# 7.7 关键路径

## 7.7.1 AOE 网

对于活动在边上的网（Activity On Edge network，AOE）可以和 AOV 网对比着来记忆。

两者的相同点：**都是有向无环图。**

两者的不同点：AOE 网的**边表示活动，边有权值，边代表活动持续时间**；顶点表示事件，事件是图中新活动开始或者旧活动结束的标志。AOV 网的**顶点表示活动，边无权值，边代表活动之间的先后关系。**

对于一个表示工程的 AOE 网，只存在一个入度为 0 的顶点，称为**源点**，表示整个工程的开始；也只存在一个出度为 0 的顶点，称为**汇点**，表示整个工程的结束。

**说明：考研中对于 AOE 网，虽然大纲中没有明确指出，但是这部分和本节其他内容联系紧密，属于大纲模糊范围内的内容，考生最好还是掌握。**

## 7.7.2 关键路径核心算法

在 AOE 网中，从源点到汇点的所有路径中，具有**最大路径长度**的路径称为关键路径。完成整个工期的最短时间就是关键路径长度所代表的时间。关键路径上的活动称为关键活动。关键路径是个特殊的概念，它**既代表了一个最短又代表了一个最长**，它是图中的**最长路径**，又是整个工期所完成的**最短时间**。这句话的含义考生要好好理解。考研中所涉及本节的题目，主要是手工求关键路径的过程，下面通过一个例子总结求关键路径的一般方法。



图 7-27 所示为一个 AOE 网，下面一步一步地求出它的关键路径，注意每一步的操作以及各步之间的联系。

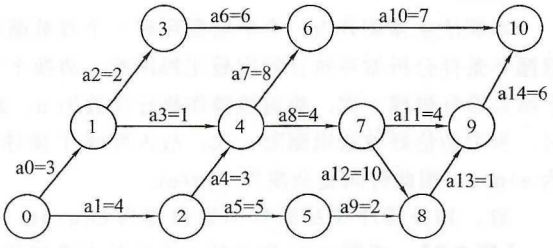


图 7-27　AOE 网

1）设 ve(k)为顶点 k 代表的事件（以下称事件 k）的最早发生时间，即从源点到顶点 k 的路径中的**最长者**（注意下边求解过程中的 max{}），即 ve(k)为 ve(j)+<j，k>权值后所得结果中的最大者，其中 j 为 k 的前驱事件，j 可能有多个。为什么是最长者呢？因为在 AOE 网中，如果事件 k 要发生，必须在事件 k 之前的活动都已经完成的情况下才可能，k 事件之前的活动是为 k 的发生做准备的；由源点到 k 的路径不止一条，每一条都代表为事件 k 的发生所做的准备工作，这些准备工作是同时开始进行的，显然 k 之前的所有准备工作的完成时间，可以用其中持续时间最长的准备工作的时间来表示，因此要取从源点到顶点 k 的路径中的最长者。在图 7-27 中，事件 4 的最早发生时间是路径<0，2，4>（长度为 7）所代表的时间，而不是路径<0，1，4>（长度为 4）所代表的时间。

