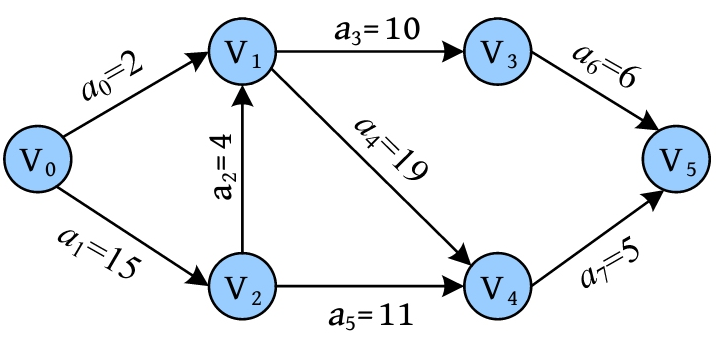
**关键路径**

**AOV网可以反映活动之间的先后制约关系**，但在实际工程中，**有时活动不仅有先后顺序，还有持续时间，必须经过多长时间该活动才可以完成**。这时需要另外一种网络——**AOE网（Activity On Edge），即以边表示活动的网**。

**AOE网是一个带权的有向无环图，节点表示事件，弧表示活动，弧上的权值表示活动持续的时间**。

例如，有一个包含6个事件、8个活动的工程，如下图所示。V0、V5分别代表工程的开始（源点）和结束（汇点），在活动a0、a2结束后，事件V1才可以开始，在V1结束后，活动a3、a4才可以开始。



在实际工程应用中通常需要解决两个问题：①估算完成整个工程至少需要多少时间；②判断哪些活动是关键活动，即如果该活动被耽搁，则会影响整个工程进度。

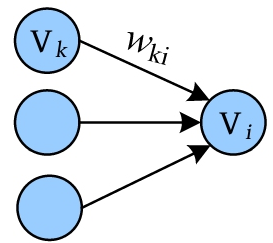
在AOE网中，从源点到汇点的带权路径长度最大的路径为关键路径。关键路径上的活动为关键活动。

**确定关键路径时首先要清楚4个问题**：**事件的最早发生时间**、**最迟发生时间**，**以及活动的最早发生时间**、**最迟发生时间**。

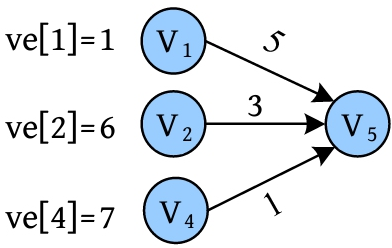
**1）事件Vi的最早发生时间ve[i]**

事件Vi的最早发生时间是从源点到Vi的最大路径长度。很多人不理解，为什么最早发生时间是最大路径长度？举例说明，小明妈妈一边炒菜，一边熬粥，炒菜需要20分钟，熬粥需要30分钟，最早什么时间开饭？肯定是最大时间。

因为进入事件Vi的所有入边活动都已完成，Vi才可以开始，因此可以根据事件的拓扑顺序从源点向汇点递推，求解事件的最早发生时间。初始化源点的最早发生时间为0，即ve[0]=0。以Vi的最早发生时间考察入边，取弧尾ve+入边权值的最大值，ve[i]=max{ve[k]+wki}，<Vk,Vi>∈T。T为以Vi为弧头的弧集合，即Vi的入边集合，如下图所示。



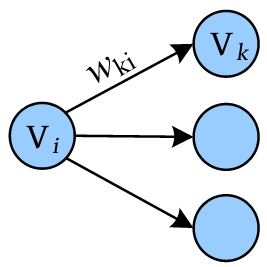
例如，一个AOE网如下图所示。已经求出V1、V2、V4三个节点的ve值，求V5的ve值。考察V5的入边，ve[5]=max{ve[1]+5,ve[2]+3,ve[4]+1}=9。



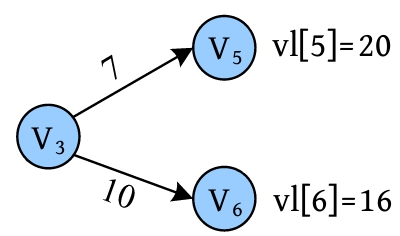
**2）事件Vi的最迟发生时间vl[i]**

事件Vi的最迟发生时间不能影响其所有后继的最迟发生时间。Vi的最迟发生时间不能大于其后继Vk的最迟发生时间减去活动<Vi,Vk>的持续时间。因此可以根据事件的逆拓扑顺序从汇点向源点递推，求解事件的最迟发生事件。

