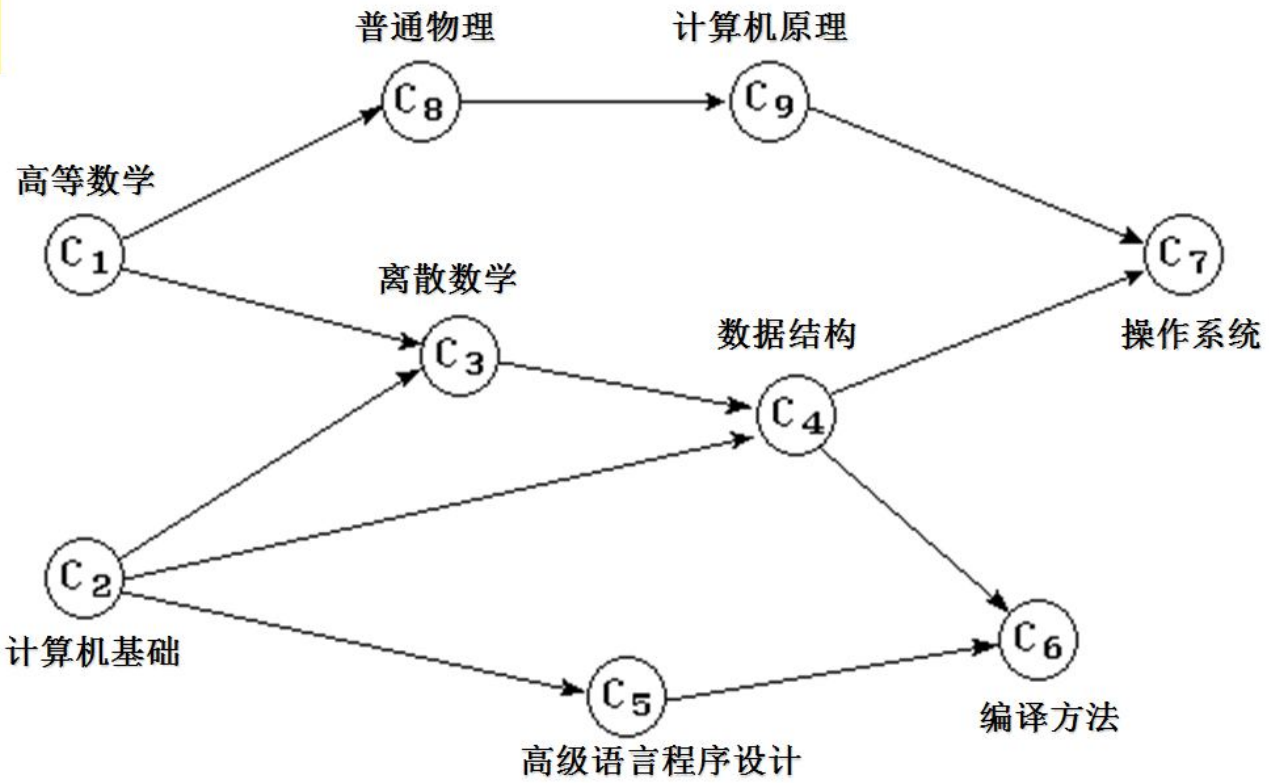
7.5 有向无环图及其应用--拓扑排序

- 研究一类有向图-- **AOV网**：用顶点表示活动，弧表示活动之间的制约（优先）关系。
- AOV网代表一项工程，工程划分成若干活动，每个活动用一个顶点表示，活动之间的先后制约关系用弧表示

课程代号	课程名称	先修课程
C_1	高等数学	
C_2	计算机基础	
C_3	离散数学	C_1, C_2
C_4	数据结构	C_3, C_2
C_5	高级语言程序设计	C_2
C_6	编译方法	C_5, C_4
C_7	操作系统	C_4, C_9
C_8	普通物理	C_1
C_9	计算机原理	C_8

例如：计算机专业的学生培养划分成9个活动（9门课程学习）。9个活动均完成（9门课都修完），工程结束，毕业，发文凭

AOV网示例

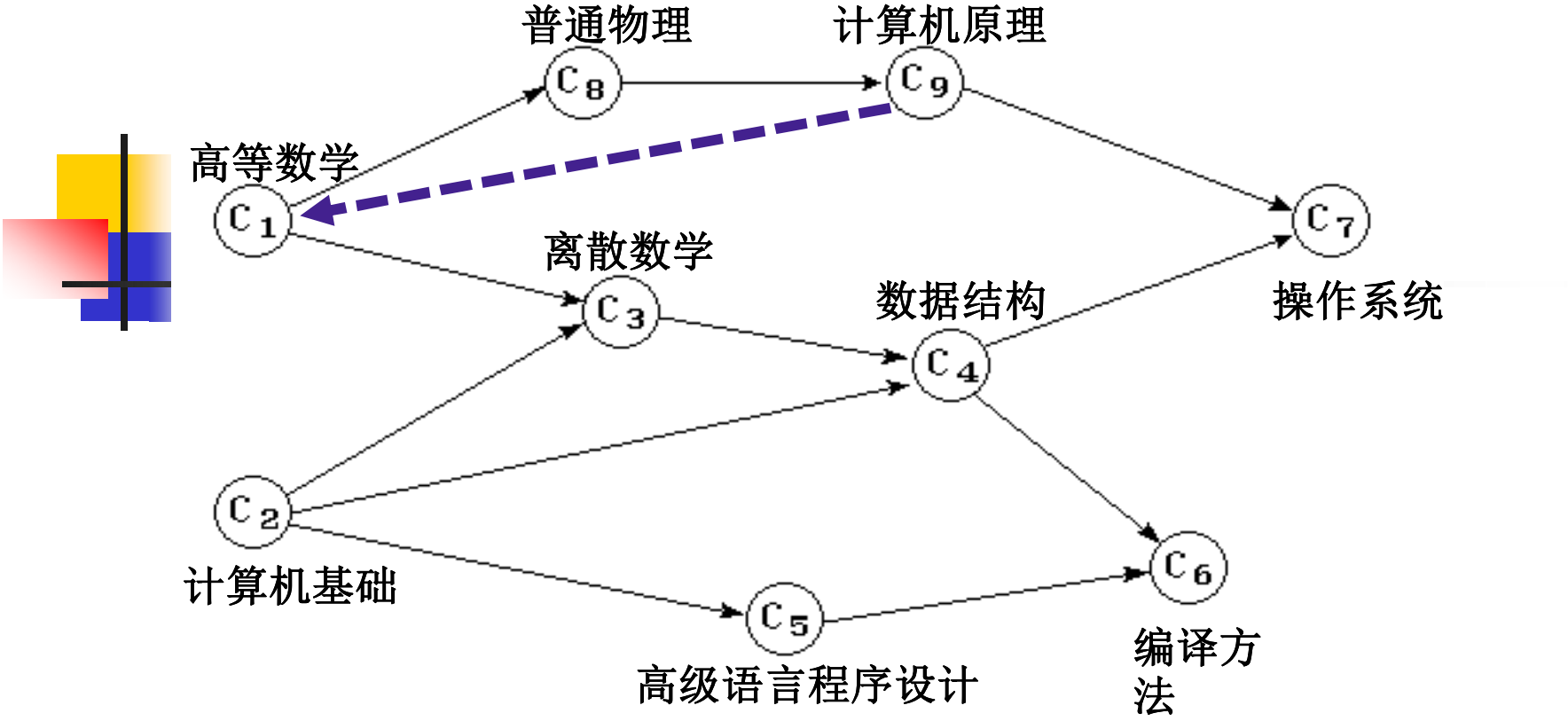


学生培养工程图（AOV网）

主要目的：1.判断一个有向图是否包含回路

2.构造拓扑排序：安排活动的进行顺序

AOV网示例



该有向图包含回路（$c_1c_8c_9c_1$）则不是一个AOV网，该学生培养工程设计不合理

主要目的：1.判断一个有向图是否包含回路

2.构造拓扑排序：安排活动的进行顺序

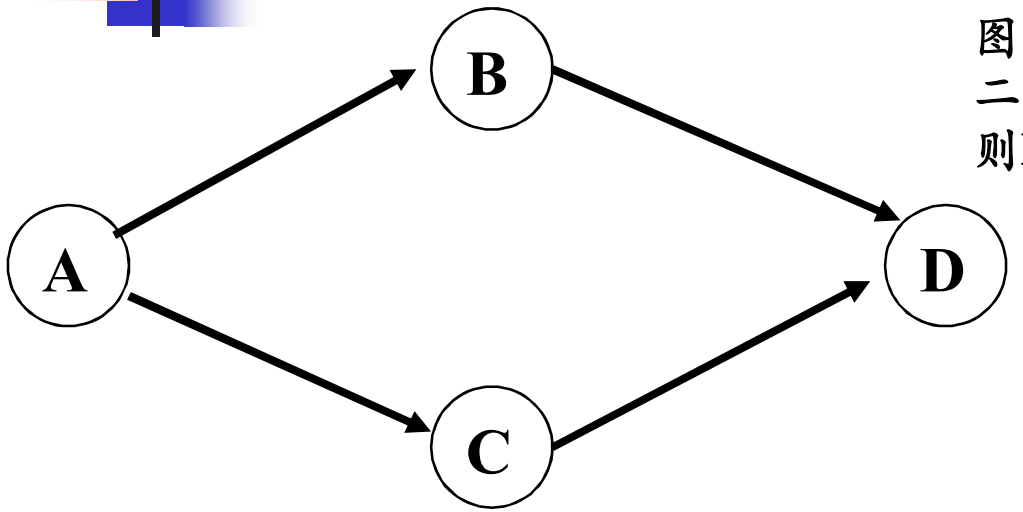
7.5 有向无环图及其应用--拓扑排序

- 测试AOV网是否具有回路的方法，就是在AOV网的偏序集合下构造一个线性序列，该线性序列具有以下性质：
  - 若顶点*i*优先于*j*，则在线性序列中*i*仍然优先于*j*；
  - 对于网中原来没有优先关系的顶点*i*与顶点*j*，在线性序列中也建立一个先后关系，或者*i*优先于*j*，或者*j*优先于*i*。
- 满足这样性质的线性序列称为**拓扑有序序列**。构造拓扑序列的过程称为拓扑排序。也可以说拓扑排序就是由某个集合上的一个偏序得到该集合上的一个全序的操作。

7.5 有向无环图及其应用--拓扑排序

- 若集合A中的二元关系R是**自反的、非对称的和传递的**，则R是A上的**偏序关系**。集合A与关系R一起称为一个**偏序集合**。
- 若R是集合A上的一个偏序关系，如果对每个a、b∈A必有aRb或bRa，则R是A上的**全序**关系。集合A与关系R一起称为一个全序集合。

7.5 有向无环图及其应用--拓扑排序



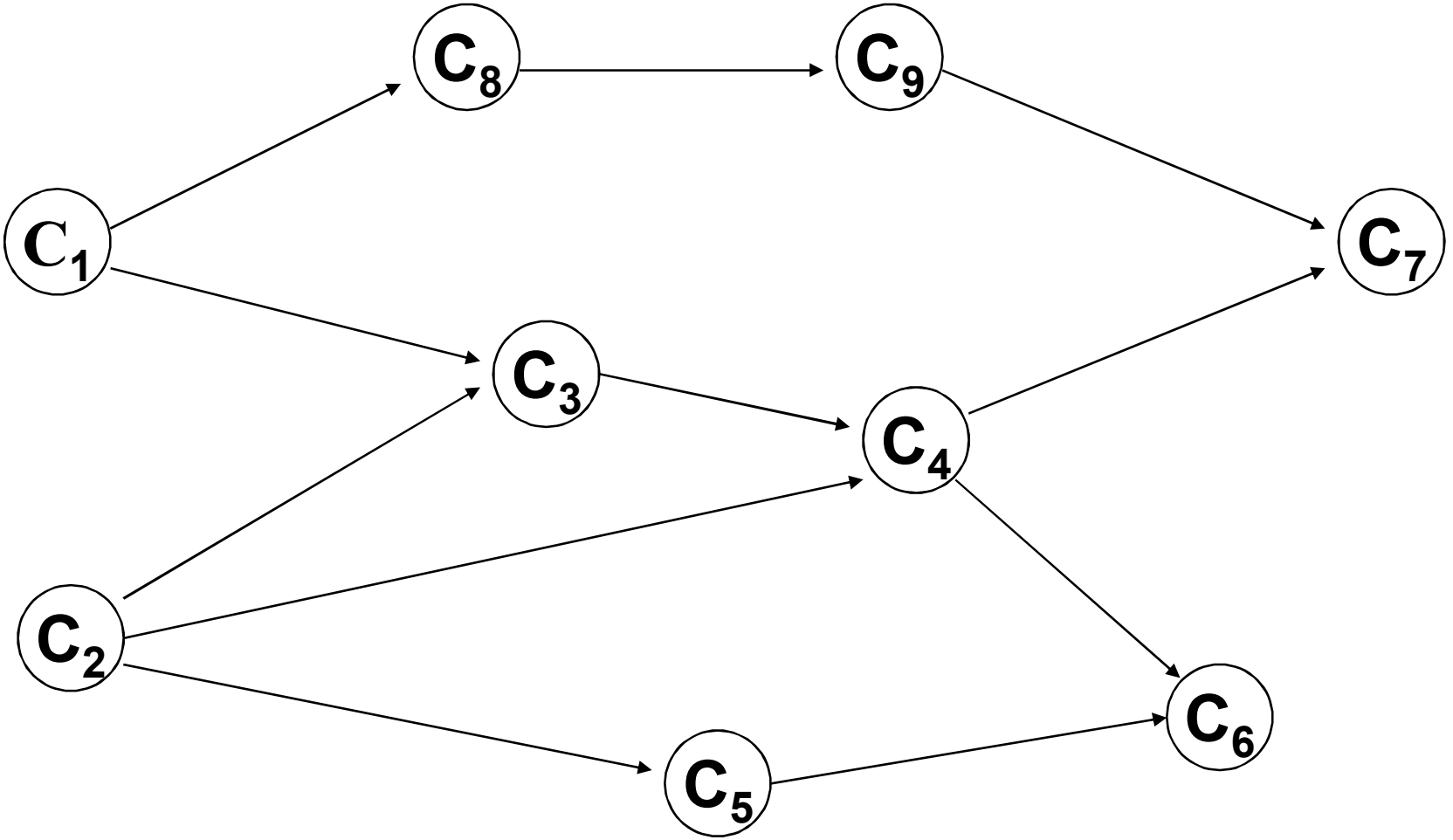
图中弧<A,B>代表的关系R为A<B,则此图的二元关系R是自反的、非对称的和传递的，则R是的偏序关系。

ABCD
ACBD

全序关系—拓扑排序序列

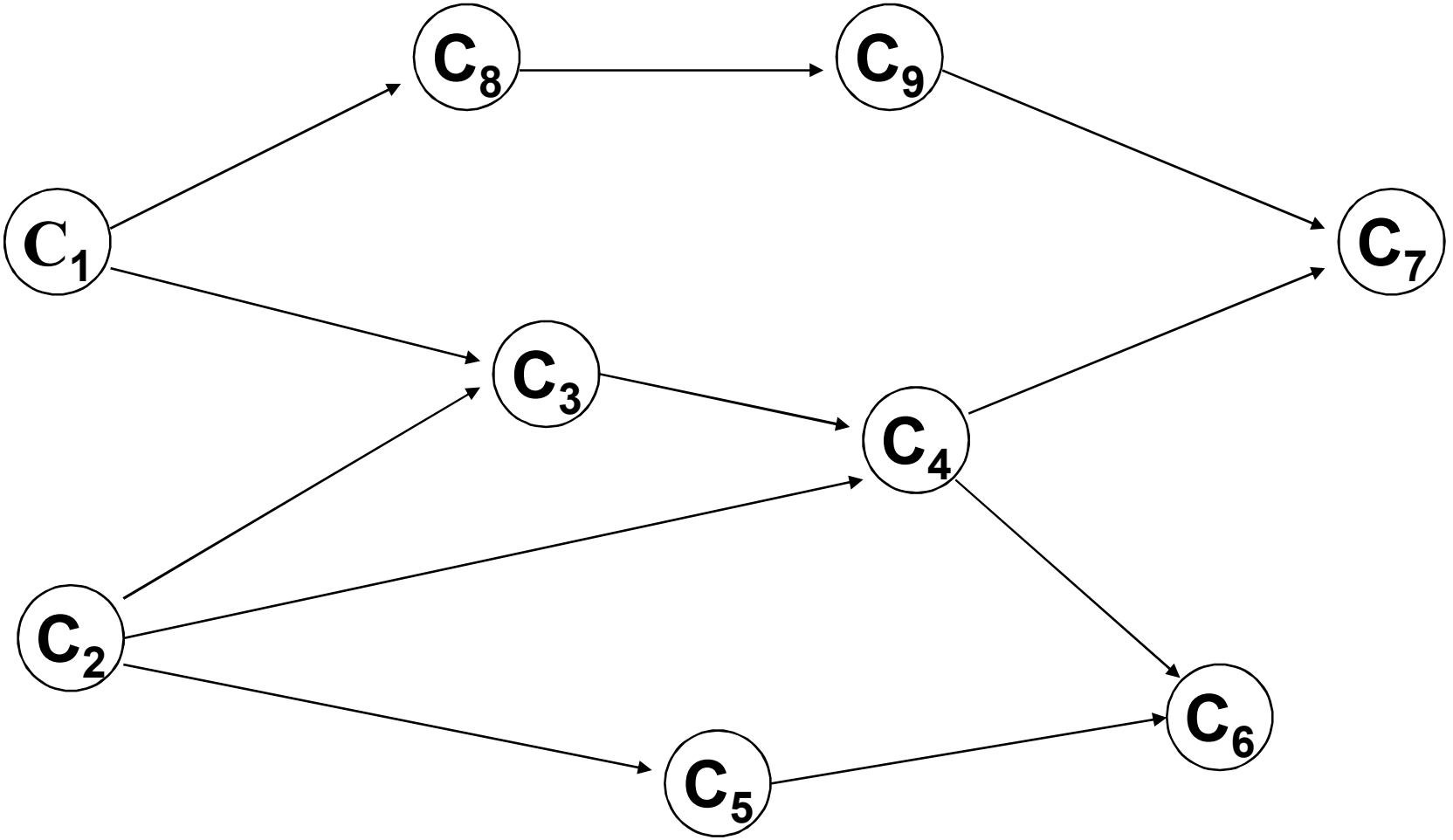
如何构造拓扑排序序列

找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



如何构造拓扑排序序列

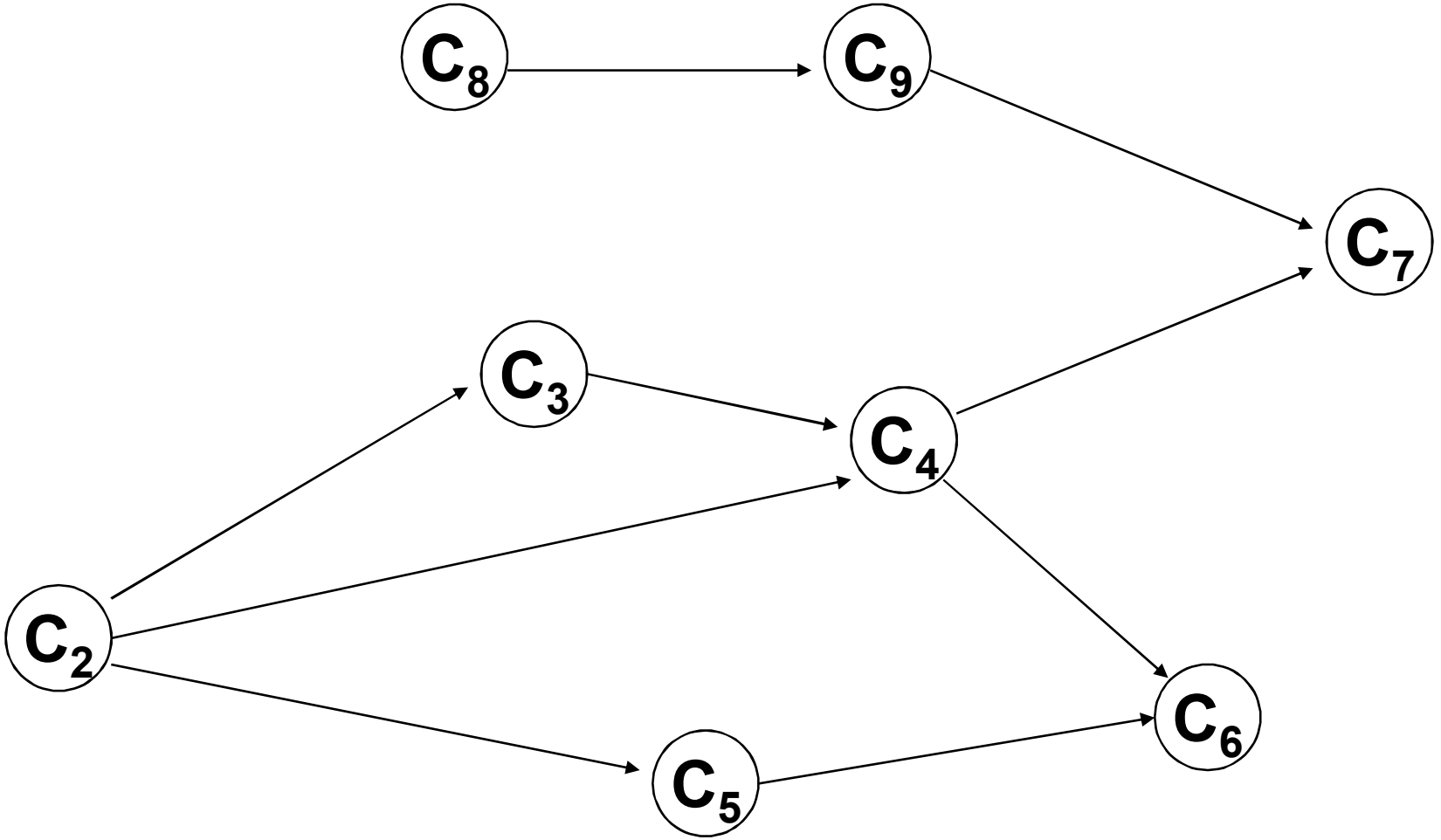
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1