为了求出关键路径，必须求出图中每个事件的最早发生时间，根据图 7-27，具体过程如下：

① 对图 7-27 进行**拓扑排序**，得到各顶点的拓扑有序序列为 0，1，2，3，4，5，6，7，8，9，10。

② 按照上述拓扑有序序列的顺序，依次求出各顶点所代表事件的最早发生时间。

初始时，将事件 0 的开始时间设置为 0，即 ve(0)=0。于是求得：

ve(1)=ve(0)+a0=0+3=3

ve(2)=ve(0)+a1=0+4=4

ve(3)=ve(1)+a2=3+2=5

ve(4)=max{ve(1)+a3，ve(2)+a4}=max{3+1，4+3}=7

ve(5)=ve(2)+a5=4+5=9

ve(6)=max{ve(3)+a6，ve(4)+a7}=max{5+6，7+8}=15

ve(7)=ve(4)+a8=7+4=11

ve(8)=max{ve(7)+a12，ve(5)+a9}=max{11+10，9+2}=21

ve(9)=max{ve(7)+a11，ve(8)+a13}=max{11+4，21+1}=22

ve(10)=max{ve(6)+a10，ve(9)+a14}=max{15+7，22+6}=28

2）设 vl(k)为事件 k 的最迟发生时间，**事件 k 的最迟发生时间是在不推迟整个工程完成的前提下，该事件最迟必须发生的时间**。vl(k)为 vl(j)-<k，j>权值后所得结果中的**最小者**（注意下边求解过程中的 min{}），其中 j 为 k 的后继事件，j 可能有多个。事件 k 的发生，一定不能推迟其所有后继事件的最迟发生时间，因此 k 要尽可能早地发生，而 vl(j)-<k，j>所得结果越小表明 k 发生越早，因此要取其中的最小者。图 7-27 中的事件 10 是个特殊的事件，它的最早发生时间是整个工程的结束时间，因此它的最早发生时间也是最迟发生时间，即 vl(10)=ve(10)。如果事件 10 推迟发生，则整个工程必定推迟完成。于是可以从顶点 10 开始逐一推算出其他事件的最迟发生时间，具体步骤如下：

① 对图 7-27 进行逆拓扑排序，得到各顶点的逆拓扑有序序列为 10，9，6，8，5，7，3，4，1，2，0。

② 按照上述逆拓扑有序序列的顺序，依次求出各顶点所代表事件的最迟发生时间。

vl(10)=ve(10)=28

vl(9)=vl(10)-a14=28-6=22

vl(6)=vl(10)-a10=28-7=21

vl(8)=vl(9)-a13=22-1=21

vl(5)=vl(8)-a9=21-2=19

vl(7)=min{vl(9)-a11，vl(8)-a12}=min{22-4，21-10}=11

vl(3)=vl(6)-a6=21-6=15

vl(4)=min{vl(6)-a7，vl(7)-a8}=min{21-8，11-4}=7

vl(1)=min{vl(3)-a2，vl(4)-a3}=min{15-2，7-1}=6

vl(2)=min{vl(5)-a5，vl(4)-a4}=min{19-5，7-3}=4

vl(0)=min{vl(1)-a0，vl(2)-a1}=min{6-3，4-4}=0

3）由 1）、2）两步，求出了图 7-27 中事件的最早和最迟发生时间。下面来求每个活动的最早和最迟发生时间。分别用 e(ak)和 l(ak)来表示当前活动 ak 的最早和最迟发生时间。图中的**事件代表一个新活动的开始或旧活动的结束，因此事件的最早发生时间就是由这个事件所发出的活动的最早发生时间**。活动的最迟发生时间怎样求得呢？我们知道图中事件的最迟发生时间代表了以它为结束点的活动的最迟结束时间，因此**用事件的最迟发生时间减去以它为结束点的活动的持续时间，就得到活动的最迟发生时间**。

由以上分析，可以求出各个活动的最早和最迟发生时间。

① 求最早发生时间：

e(a0)=e(a1)=ve(0)=0

e(a2)=e(a3)=ve(1)=3

e(a4)=e(a5)=ve(2)=4

e(a6)=ve(3)=5

e(a7)=e(a8)=ve(4)=7

e(a9)=ve(5)=9

e(a10)=ve(6)=15

e(a11)=e(a12)=ve(7)=11

e(a13)=ve(8)=21

e(a14)=ve(9)=22

② 求最迟发生时间：

l(a0)=vl(1)-3=3

l(a1)=vl(2)-4=0

l(a2)=vl(3)−2=13
l(a3)=vl(4)−1=6
l(a4)=vl(4)−3=4
l(a5)=vl(5)−5=14
l(a6)=vl(6)−6=15
l(a7)=vl(6)−8=13
l(a8)=vl(7)−4=7
l(a9)=vl(8)−2=19
l(a10)=vl(10)−7=21
l(a11)=vl(9)−4=18
l(a12)=vl(8)−10=11
l(a13)=vl(9)−1=21
l(a14)=vl(10)−6=22

把①和②两步中所得数据整理成一个表，见表 7-7。

表 7-7 图 7-27 中活动的最早发生时间和最迟发生时间

| 活动 | 最早发生时间 | 最迟发生时间 | 关键活动 |
|---|---|---|---|
| a0 | 0 | 3 | |
| a1 | 0 | 0 | ▲ |
| a2 | 3 | 13 | |
| a3 | 3 | 6 | |
| a4 | 4 | 4 | ▲ |
| a5 | 4 | 14 | |
| a6 | 5 | 15 | |
| a7 | 7 | 13 | |
| a8 | 7 | 7 | ▲ |
| a9 | 9 | 19 | |
| a10 | 15 | 21 | |
| a11 | 11 | 18 | |
| a12 | 11 | 11 | ▲ |
| a13 | 21 | 21 | ▲ |
| a14 | 22 | 22 | ▲ |