初始化汇点的最迟发生时间为汇点的最早发生时间，即vl[n-1]=ve[n-1]。以Vi的最迟发生时间考察出边，取弧头vl-出边权值的最小值，ve[i]=min{vl[k]-wki}，<Vk,Vi>∈T。T为以Vi为弧尾的弧集合，即Vi的出边集合。

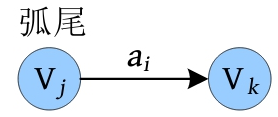


例如，一个AOE网如下图所示。已经求出V5、V6两个节点的vl值，求V3的vl值。考察V3的出边，vl[3]=min{vl[5]-7,vl[6]-10}=6。

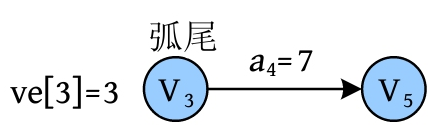


**3）活动ai=<Vj,Vk>的最早发生时间e[i]**

只要事件Vj发生了，活动ai就可以开始，因此活动ai的最早发生时间等于事件Vj的最早发生时间。即ai的最早发生时间为其弧尾的最早发生时间，e[i]=ve[j]。

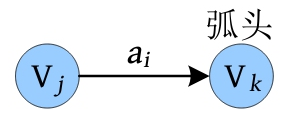


例如，一个AOE网如下图所示。已经求出V3节点的ve值，求a4的e值。a4 的e值等于弧尾V3的ve值，e[4]=ve[3]=3。

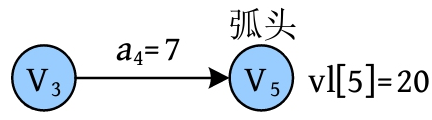


**4）活动ai=<Vj,Vk>的最迟发生时间l[i]**

活动ai的最迟发生时间不能耽误事件Vk的最迟发生时间，因此活动ai的最迟发生时间等于事件Vk的最迟发生时间减去活动ai的持续时间wjk。即活动ai的最迟发生时间等于弧头的最迟发生时间减去边值，l[i]=vl[k]-wjk。



