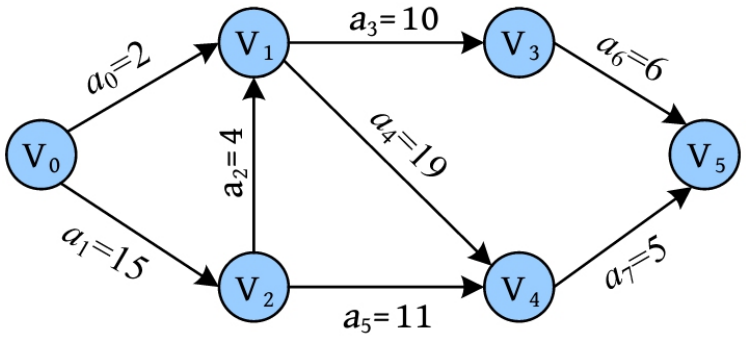
关键路径

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。V_0、V_5 分别代表工程的开始（源点）和结束（汇点），在活动 a_0、a_2 结束后，事件 V_1 才可以开始，在 V_1 结束后，活动 a_3、a_4 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。

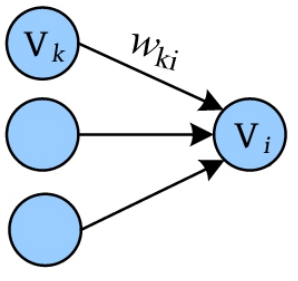
在 AOE 网中，从源点到汇点的带权路径长度最大的路径为关键路径。关键路径上的活动为关键活动。

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。**

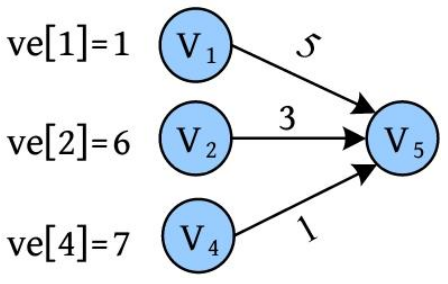
1）事件 V_i 的最早发生时间 ve[i]

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 V_i 的所有入边活动都已完成，V_i 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 V_i 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}，$<V_k,V_i>\in T$。T 为以 V_i 为弧头的弧集合，即 V_i 的入边集合，如下图所示。



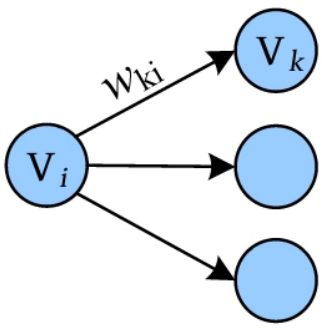
例如，一个 AOE 网如下图所示。已经求出 V_1、V_2、V_4 三个节点的 ve 值，求 V_5 的 ve 值。考察 V_5 的入边，ve[5]=max{ve[1]+5,ve[2]+3,ve[4]+1}=9。



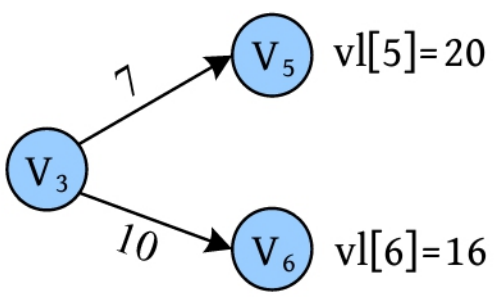
2）事件 V_i 的最迟发生时间 vl[i]

事件 V_i 的最迟发生时间不能影响其所有后继的最迟发生时间。V_i 的最迟发生时间不能大于其后继 V_k 的最迟发生时间减去活动$<V_i,V_k>$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 V_i 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]-w_{ki}}，$<V_k,V_i>\in T$。T 为以 V_i 为弧尾的弧集合，即 V_i 的出边集合。

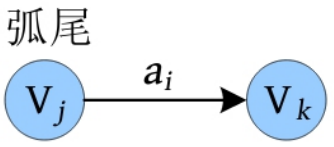


例如，一个 AOE 网如下图所示。已经求出 V_5、V_6 两个节点的 vl 值，求 V_3 的 vl 值。考察 V_3 的出边，vl[3]=min{vl[5]-7,vl[6]-10}=6。

