



7.5 有向无环图及其应用--拓扑排序

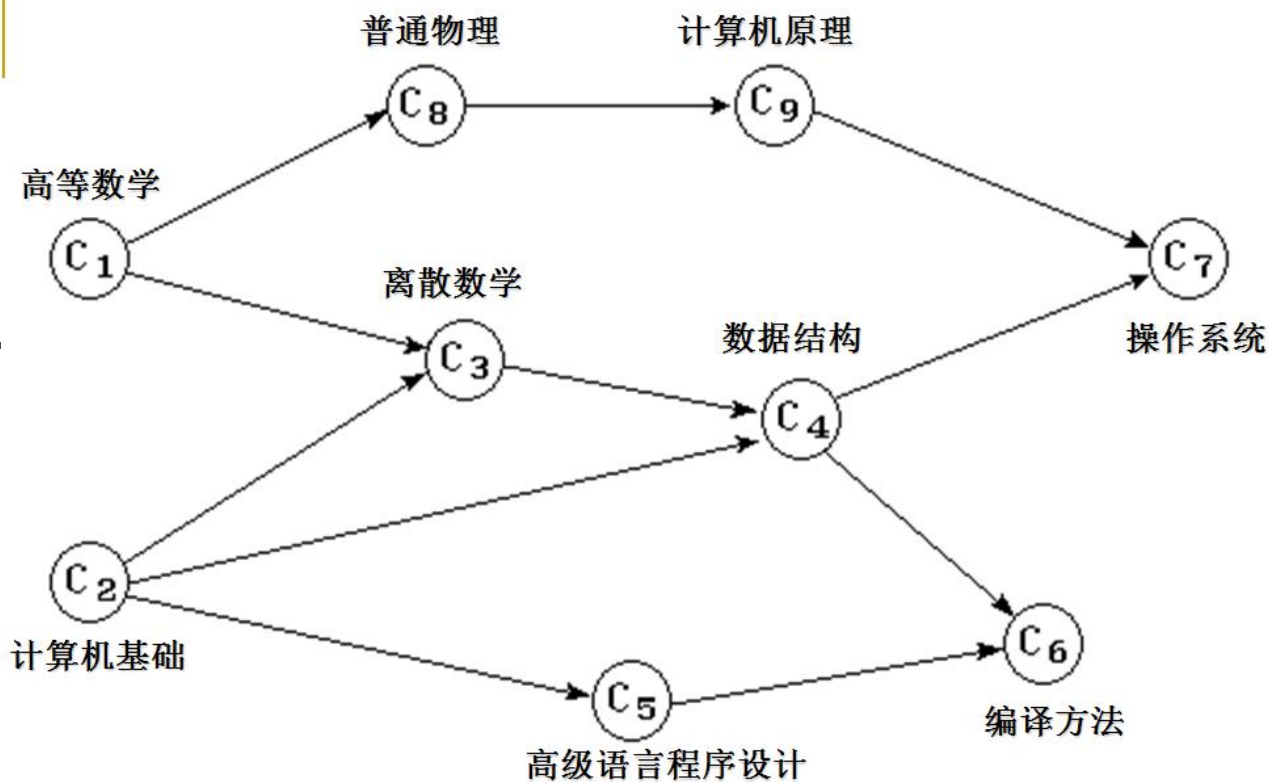
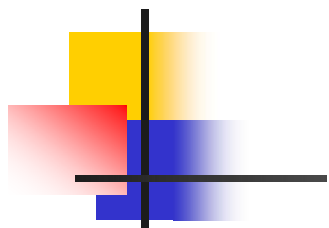
- 研究一类有向图--**AOV网**:用顶点表示活动, 弧表示活动之间的制约 (优先) 关系。
- AOV网代表一项工程, 工程划分成若干活动, 每个活动用一个顶点表示, 活动之间的先后制约关系用弧表示

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

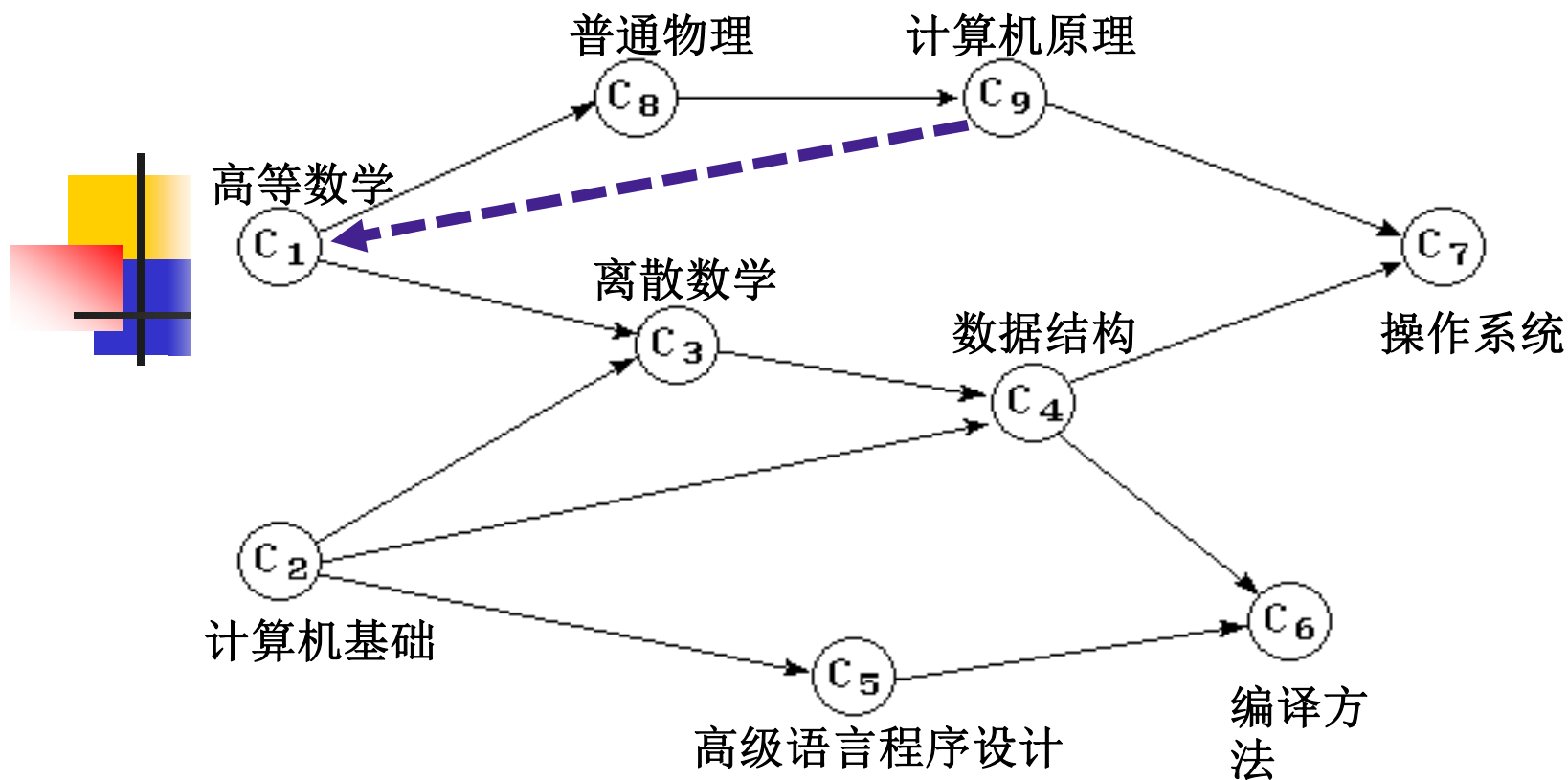
AOV网示例



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



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

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



7.5 有向无环图及其应用--**拓扑排序**

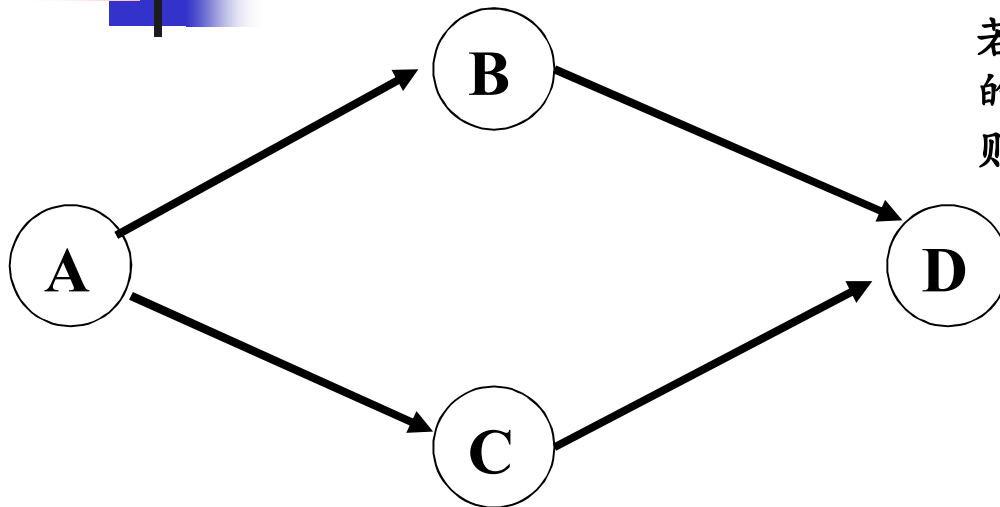
- 测试AOV网是否具有回路的方法，就是在AOV网的偏序集合下构造一个线性序列，该线性序列具有以下性质：
 - 若顶点 i 优先于 j ，则在线性序列中 i 仍然优先于 j ；
 - 对于网中原来没有优先关系的顶点 i 与顶点 j ，在线性序列中也建立一个先后关系，或者 i 优先于 j ，或者 j 优先于 i 。
- 满足这样性质的线性序列称为**拓扑有序序列**。构造拓扑序列的过程称为**拓扑排序**。也可以说拓扑排序就是由某个集合上的一个偏序得到该集合上的一个全序的操作。



