应用

一. 有向无环图

- 有向图中，可以用深度优先搜索(DFS)，找出是否存在环：从某个顶点v出发，进行DFS，如果存在一条从顶点u到v的回边，则有向图中存在环
  - 下图DFS：0, 1, 2, 4, 3

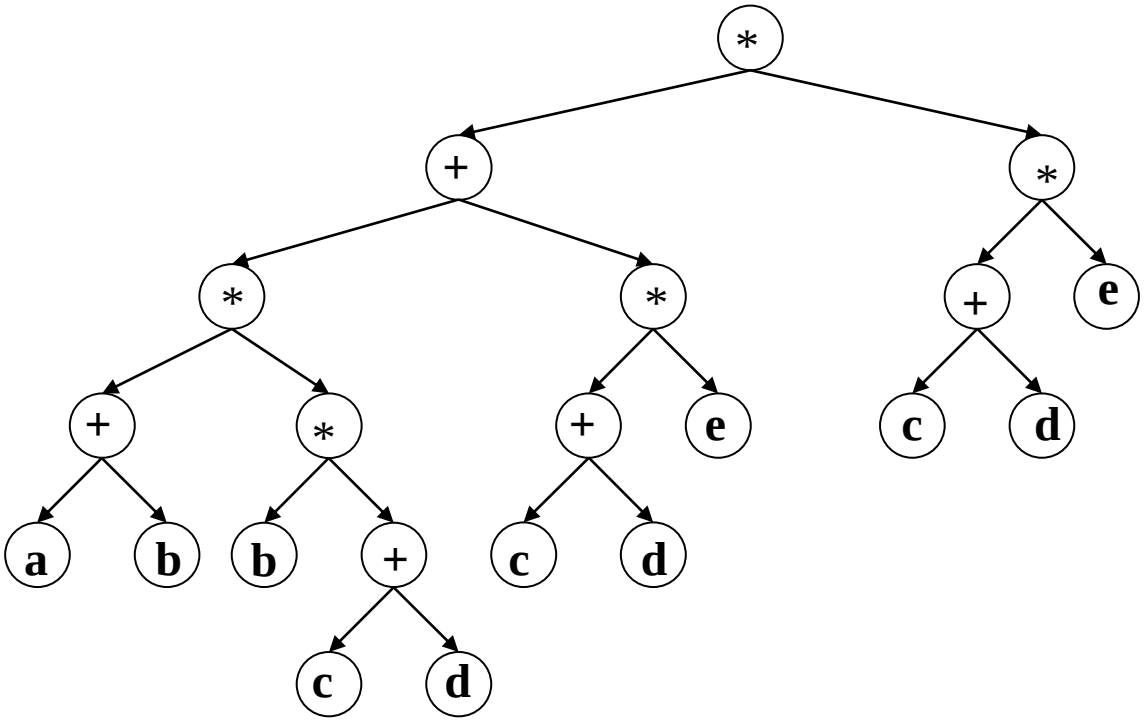


非DAG

7.5 有向无环图及其应用

一. 有向无环图

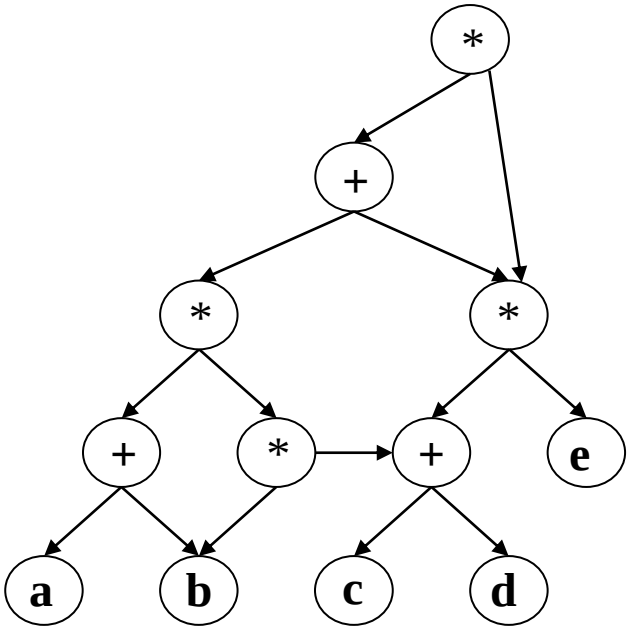
- DAG图用于表达式的共享
  - 表达式((a +b) * (b * (c + d)) + (c + d) * e) * ((c + d) * e)，用二叉树表示



