# 实验6 关键路径实验指导

## 1题目

计算工程项目的关键路径。

（1）AOE和AOV网

AOE-网：指用边表示活动的网，是一个带权的有向无环图，其中，顶点表示事件，弧表示活动，权表示活动持续的时间，通常一个AOE-网可用来估算工程的完成时间。

AOE网具有以下几个性质：

① 只有在进入某一顶点的各有向边所代表的活动都已经结束，该顶点所代表的事件才能发生；

② 只有在某顶点所代表的事件发生后，从该顶点出发的各有向边所代表的活动才能开始；

③ 表示实际工程计划的AOE网应该是无环的，并且存在惟一的入度为0的开始顶点(源点）和惟一的出度为0的完成顶点（汇点）。

对于AOE-网，我们不妨采用与AOV-网一样的邻接表的存储方式，其中邻接表中边结点增设一个dut域存放该边的权值，即该有向边代表的活动所持续的时间。

（2）AOE网有关问题

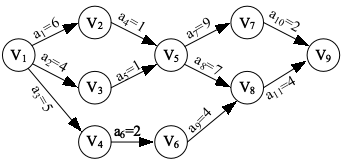
① 完成整个工程至少需要多少时间；

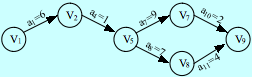
② 哪些活动是影响工程进度的关键。

（3）关键路径

由于在AOE-网中某些活动可以并行地进行，因此完成工程的最短时间是从开始顶点（源点）到完成顶点（汇点）的最大路径长度（路径长度是指这条路径上完成各个活动所需时间之和，不是路径上弧的数目）；具有最大路径长度的路径称为关键路径。

## 2 图的关键路径用例



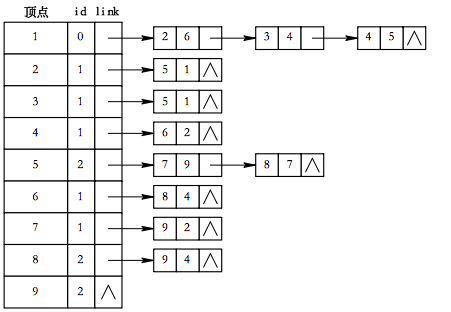


关键活动是：a1, a4, a7, a8, a10, a11

它们构成了两条关键路径：(v1，v2，v5，v7，v9)和(v1，v2，v5，v8，v9)

## 3 关键路径算法

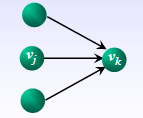
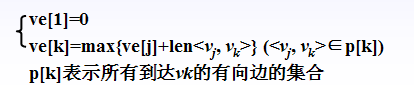
（1）AOE 网的存储结构



（2）与关键路径有关的几个术语

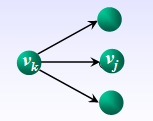
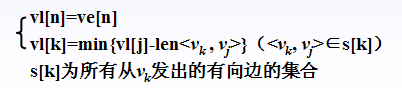
① 事件的最早发生时间ve[k]

　　ve[k]是指从始点开始到顶点*vk*的最大路径长度，这个长度决定了所有从顶点*vk*发出的活动能够开工的最早时间。

② 事件的最迟发生时间vl[k]

　　vl[k]是指在不推迟整个工期的前提下,事件*vk*允许的最晚发生时间。

③ 活动的最早开始时间e[i]

　　若活动*ai*是由弧<*vk*, *vj*>表示，则活动*ai*的最早开始时间e[i]应等于事件*vk*的最早发生时间。因此，有：e[i]=ve[k]

④ 活动的最晚开始时间l[i]

　　活动*ai*的最晚开始时间l[i]是指，在不推迟整个工期的前提下， *ai*必须开始的最晚时间。若*ai*由弧<*vk*，*vj*>表示，则*ai*的最晚开始时间要保证事件*vj*的最迟发生时间不拖后。因此，有：l[i]=vl[j]-len<*vk*,*vj*>

（3）求关键路径的算法描述

① 建立图的邻接表存储结构，计算每个顶点的入度；

② 对图进行拓扑排序，排序过程中求顶点的ve[i]，将得到的拓扑序列进栈；

③ 从汇点vn出发，令vl[n]=ve[n],按逆拓扑序列求其余各顶点的vl[i]；

④ 根据各顶点的ve和vl值，计算每条弧的e[i]和l[i]，找出e[i]=l[i]的关键活动。

## 4 测试用例执行过程

根据给定的上图求该AOE-网的关键活动及关键路径。

第一行表示顶点数和弧数；后面每一行就是权值，弧尾，弧头；

6 8

3 1 2

2 1 3

2 2 4

3 2 5

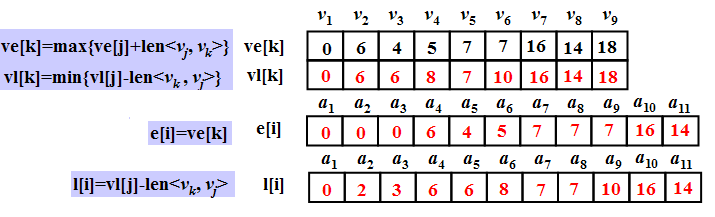
4 3 4

3 3 6

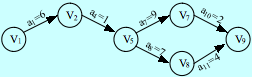
2 4 6

1 5 6

根据3.算法描述（3），从顶点V1到汇点V9 计算ve、vl和活动的e、l，判定关键路径。



得到的关键路径的图如下：



关键活动是：a1, a4, a7, a8, a10, a11

它们构成了两条关键路径：(v1，v2，v5，v7，v9)和(v1，v2，v5，v8，v9)

版权声明：本文为CSDN博主「Randyhe\_」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/randyhe\_/article/details/78659453

## 5关键代码

Status TopOrder(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为拓扑序列定点栈，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。

Stack S;

intcount=0,k;

charindegree[40];

ArcNode \*p;

InitStack(S);

FindInDegree(G, indegree); // 对各顶点求入度indegree[0..vernum-1]

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

if(indegree[j]==0)

Push(S, j); // 入度为0者进栈

InitStack(T);//建拓扑序列顶点栈T

count = 0;

for(inti=0; i<G.vexnum; i++)

ve[i] = 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; // 对j号顶点的每个邻接点的入度减1

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;

}//for \*(p->info)=dut(<j,k>)

}

if(count<G.vexnum)

returnERROR; // 该有向网有回路

else

returnOK;

}

Status CriticalPath(ALGraph G)

{

// G为有向网，输出G的各项关键活动。

Stack T;

inta,j,k,el,ee,dut;

chartag;

ArcNode \*p;

if(!TopOrder(G, T))

return ERROR;

for(a=0; a<G.vexnum; a++)

vl[a] = ve[G.vexnum-1]; // 初始化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; //dut<j,k>

if(vl[k]-dut < vl[j])

vl[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[j]; el = vl[k]-dut;

tag = (ee==el) ?'\*' : ' ';

printf(j, k, dut, ee, el, tag); // 输出关键活动

}

return OK;

}