7.4图的应用（续）

DS/SS

连通网的最小生成树

•问题：

用连通网表示的网络上如何构造代价最小的通信网。

对于n个顶点的连通网可以建立许多不同的生成树，每一棵生成树都是一个通信网。一个生成树的代价就是树上各边的代价之和，表示着通信网上总通信耗费量。使通信网上总通信耗费量最小的问题就是求解最小生成树的问题，即如何构造连通网的最小代价生成树

•思路：

利用最小生成树的MST性质。

•方法：

*普里姆(prim)算法；

*克鲁斯卡尔(kruskal)算法。

7.4图的应用（续）

DS/SS

连通网的最小生成树

最小生成树的MST性质：

假设N = (V,{E})是一个连通网，U是顶点集V的一个非空子集。若(u,v)是一条具有最小权值（代价）的边，其中u属于U,v属于V-U，则必存在一棵包含边(u,v)的最小生成树。

证明：（反证法）

假设N = (V,{E})的任何一棵最小生成树都不含边(u,v)。设T是连通网上的一棵最小树，当将边(u,v) 加入到T中时，由生成树的定义，T中必存在一条包含(u,v)的回路。另一方面，由于T是生成树，则在T上必存在另一条边(u',v')，其中u'属于U,v'属于V-U，且u和u'，v和v'之间均有路径相通。删去边(u',v')，便可消除上述回路，同时得到另一棵生成树T'。因为(u,v)的代价不高于(u',v')，则T'的代价亦不高于T，T'是一棵包含(u,v)的最小生成树。矛盾！

7.4图的应用（续）

DS/SS

连通网的最小生成树---prim算法

输入：连通网N = (V, {E})，最小生成树边的集合TE =空集

输出：一棵最小生成树T = (V, {TE})