表 7-7 中用“▲”指出了关键活动，最早发生时间和最迟发生时间相同的活动就是关键活动。这里还要介绍一个量，就是活动的**剩余时间**。**剩余时间等于活动的最迟发生时间减去活动的最早发生时间**。剩余时间反映了活动完成的一种**松弛度**。例如，导师交给你一项任务，以你的水平可以 3 天完成，而导师给了你 5 天的时间，这样你就有 2 天的剩余时间。你只要在前两天内的任何一个时刻开始执行任务都不会影响最终任务的完成。通过表 7-7 可以看出，**关键活动的剩余时间为 0**，这体现了关键活动在整个工程中的重要性，关键活动没有缓期执行的余地。关键活动组成了关键路径，本例中关键路径如图 7-28 所示。

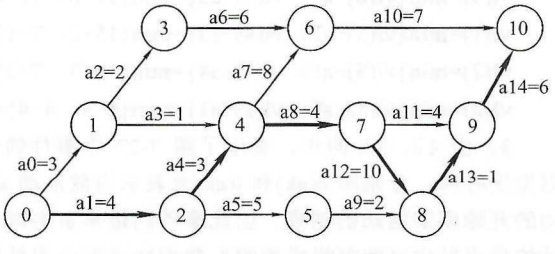


图 7-28 关键路径

由图 7-28 可知：a1+a4+a8+a12+a13+a14=28。

图 7-28 反映了关键路径所持续的时间就是整个工程所持续的时间。

**说明：上述步骤写得比较详细，是为了方便大家理解。考试中如果出现这种手工求关键路径的题目大可不必这样写，大家可以在理解的基础上总结出适合自己的简洁的答题步骤。**

由以上求解过程可以总结出求关键路径的一般方法，具体内容如下：

1）根据图求出拓扑有序序列 a 和逆拓扑有序序列 b。

2）根据序列 a 和 b 分别求出每个事件的最早发生时间和最迟发生时间，求解方法如下：

① 一个事件的最早发生时间为指向它的边（假设为 a）的权值加上发出 a 这条边的事件的最早发生时间。如果有多条边，则逐一求出对应的时间并选其中最大的结果作为当前事件的最早发生时间。

② 一个事件的最迟发生时间为由它所发出的边（假设为 b）所指向的事件的最迟发生时间减去 b 这条边的权值。如果有多条边，则逐一求出对应的时间并选其中最小的结果作为当前事件的最迟发生时间。

3）根据 2）中结果求出每个活动的最早发生时间和最迟发生时间。

4）根据 3）中结果找出最早发生时间和最迟发生时间相同的活动，即为关键活动。由关键活动所连成的路径即为关键路径。

**说明：考试中对于关键路径这一部分，可能出选择题，也可能出大题，只要根据上述讲解熟练掌握了关键路径的手工求解过程，无论是选择题还是大题，都可以迎刃而解。**

微信答疑

提问：

一个工程的完成仅仅需要执行关键活动吗？还是说关键活动完成的同时其他普通活动也已经完成了呢？

回答：

不是的，图中所表示的所有活动都要执行，只不过关键路径执行所需的时间就是整个图中所有活动所完成的时间。

## ▲真题仿造

1．试写一算法，判断以邻接表方式存储的有向图中是否存在由顶点 $v_i$ 到顶点 $v_j$ 的路径。

（1）给出算法的基本设计思想。

（2）根据设计思想，采用 C 或 C++语言描述算法，并在关键之处给出注释。

（3）分析算法的时间复杂度。

2．在有向图 G 中，如果 r 到 G 中的每个结点都有路径可达，则称结点 r 为 G 的根结点。编写一个算法判断有向图 G 是否有根，若有，则打印出所有根结点的值。假设图已经存在于邻接表 g 中。

（1）给出算法的基本设计思想。

（2）根据设计思想，采用 C 或 C++语言描述算法，并在关键之处给出注释。



## 真题仿造答案与解析

**1．解**

**（1）算法基本设计思想**

判断图中从 $v_i$ 到 $v_j$ 是否有路径，可以采取遍历的方法。遍历的起点为 $v_i$，在一次 BFS 退出之前遇到 $v_j$，则证明有路径，否则没有路径。