7.5 有向无环图及其应用

一. 有向无环图

- DAG图用于表达式的共享
  - 表达式((a +b) * (b * (c + d)) + (c + d) * e) * ((c + d) * e)，DAG图表示.
  - 可以节省存储空间



7.5 有向无环图及其应用

一. 有向无环图

- 有向无环是一类具有代表性的图，主要用于研究工程项目的工序问题、工程时间进度问题等。
- 一个工程(project)都可分为若干个称为活动(active)的子工程(或工序)，各个子工程受到一定的条件约束：某个子工程必须开始于另一个子工程完成之后；整个工程有一个开始点(起点)和一个终点。
- 一个工程活动可以用有向无环图来描述

7.5 有向无环图及其应用

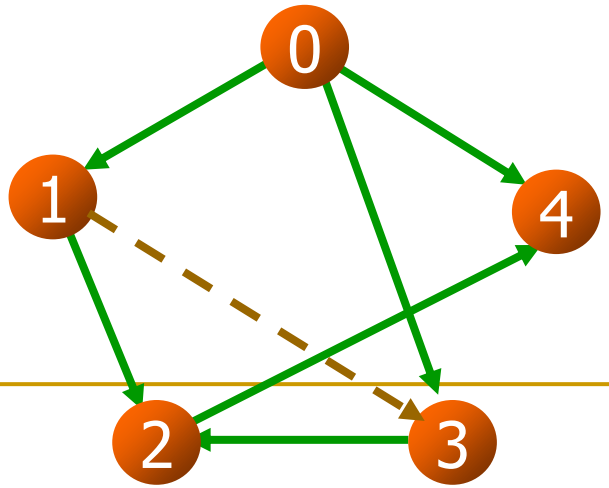
二. 拓扑排序

- 偏序关系
- 若集合X上的关系R是：
  - 自反的：x R x
  - 反对称的：x R y and y R x => x=y
  - 传递的：xRy & yRz => xRz
  - 则称R是集合X上的偏序关系，例如小于等于关系
  - 若P是A上的一个偏序关系，我们用a≤b来表示（a,b）∈P

7.5 有向无环图及其应用

二. 拓扑排序

- 设关系R是集合X上的偏序，如果对每个x,y∈X，必有xRy或者yRx，则称R是X上的全序关系
- 偏序指集合中仅有部分成员之间可比较
- 全序指集合中全体成员之间均可比较
  - 下图是一个偏序关系，因为1,3没有先后关系
  - 如果人为地增加1,3先后关系，如1先于3，则右图变为全序，称为拓扑有序
  - 从图来说，全序表示任意两点之间至少存在一条路径