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 $a, b \in A$ 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



若图中弧 $\langle A, B \rangle$ 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**偏序关系**。

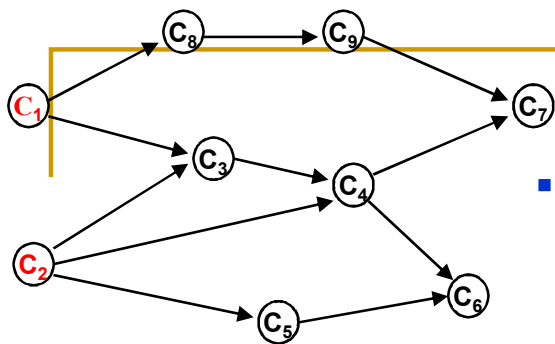
$\left. \begin{array}{l} ABCD \\ ACBD \end{array} \right\}$

全序关系——拓扑排序序列

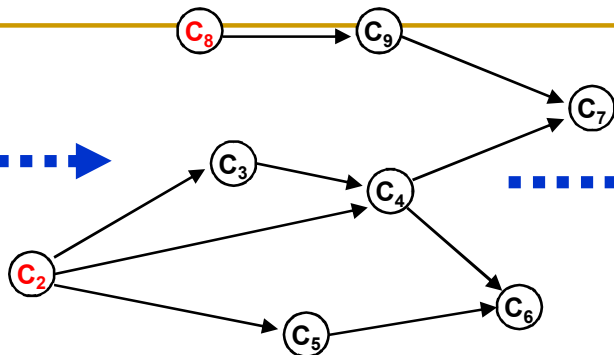


拓扑排序算法

- 1. 选一个入度为0的顶点,放到拓扑排序序列中;
- 2. 删除该顶点以及由它出发的所有弧
- 3. 重复1,2, 直到全部顶点均放到拓扑排序序列或不存在入度为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。

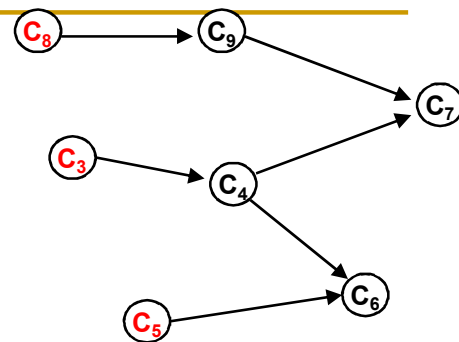


入度为0的顶点为C1,C2



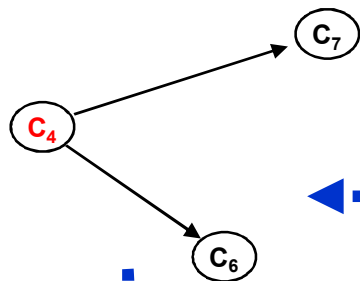
拓扑排序序列: C1

将C1放入拓扑排序序列, 删去C1及其发出去的弧

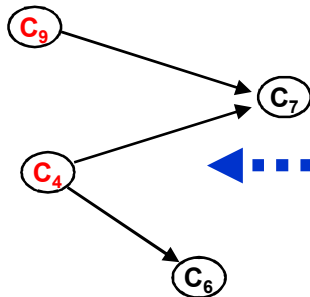


拓扑排序序列: C1,C2

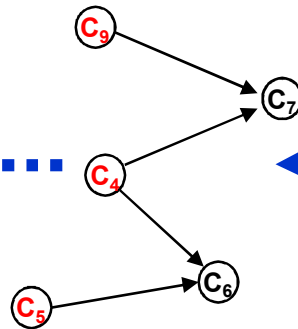
将C2放入拓扑排序序列, 删去C2及其发出去的弧



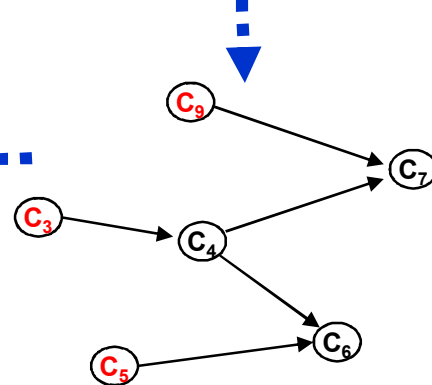
拓扑排序序列: C1,C2,C8,C3,C5,C9



拓扑排序序列: C1,C2,C8,C3,C5



拓扑排序序列: C1,C2,C8,C3



拓扑排序序列: C1,C2,C8



空图, 图中所有顶点均输出到拓扑排序序列中, 最后得到拓扑排序序列:
C1,C2,C8,C3,C5,C9,C4,C6,C7



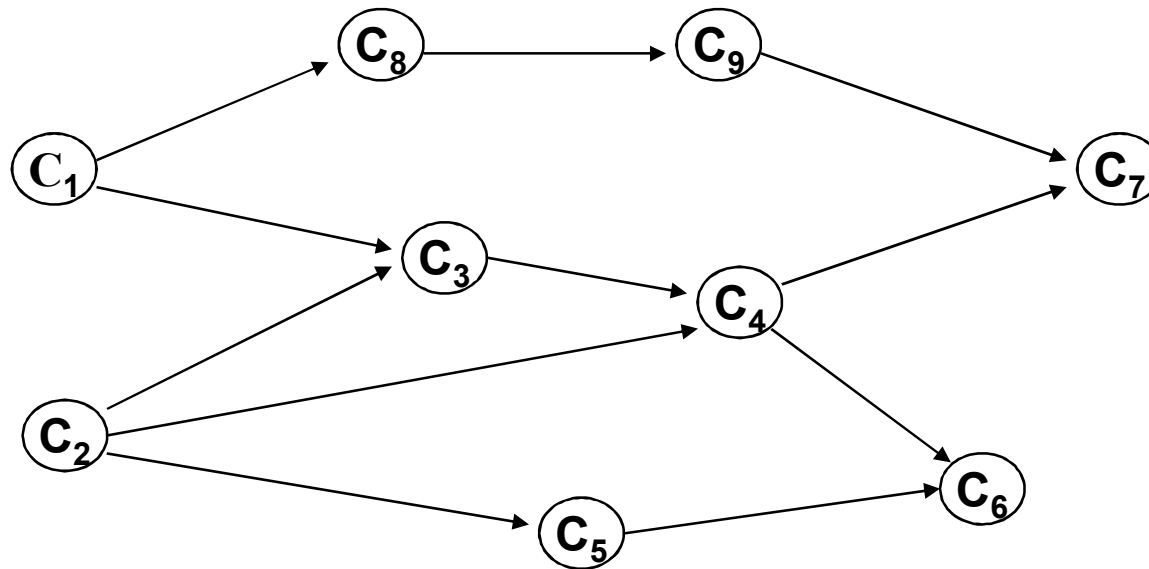
拓扑排序序列: C1,C2,C8,C3,C5,C9,C4

拓扑排序序列: C1,C2,C8,C3,C5,C9,C4,C6

拓扑排序过程演示

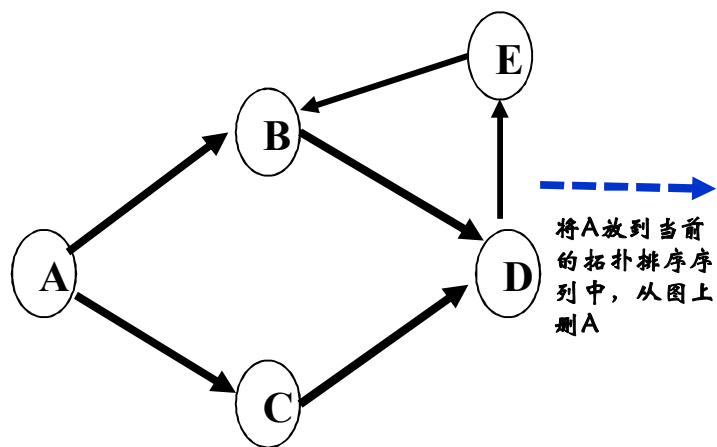
图中红色顶点为入度为0的顶点

一个AOV网的拓扑排序序列可能存在多个，不唯一

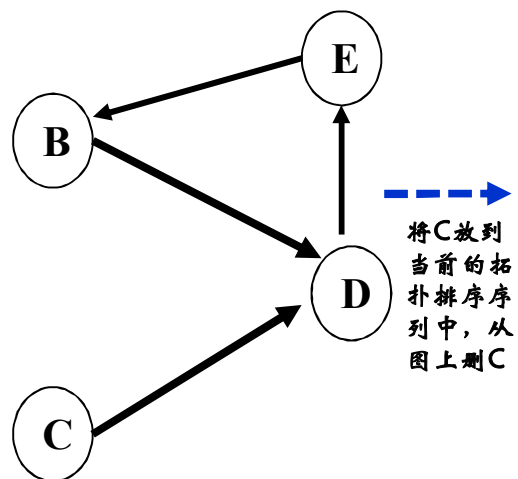


该图的2个拓扑排序序列:

