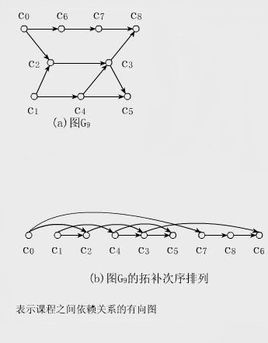
# 拓扑排序与关键路径：

## 概念

给你一个DAG(有向无环图) 进行拓扑排序，是将G中所有顶点排成一个线性序列，使得图中任意一对顶点u和v，若边(u,v)∈E(G)，则u在线性序列中出现在v之前。通常，这样的线性序列称为满足拓扑次序(Topological Order)的序列，简称**拓扑序列**。简单的说，由某个集合上的一个[偏序](https://baike.baidu.com/item/%E5%81%8F%E5%BA%8F/2439087)得到该集合上的一个[全序](https://baike.baidu.com/item/%E5%85%A8%E5%BA%8F/10577699)，这个操作称之为拓扑排序。 拓扑排序是对点来说的，输出是一组拓扑序的点。



**AOV网：**是顶点表示活动(点权代表活动时间)，用弧表示活动之间的关系的DAG。注意AOV网边没有权，它仅仅用方向来代表关系。

**AOE网：**是顶点表示事件(活动之间的关系)，用弧表示活动(边权代表活动时间)的DAG。注意AOE网是带边权的，但没有点权。

在AOE网里，如果只有：1个入度为0的点(源点S)和1个出度为0的点(汇点T)

定义S到T的最长路径为**关键路径**，关键路径上的活动(边)叫**关键活动。**

**关键路径的实际意义**：整个工程完成需要的时间是最长路径长度，而关键路径上的活动时间(边权)会影响工程进度(增大或减小总时间)

定义如下概念：

|  |  |  |
| --- | --- | --- |
| 事件(顶点) | 最早发生时间ve(u) | 最晚发生时间vl(u) |
| 活动(边) w(u,v) | 最早开始时间e(u,v) | 最晚开始时间l(u,v) |

要想求得边的l(u,v)与e(u,v)，需要先求点的ve(u)与vl(u).

先对AOE网拓扑排序，再进行操作,。

**求ve:** ve(S)=0, ve(u)=max{ve(v)+prew(u,v)}

表述：原点的最早发生时间是0，其他点的ve,是和自己前驱点的边权，取最大。

**求vl:** vl(T)=ve(T), vl (u)=min{ vl (v)+nextw(u,v)}

表述：汇点最晚发生时间和ve相同，其他点的vl,是和自己后继点的边权，取最小。

**求e(u,v):** e(u,v)=ve(u) //一条边的最早开始时间，是有向边起点的ve值

**求l(u,v) :** l(u,v)=vl(v)-w(u,v) //一条边的最晚开始时间，是有向边末点的ve值 减去 边权。

* 对于一个活动(边)，完成它的**剩余时间**是：l(u,v)-e(u,v)， 特别的，对于所有关键活动l(u,v)-e(u,v)==0，即：e(u,v)==l(u,v)
* **工程总时间**就是ve(T)
* **关键路径**是最长路径，用拓扑排序即可求出相关的

## Kahn算法:

**普通拓扑排序**

显然拓扑排序的顺序是每次取入度是0的点，然后删除与其相邻的边。

Kahn的算法的思路其实就是，手动模拟的拓扑排序，我们先使用一个栈保存入度为0 的顶点，然后输出栈顶元素并且将和栈顶元素有关的边删除(代码中不必真的删除边，而是改变其他点入度)，减少和栈顶元素有关的顶点的入度数量并且把入度减少到0的顶点也入栈。

**关键路径**

找关键路径和求l(u,v)和vl(u)都是要从后往前找的，所以要进行正向反向拓扑排序各一次。正向拓扑排序如前面介绍，反向拓扑排序就是记录每个点出度，和正向类似，只是从出度为0的点开始入栈，每次”删去”前驱边即可。

至于如何求关键路径，如果学过迪杰斯特拉算法就很容易想到了.就是维护数组dist[]和next[]，dist[u]表示：点u到达汇点T的最长路径的长度。next[u]==0表示这个节点不是关键路径的节点，next[u]>0表示下一个节点的编号。

## Dfs搜索法:

由于图是有向无环的，显然可以以每个入度是0的点开始dfs,搜到一个点如果他入度是0，和 Kahn很像就删除边(代码中不必真的删除边，而是改变其他点入度) ，dfs如果判断环路要比Kahn麻烦。但是dfs寻找关键路径和相关属性(e,l,ve,vl)的代码比kahn简单，此时考虑dfs法。

* 拓扑排序一定要考虑重边自环。因为这会影响度数。

# 模型和应用

## 权值传播：

Hihocoder1175:给定图，图每个点有权值代表病毒，病毒会沿着有向图传播问传播完成后病毒总数，用拓扑序把每访问到点的权值向前加和。

## 有向无环图路径计数：

就是找入度为0的点按照Kahn算法的步骤，并且像权值传播那样，往前传播路径数量。起点权值初始化是1其余点都是0。