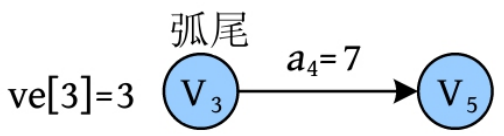


3）活动 $a_i=<V_j,V_k>$ 的最早发生时间 e[i]

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

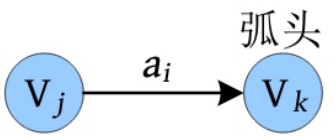


例如，一个 AOE 网如下图所示。已经求出 V_3 节点的 ve 值，求 a_4 的 e 值。a_4 的 e 值等于弧尾 V_3 的 ve 值，e[4]=ve[3]=3。

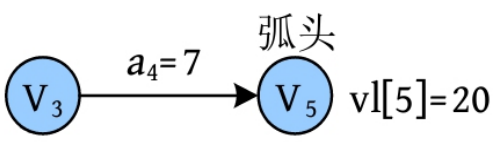


4）活动 $a_i=<V_j,V_k>$ 的最迟发生时间 l[i]

活动 a_i 的最迟发生时间不能耽误事件 V_k 的最迟发生时间，因此活动 a_i 的最迟发生时间等于事件 V_k 的最迟发生时间减去活动 a_i 的持续时间 w_{jk}。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]-w_{jk}。



例如，一个 AOE 网如下图所示。已经求出 V_5 节点的 vl 值，求 a_4 的 l 值。a_4 的 l 值=弧头 V_5 的 vl 值-边值，l[4]= vl[5]-7=20-7=13。



1. 求解过程

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

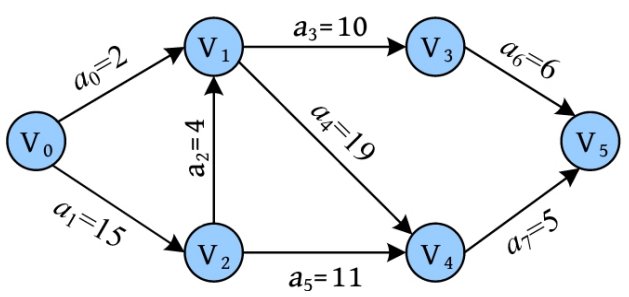
（2）事件 V_i 的最迟发生时间 vl[i]：考察出边，弧头 vl-出边权值的最小值。

（3）活动 a_i 的最早发生时间 e[i]：弧尾的最早发生时间。

（4）活动 a_i 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。



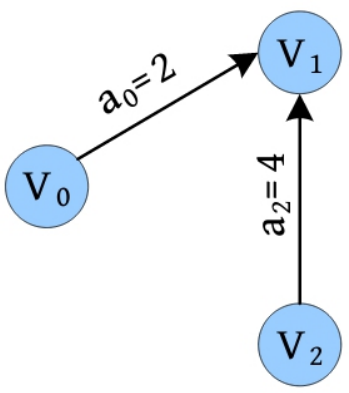
（1）求拓扑排序序列，将其保存在 topo[]数组中。

	0	1	2	3	4	5
topo[]	0	2	1	3	4	5

（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。

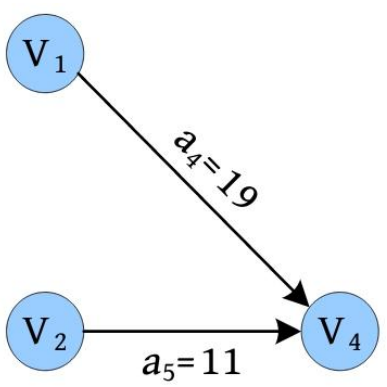
- ve[0]=0。
- ve[2]=ve[0]+15=15。

V_1 有两个入边，弧尾 ve+入边权值，取最大值。



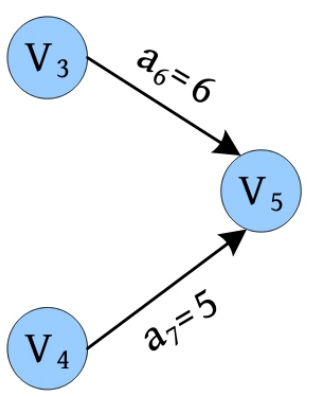
- ve[1]=max{ve[2]+4,ve[0]+2}=19。
- ve[3]=ve[1]+10=29。