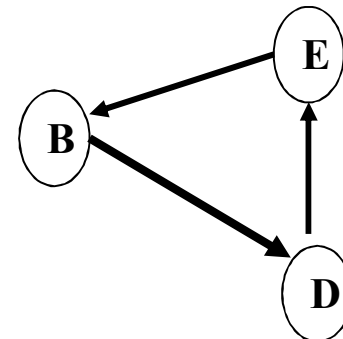
C1	C2	C8	C3	C5	C9	C4	C6	C7
C1	C2	C3	C4	C5	C6	C8	C9	C7



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

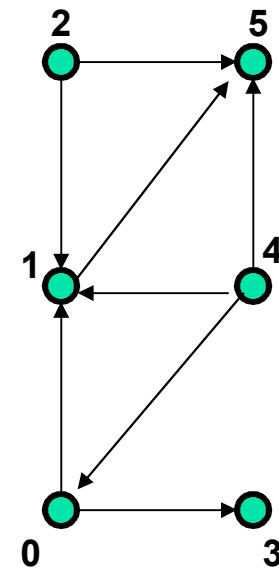
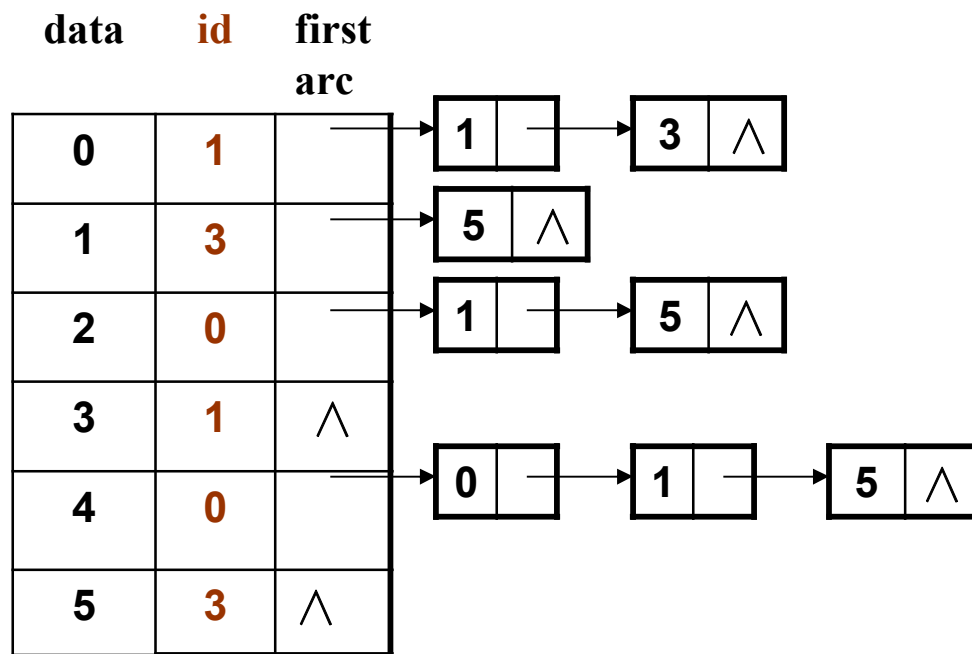
拓扑排序过程演示