例如，一个AOE网如下图所示。已经求出V5节点的vl值，求a4的l值。a4的l值=弧头V5的vl值-边值，l[4]= vl[5]-7=20-7=13。



**1. 求解过程**

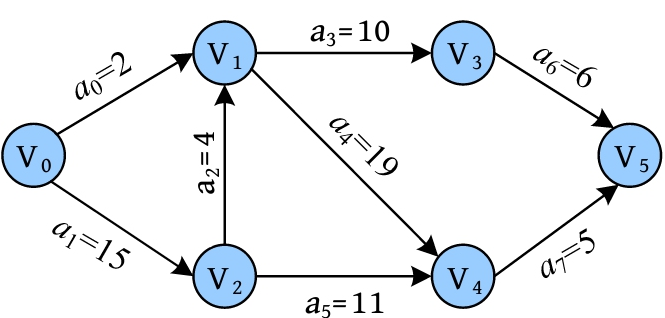
（1）事件Vi的最早发生时间ve[i]：考察入边，弧尾ve+入边权值的最大值。

（2）事件Vi的最迟发生时间vl[i]：考察出边，弧头vl-出边权值的最小值。

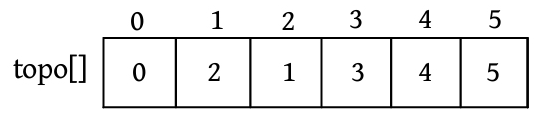
（3）活动ai的最早发生时间e[i]：弧尾的最早发生时间。

（4）活动ai的最迟发生时间l[i]：弧头的最迟发生时间减去边值。

**2. 图解**

例如，一个AOE网如下图所示，求其关键路径。

（1）求拓扑排序序列，将其保存在topo[]数组中。

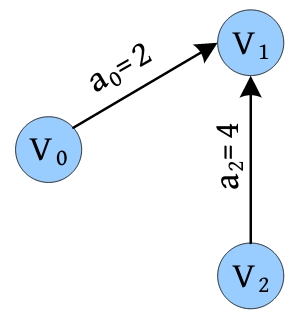


（2）按照拓扑排序序列（0,2,1,3,4,5），从前向后求解每个节点的最早发生时间ve[]。考察节点的入边，即求弧尾ve+入边权值的最大值。

• ve[0]=0。

• ve[2]=ve[0]+15=15。

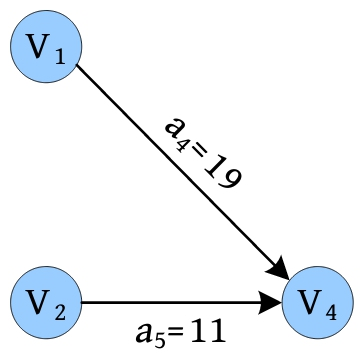
V1有两个入边，弧尾ve+入边权值，取最大值。



• ve[1]=max{ve[2]+4,ve[0]+2}=19。

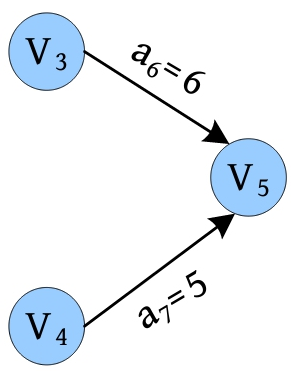
• ve[3]=ve[1]+10=29。

V4有两个入边，弧尾ve+入边权值，取最大值。



ve[4]=max{ve[2]+11,ve[1]+19}=38。

V5有两个入边，弧尾ve+入边权值，取最大值。



ve[5]=max{ve[4]+5,ve[3]+6}=43。

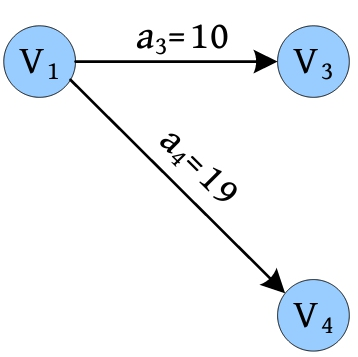
（3）按照逆拓扑顺序(5,4,3,1,2,0)，从后向前求解每个节点的最迟发生时间vl[]。初始化汇点的最迟发生时间为汇点的最早发生时间，即vl[n-1]=ve[n-1]。对其他节点考察出边，弧头vl-出边权值的最小值。

• vl[5]=ve[5]=43。

• vl[4]=vl[5]-5=38。

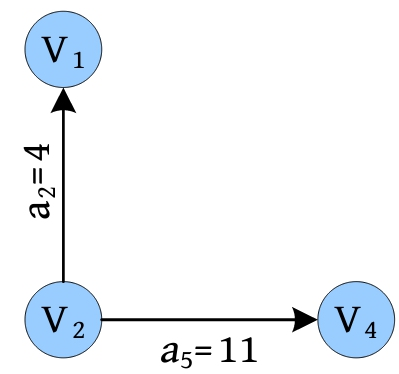
• vl[3]=vl[5]-6=37。

V1有两个出边，弧头vl-出边权值，取最小值。



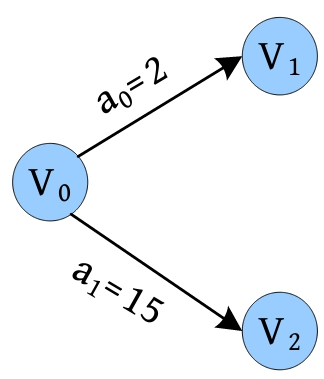
vl[1]=min{vl[4]-19,vl[3]-10}=19。

V2有两个出边，弧头vl-出边权值，取最小值。



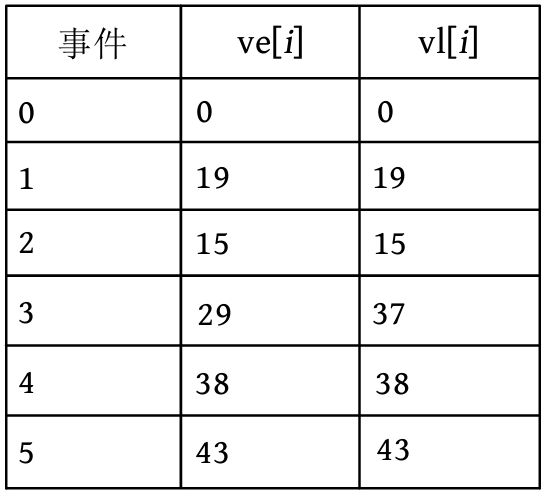
vl[2]= min{vl[4]-11,vl[1]-4}=15。

V0有两个出边，弧头vl-出边权值，取最小值。



vl[0]=min{vl[2]-15,vl[1]-2}=0。

求解完毕后，事件的最早发生时间和最迟发生时间如下表所示。



（4）计算每个活动的最早发生时间和最迟发生时间。活动ai的最早发生时间e[i]等于弧尾的最早发生时间。活动ai的最迟发生时间l[i]等于弧头的最迟发生时间减去边值。