如何构造拓扑排序序列

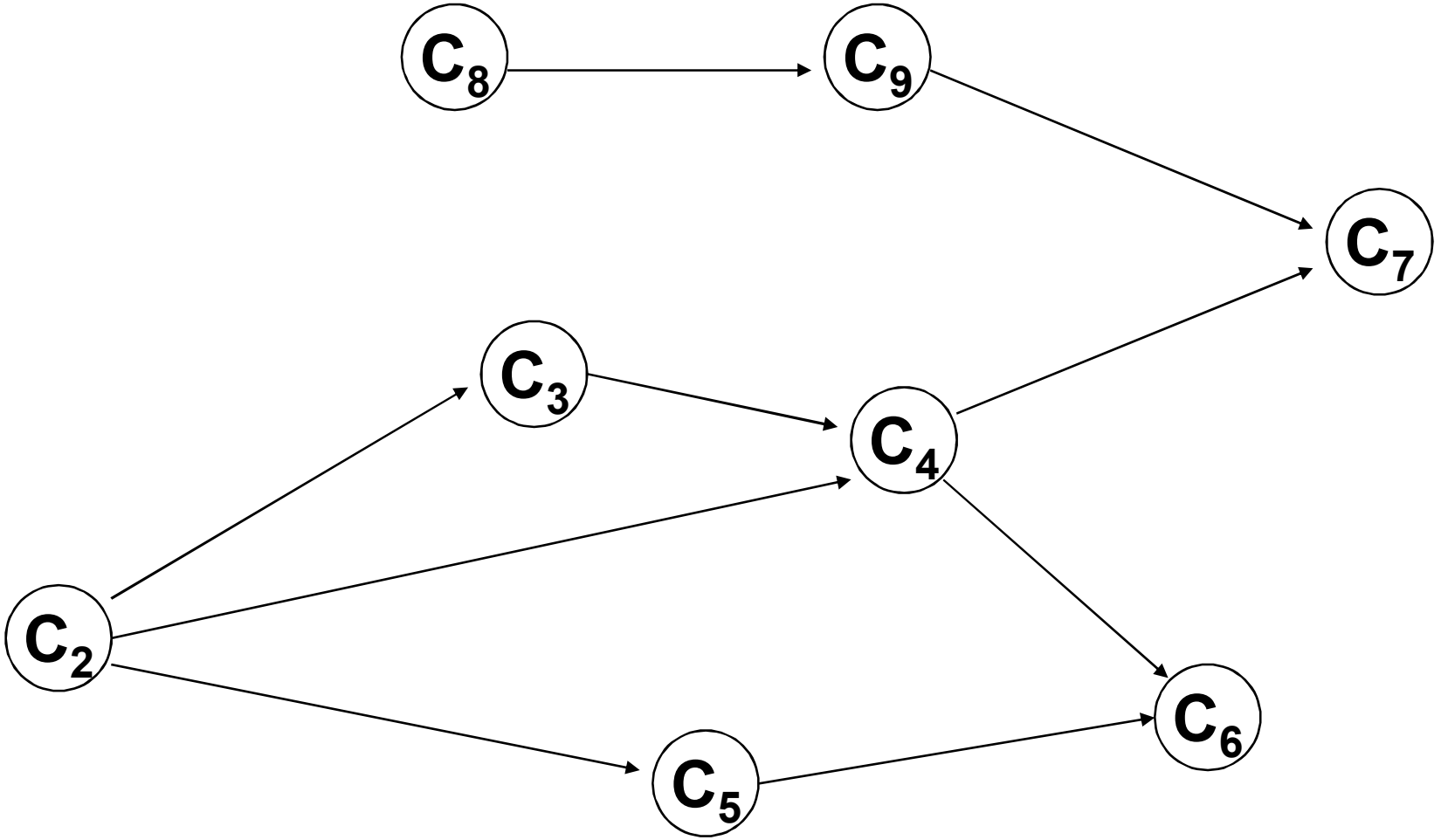
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1

如何构造拓扑排序序列

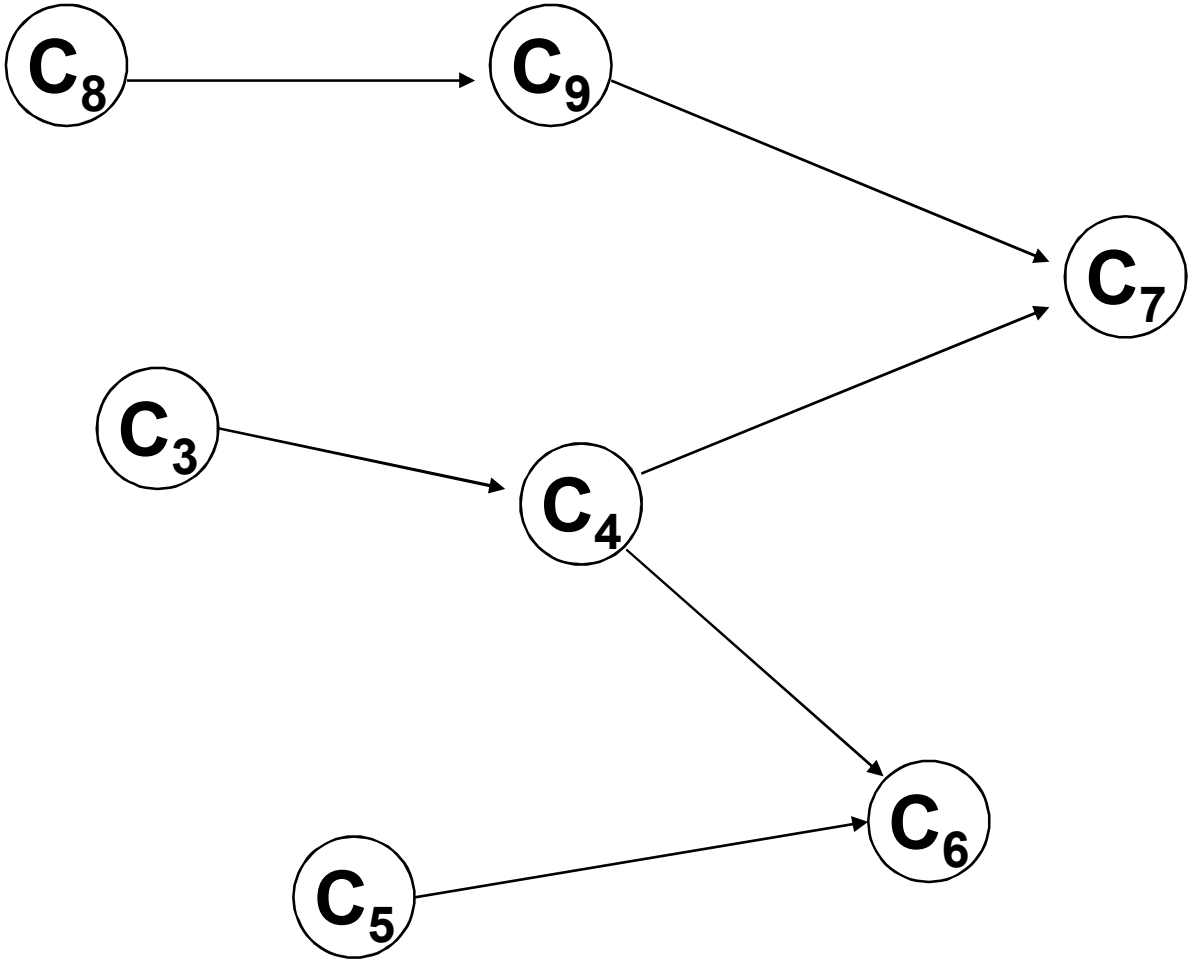
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2

如何构造拓扑排序序列

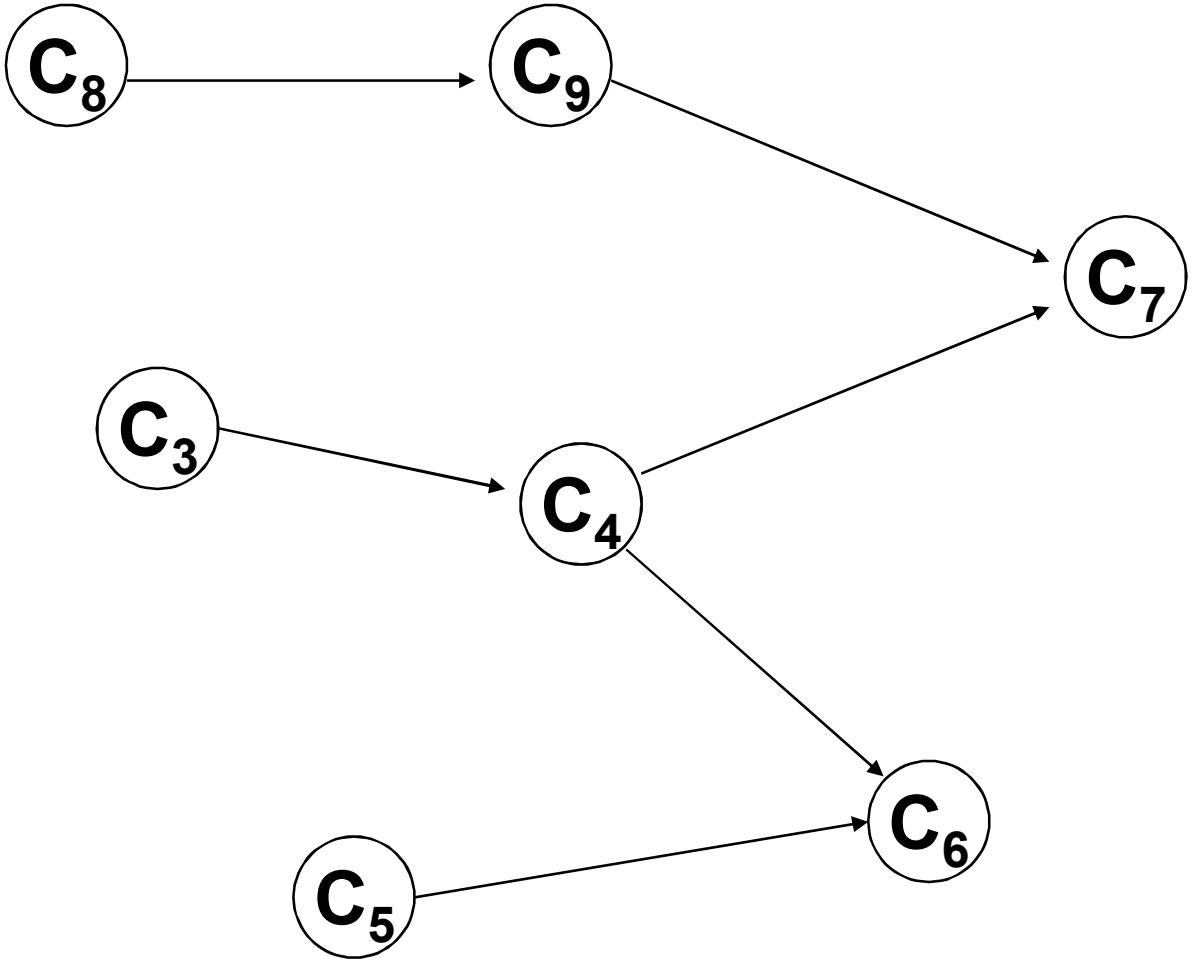
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2

如何构造拓扑排序序列

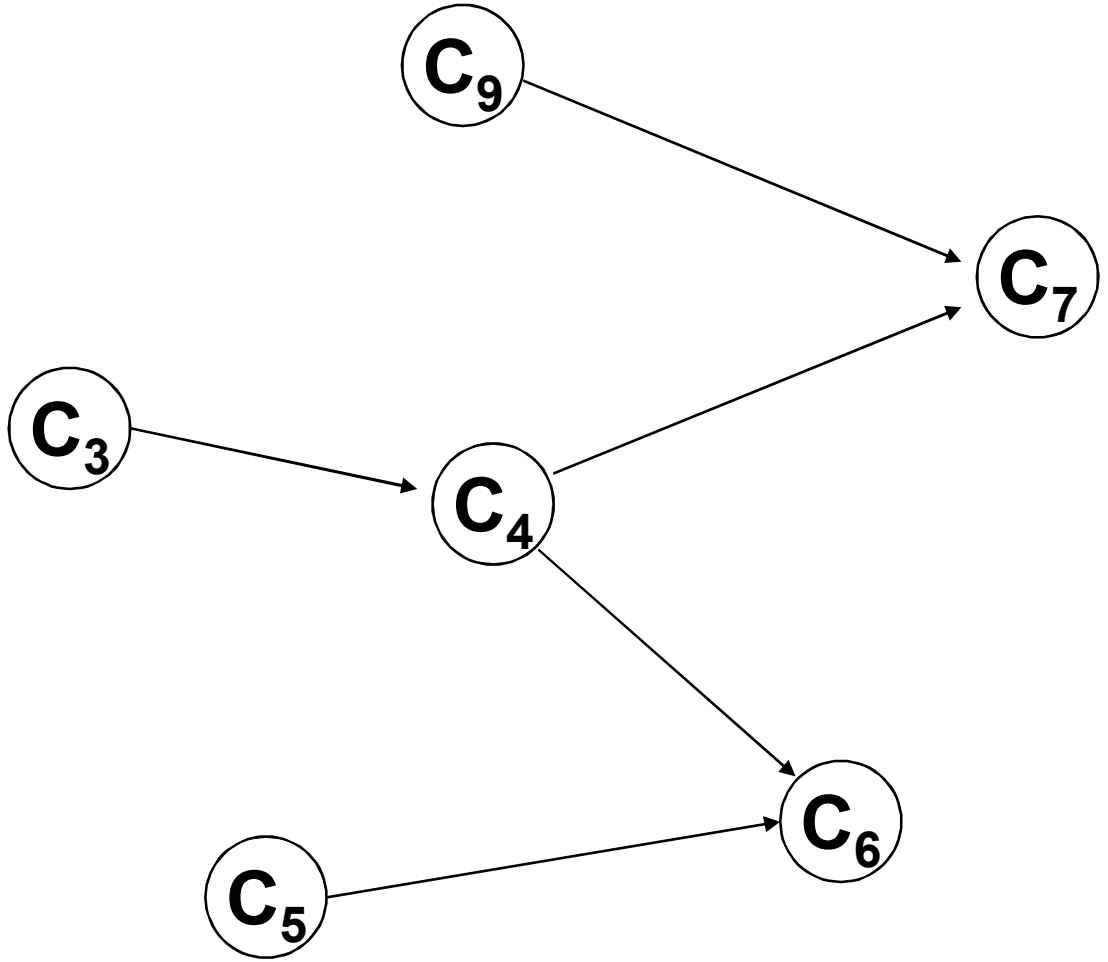
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8

如何构造拓扑排序序列

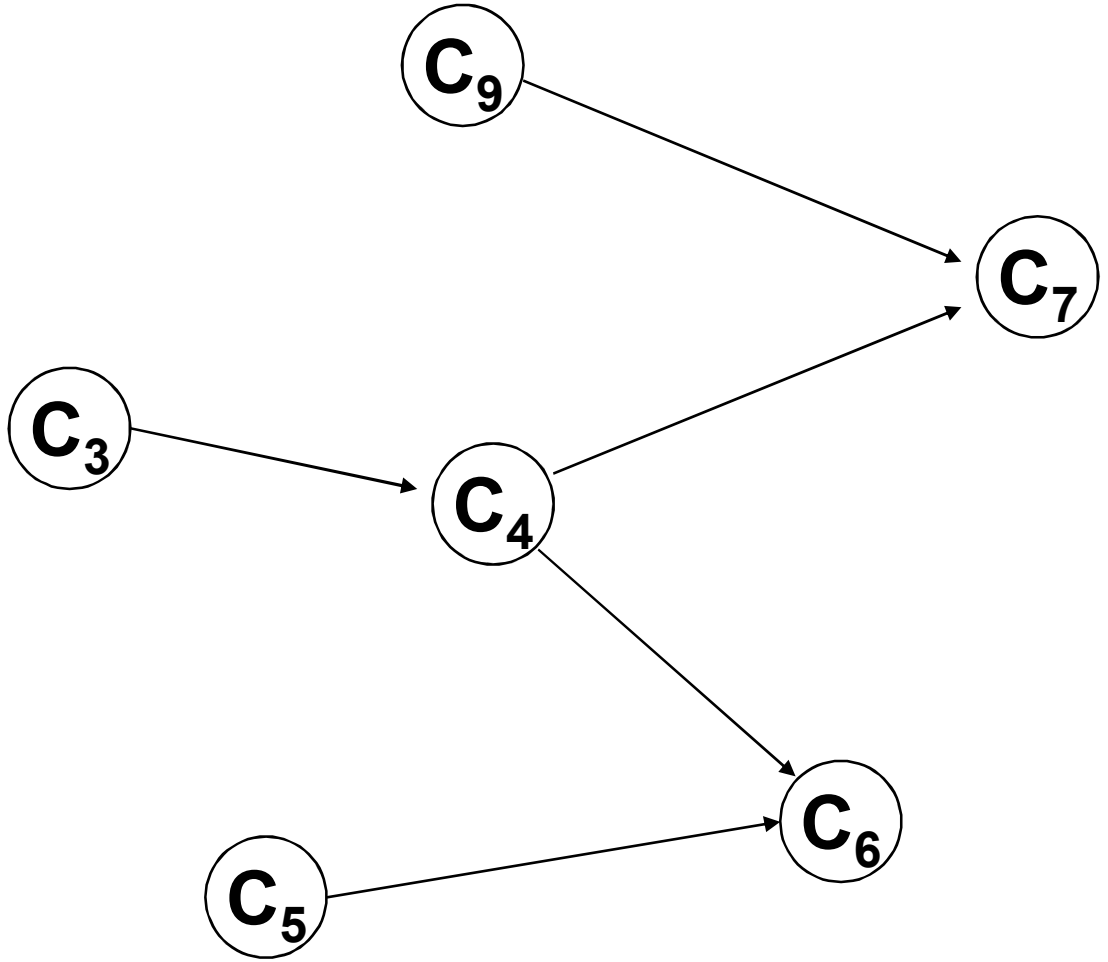
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8