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

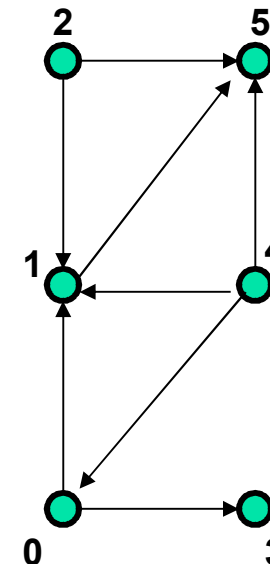
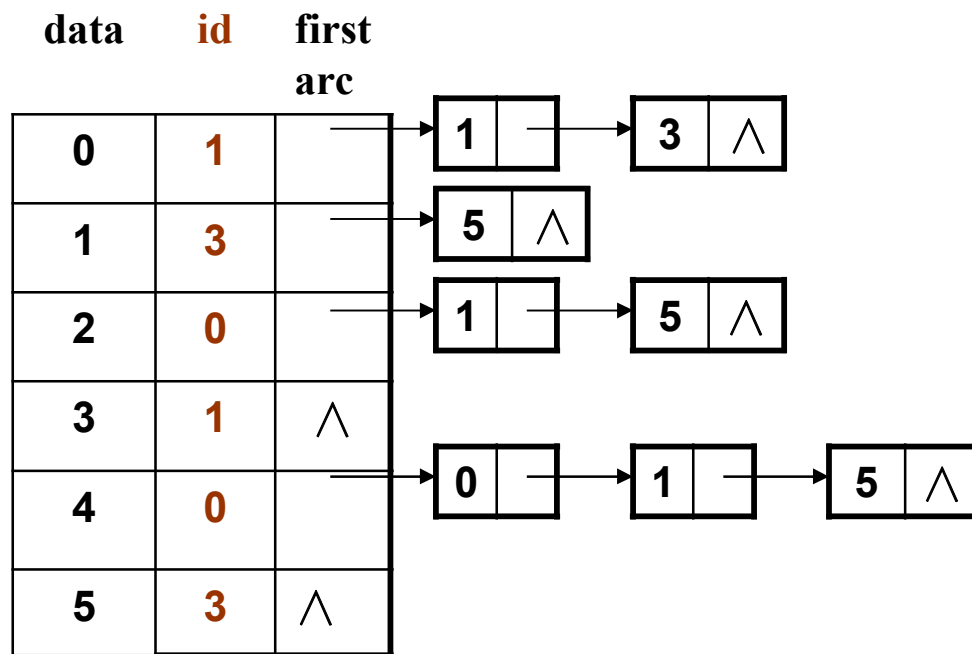


```
#define MAXSIZE 100
```

```
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;

//计算器count中存放已经放到拓扑排序序列中的顶点数

2. while (队列非空)

{ 将队头顶点 v 输出; count++;

for(每条弧 $\langle v, u \rangle$)

{ 将点 u 的入度减1;

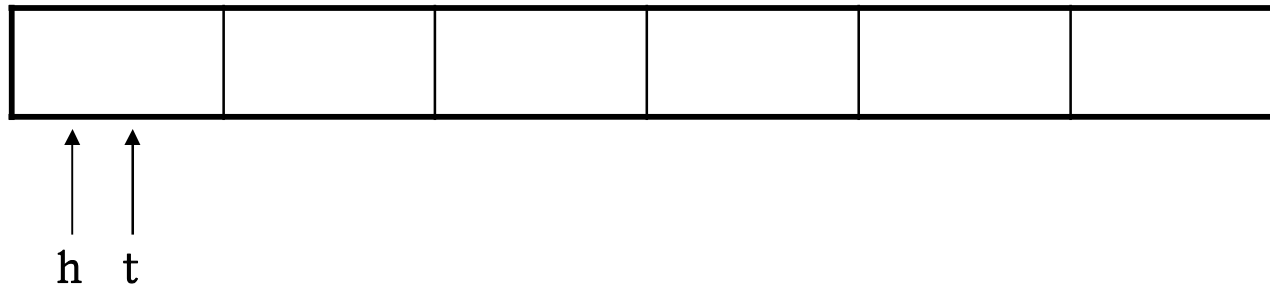
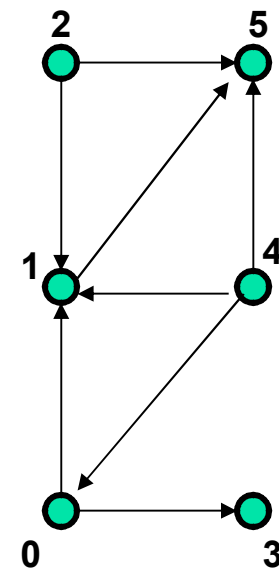
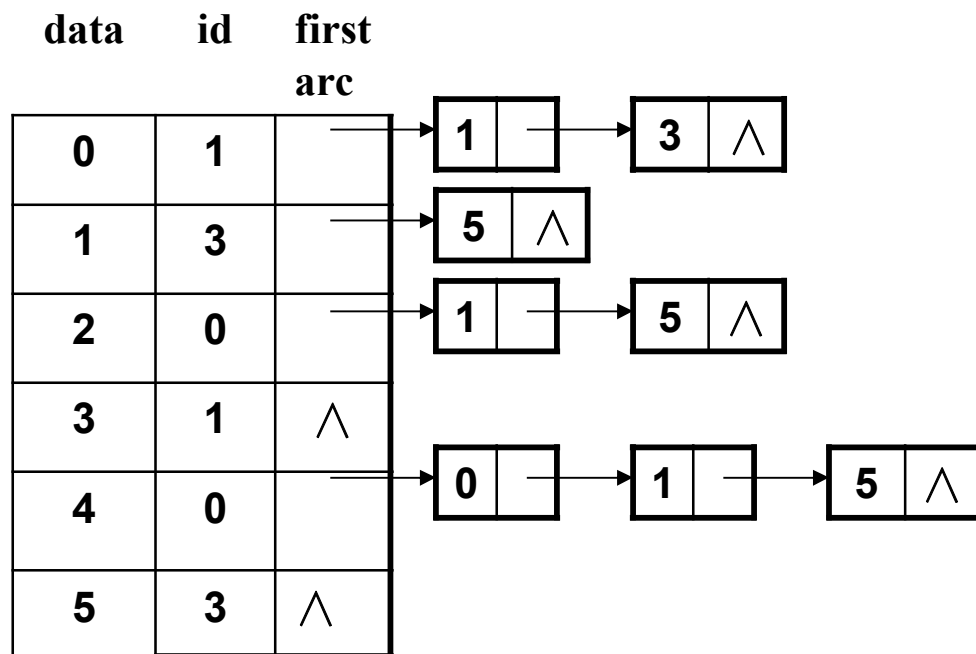
若 u 的入度变为0，则把 u 放入队尾; } }

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

//删除 v 和 v 发出去的弧这一操作没有做真正的删除，只是将 v 发出去的弧流向的顶点的入度减“1”

拓扑排序算法

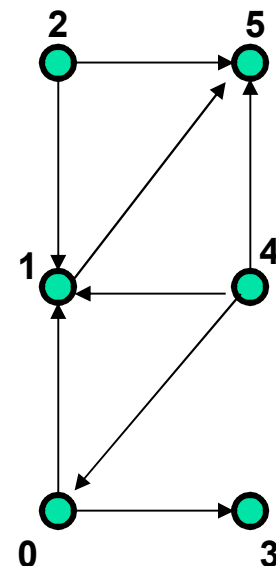
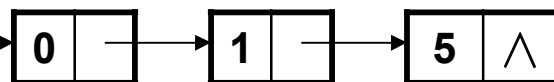
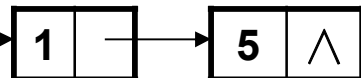
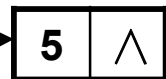
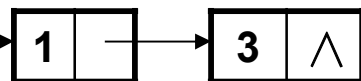
初始化一个空队列



拓扑排序算法

图中顶点2和4入度为0，依次加入队列

data	id	first arc
0	1	1
1	3	5
2	0	1
3	1	^
4	0	0
5	3	^



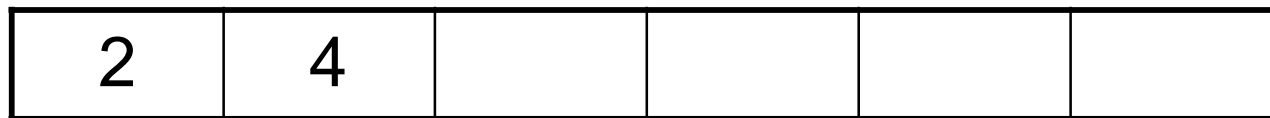
↑
h

↑
t

拓扑排序算法