• 活动a0=<V0,V1>：e[0]=ve[0]=0；l[0]=vl[1]-2=17。

• 活动a1=<V0,V2>：e[1]=ve[0]=0；l[1]=vl[2]-15=0。

• 活动a2=<V2,V1>：e[2]=ve[2]=15；l[2]=vl[1]-4=15。

• 活动a3=<V1,V3>：e[3]=ve[1]=19；l[3]=vl[3]-10=27。

• 活动a4=<V1,V4>：e[4]=ve[1]=19；l[4]=vl[4]-19=19。

• 活动a5=<V2,V4>：e[5]=ve[2]=15；l[5]=vl[4]-11=27。

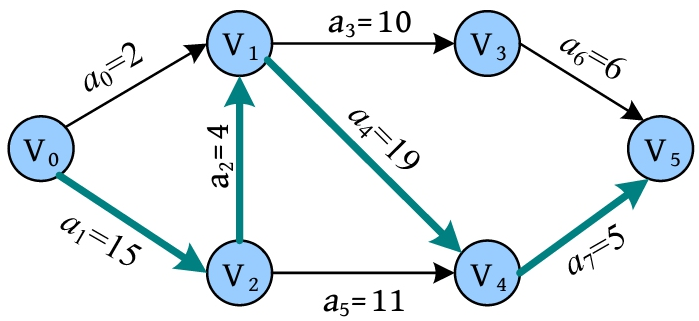
• 活动a6=<V3,V5>：e[6]=ve[3]=29；l[6]=vl[5]-6=37。

• 活动a7=<V4,V5>：e[7]=ve[4]=38；l[7]=vl[5]-5=38。

如果活动的最早发生时间等于最迟发生时间，则该活动为关键活动，如下表所示。



（5）由关键活动组成的从源点到汇点的路径为关键路径V0-V2-V1-V4-V5，如下图所示。



**3. 算法步骤**

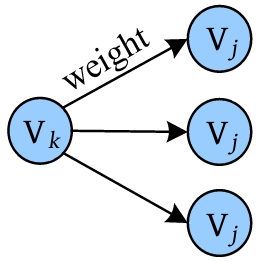
（1）利用拓扑排序算法，将拓扑排序结果保存在topo[]数组中。

（2）将每个事件的最早发生时间都初始化为0，即v[i]=0，i=0,1,…,n-1。

（3）根据拓扑顺序从前向后依次求每个事件的最早发生时间，循环执行这些操作：①取出拓扑序列中的节点k，k=topo[i]，i=0,1,…,n-1；②用指针p依次指向k的每个邻接点，取得邻接点的序号j=p->v，更新节点j的最早发生时间ve[j]，即



相当于求弧尾ve+入边的最大值，如下图所示。



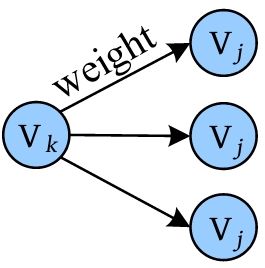
**这里的程序处理并不是一下子考察所有入边，但效果是一样的，想一想为什么？**

（4）将每个事件的最迟发生时间vl[i]都初始化为汇点的最早发生时间，即vl[i]=ve[n-1]。

（5）按照逆拓扑顺序从后向前求解每个事件的最迟发生时间，循环执行这些操作：①取出逆拓扑序列中的序号k，k=topo[i]，i=n-1,…,1,0；②用指针p依次指向k的每个邻接点，取得邻接点的序号j=p->v，更新节点k的最迟发生时间vl[k]，即