如何构造拓扑排序序列

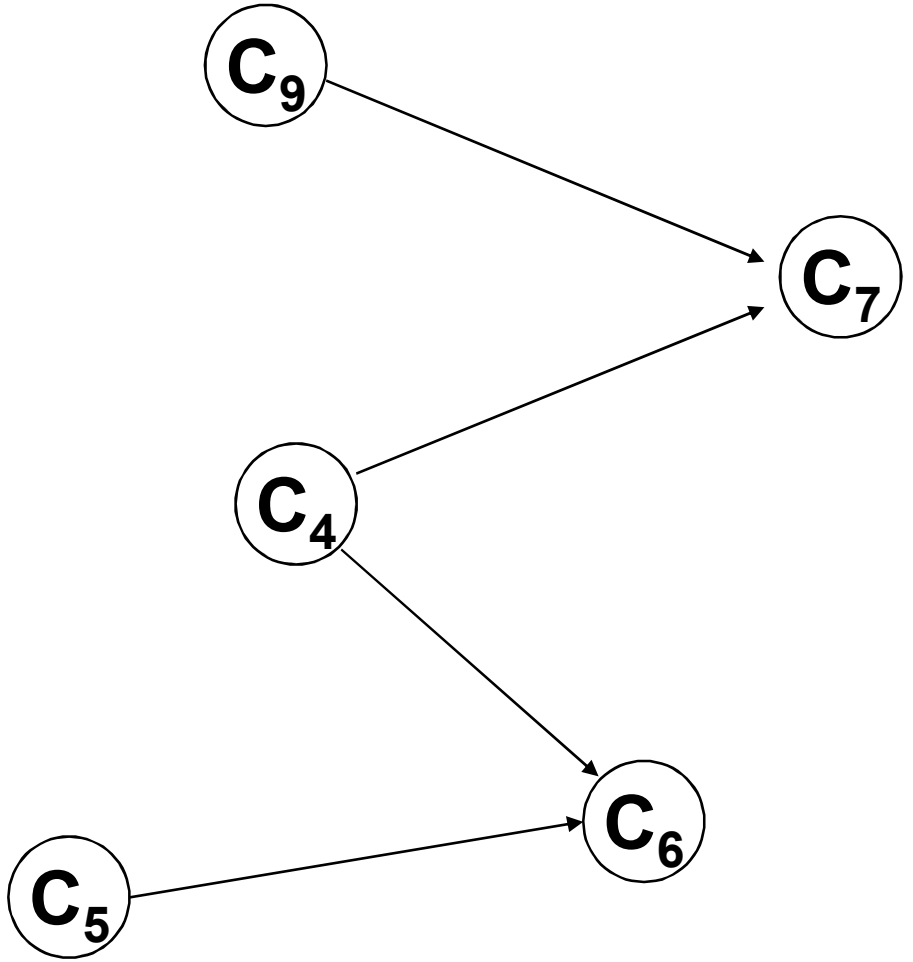
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8　　C3

如何构造拓扑排序序列

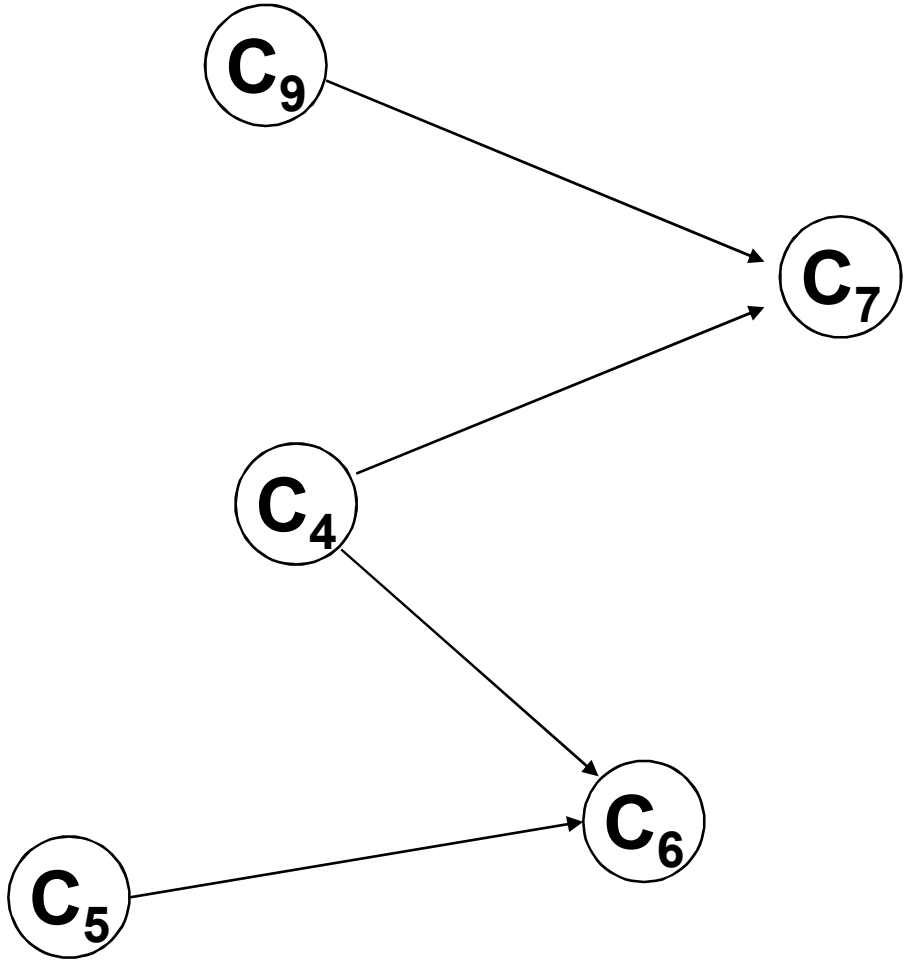
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8　　C3

如何构造拓扑排序序列

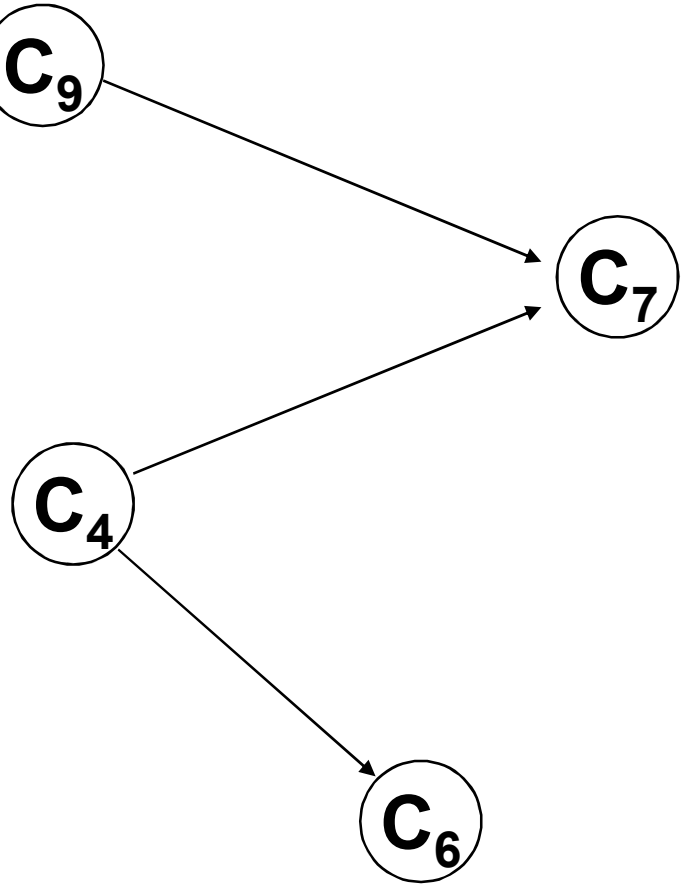
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8　　C3　　C5

如何构造拓扑排序序列

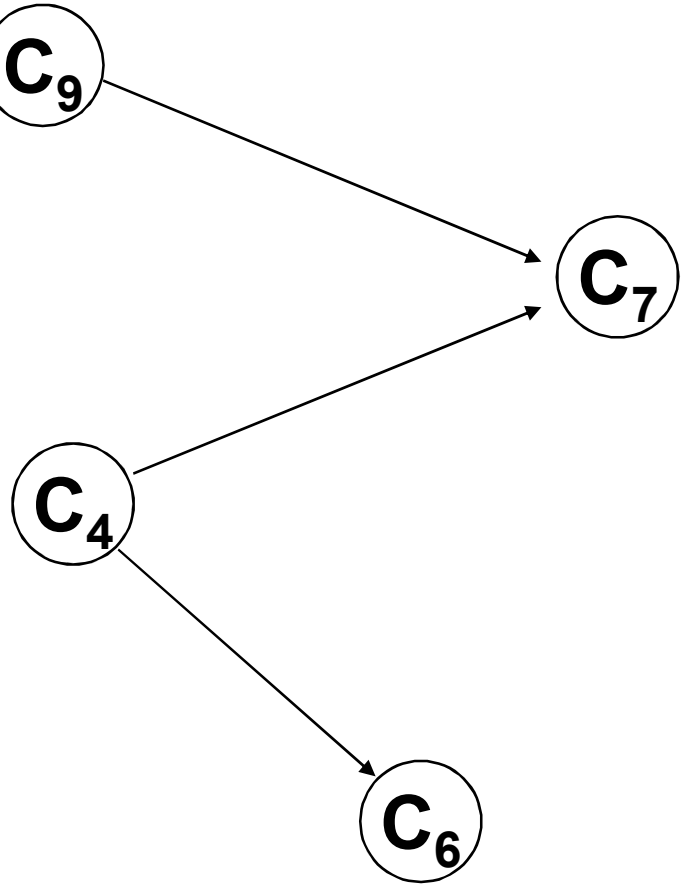
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8　　C3　　C5

如何构造拓扑排序序列

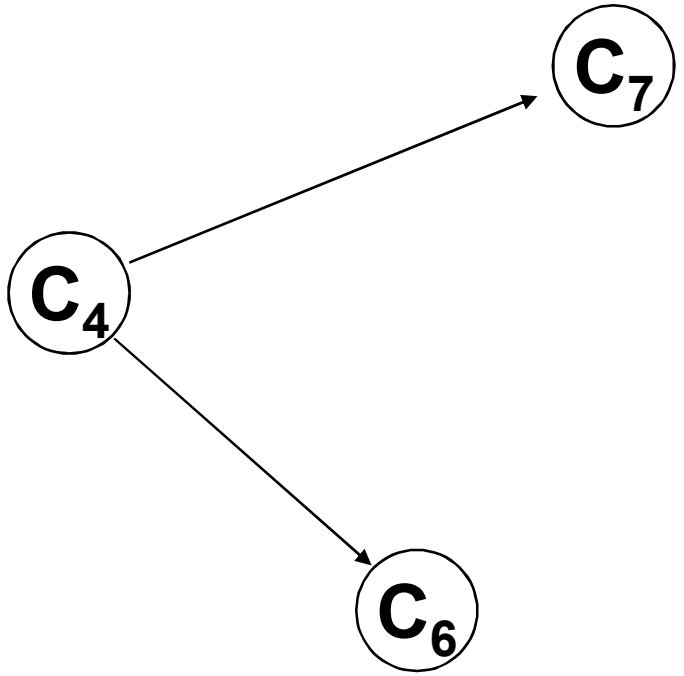
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8　　C3　　C5　　C9

如何构造拓扑排序序列

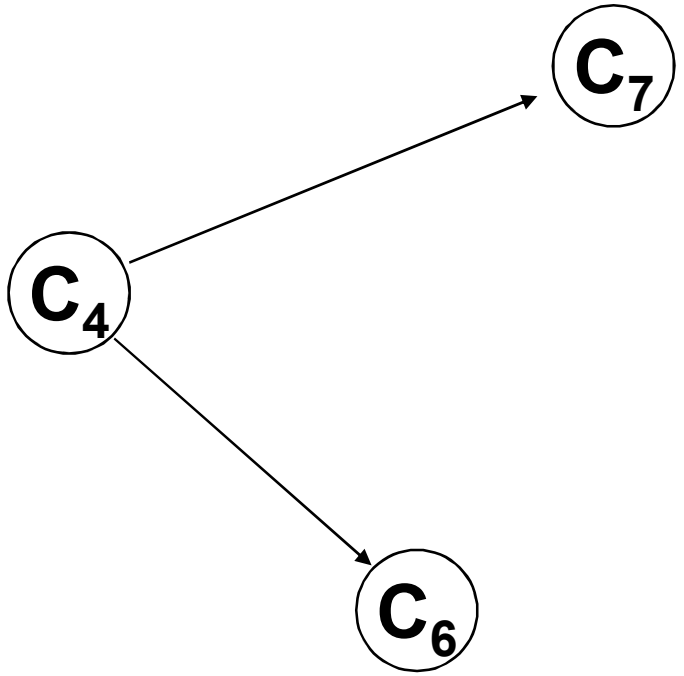
找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8　　C3　　C5　　C9

如何构造拓扑排序序列

找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中



C1　　C2　　C8　　C3　　C5　　C9　　C4

如何构造拓扑排序序列

找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中

C1　　C2　　C8　　C3　　C5　　C9　　C4

如何构造拓扑排序序列

找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中

C1 C2 C8 C3 C5 C9 C4 C6

如何构造拓扑排序序列

找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中

C1　　C2　　C8　　C3　　C5　　C9　　C4　　C6

如何构造拓扑排序序列

找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中

C_7

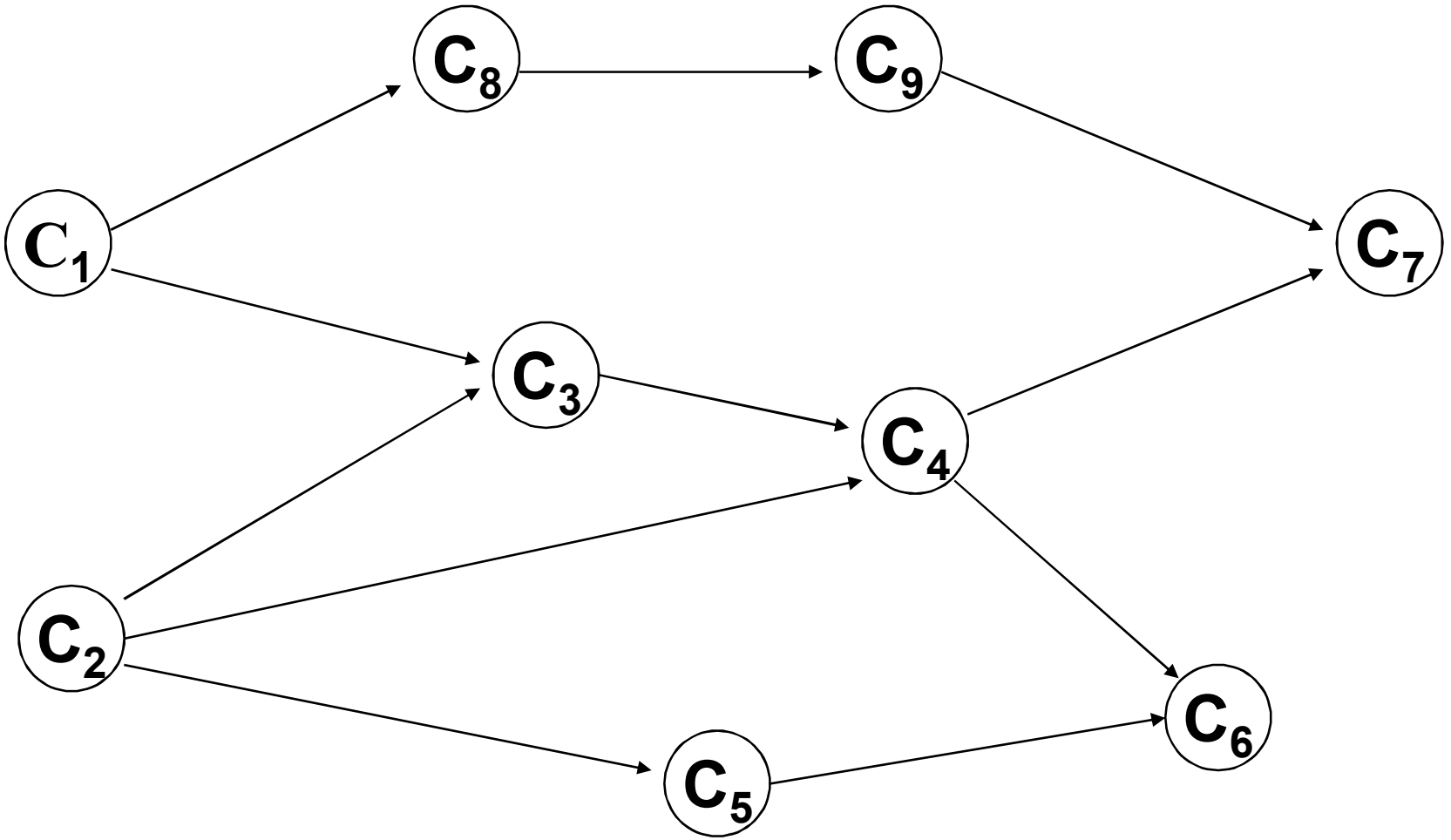
C1 C2 C8 C3 C5 C9 C4 C6 C7

如何构造拓扑排序序列

找当前图中入度为0的顶点（没有制约活动，或其制约活动已经安排进行）放到拓扑排序序列中

C1 C2 C8 C3 C5 C9 C4 C6 C7

C1 C2 C8 C3 C5 C9 C4 C6 C7

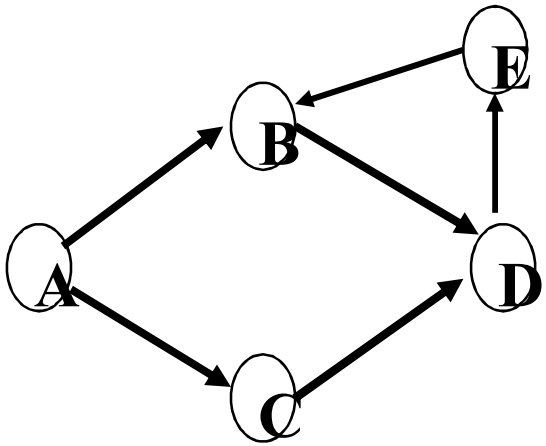


C1 C2 C3 C4 C5 C6 C8 C9 C7

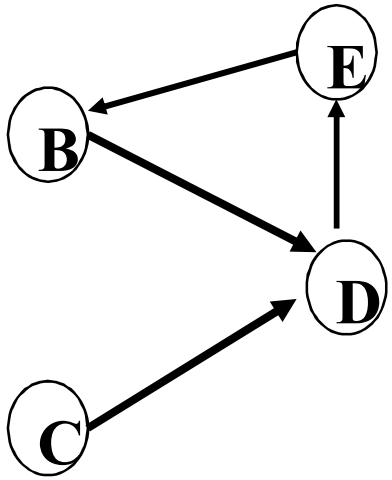
一个AOV网的拓扑排序序列可能存在多个，不唯一

拓扑排序算法

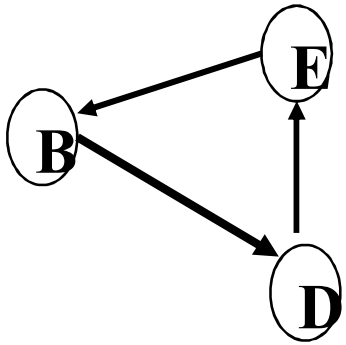
- 1. 选一个入度为0的顶点,输出;
- 2. 删除该顶点以及由它出发的所有边
- 3. 重复1,2, 直到全部顶点输出或不存在入度为0的顶点;
- 4.若图中还有剩余顶点未被删除，说明图中有回路，不是一个AOV网。