V_4 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。

V_5 有两个入边，弧尾 ve+入边权值，取最大值。

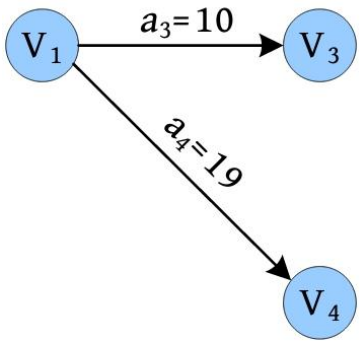


ve[5]=max{ve[4]+5,ve[3]+6}=43。

（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。

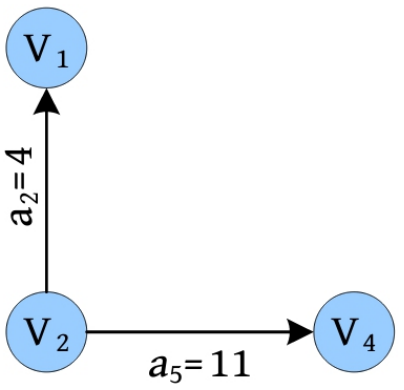
- vl[5]=ve[5]=43。
- vl[4]=vl[5]-5=38。
- vl[3]=vl[5]-6=37。

V_1 有两个出边，弧头 vl-出边权值，取最小值。



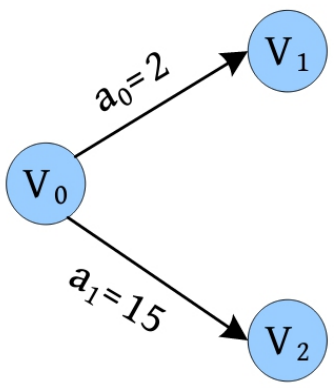
vl[1]=min{vl[4]-19,vl[3]-10}=19。

V_2 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。

V_0 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。

事件	ve[i]	vl[i]
0	0	0
1	19	19
2	15	15
3	29	37
4	38	38
5	43	43

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

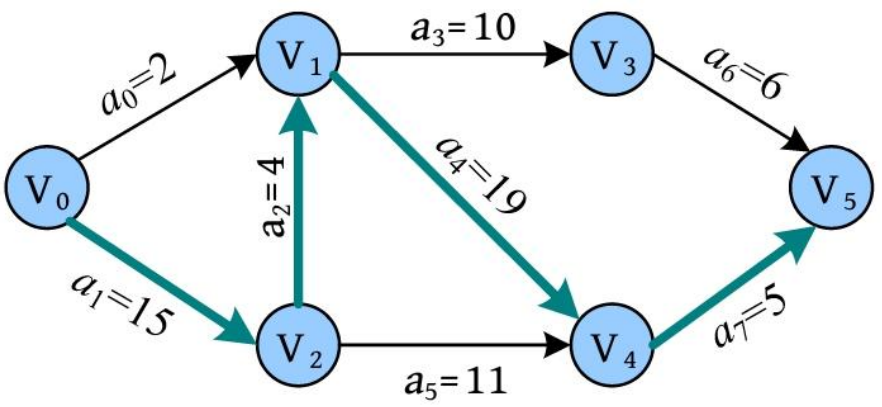
- 活动 a_0=<V_0,V_1>：e[0]=ve[0]=0；l[0]=vl[1]-2=17。
- 活动 a_1=<V_0,V_2>：e[1]=ve[0]=0；l[1]=vl[2]-15=0。
- 活动 a2=<V_2,V_1>：e[2]=ve[2]=15；l[2]=vl[1]-4=15。

- 活动 a_3=<V_1,V_3>：e[3]=ve[1]=19；l[3]=vl[3]-10=27。
- 活动 a_4=<V_1,V_4>：e[4]=ve[1]=19；l[4]=vl[4]-19=19。
- 活动 a_5=<V_2,V_4>：e[5]=ve[2]=15；l[5]=vl[4]-11=27。
- 活动 a_6=<V_3,V_5>：e[6]=ve[3]=29；l[6]=vl[5]-6=37。
- 活动 a_7=<V_4,V_5>：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。