7.5 有向无环图及其应用

二. 拓扑排序

- 从偏序得到的全序称为拓扑有序
- 由偏序得到拓扑有序的操作称为拓扑排序
- 拓扑排序算法：

  (1) 在有向图中选一个没有前驱的顶点且输出之

  (2) 从图中删除该顶点和所有以它为尾的弧

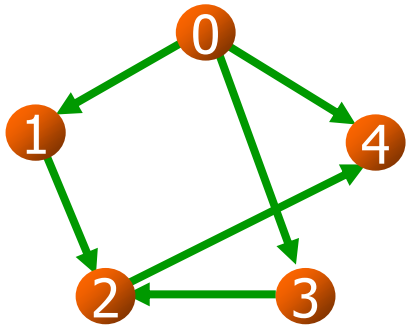
  重复(1)(2)两步，直到所有顶点输出为止

- 关系：通过拓扑排序操作得到拓扑有序序列

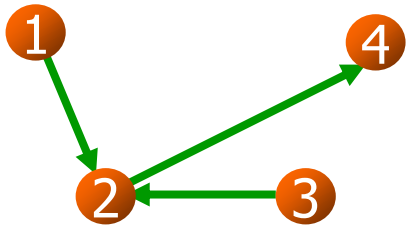
7.5 有向无环图及其应用

二. 拓扑排序

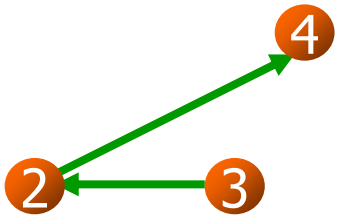
- 拓扑排序举例，最后输出拓扑排序结果：0, 1, 3, 2, 4



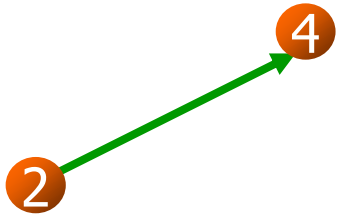
原图



输出0之后



输出0,1之后



输出0,1,3之后

输出0,1,3,2之后

7.5 有向无环图及其应用

二. 拓扑排序

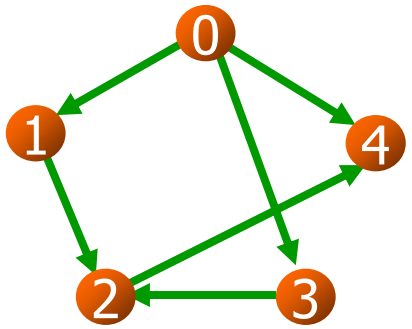
- 拓扑排序程序算法：给出有向图邻接矩阵，求拓扑有序序列

(1) 逐列扫描矩阵，找出入度为0且编号最小的顶点v

(2) 输出v，并标识v已访问

(3) 把矩阵第v行全清0

重复上述步骤，直到所有顶点输出为止



0	1	0	1	1
0	0	1	0	0
0	0	0	0	1
0	0	1	0	0
0	0	0	0	0

7.5 有向无环图及其应用

二. 拓扑排序

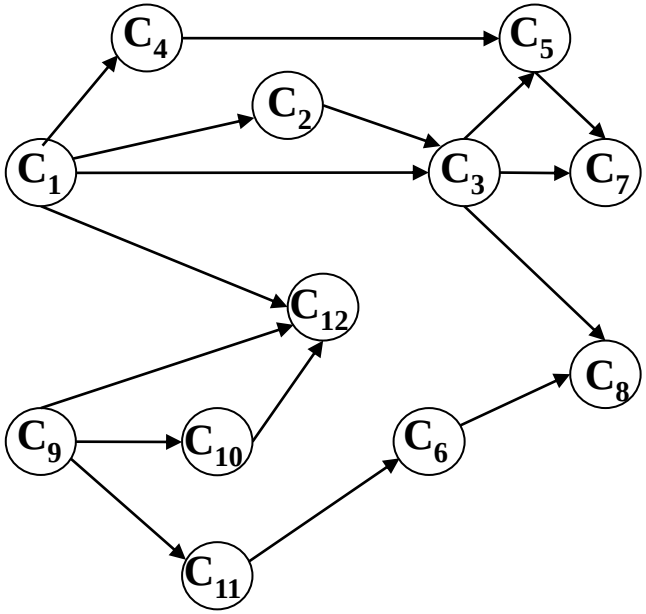
- 拓扑排序的作用：我们可以通过求解一个DAG图的拓扑有序序列，来求得一个项目中各个子活动之间的次序关系
  - 例如软件专业的学生必须学习一系列课程，某些课程是其他课程的先修课，即课程之间有先后关系，求学生修课顺序