图



拓扑序列：A



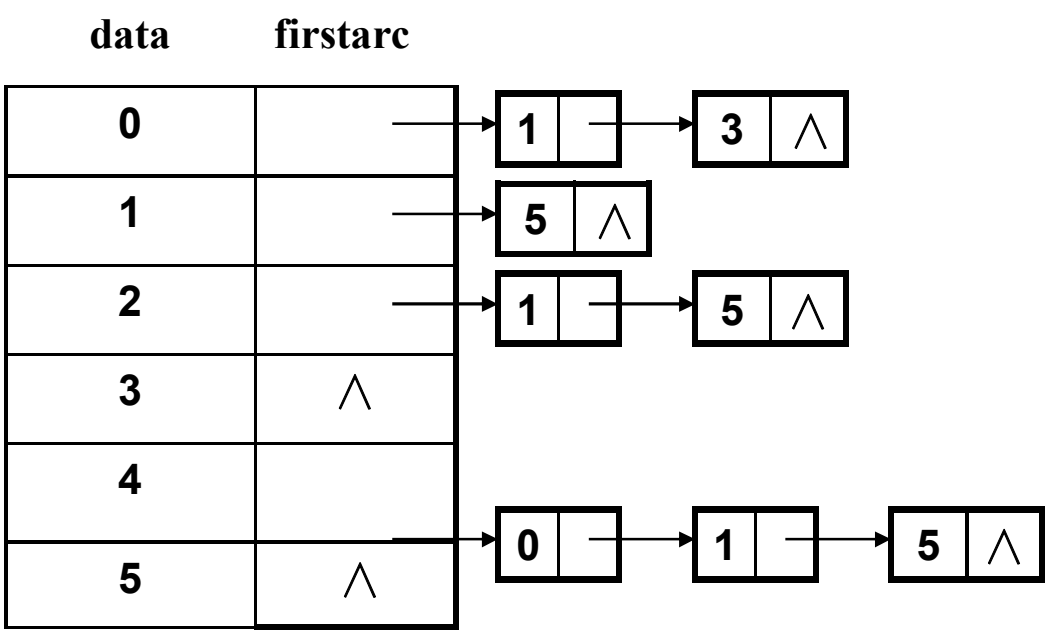
拓扑序列：AC

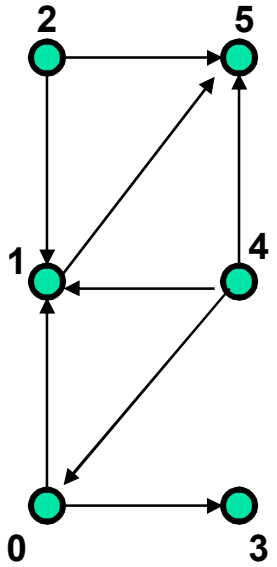
图中不再存在入度为0的顶点，且图中有剩余顶点没有放到拓扑排序序列说明图中有回路

拓扑排序示例

拓扑排序算法

- 图采用邻接表存放；
- 计算所有顶点的入度，存放于一维数组中；

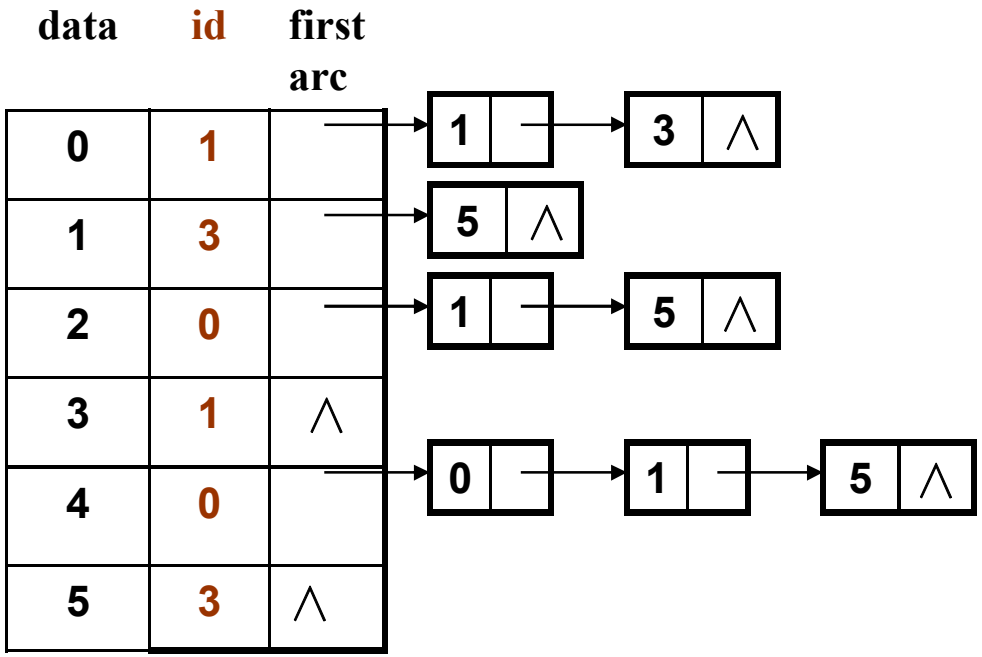


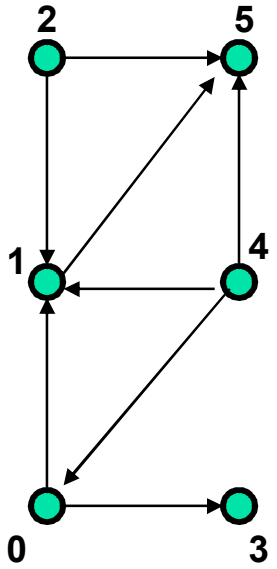


拓扑排序算法

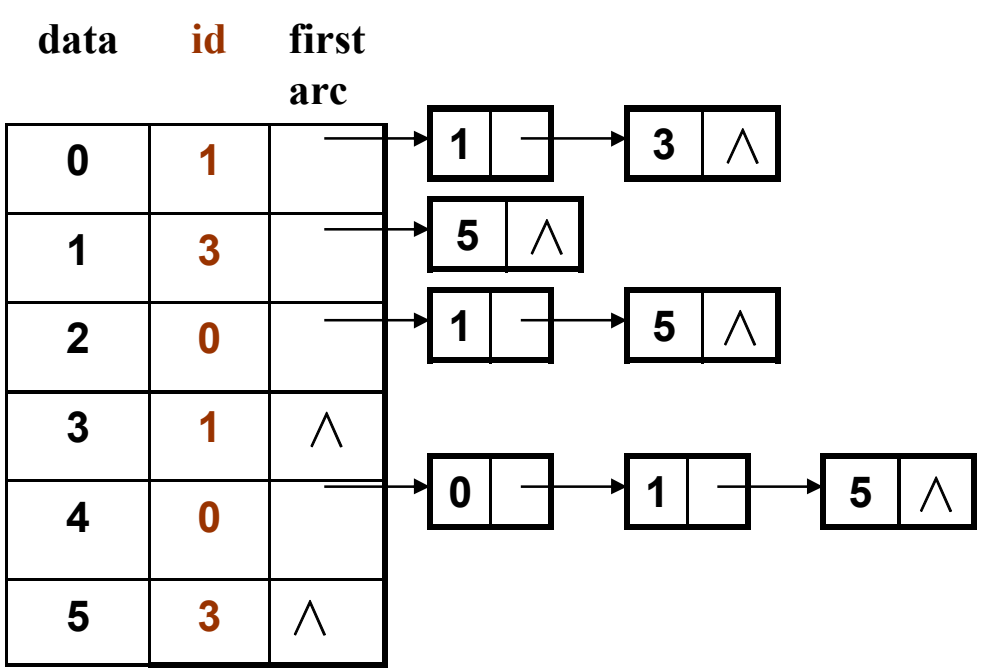
- 图采用邻接表存放；
- 计算所有顶点的入度，存放于一维数组中；

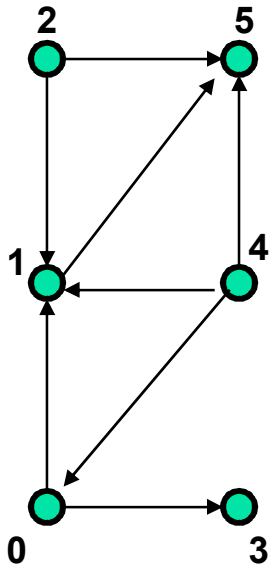
在图的邻接表中每个数组元素增加一个成员(id)存放顶点的入度





```
#define MAXSIZE 100
  typedef struct ArcNode{ int vex;
                          struct ArcNode * link;} ArcNode;
  typedef struct VNode{  VertexType  data;
                         int id;
                         ArcNode * firstarc;}Vnode;
  typedef struct  {     VNode arc[MAXSIZE];
                        int vexnum,arcnum;}Graphs;
```





拓扑排序算法

1. 计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while（队列非空）

{ 将队头顶点*v*输出,count++;

for(每条弧<*v*,*u*>)

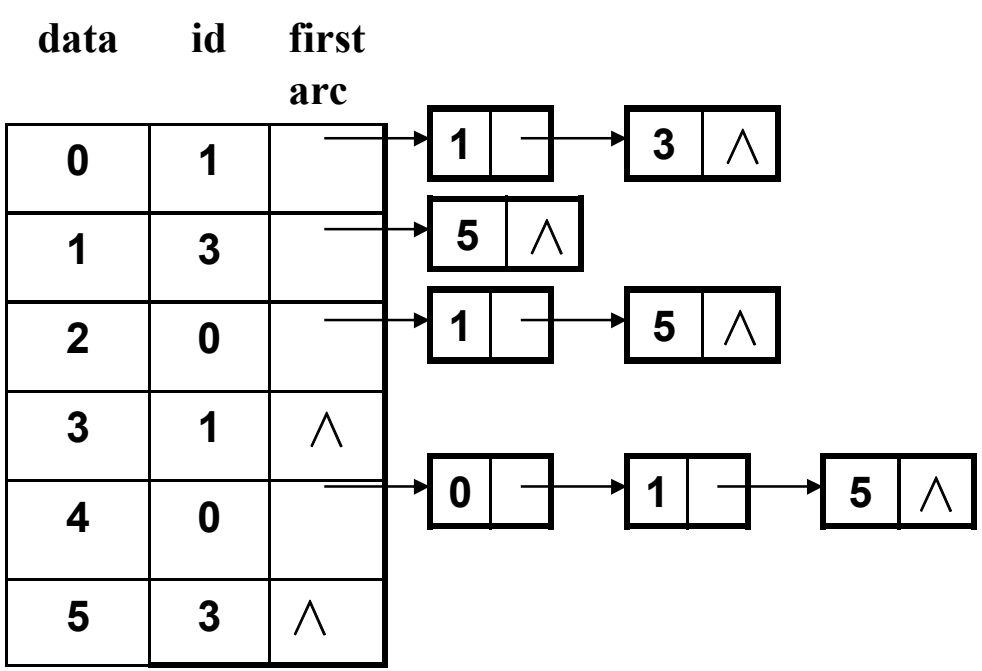
{ 将点*u*的入度减1;

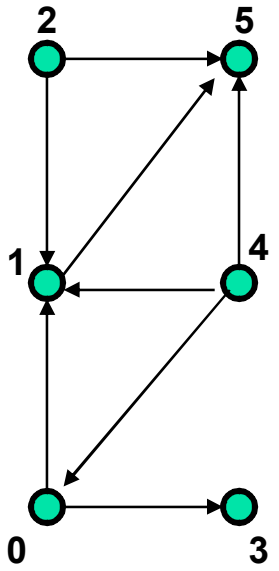
若*u*的入度变为0，则把*u*放入队尾； } }

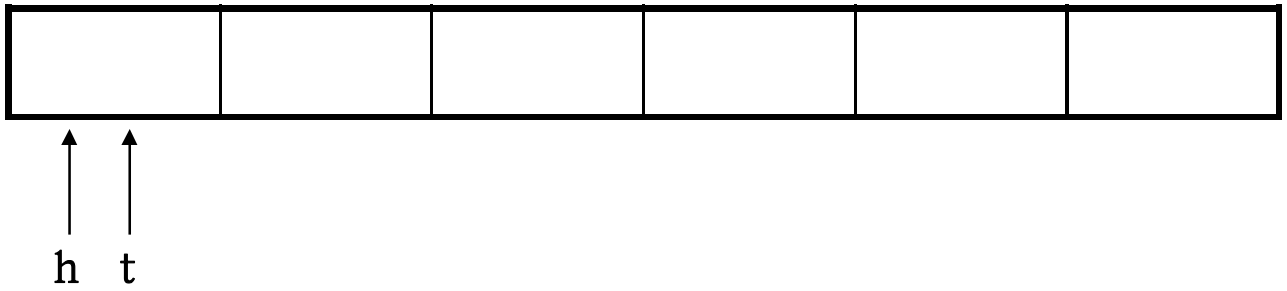
3. 若输出的顶点数小于*n*，则输出“存在有向回路”；否则拓扑排序正常结束。

- //删除不是真正的删除，只是将相应顶点的入度减“1”