活动	$e[i]$	$l[i]$	关键活动
a_0	0	17	
a_1	0	0	√
a_2	15	15	√
a_3	19	27	
a_4	19	19	√
a_5	15	27	
a_6	28	37	
a_7	38	38	√

（5）由关键活动组成的从源点到汇点的路径为关键路径 V_0-V_2-V_1-V_4-V_5，如下图所示。



3. 算法步骤

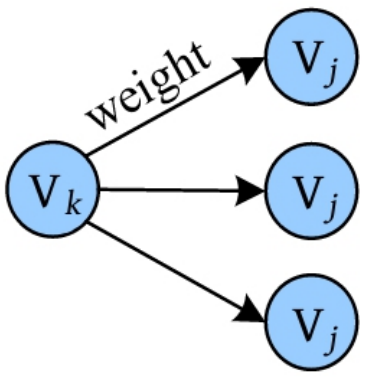
（1）利用拓扑排序算法，将拓扑排序结果保存在 topo[]数组中。

（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

```
if(ve[j]<ve[k]+p->weight)  ve[j]=ve[k]+p->weight
```

相当于求弧尾 ve+入边的最大值，如下图所示。



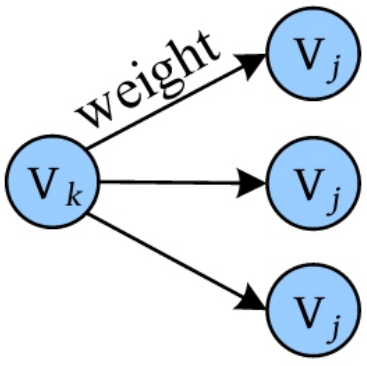
这里的程序处理并不是一下子考察所有入边，但效果是一样的，想一想为什么？

（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

```
if(vl[k]>vl[j]-p->weight)  vl[k]=vl[j]-p->weight
```

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<V_i,V_j>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<V_i,V_j>为关键活动，即

```
e=ve[i];  l=vl[j]-p->weight
```

4. 算法实现

```
bool CriticalPath(){//关键路径
   if(TopoSort()){
      cout<<"拓扑序列为："<<endl;
      for(int i=0;i<n;i++)//输出拓扑序列
         cout<<topo[i]<<"\t";
      cout<<endl;
   }
   else{
      cout<<"该图有环，无拓扑序列！"<<endl;
      return 0;
   }
   for(int i=0;i<n;i++)//初始化最早发生时间为 0
      ve[i]=0;
   //按拓扑次序求每个事件的最早发生时间
   for(int j=0;j<n;j++){
      int u=topo[j];  //取得拓扑序列中的节点
      for(int i=head[u];~i;i=e[i].next){
         int v=e[i].to,w=e[i].w;
         if(ve[v]<ve[u]+w)
            ve[v]=ve[u]+w;
      }
   }
   for(int i=0;i<n;i++) //初始化每个事件的最迟发生时间为 ve[n]
      vl[i]=ve[n-1];
   for(int j=n-1;j>=0;j--){//按逆拓扑序求每个事件的最迟发生时间
      int u=topo[j];  //取得拓扑序列中的节点
      for(int i=head[u];~i;i=e[i].next){
         int v=e[i].to,w=e[i].w;
         if(vl[u]>vl[v]-w)
            vl[u]=vl[v]-w;
      }
   }
   ccut<<"事件的最早发生时间和最迟发生时间："<<endl;
   for(int i=0;i<n;i++)
     cout<<ve[i]<<"\t"<<vl[i]<<endl;
   ccut<<"关键活动路径为:"<<endl;
   for(int u=0;u<n;u++){ //每次循环都针对以 vi 为弧尾的所有活动
      for(int i=head[u];~i;i=e[i].next){
         int v=e[i].to,w=e[i].w;
         int e=ve[u];    //计算活动<vi,vj>的最早开始时间 e
         int l=vl[v]-w; //计算活动<vi,vj>的最迟开始时间 l
         if(e==l)   //若为关键活动，则输出<vi,vj>
            cout<<"<"<<u<<","<<v<<">"<<endl;
      }
   }
   return 1;
}
```

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)。