课程编号	课程名称	先决条件
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	语言的设计和分析	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_9，C_{10}，C_1

7.5 有向无环图及其应用

二. 拓扑排序

- 拓扑排序的作用
  - 根据表格得到课程之间优先关系图
  - 根据DAG图，求得拓扑有序序列



1. (C1, C2, C3, C4, C5, C7, C9, C10, C11, C6, C12, C8)
2. (C9, C10, C11, C6, C1, C12, C4, C2, C3, C5, C7, C8)

7.5 有向无环图及其应用

二. 拓扑排序

```
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);
    // 建零入度顶点栈 S
    for (i=0; i<G.vexnum; ++ i)
        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
```

7.5 有向无环图及其应用

三. 关键路径

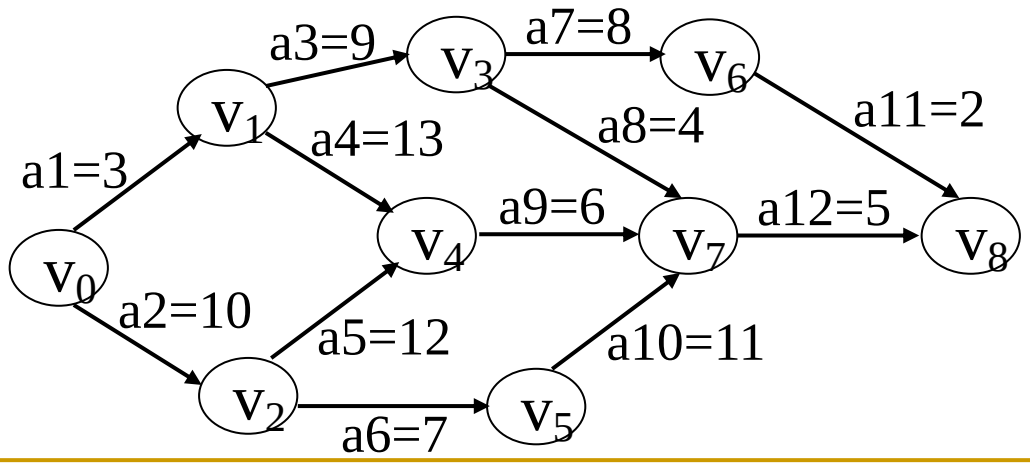
- 对工程的活动加以抽象：图中顶点表示活动，有向边表示活动之间的优先关系，这样的有向图称为顶点表示活动的网(**Activity On Vertex Network ， AOV网**) 。
- 与AOV网对应的还有AOE网，即以边表示活动的网

- 对于工程活动人们关心：
  - 工程能否顺利完成?影响工程的关键活动是什么?
  - 估算整个工程完成所必须的最短时间是多少?
  - 因此常采用AOE网估算工程的进度

7.5 有向无环图及其应用

三. 关键路径

- 图中顶点表示事件(Event)，每个事件表示在其前的所有活动已经完成，其后的活动可以开始；弧表示活动，弧上的权值表示相应活动所需的时间或费用
  - 工程完成最短时间：从起点到终点的最长路径长度(路径上各活动持续时间之和)
  - 长度最长的路径称为关键路径，关键路径上的活动称为关键活动。关键活动是影响整个工程的关键



7.5 有向无环图及其应用

三. 关键路径

- 求AOE网中完成最短时间和关键路径
  - 设v_0是起点，从v_0到v_i的最长路径长度称为事件v_i的最早发生时间，即是以v_i为尾的所有活动的最早发生时间
  - e(i)：表示活动a_i的最早开始时间
  - *l*(i)：在不影响进度的前提下，表示活动ai的最晚开始时间
  - *l*(i)-e(i)表示活动ai的时间余量，若*l*(i)-e(i)=0，表示活动ai是关键活动
  - ve(i)：表示事件vi的最早发生时间，即从起点到顶点vi的最长路径长度；
  - v*l*(i)：表示事件vi的最晚发生时间，即从终点到顶点vi的最短路径长度；