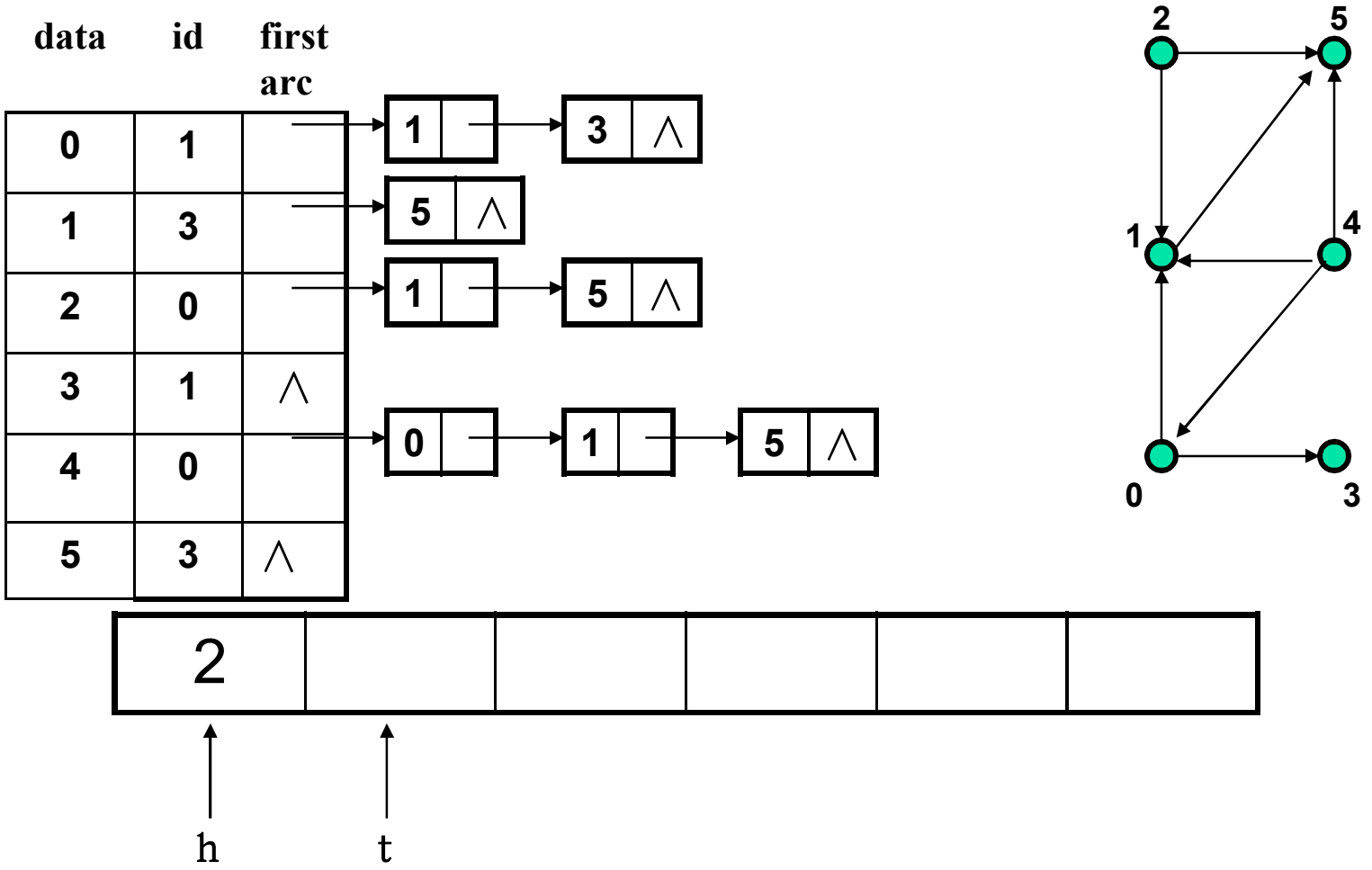
拓扑排序算法



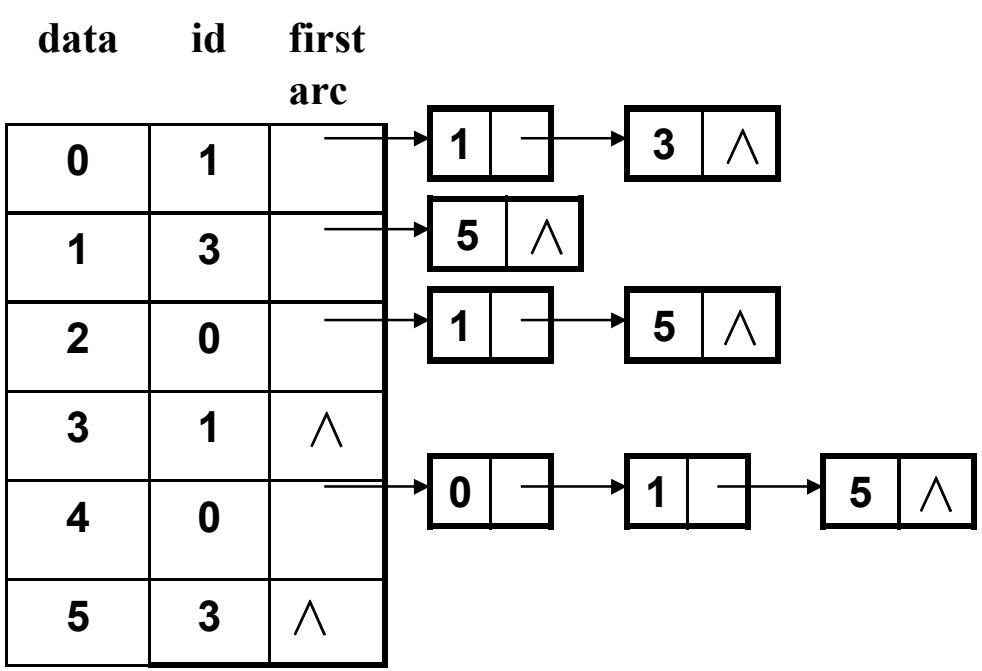


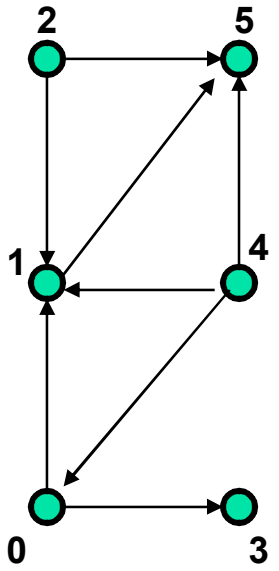


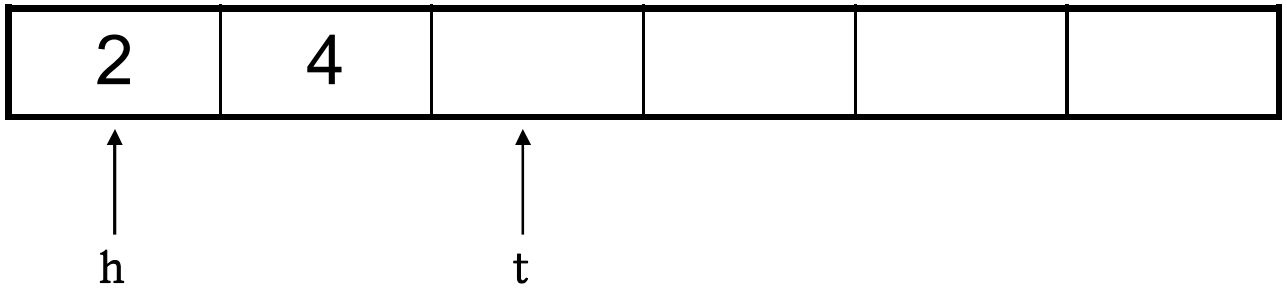
拓扑排序算法



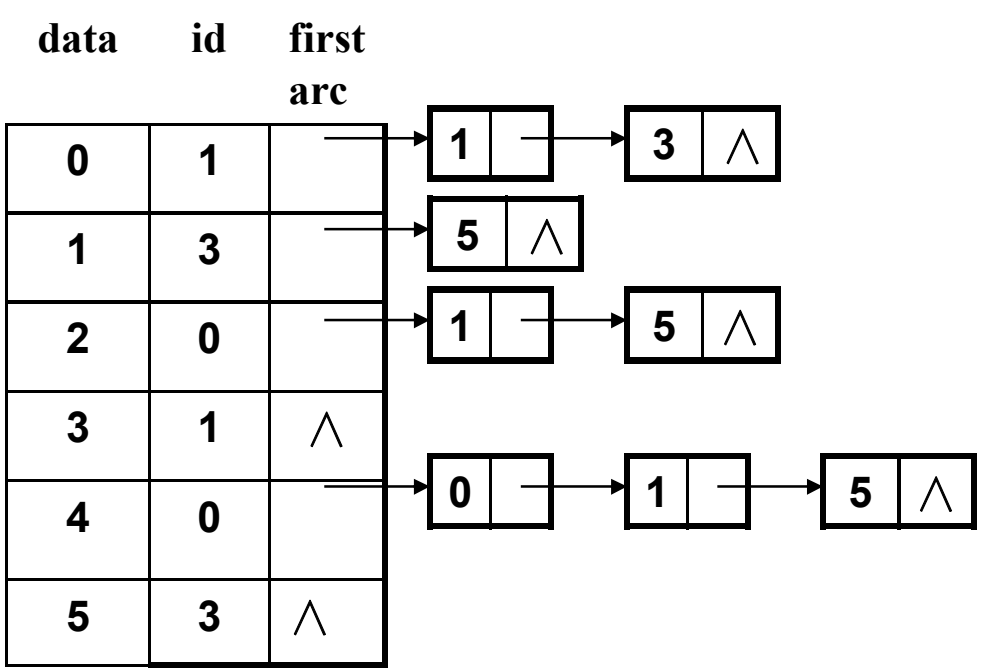
拓扑排序算法

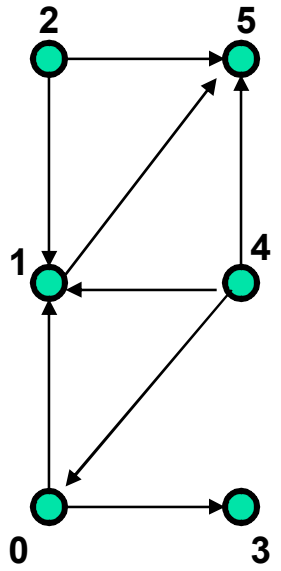


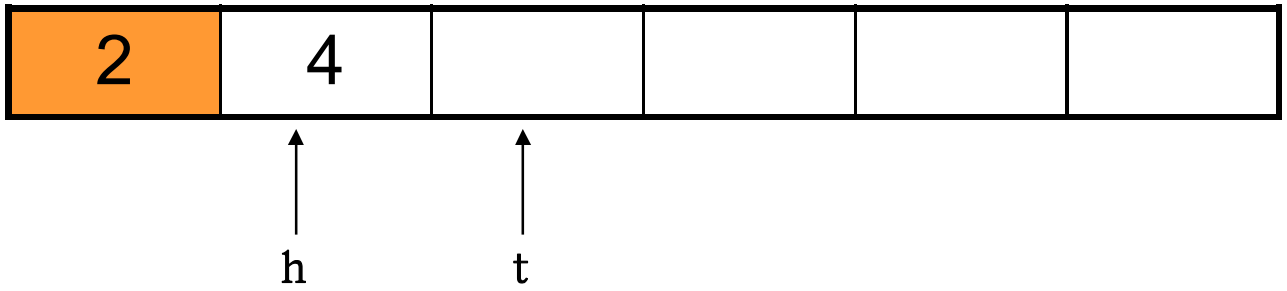




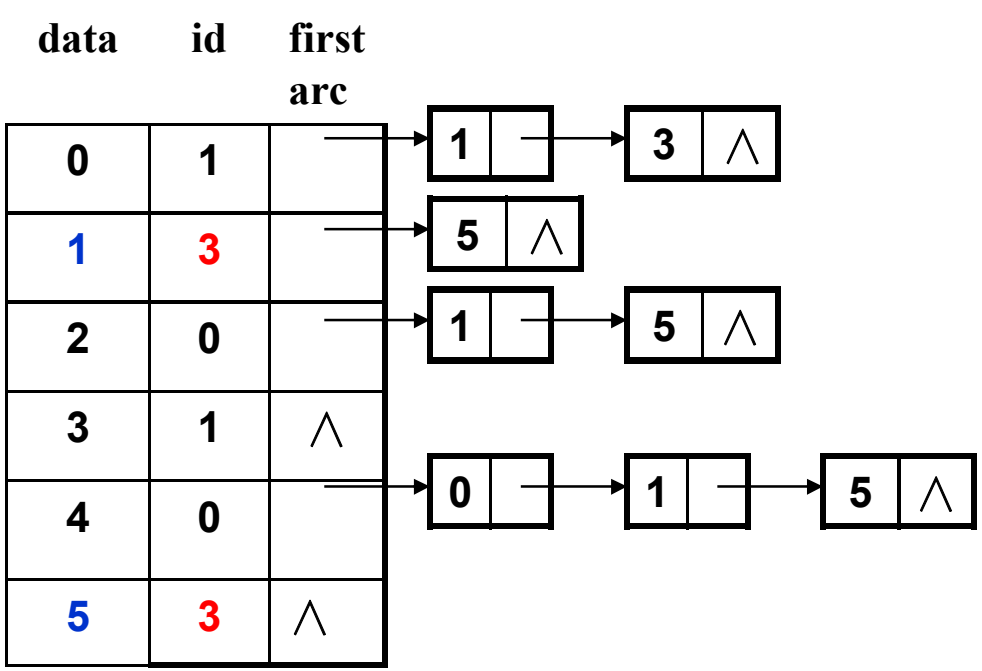
拓扑排序算法

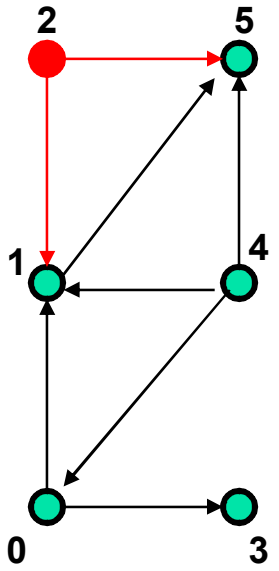


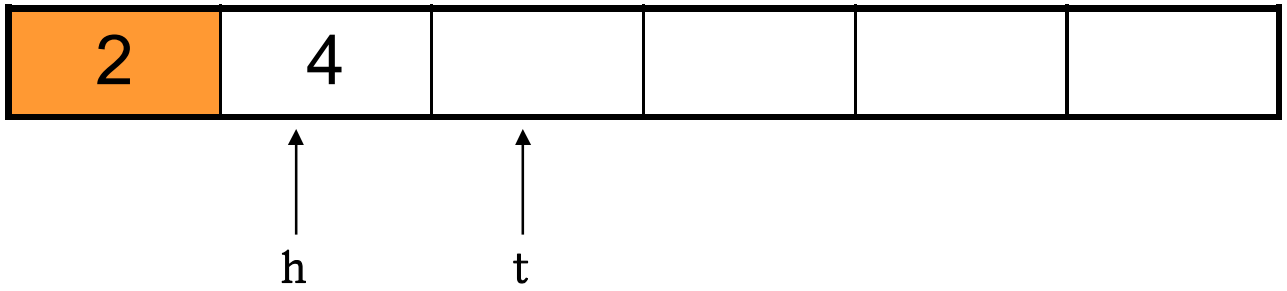




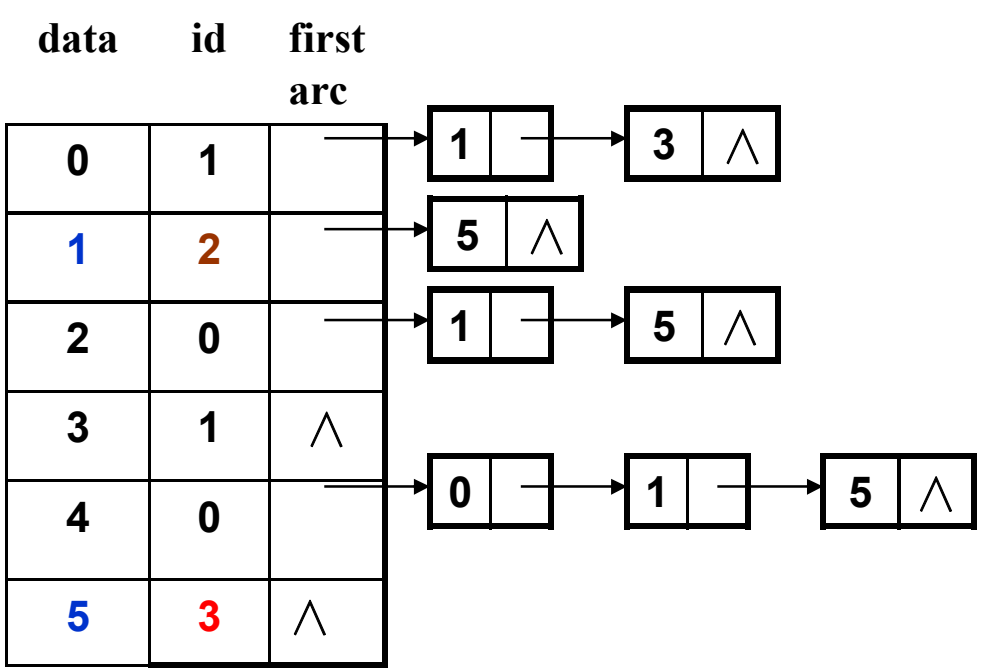
拓扑排序算法

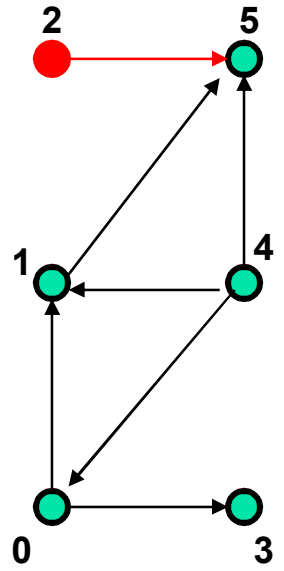




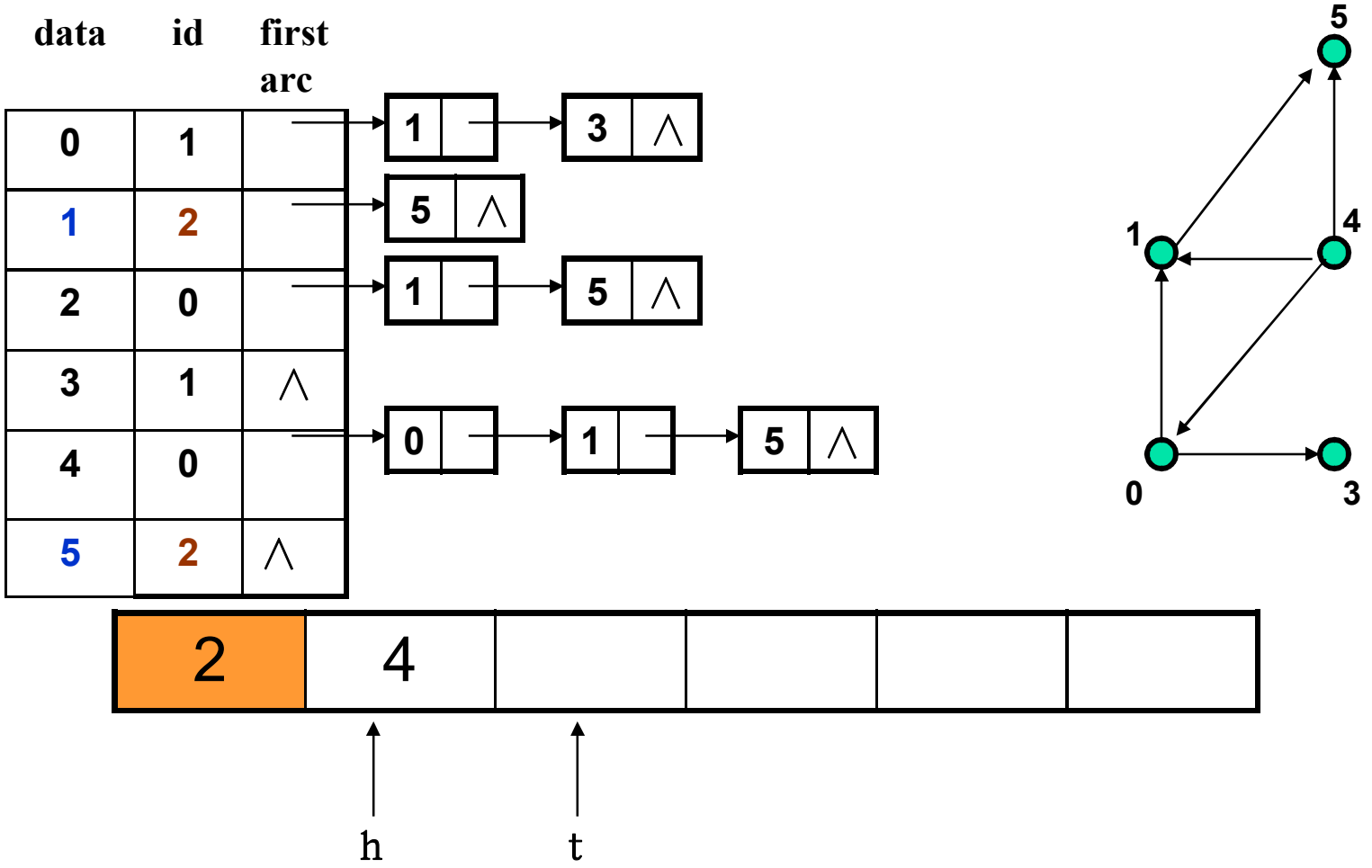


拓扑排序算法

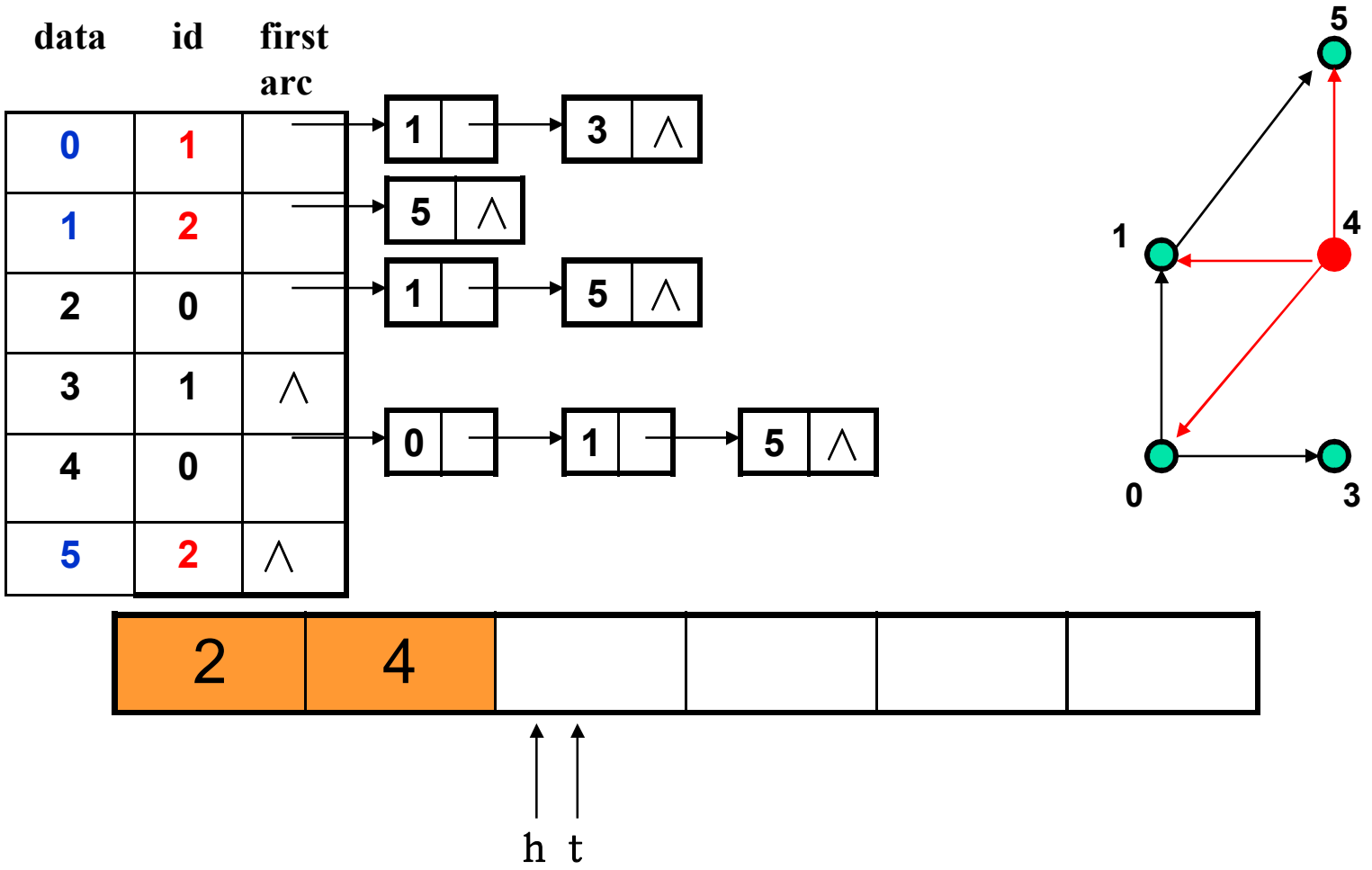




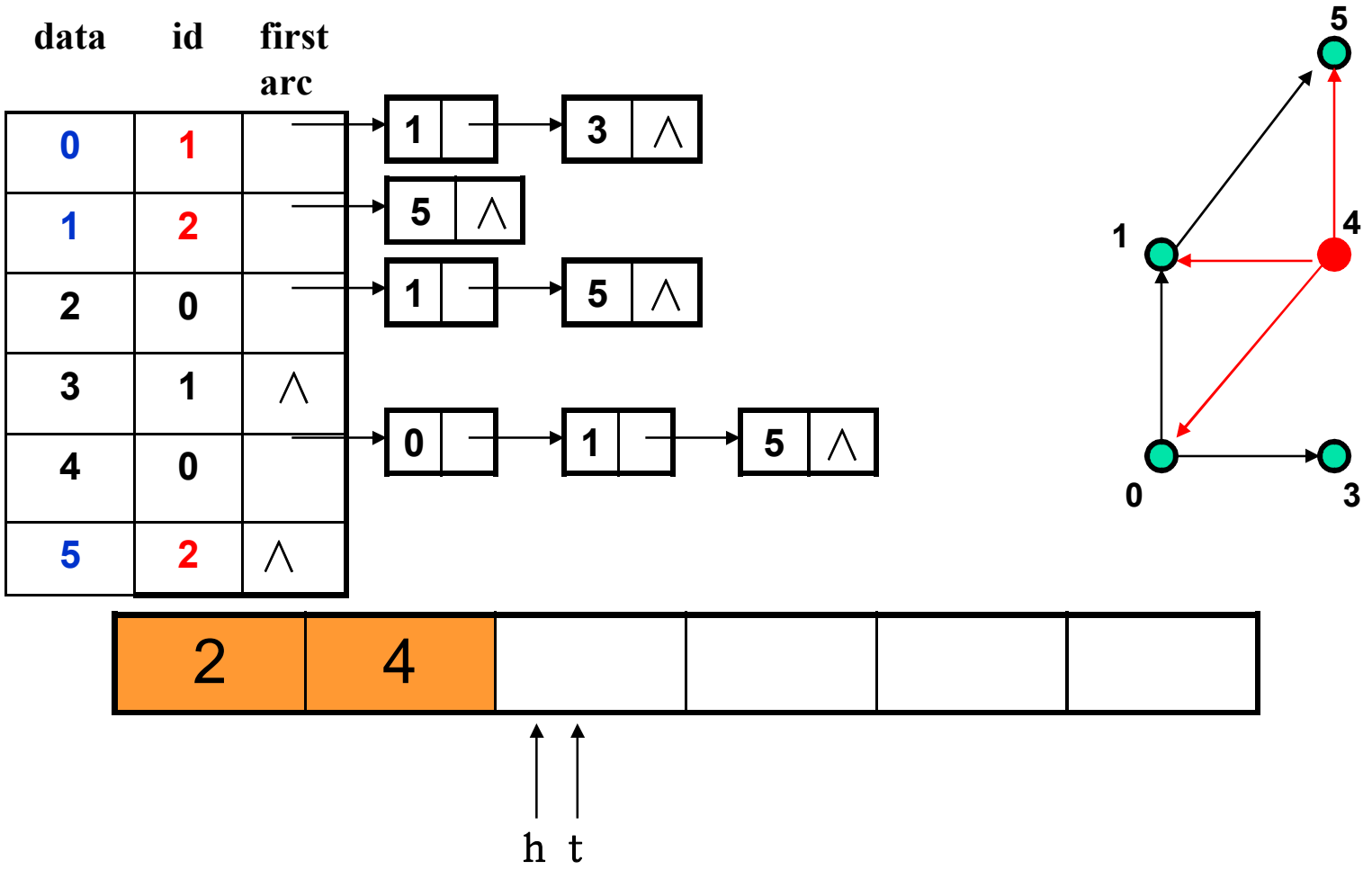
拓扑排序算法



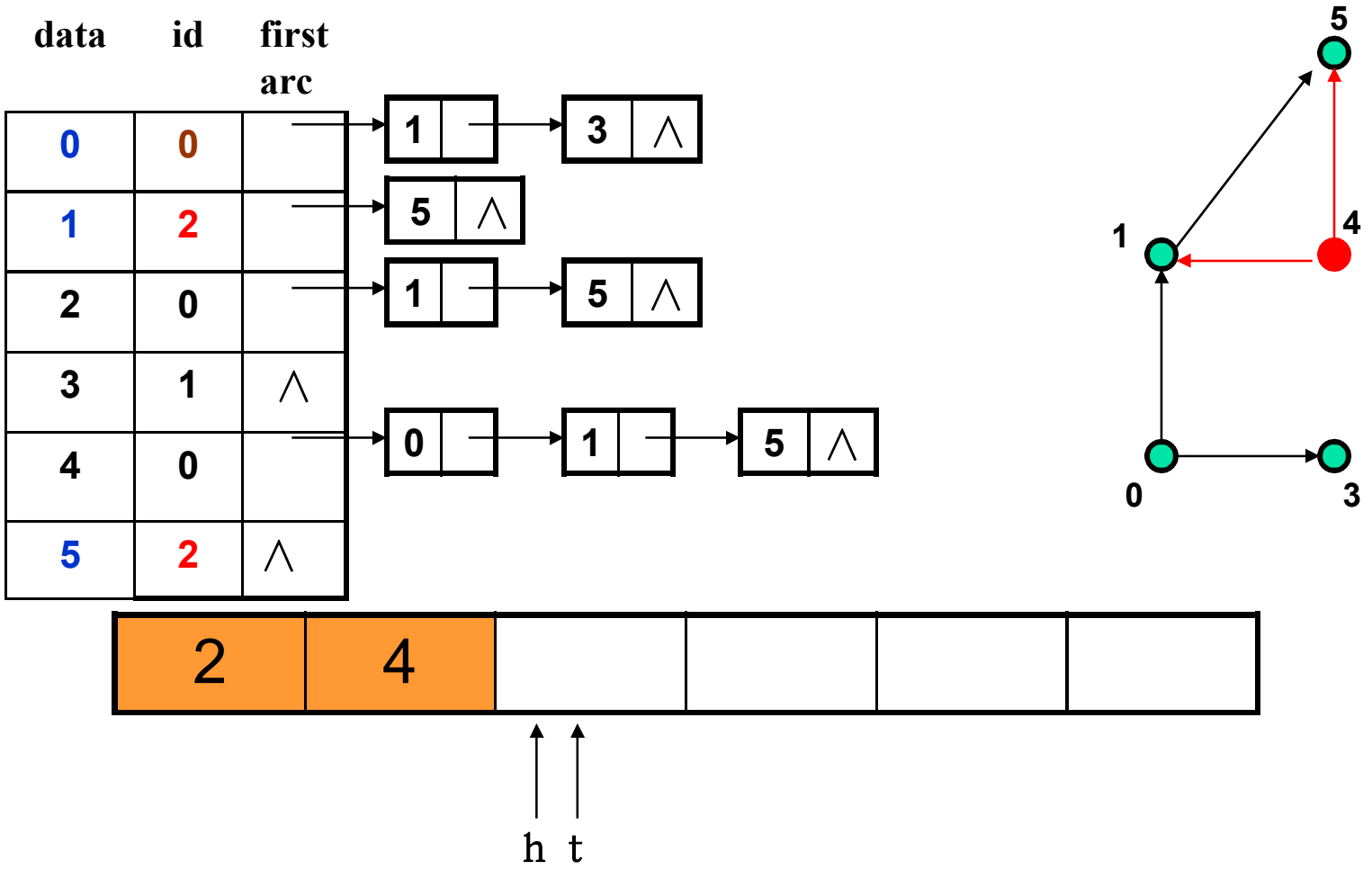
拓扑排序算法



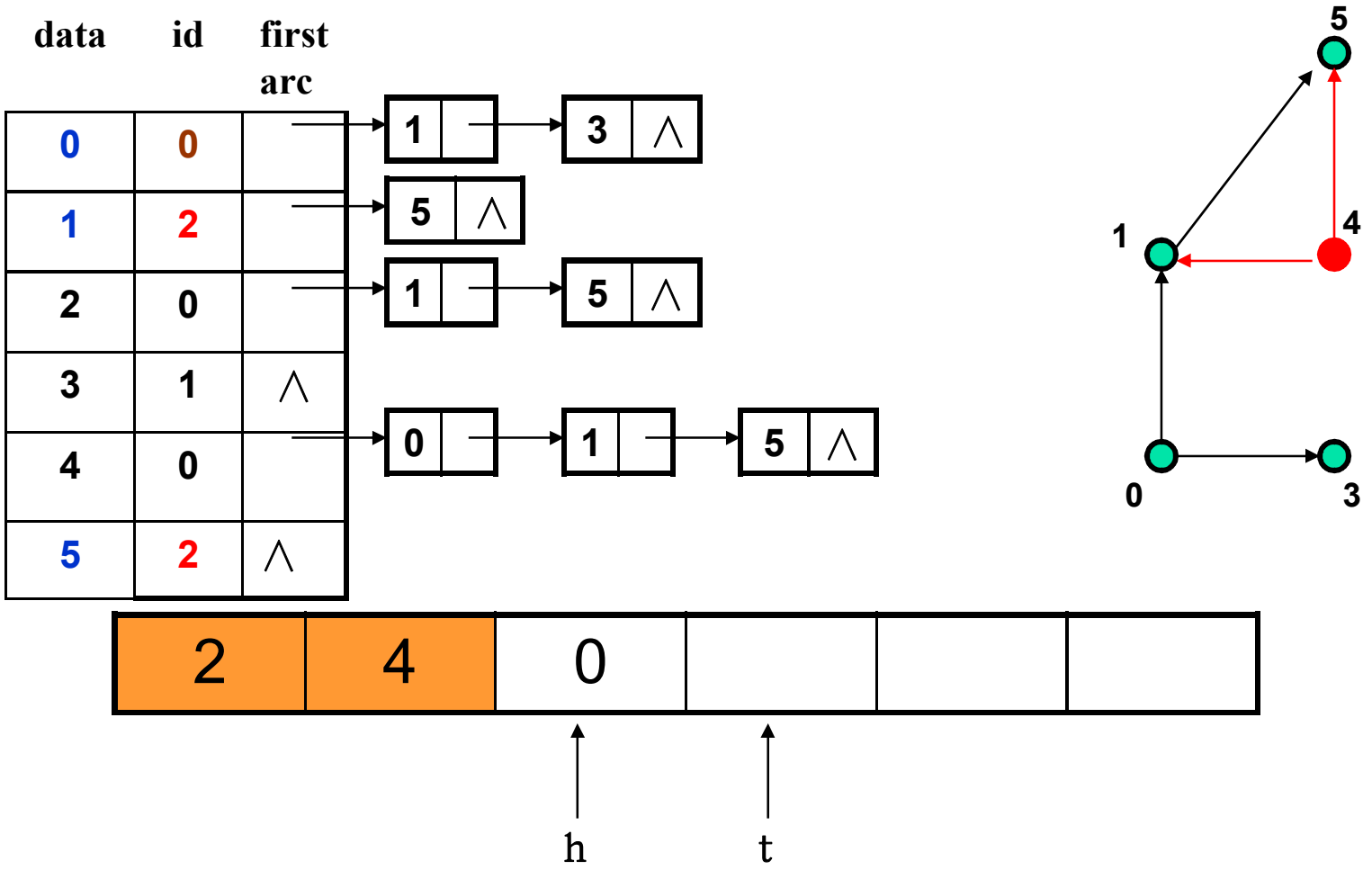
拓扑排序算法



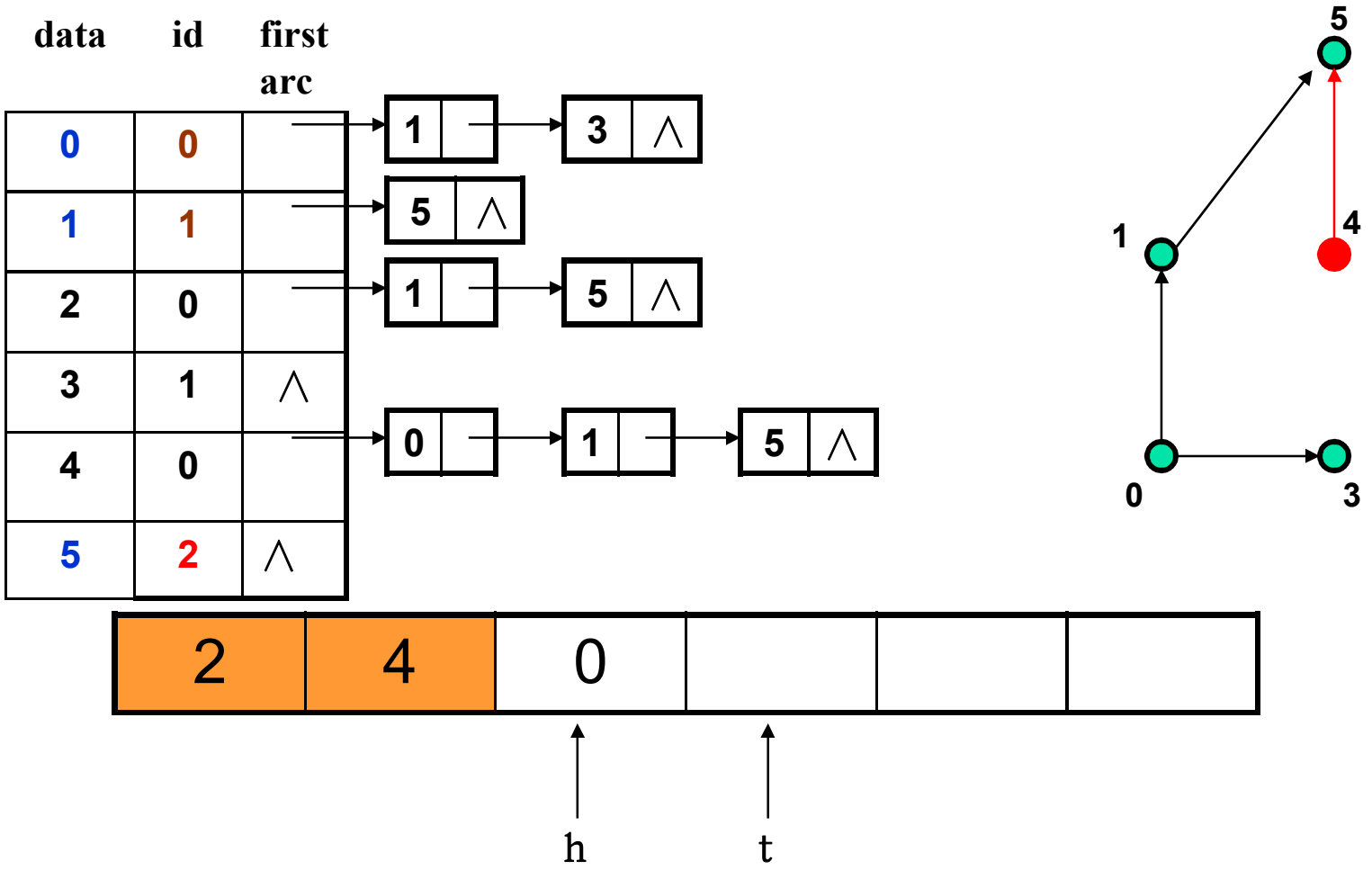
拓扑排序算法



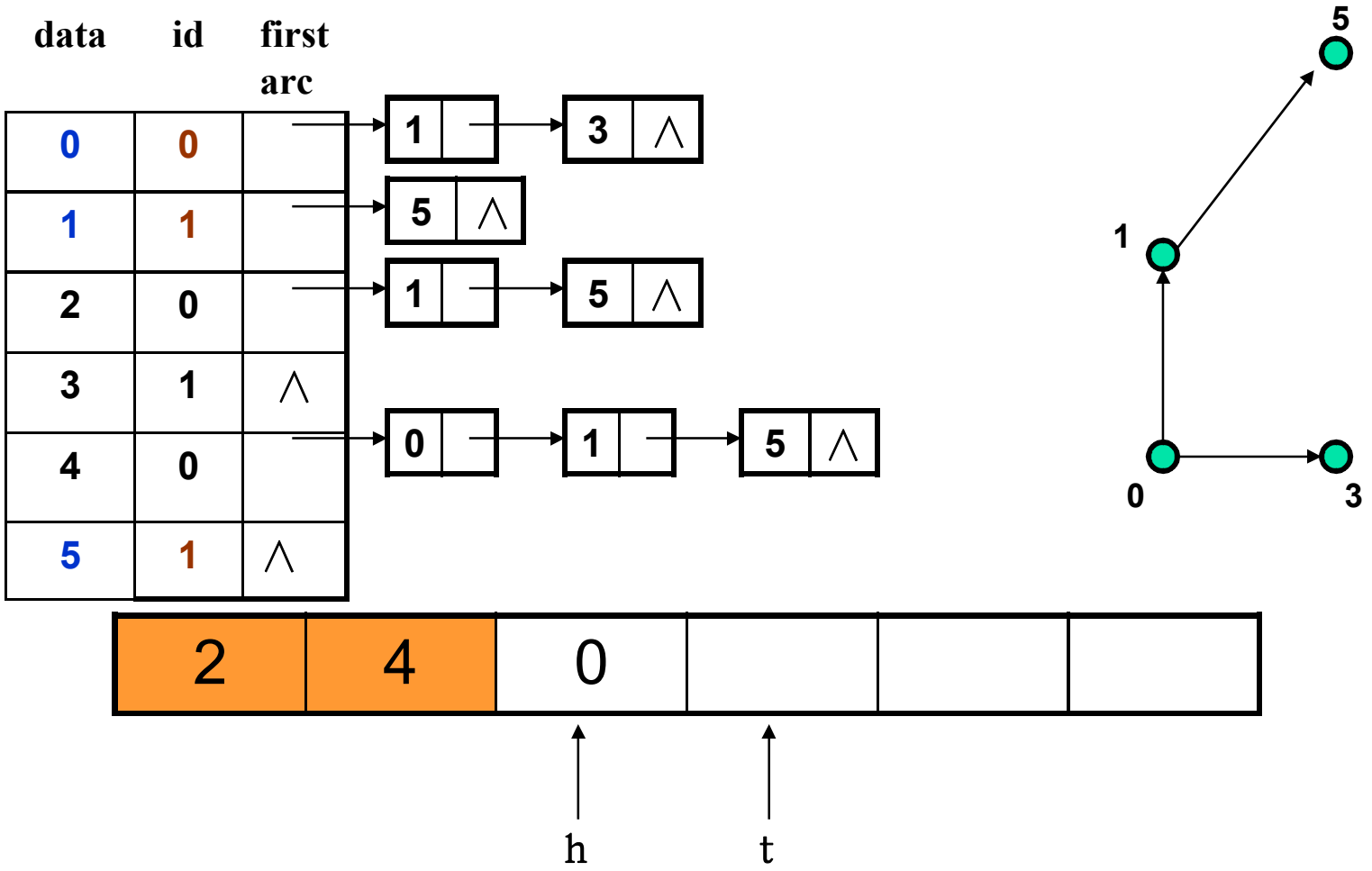
拓扑排序算法



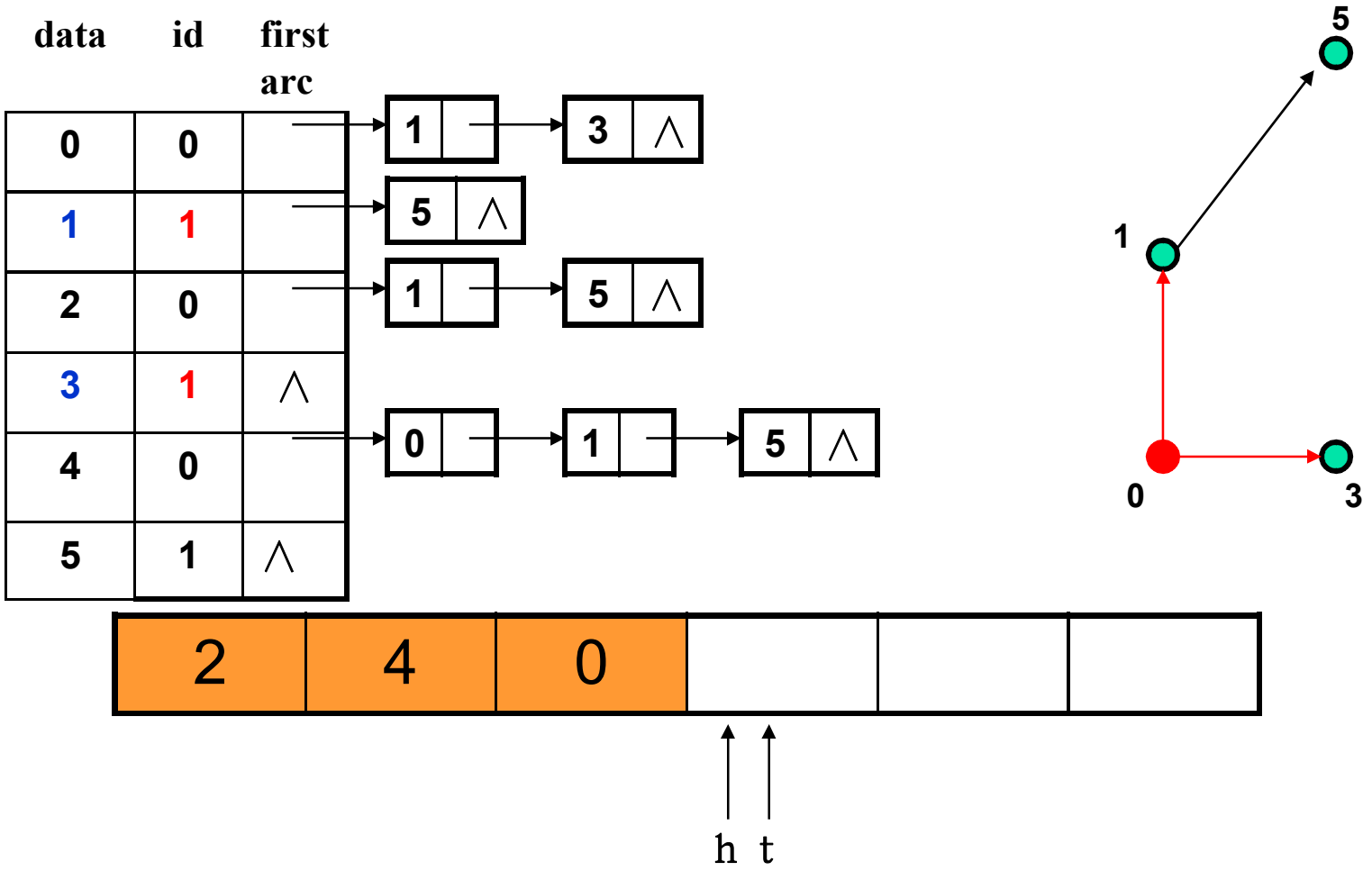
拓扑排序算法



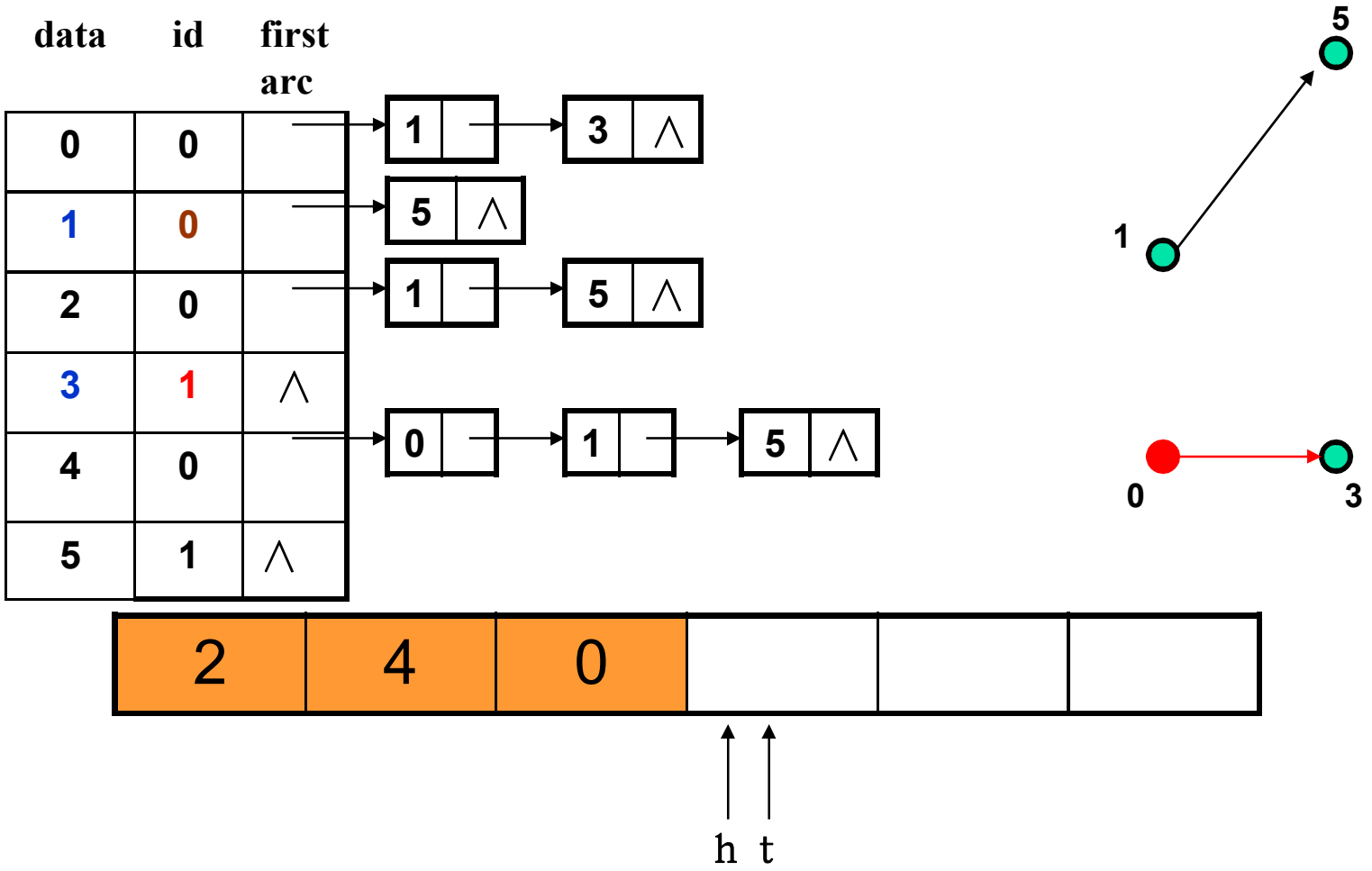