相当于求弧头vl-出边的最小值，如下图所示。

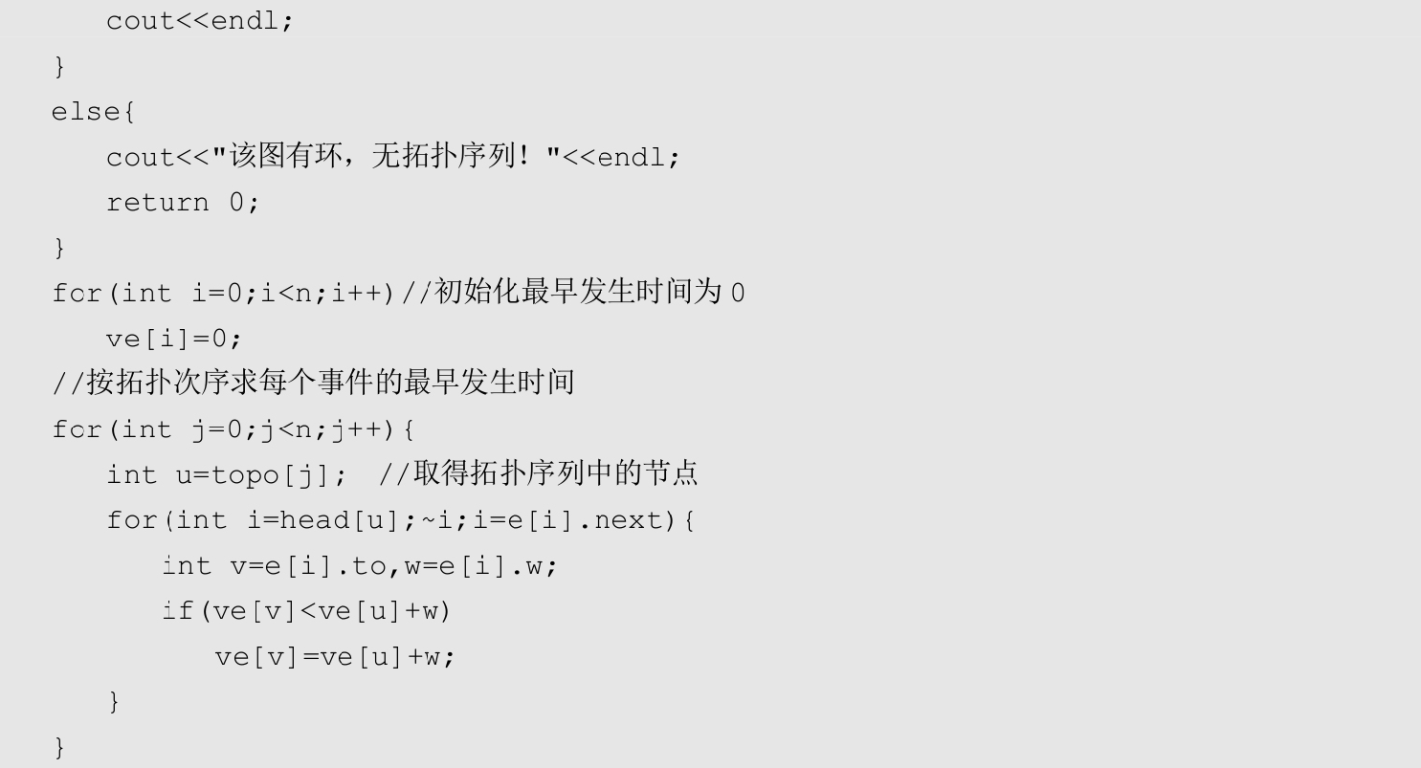


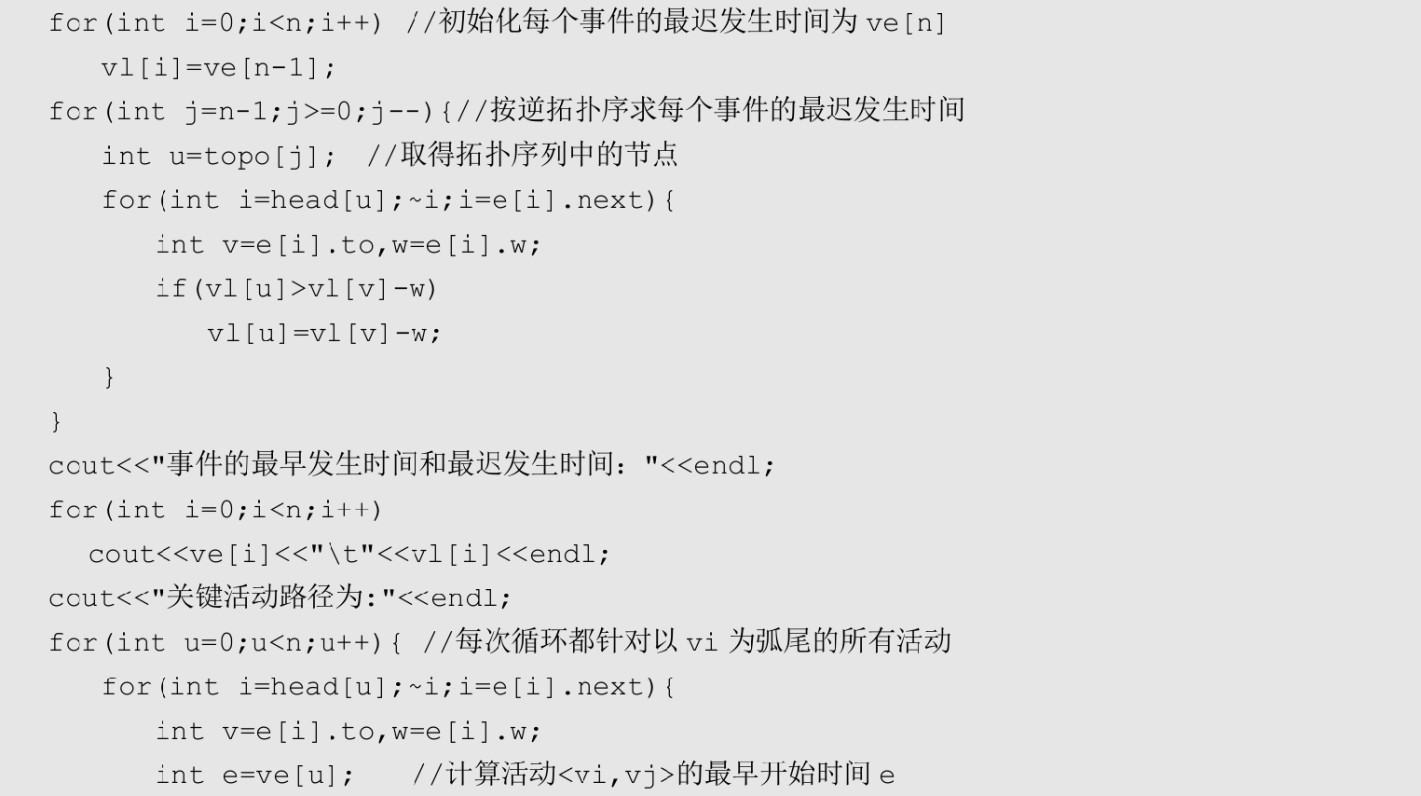
（6）判断活动是否为关键活动。对每个节点i，都用指针p依次指向i的每个邻接点，取得邻接点的序号j=p->v，计算活动<Vi,Vj>的最早发生时间和最迟发生时间，如下图所示，如果e和l相等，则活动<Vi,Vj>为关键活动，即

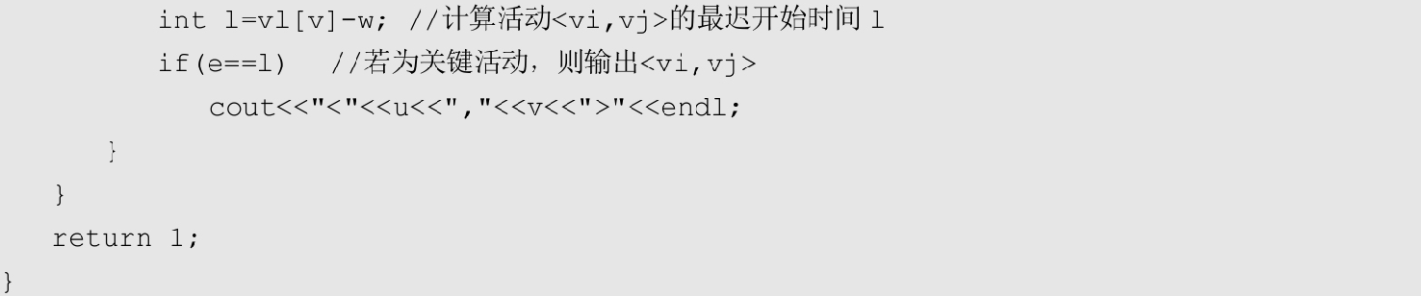


**4. 算法实现**

****

****

****

****

**5. 算法分析**

**时间复杂度：**求事件的最早发生时间和最迟发生时间及活动的最早发生时间和最迟发生时间时，要对所有节点及邻接表进行检查，因此求关键路径算法的时间复杂度为O(n+e)。

**空间复杂度：**算法所需的辅助空间包含拓扑排序算法中的入度数组indegree[]、拓扑序列数组topo[]、栈S及关键路径算法中的ve[]、vl[]、e[]、l[]，算法的空间复杂度是O(n+e)。