7.5 有向无环图及其应用

三. 关键路径

- 求AOE网中完成最短时间和关键路径
  - 假设活动i由弧<j, k>表示，则j和k表示两个顶点
  - 顶点表示事件，即事件j表示活动i的开始，事件k表示活动i的结束
  - e(i)、*l*(i)、ve(j)、v*l*(k)的关系

    e(i) = ve(j)

    l(i) = vl(k) －dut(<j, k>)
  - 以顶点k为弧头的多条弧<j, k>，j=0,1,……，存在公式

    ve(k) = Max{ve(j) + dut(<j, k>)}
  - 以顶点j为弧尾的多条弧<j,k>，k=0,1,……，存在公式

    vl(j) = Min{vl(k)－dut(<j, k>)}

7.5 有向无环图及其应用

三. 关键路径

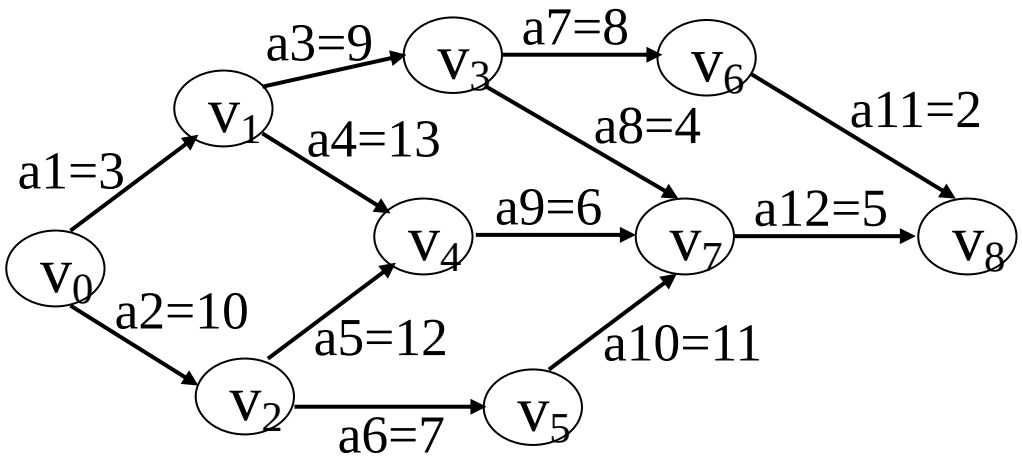
- 求完成最短时间和关键路径的方法
  - 利用拓扑排序求出AOE网的一个拓扑序列；
  - 从拓扑排序的序列的第一个顶点(源点)开始，按拓扑顺序依次计算每个事件的最早发生时间ve(i)；
  - 从拓扑排序的序列的最后一个顶点(汇点)开始，按逆拓扑顺序依次计算每个事件的最晚发生时间vl(i)；

7.5 有向无环图及其应用

三. 关键路径

- 求完成最短时间和关键路径
  - 拓扑排序序列：(v_0, v_1, v_2, v_3 , v_4, v_5 , v_6 , v_7 , v_8)
  - 计算各个事件的ve(i)和v*l*(i)值

顶点	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
ve(i)	0	3	10	12	22	17	20	28	33
v*l*(i)	0	9	10	23	22	17	31	28	33



- 计算各个活动的**e(i)**和**/(i)**值

边	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
E(i)	0	0	3	3	10	10	12	12	22	17	20	28
L(i)	6	0	14	9	10	10	24	24	22	17	31	28

7.5 有向无环图及其应用

三. 关键路径

- 求完成最短时间和关键路径的方法
  - 根据 $l(i)-e(i)=0$找出关键路径

$(v_0, v_2, v_4, v_7, v_8)$ 和$(v_0, v_2, v_5, v_7, v_8)$

  - 关键路径活动

$\langle v_0, v_2\rangle$，$\langle v_2, v_4\rangle$，$\langle v_2, v_5\rangle$，$\langle v_4, v_7\rangle$，$\langle v_5, v_7\rangle$，$\langle v_5, v_8\rangle$