拓扑排序算法



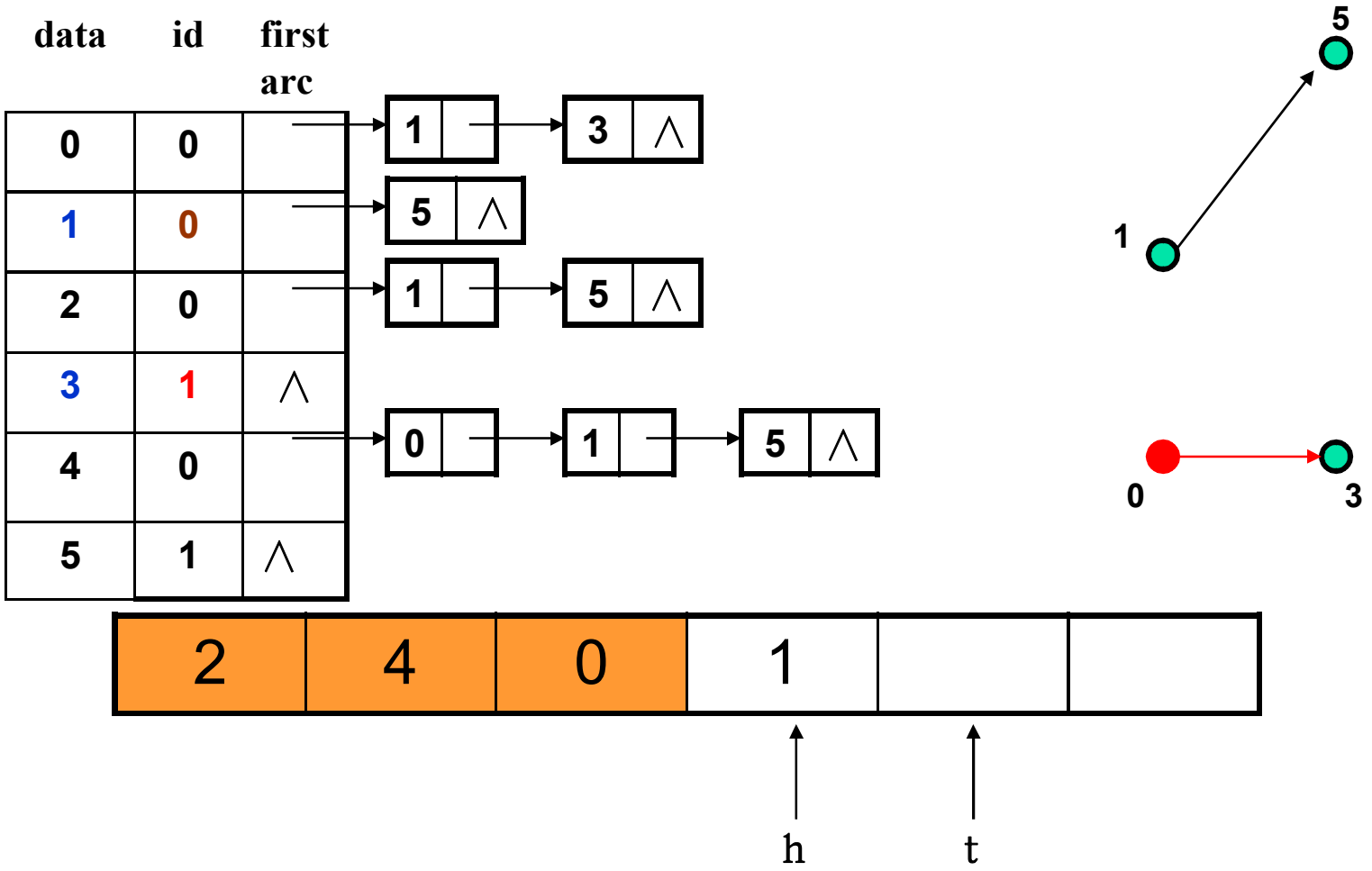
拓扑排序算法



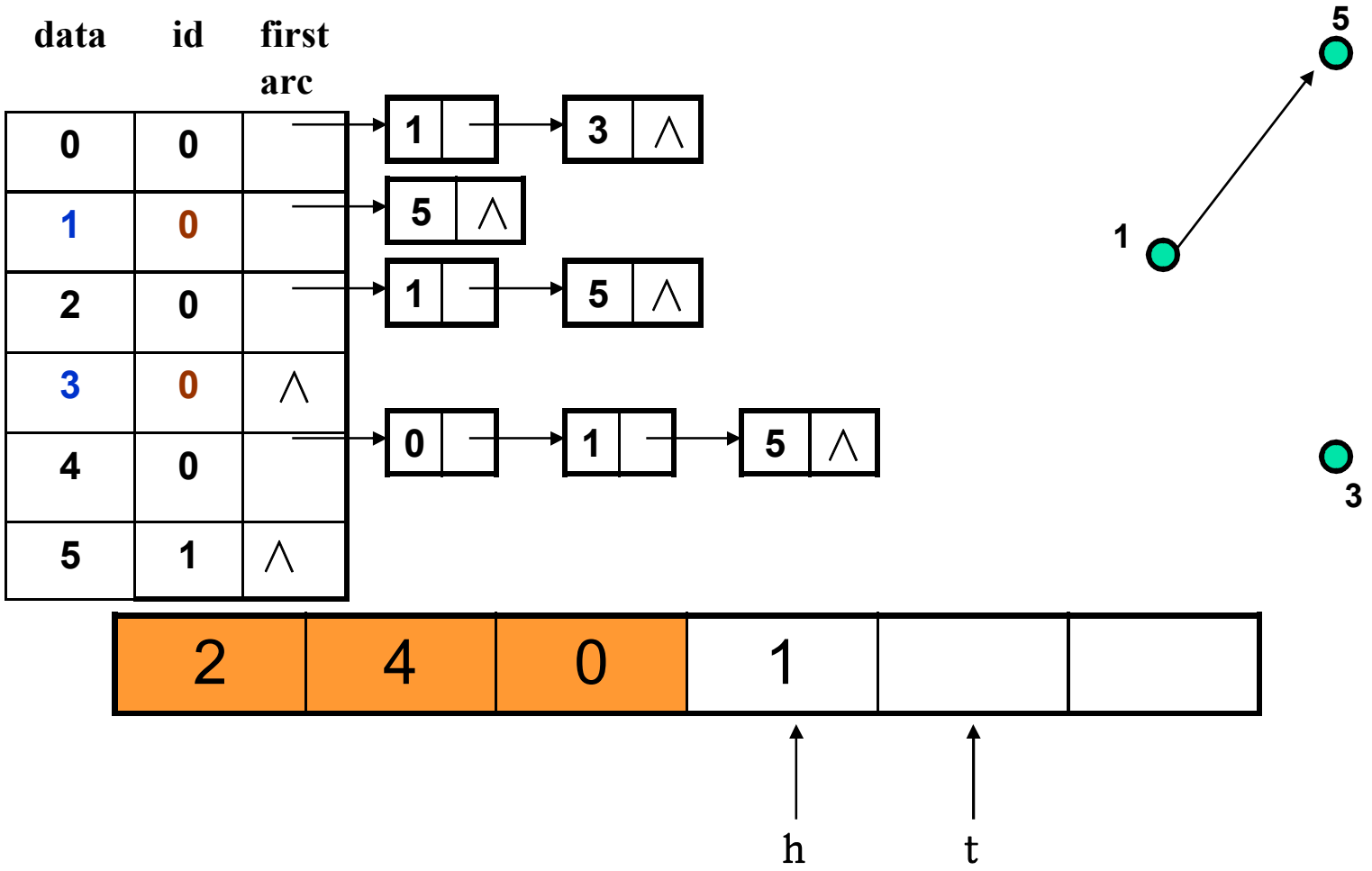
拓扑排序算法



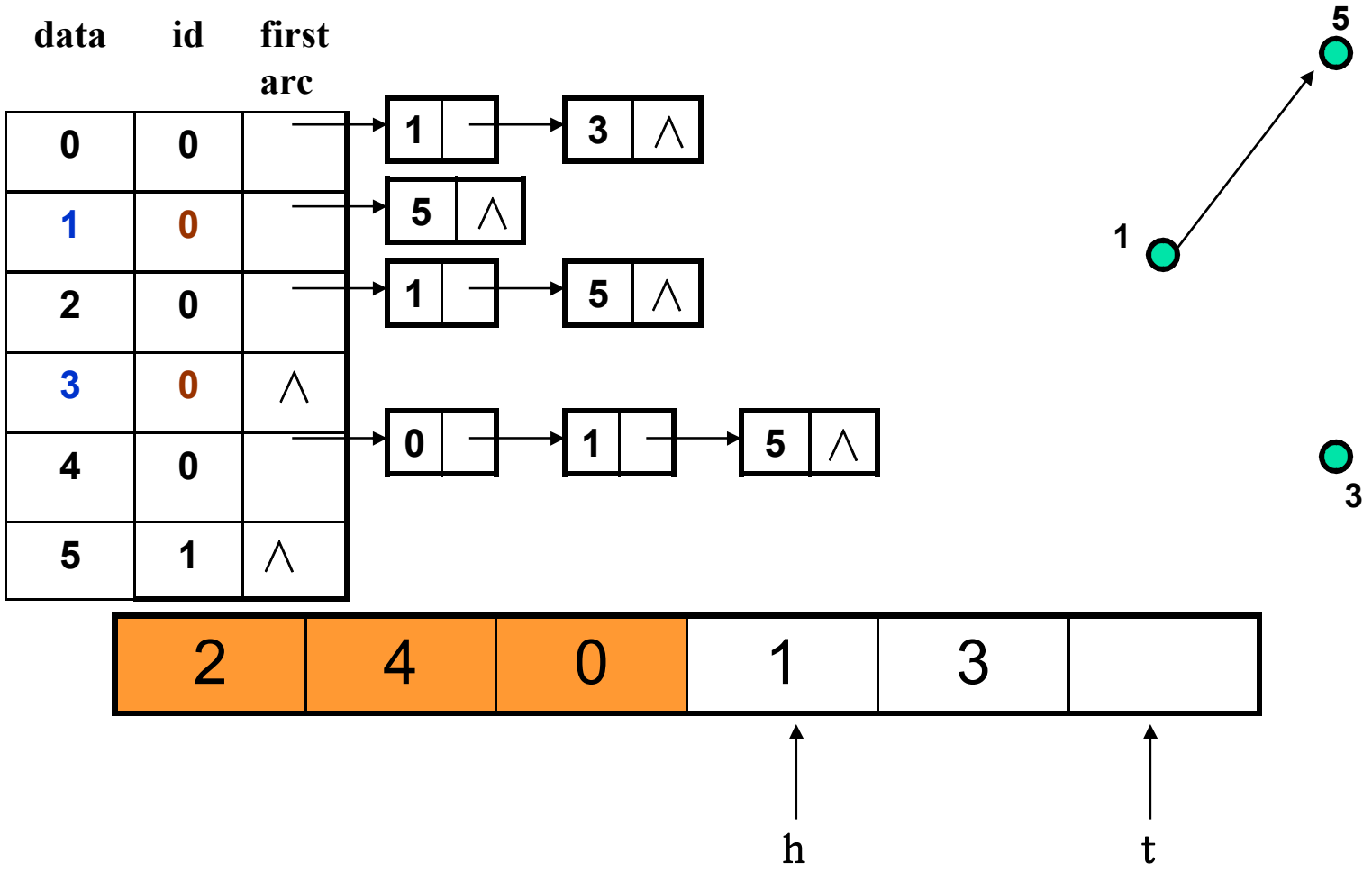
拓扑排序算法



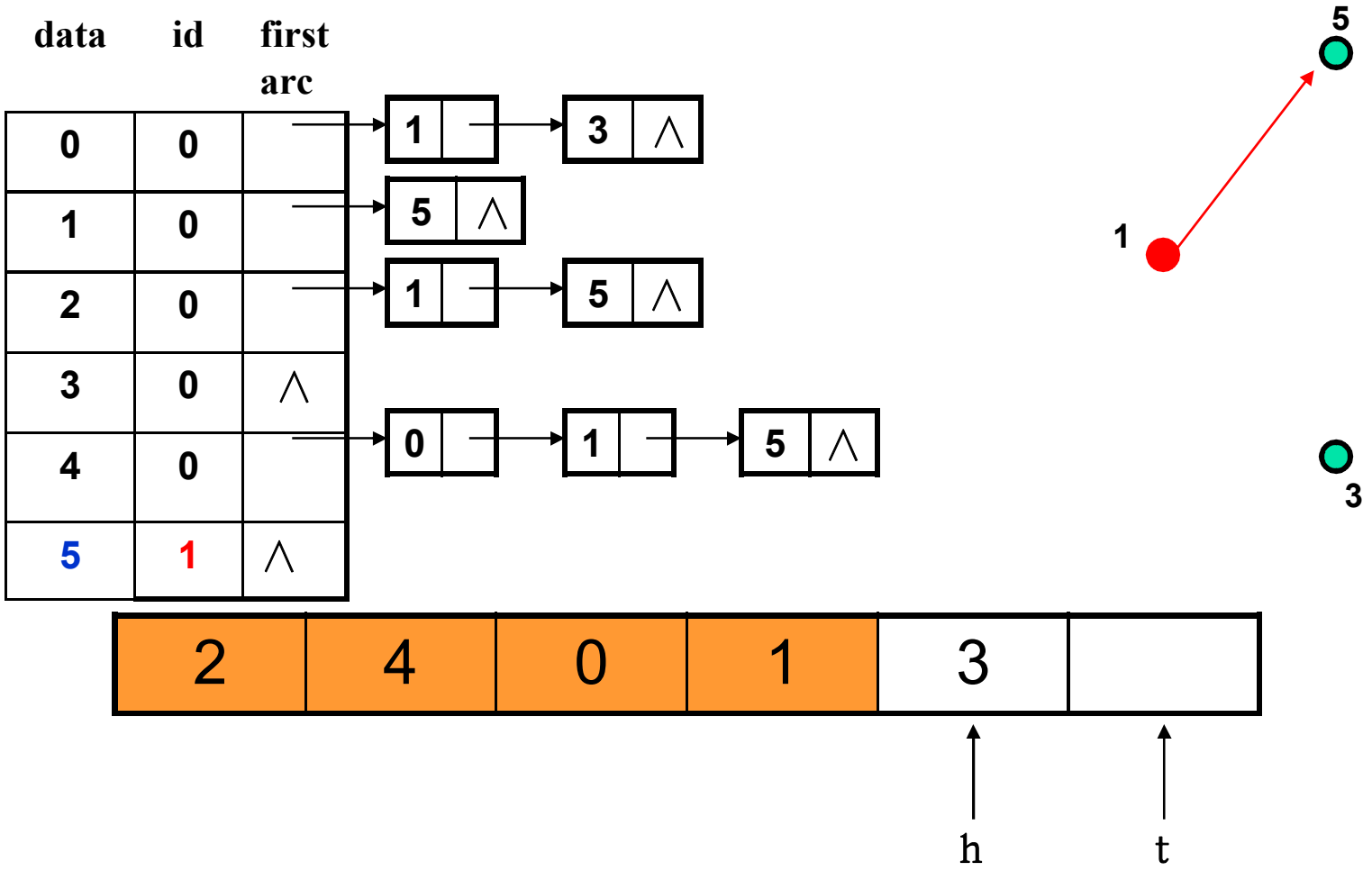
拓扑排序算法



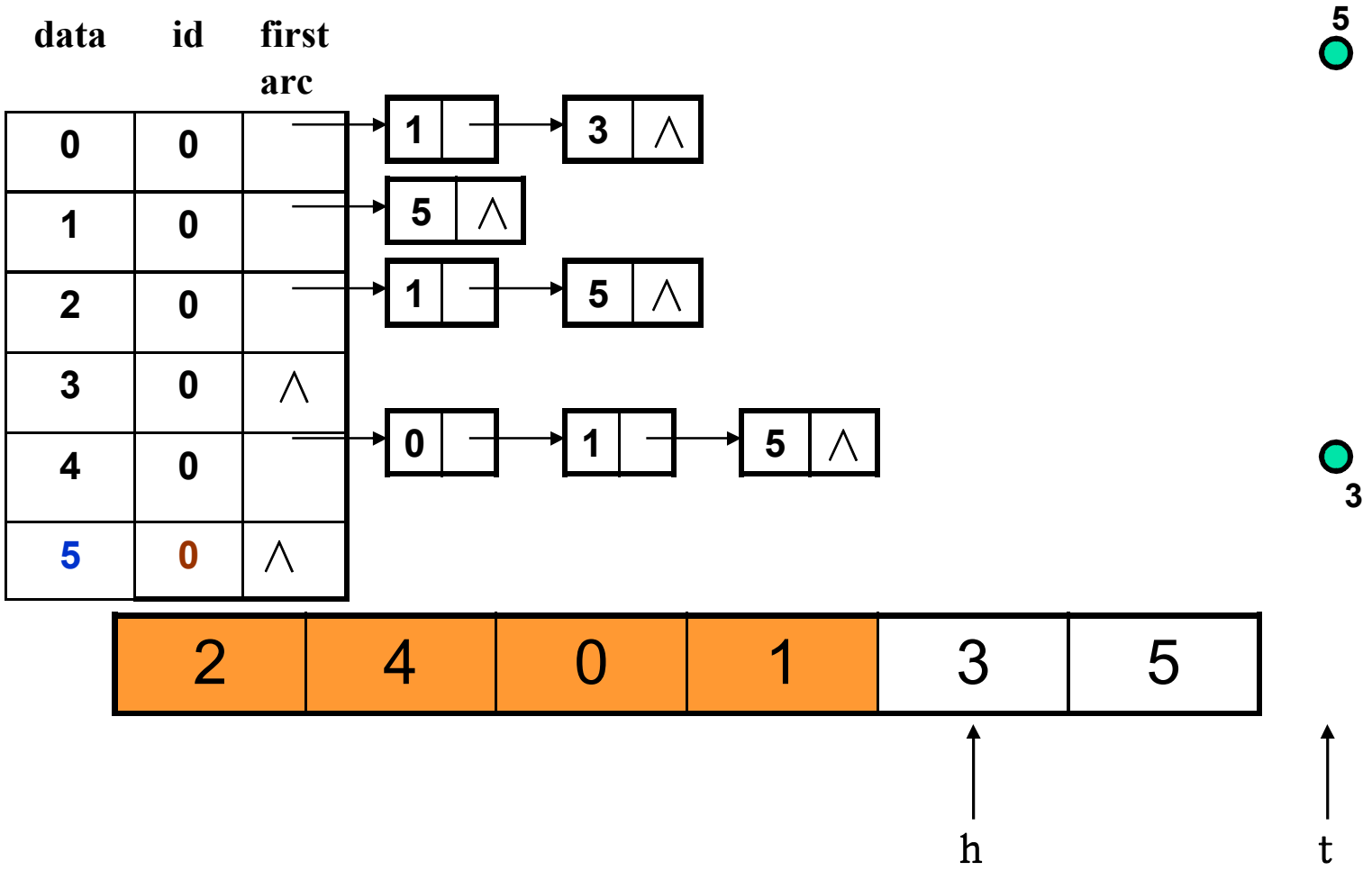
拓扑排序算法



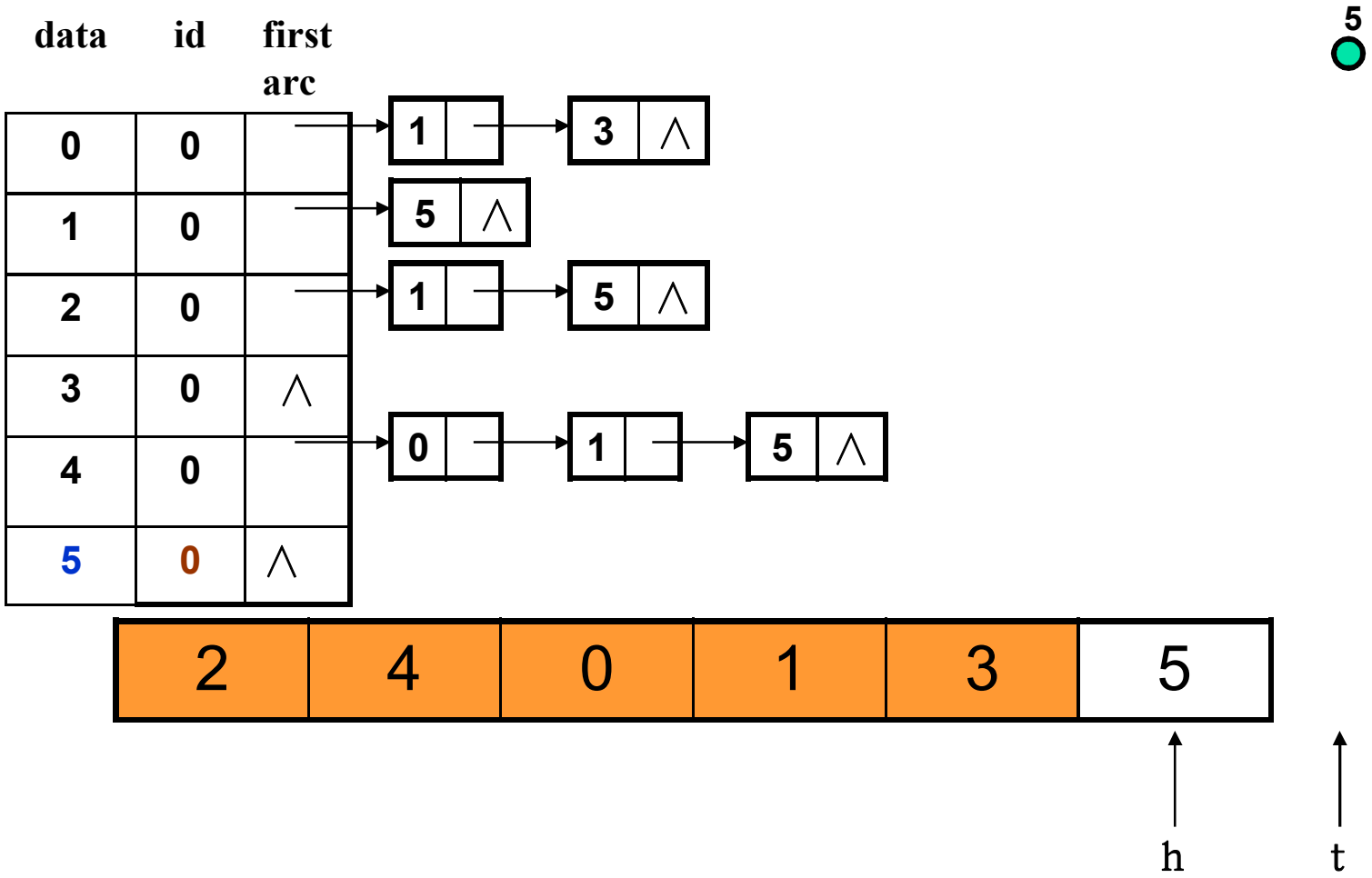
拓扑排序算法



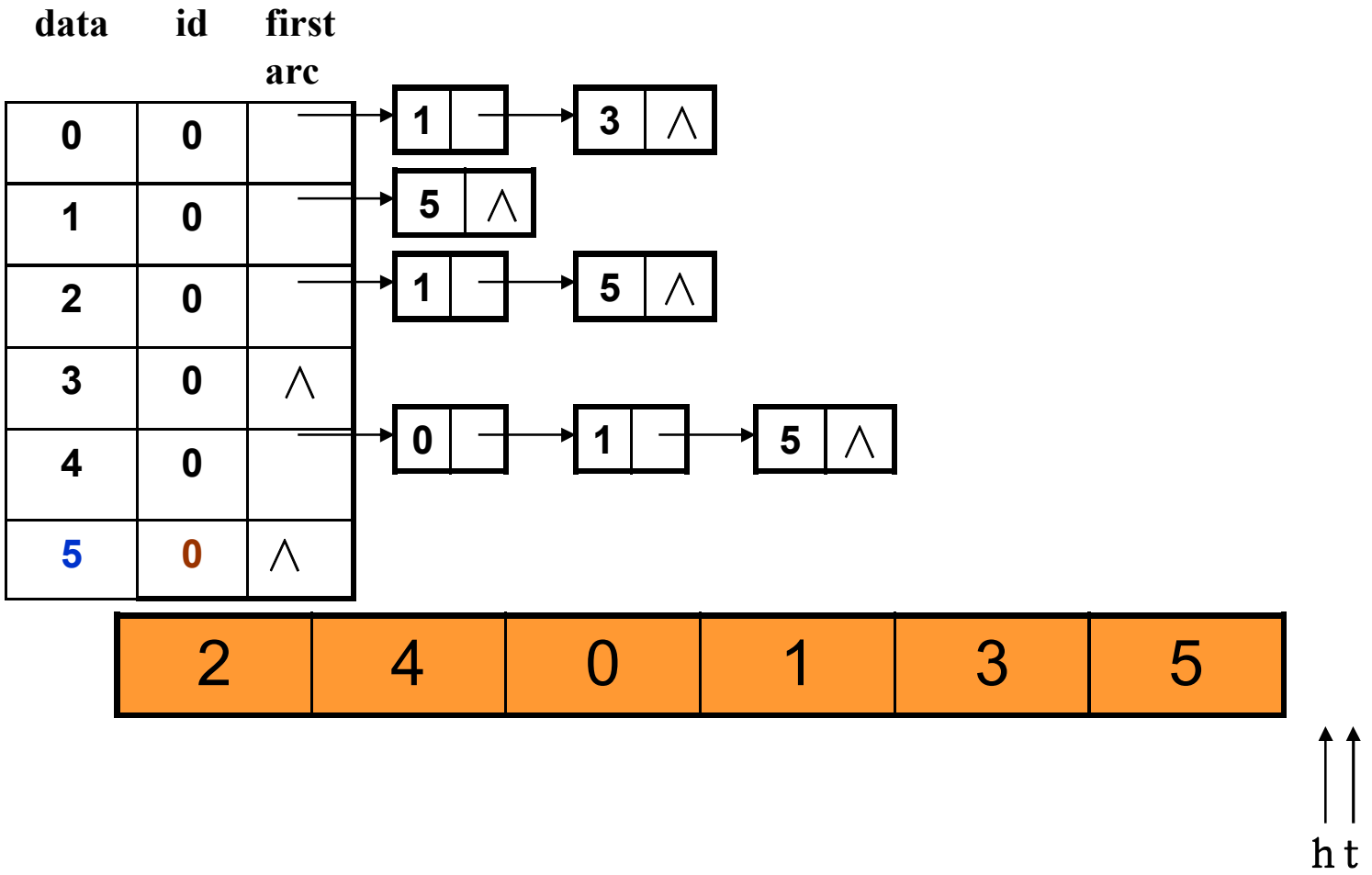
拓扑排序算法

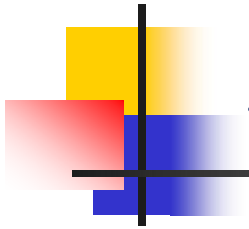


拓扑排序算法



拓扑排序算法





拓扑排序算法

```
int topsort(Graphs T)
  {   int q[MAXSIZE], count, h=t=0;
      ArcNode * p;
      int u,v;
      ……
  }
```

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while（队列非空）

{将队头顶点 *v* 输出,count++;

for(每条弧<*v*,*u*>)

{将点*u*的入度减1;

若*u*的入度变为0，则把*u*放入队尾;

}

}

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

{ 将队头顶点 *v* 输出,count++;

for(每条弧<*v*,*u*>)

{ 将点*u*的入度减1;

若*u*的入度变为0，则把*u*放入队尾;

}

}

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

{ *v*=q[*h*++]; printf(“%d”,v); count++; ;

for(每条弧<*v*,*u*>)