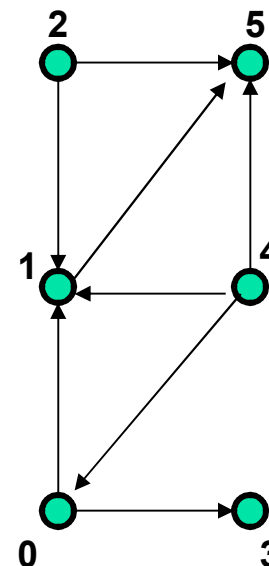
图中顶点2和4入度为0，依次加入队列

data	id	first arc
0	1	1 → 3 ^
1	3	5 ^
2	0	1 → 5 ^
3	1	^
4	0	0 → 1 → 5 ^
5	3	^



↑
h

↑
t



拓扑排序算法：2

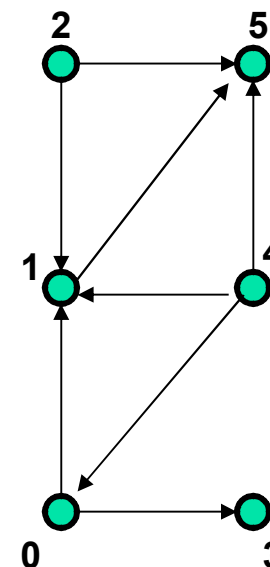
队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“2”加入拓扑排序序列中

data	id	first arc
0	1	1 → 3 ^
1	3	5 ^
2	0	1 → 5 ^
3	1	^
4	0	0 → 1 → 5 ^
5	3	^



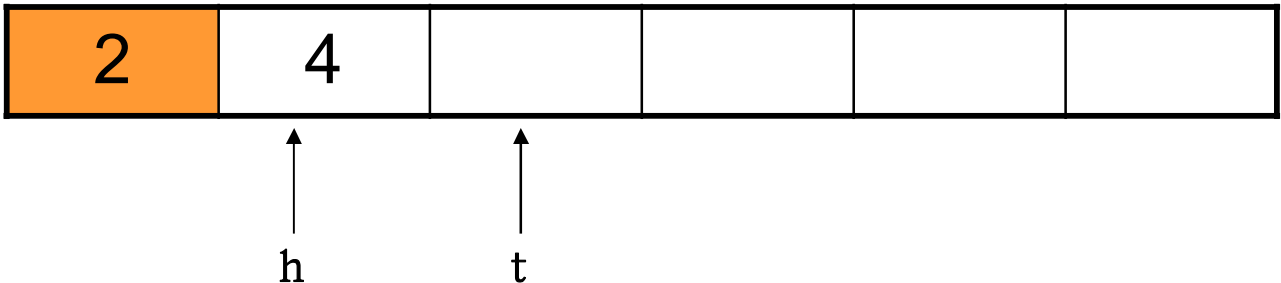
h

t





data	id	first arc
0	1	→ [1 → [3 ^]
1	3	→ [5 ^]
2	0	→ [1 → [5 ^]
3	1	^
4	0	→ [0 → [1 → [5 ^]
5	3	^



t

拓扑排序算法：2

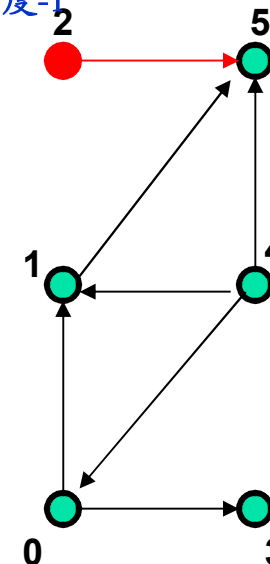
队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“2”加入拓扑排序序列中
“删去”顶点“2”及其发出去的弧----“删”第一条弧<2,1>,顶点1的入度-1

data	id	first	arc
0	1		1 → 3 ^
1	2		5 ^
2	0		1 → 5 ^
3	1	^	
4	0		0 → 1 → 5 ^
5	3	^	



h

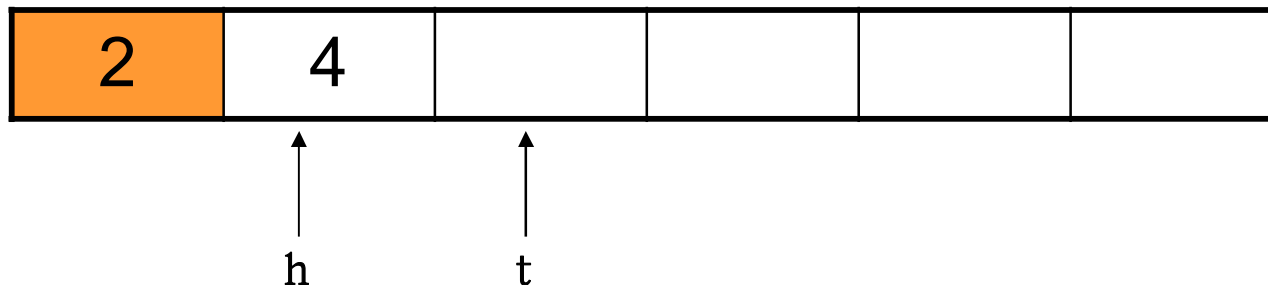
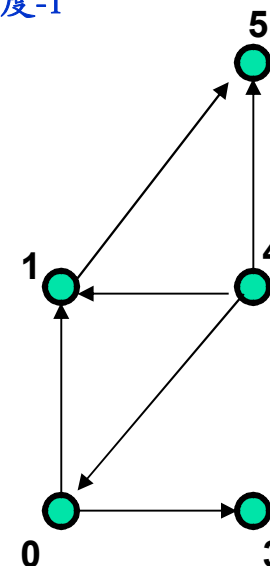
t



拓扑排序算法：2

队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“2”加入拓扑排序序列中
“删去”顶点“2”及其发出去的弧----“删”第二条弧<2,5>,顶点5的入度-1

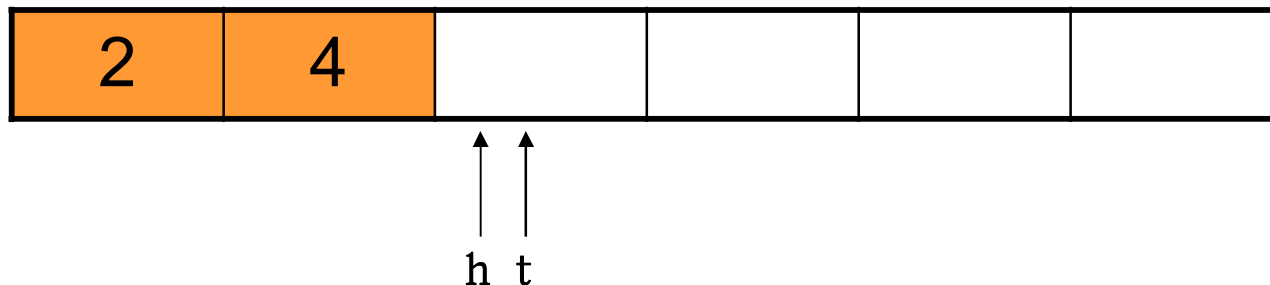