- 用AOE网估算工程完成时间是非常有用的，并可以提高几条关键活动的速度来缩短工期

7.5 有向无环图及其应用

三. 关键路径

```
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[o..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;
        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);
      }
    if(count<G.vexnum) return ERROR;
      else return OK;
}
```

7.5 有向无环图及其应用

三. 关键路径

```
Status CriticalPath(ALGraph G){
    if(!TopologicalOrder(G, T)) return ERROR;
    vl[0..G.vexnum-1] = ve[G.vexnum-1];
    // 初始化各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);
        if(vl[k]-dut<vl[j]) v[j]=vl[k]-dut;
    }
    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[jl]; el=vl[k]-dut;
            tag=(ee == el)?'*':' ';
            printf(j,k, dut, ee, el, tag);  // 输出关键活动
        }
} //CriticalPath
```

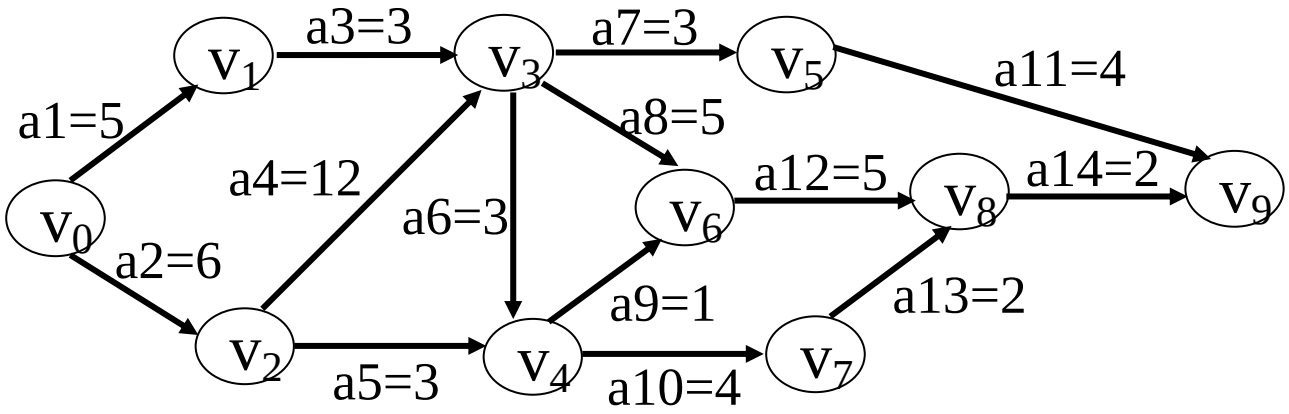
练习

- 假设一个工程的进度计划用**AOE**网表示，如图所示

① 求出每个事件的最早发生时间和最晚发生时间。

② 该工程完工至少需要多少时间?

③ 求出所有关键路径和关键活动。



顶点	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
ve(i)	0	5	6	18	21	21	23	25	28	30
vl(i)	0	15	6	18	22	26	23	26	28	30

第七章总结

- 图的数据结构：G＝(V, E) ，V是顶点，E是边或弧
  - 无向图的边(x,y)，有向图的弧<x,y>，带权值的图称为网
- 图的术语：
  - 度、入度、出度的计算
  - 路径、回路（环）、简单路径、连通、生成树
- 图的存储结构：
  - 邻接矩阵、邻接表和逆邻接表
  - 十字链表、邻接多重表
- 图的遍历：深度优先搜索、广度优先搜索
- 生出树：DFS生成树和BFS生成树
- 最小生成树：
  - 普里姆(Prim)算法，从点出发
  - 克鲁斯卡尔(Kruskal)算法，从边中选择
- 拓扑排序，根据AOV网求拓扑有序序列
- 关键路径，根据AOE图求关键路径、最早和最晚开始时间
- 最短路径：迪杰斯特拉(Dijkstra)算法