步骤：

step1.初始：U ={u_0}(u_0属于V),TE = {}；

step2.在所有u属于U，v属于V – U的边(u,v)中找一条代价最小的边(u_0, v_0)并入集合TE，同时v_0 并入U；

step3.重复step2，直至 U = V为止

最后TE中有n-1条边， T = (V, {TE})即为N的一棵最小生成树。

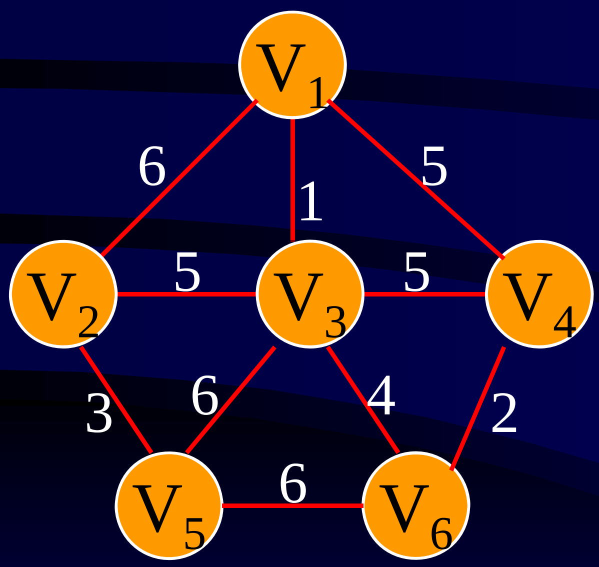
效率：

T(n) = O(n^2)，与e无关（其中n为网中结点个数， e为边的个数）

7.4图的应用（续）

DS/SS

连通网的最小生成树---prim算法



$U=\{v_1\}$, $V-U=\{v_2, v_3, v_4, v_5, v_6\}$, $TE=\{\ \}$

$U=\{v_1, v_3\}$, $V-U=\{v_2, v_4, v_5, v_6\}$, $TE=\{(v_1, v_3)\}$

$U=\{v_1, v_3, v_6\}$, $V-U=\{v_2, v_4, v_5\}$,

$TE=\{(v_1, v_3), (v_3, v_6)\}$

$U=\{v_1, v_3, v_6, v_4\}$, $V-U=\{v_2, v_5\}$,

$TE=\{(v_1, v_3), (v_3, v_6), (v_4, v_6)\}$

$U=\{v_1, v_3, v_6, v_4, v_2\}$, $V-U=\{v_5\}$,

$TE=\{(v_1, v_3), (v_3, v_6), (v_4, v_6), (v_2, v_3)\}$

$U=\{v_1, v_3, v_6, v_4, v_2, v_5\}$, $V-U=\{\}$,

$TE=\{(v_1, v_3), (v_3, v_6), (v_4, v_6), (v_2, v_3), (v_2, v_5)\}$

7.4图的应用（续）

连通网的最小生成树---kruskal算法

输入：连通网N = (V, {E})，最小生成树边的集合TE =空集

输出：一棵最小生成树T = (V, {TE})

步骤：

step1.初始：只有n个顶点而无边的非连通图T = (V,TE={})；

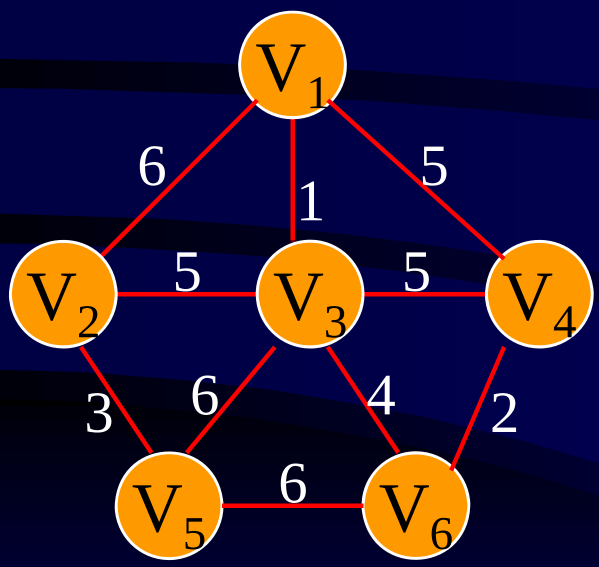
step2.在E中选择代价最小的边，若该边依附的顶点落在T中不同的连通分量上，则将此边加入到T的TE集合中，否则舍去此边而选择下一条代价最小的边；

step3.重复step2，直至 T中所有顶点都在同一个连通分量上为止

效率：

T(n) = O(eloge)，与n无关（其中n为网中结点个数， e为边的个数）

连通网的最小生成树---kruskal算法



$V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$, TE= { }

$V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$, TE= { (v_1,v_3) }

$V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$,

TE= { (v_1,v_3), (v_4,v_6) }

$V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$,

TE= { (v_1,v_3), (v_4,v_6), (v_2,v_5) }

$V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$,

TE= { (v_1,v_3), (v_4,v_6), (v_2,v_5), (v_3,v_6) }

$V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$,

TE ={ (v_1,v_3),(v_3,v_6),(v_4,v_6),(v_2,v_3),(v_2,v_5) }

7.4图的应用（续）

DS/SS

有向无环图中基本概念

- 有向无环图(DAG)：一个无环的有向图。其作用：

  *描述含有公共子式的表达式

  *描述一项工程或系统的进行过程

  #工程能否顺利进行；

  #工程完成所必需的最短时间

- 拓扑排序(Topological Sort)：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。这个全序称为拓扑有序(Topological Order)。
- AOV-网：用顶点表示活动，用弧表示活动间的优先关系的有向图称为顶点表示活动的图(Activity On Vertex Network)。
- AOE-网：顶点表示事件，弧表示活动，权表示活动持续时间的带权有向无环图称为边表示活动的网(Activity On Edge)

7.4图的应用（续）

DS/SS

有向无环图的拓扑排序

为了工程能进行，其对应的AOV-网中不应该存在环。检测的办法有：

*DFS遍历：从有向图上某个顶点v出发的遍历，在DFS(v)结束之前如果出现从顶点u到顶点v的回边，由于u在生成树上是v的子孙，则有向图中必定存在包含顶点v和u的环。