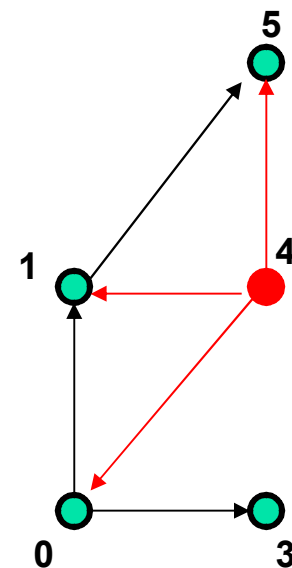
data	id	first	arc
0	1		1 → 3
1	2		5 ^
2	0		1 → 5
3	1	^	
4	0		0 → 1 → 5
5	2	^	



拓扑排序算法：2,4

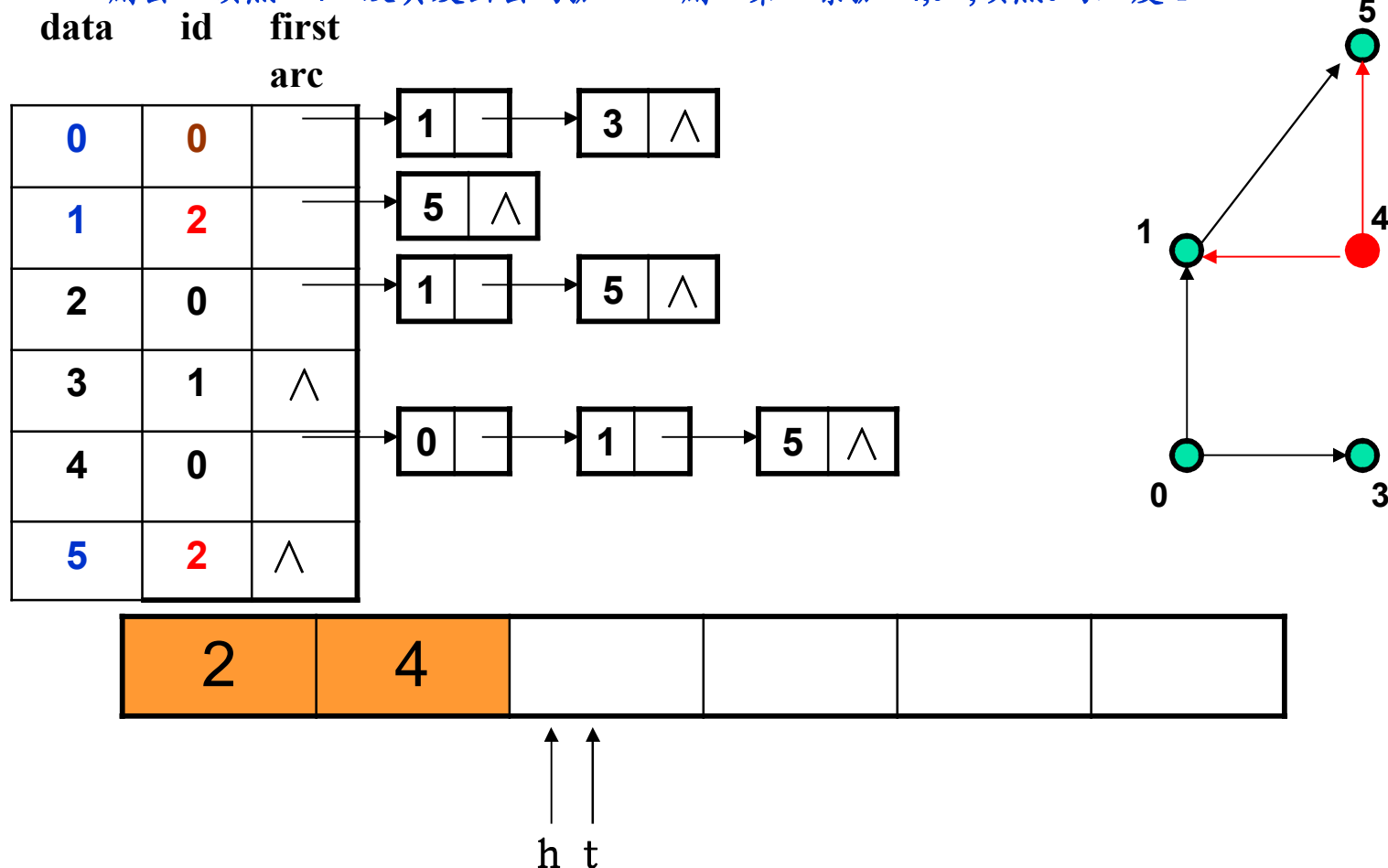
队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“4”加入拓扑排序序列中
“删去”顶点“4”及其发出去的弧

data	id	first	arc
0	1		1 → 3
1	2		5
2	0		1 → 5
3	1	^	
4	0		0 → 1 → 5
5	2	^	



拓扑排序算法：2,4

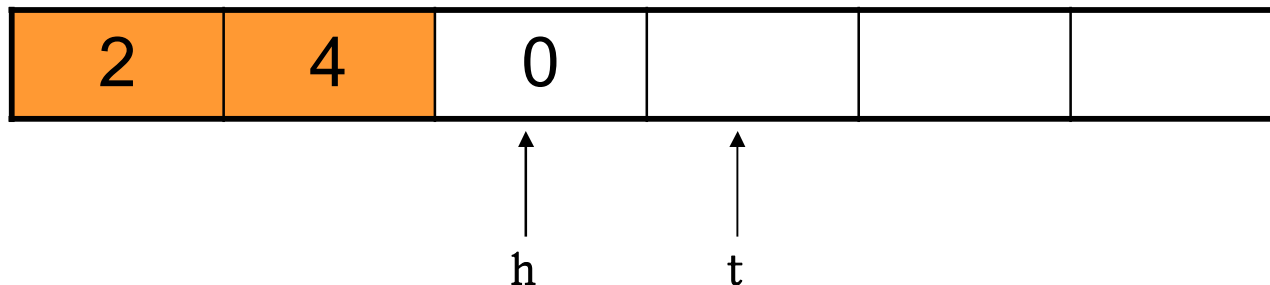
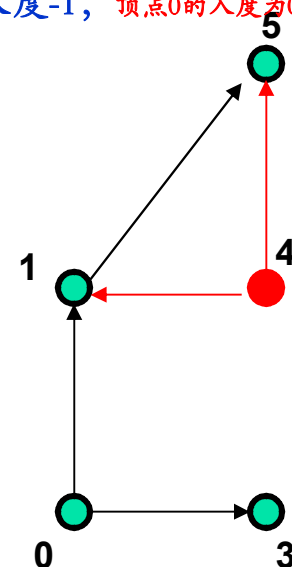
队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“4”加入拓扑排序序列中
“删去”顶点“4”及其发出去的弧----“删”第一条弧<4,0>,顶点0的入度-1



拓扑排序算法：2,4

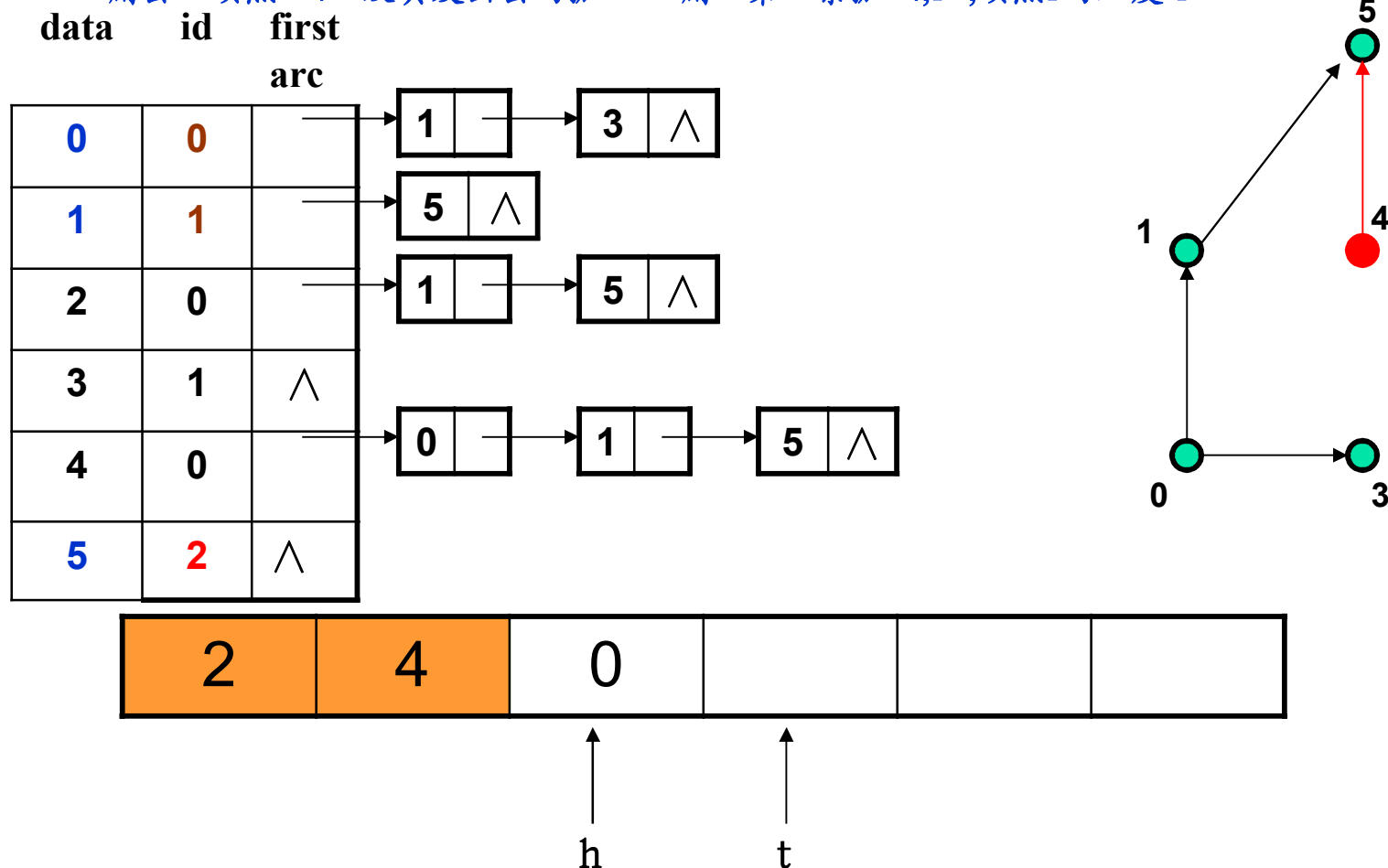
队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“4”加入拓扑排序序列中
“删去”顶点“4”及其发出去的弧----“删”第二条弧<4,0>,顶点0的入度-1, 顶点0的入度为0, 入队

data	id	first	arc
0	0		1 — 3 ^
1	2		5 ^
2	0		1 — 5 ^
3	1	^	
4	0		0 — 1 — 5 ^
5	2	^	



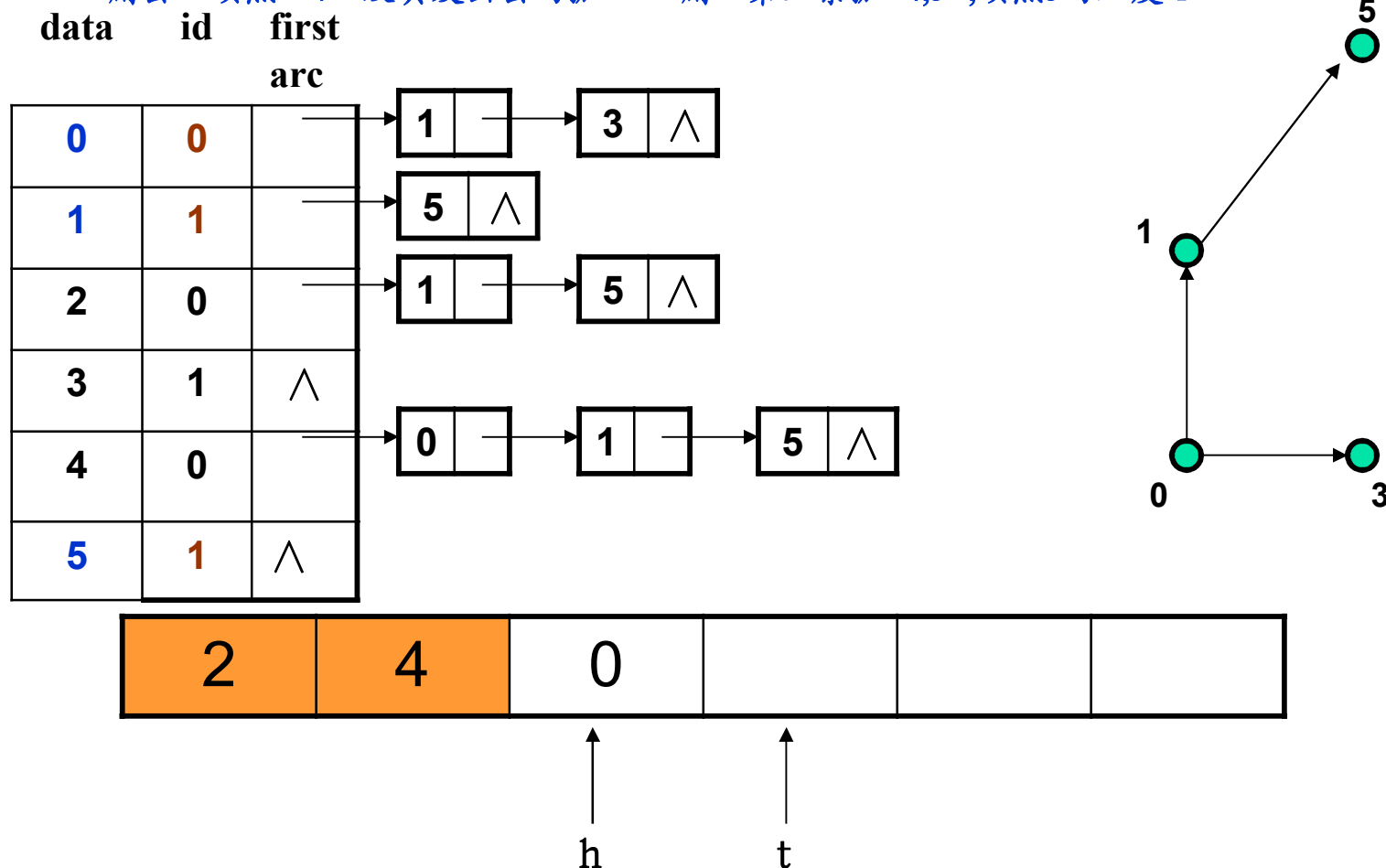
拓扑排序算法：2,4

队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“4”加入拓扑排序序列中
“删去”顶点“4”及其发出去的弧----“删”第二条弧<4,1>,顶点1的入度-1