{ 将点*u*的入度减1;

若*u*的入度变为0，则把*u*放入队尾;

}

}

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

{ *v*=q[*h*++]; printf(“%d”,v); count++; ;

for(p=T.arc[*v*].firstarc; p!=Null;p=p->link)

{ 将点*u*的入度减1;

若*u*的入度变为0，则把*u*放入队尾;

}

}

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

{ *v*=q[*h*++]; printf(“%d”,v); count++; ;

for(p=T.arc[*v*].firstarc; p!=Null;p=p->link)

{ *u*=p->*vex*; T. arc[*u*].id--;;

若*u*的入度变为0，则把*u*放入队尾;

}

}

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

{ *v*=q[*h*++]; printf(“%d”,v); count++; ;

for(p=T.arc[*v*].firstarc; p!=Null;p=p->link)

{ *u*=p->*vex*; T. arc[*u*].id--;;

if (T. arc[*u*].id==0) ，则把*u*放入队尾；

}

}

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

```
{  v=q[h++]; printf("%d",v); count++;   ;
  for(p=T.arc[v].firstarc; p!=Null;p=p->link)
    {  u=p->vex; T. arc[u].id--;;
       if (T. arc[u].id==0) q[t++]=u ;
     }
}
```

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

{ *v*=q[*h*++]; printf(“%d”,v); count++; ;