**（2）算法描述**

```
int BFS(AGraph *G,int vi, int vj)      //将函数返回值类型改为 int 型
{
    ArcNode *p;
    int que[maxSize],front=0,rear=0;
    int visit[maxSize];
    int i,j;
    for(i=0;i<G->n;++i)visit[i]=0;
    rear=(rear+1)%maxSize;
    que[rear]=vi;
    visit[vi]=1
    while(front!=rear)
    {
        front=(front+1)%maxSize;
        j=que[front];
        if(j==vj)//此处为修改处，将对顶点的访问函数换成判断当前顶点是否为 vj 即可
            return 1;
        p=G->adjlist[j].firstarc;
        while(p!=NULL)
        {
```

```
            if(visit[p->adjvex]==0)
            {
                rear=(rear+1)%maxSize;
                que[rear]=p->adjvex;
                visit[p->adjvex]=1;
            }
            p=p->nextarc;                    //p 指向 j 的下一条边
        }
    }
    return 0;
}
```

**（3）时间复杂度分析**

本算法的主体部分为一个双重循环，基本操作有两种：一种是顶点进队，另一种是边的访问。最坏情况为遍历图中的所有顶点后才找到通路，此时所有顶点都进队一次，所有边都被访问一次，因此基本操作的总次数为 n+e（其中，n 为图中顶点数，e 为图中边数），即本题的时间复杂度为 O(n+e)。

**2．解**



**（1）算法基本设计思想**

判断顶点 r 到 G 中的每个顶点是否有路径可达，可以通过深度优先搜索遍历的方法。以 r 为起点进行深度优先搜索遍历，若在函数 dfs()退出前已经访问过所有顶点，则 r 为根。要打印出所有的根结点，可以对图中的每个顶点都调用一次函数 dfs()，如果是根则打印。

**（2）算法描述**

```
int visit[maxSize],sum;//假设常量 maxSize 已经定义
/*以下是深度优先搜索遍历算法*/
void DFS(AGraph *G,int v)
{
    ArcNode *p;
    visit[v]=1;
    ++sum;                  //此处为 DFS 算法的修改部分，将函数 visit()换成了++sum
                            //即每次访问一个顶点时计数器加 1
    p=G->adjlist[v].firstarc;
    while(p!=NULL)
    {
        if(visit[p->adjvex]==0)
            DFS(G,p->adjvex);
            p=p->nextarc;
    }
}
void print(AGraph *G)
{
    int i,j;
    for(i=0;i<G->n;++i)
    {
        sum=0;                      //每次选取一个新起点时计数器清零
        for(j=0;j<G->n;++j)         //每次进行 DFS 时访问标记数组清零
```

```
            visit[j]=0;
        DFS (G,i);
        if(sum==G->n)                    //当图中顶点全部被访问时则判断为根，输出
            cout<<i<<endl;
    }
}
```

# 习题+真题精选

微信扫码看本章题目讲解视频：



一、习题

（一）选择题

1．图中有关路径的定义是（　　）。

A．由相邻顶点序偶所形成的序列　　B．由不同顶点所形成的序列

C．由不同边所形成的序列　　D．上述定义都不是

2．设无向图的顶点个数为n，则该图最多有（　　）条边。

A．n-1　　B．n(n-1)/2　　C．n(n+1)/2　　D．0　　E．$n^2$

3．含有n个顶点的连通无向图，其边的个数至少为（　　）。

A．n-1　　B．n　　C．n+1　　D．$n\log_2 n$

4．要连通具有n个顶点的有向图，至少需要（　　）条边。

A．n-1　　B．n　　C．n+1　　D．2n

5．n个结点的完全有向图含有边的数目为（　　）。

A．$n^2$　　B．n(n+1)　　C．n/2　　D．n(n-1)

6．一个有n个结点的无向图，最少有（　　）个连通分量，最多有（　　）个连通分量。

A．0　　B．1　　C．n-1　　D．n

7．在一个无向图中，所有顶点的度数之和等于所有边数的（　　）倍；在一个有向图中，所有顶点的入度之和等于所有顶点出度之和的（　　）倍。

A．1/2　　B．2　　C．1　　D．4

8．用有向无环图描述表达式(A+B)*((A+B)/A)，至少需要顶点的数目为（　　）。

A．5　　B．6　　C．8　　D．9

9．用DFS遍历一个无环有向图，并在DFS算法退栈返回时打印相应的顶点，则输出的顶点序列是（　　）。

A．逆拓扑有序　　B．拓扑有序　　C．无序的

10．（　　）的邻接矩阵是对称矩阵。

A．有向图　　B．无向图　　C．AOV网　　D．AOE网

11．由邻接矩阵 $\mathbf{A}=\begin{pmatrix}0&1&0\\1&0&1\\0&1&0\end{pmatrix}$ 可以看出，该图共有①中的（　　）个顶点；如果是有向图，该图共