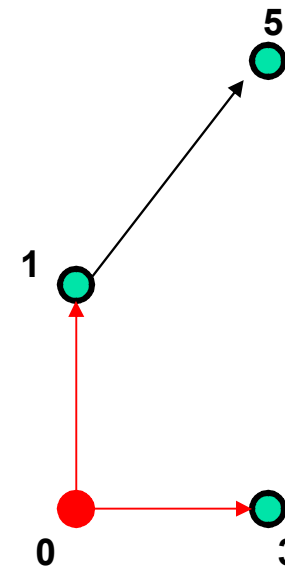
拓扑排序算法：2,4

队列不是空的，还有入度为0的顶点没加入拓扑排序序列，取队首“4”加入拓扑排序序列中
“删去”顶点“4”及其发出去的弧----“删”第三条弧<4,5>,顶点5的入度-1



拓扑排序算法: 2,4,0

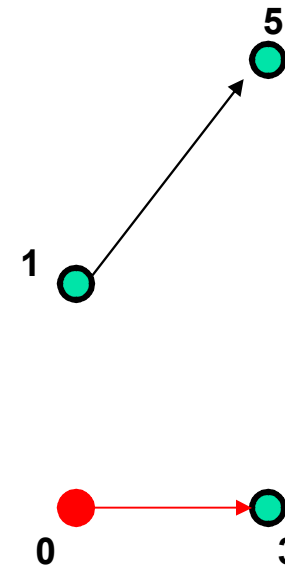
data	id	first arc
0	0	→ [1 -] → [3 ^]
1	1	→ [5 ^]
2	0	→ [1 -] → [5 ^]
3	1	^
4	0	→ [0 -] → [1 -] → [5 ^]
5	1	^



↑ ↑
h t

拓扑排序算法: 2,4,0

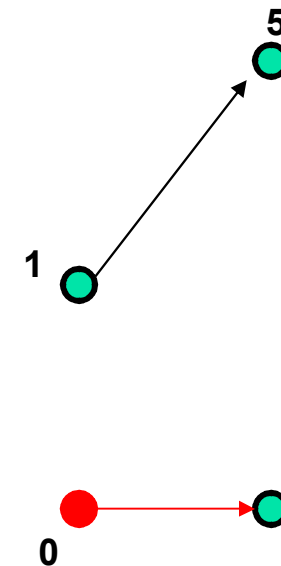
data	id	first	arc
0	0		1 → 3 ^
1	0		5 ^
2	0		1 → 5 ^
3	1	^	
4	0		0 → 1 → 5 ^
5	1	^	



↑ ↑
h t

拓扑排序算法: 2,4,0

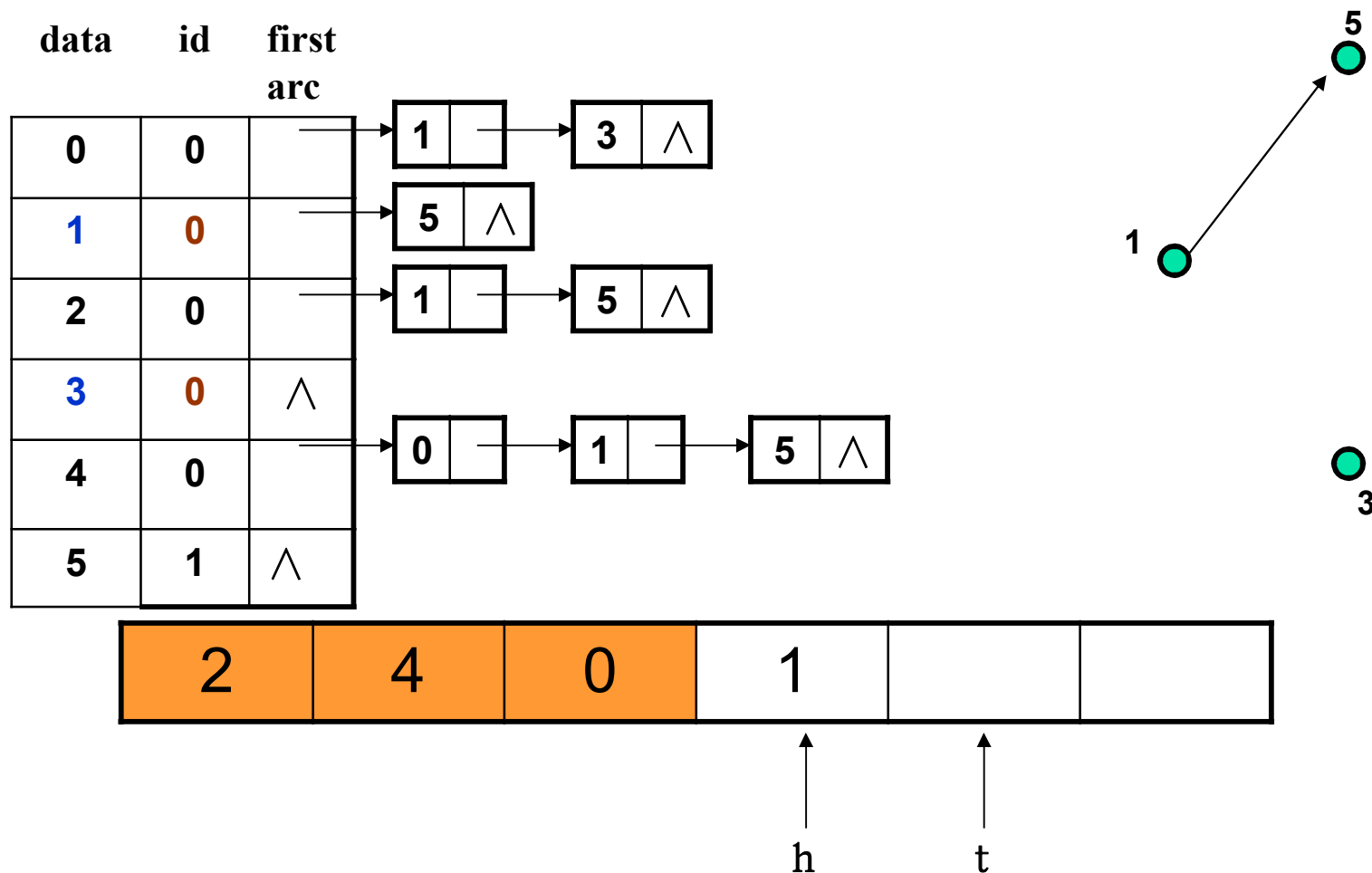
data	id	first arc
0	0	→ [1 -] → [3 ^]
1	0	→ [5 ^]
2	0	→ [1 -] → [5 ^]
3	1	^
4	0	→ [0 -] → [1 -] → [5 ^]
5	1	^



↑
h

↑
t

拓扑排序算法: 2,4,0



拓扑排序算法: 2,4,0

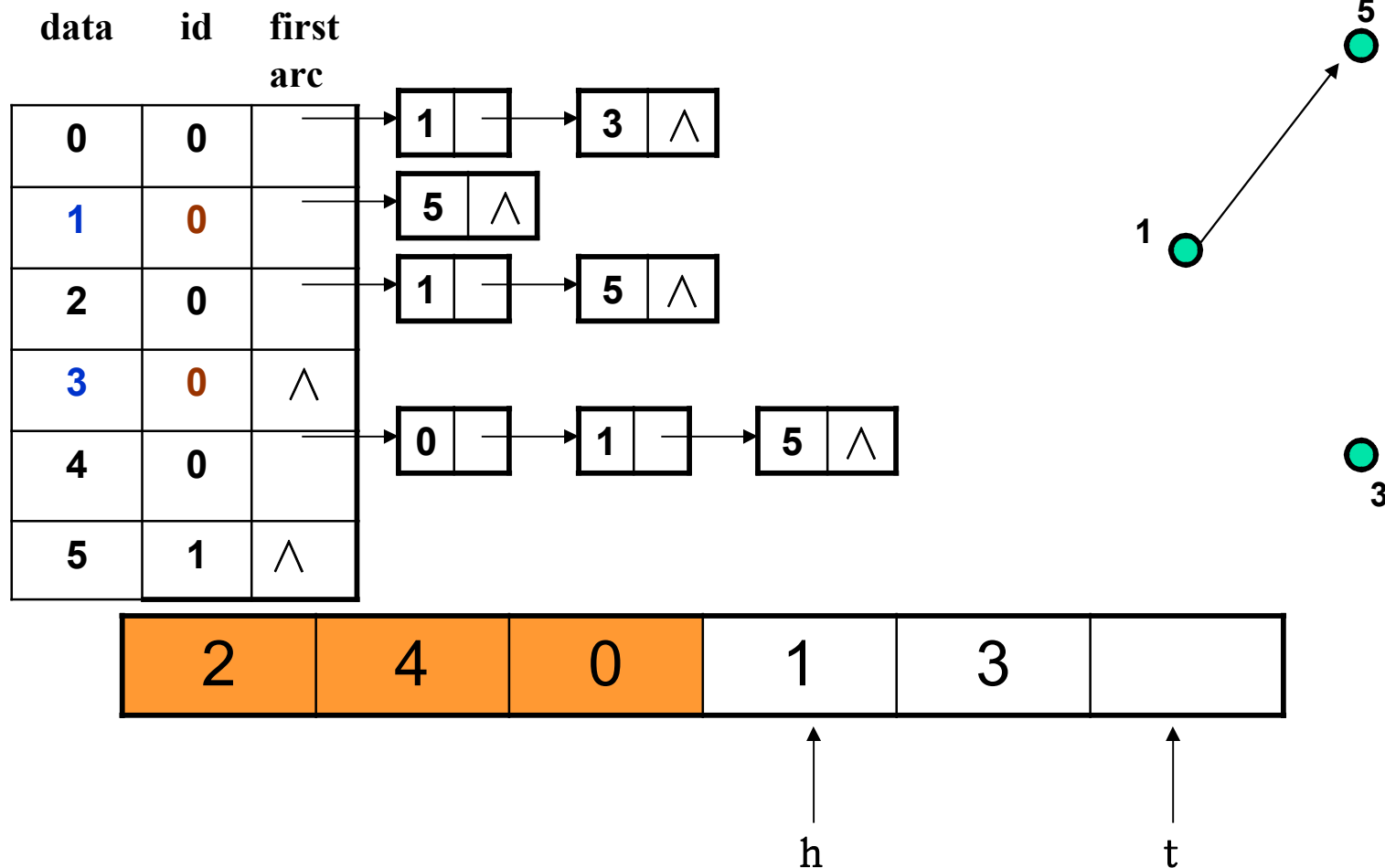
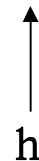
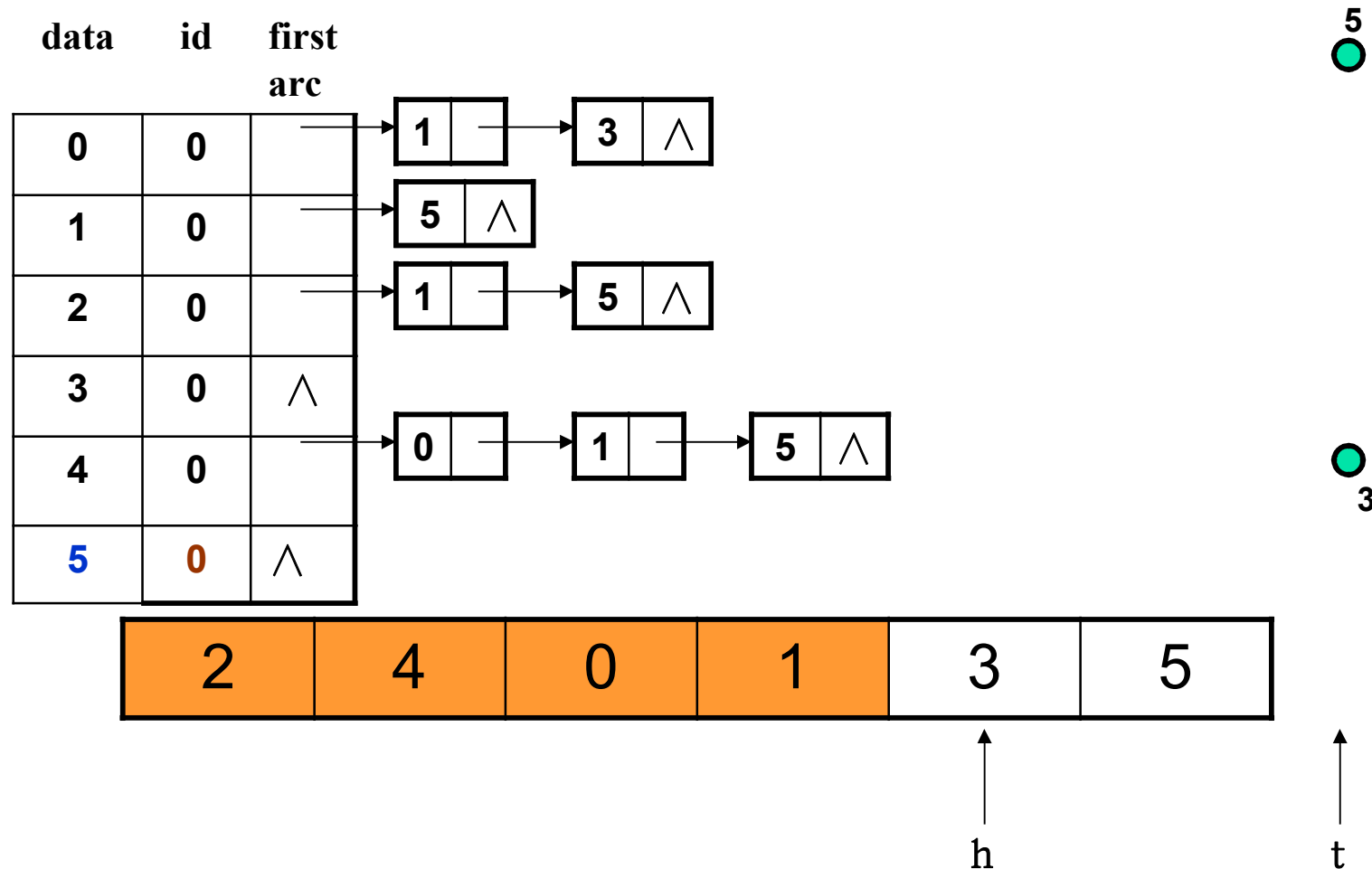


Diagram illustrating three separate linked lists:

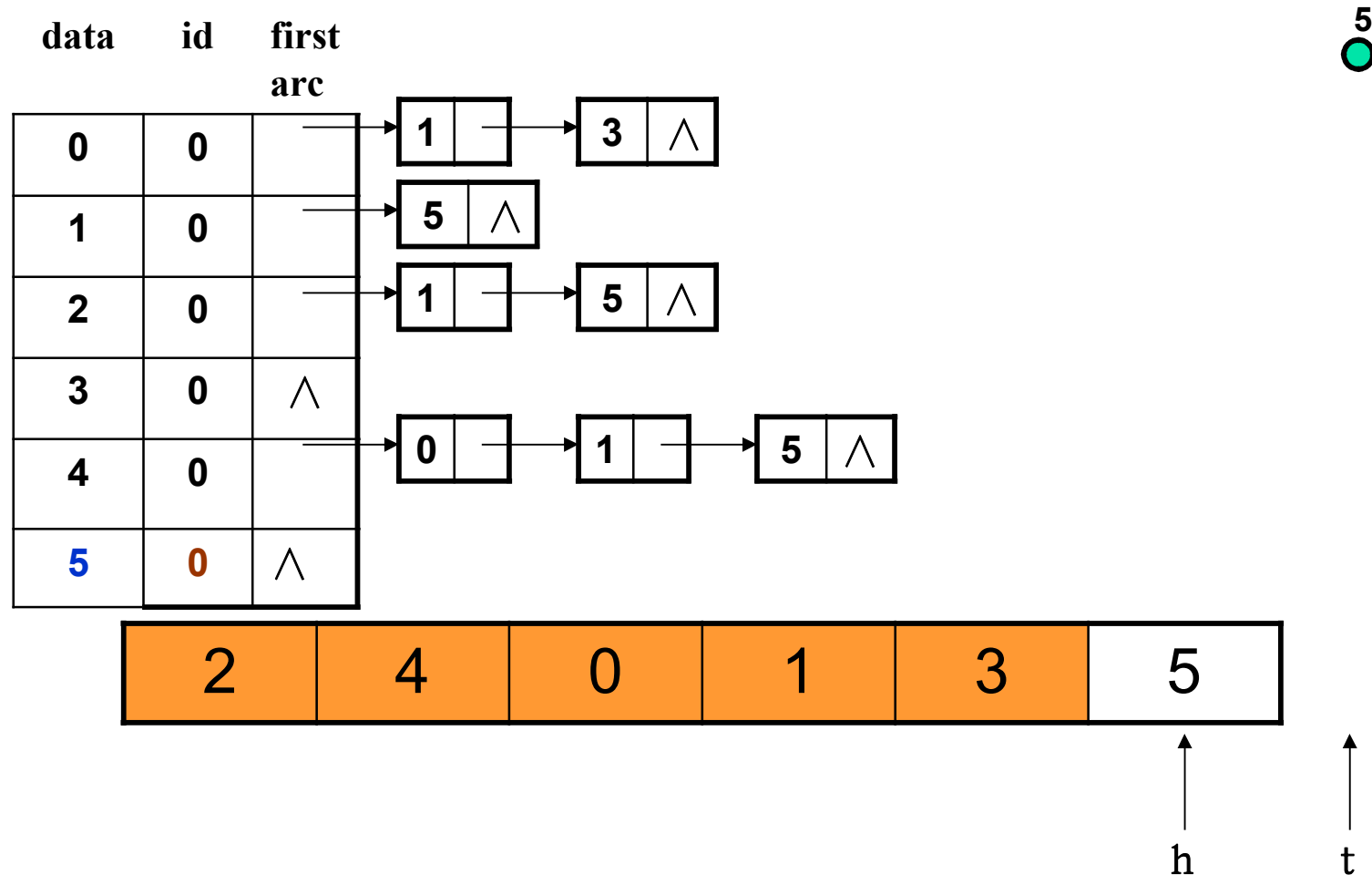
- Node 1 (value 1) points to Node 3 (value 3, null).
- Node 5 (value 5, null).