for(p=T.arc[*v*].firstarc; p!=Null;p=p->link)

{ *u*=p->*vex*; T. arc[*u*].id--;;

if (T. arc[*u*].id==0) q[*t*++]=*u* ； //自己加判断队列是否会溢出.

}

}

3. 若输出的顶点数小于*n*，则输出“存在有向回路”；

否则拓扑 排序正常结束。

拓扑排序算法

1.计算所有顶点入度，将入度为0 的顶点放入队列；count=0;

2. while (*h*!=*t*)

{ *v*=q[*h*++]; printf("%d",v); count++; ;

for(p=T.arc[*v*].firstarc; p!=Null;p=p->link)

{ *u*=p->*vex*; T. arc[*u*].id--;;

if (T. arc[*u*].id==0) q[*t*++]=*u* ； //自己加判断队列是否会溢出.

}

}

3. if (count<T.vexnum) {printf("There is a cycle")； return 0;}

else return 1;

拓扑排序算法

1. 计算所有顶点入度，将入度为0 的顶点放入队列；

```
for(v=0;v<T.vexnum;v++)
   T.arcs[v].id=0;
for (v=0;v<T.vexnum;v++)
    for(p=T.arc[v].firstarc; p!=Null;p=p->link)
   {
        u=p->vex;
        T. arcs[u].id++;
    }
   for (v=0;v<T.vexnum;v++)
          if (T. arc[v].id==0) q[t++]=v ; // 自己加判断队列是否会溢出！
```

拓扑排序算法

- 分析： 设图有n个顶点，m条边。
- 计算所有定点的入度，执行时间为o(m)
- 初始建立入度为0的顶点队列，要检查所有顶点一次，执行时间为o(n)
- 每个顶点入、出队各一次，执行时间为o(n)
- 邻接表的每条边执行一次入度减1操作，执行时间为o(m)
- 总的时间复杂度为o(n+m)