- Node 1 (value 1) points to Node 5 (value 5, null).



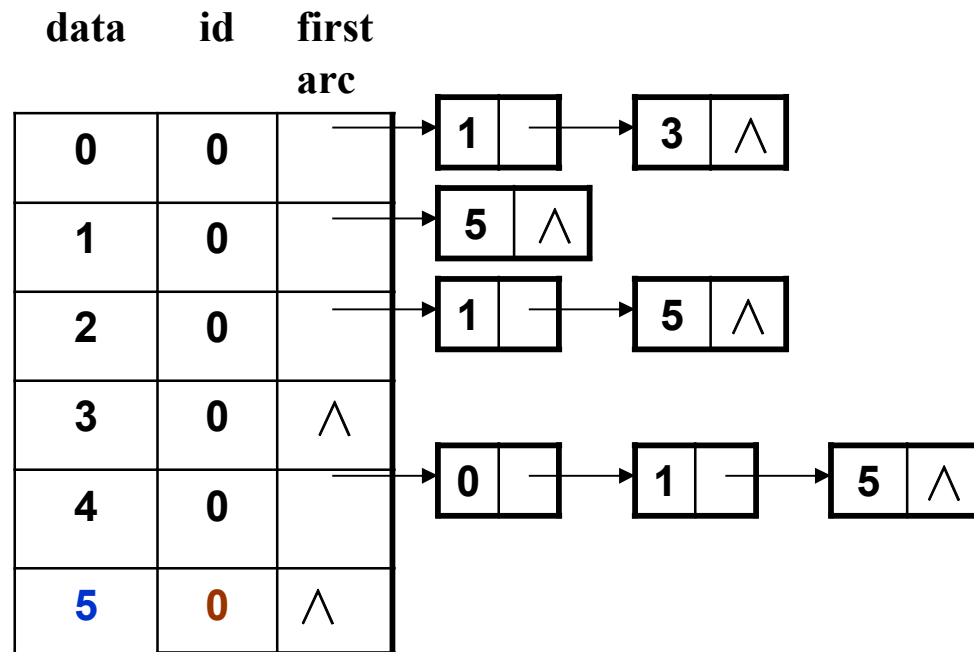
拓扑排序算法: 2,4,0,1



拓扑排序算法: 2,4,0,1,3



拓扑排序算法: 2,4,0,1,5



↑↑
h t



拓扑排序算法

```
int topsort(Graphs T)
{
    int q[MAXSIZE], count, h=t=0;
    ArcNode * p;
    int u,v;
    计算所有顶点入度, 将入度为0 的顶点放入队列; count=0;
    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;
        }
    }
    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

int topsort(Graphs T)

```
{ int q[MAXSIZE], count, h=t=0;    ArcNode * p;    int u,v;

    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. arc[u].id++; } //计算顶点入度
    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!

    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!
      }
    }

    if (count<T.vexnum) {printf(“There is a cycle”); return 0;}
    else return 1;
}
```



拓扑排序算法

- **分析：** 设图有 n 个顶点， e 条边。
- 计算所有顶点的入度，执行时间为 $O(e)$
- 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 $O(e)$
- 总的时间复杂度为 $O(n+e)$