*拓扑排序：对有向图构造其顶点的拓扑有序序列，若网中所有顶点都在它的拓扑有序序列中，则该AOV-网中必定不存在环。

7.4图的应用（续）

DS/SS

有向无环图的拓扑排序算法

输入：AOV-网

输出：包含全部顶点的一个拓扑序列或者部分顶点的序列（存在环）

步骤：

step1.在图中选取一个没有前驱的顶点且输出之；

step2.在图中删除该顶点和所有以它为尾的弧；

step3.重复step1、step2，直至全部顶点均已输出，或者当前图中不存在无前驱的顶点为止（存在环）。

逻辑上：拓扑序列不唯一；

物理上：拓扑序列唯一

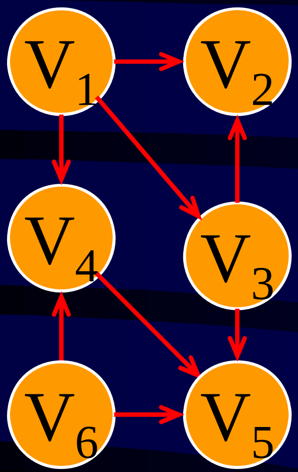
7.4图的应用（续）

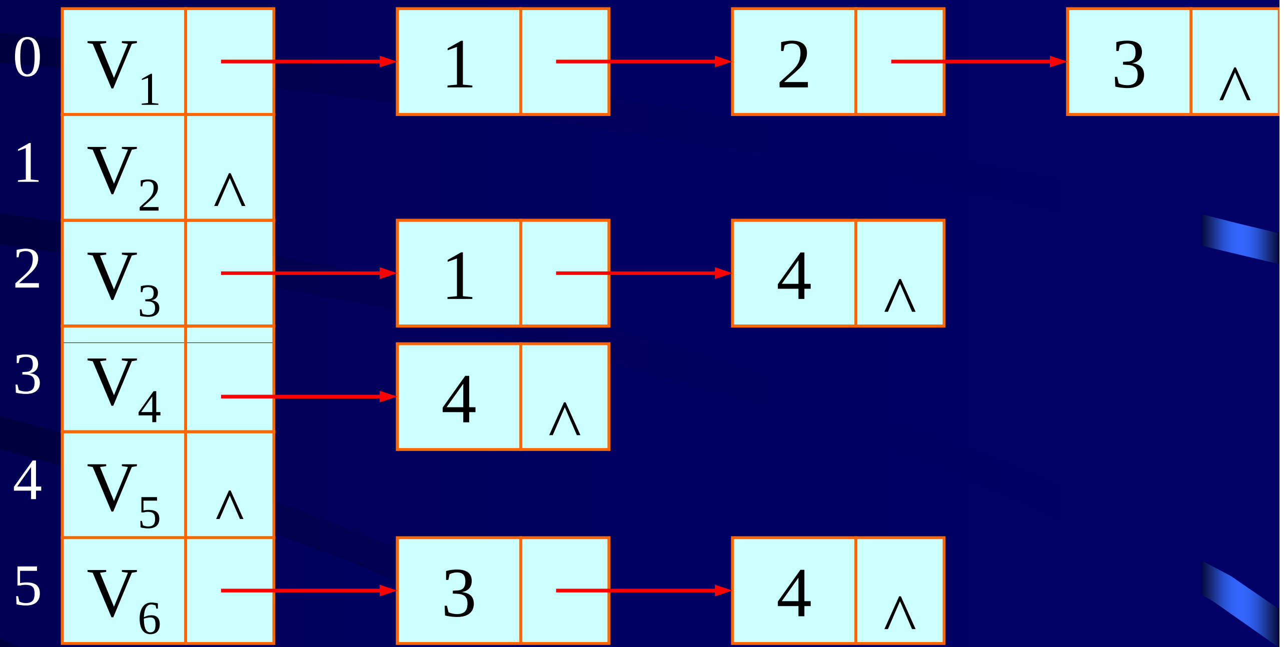
DS/SS

```
Status TopologicalSort(ALGraph G){  //有向图G采用邻接表存储结构
    FindInDegree(G,indegree);        //对各顶点求入度Indegree[0..vexnum-1]--O(e)
    InitStack(S);    //用栈存放所有入度为零的顶点
    for (i =0;i<G.vexnum;++i)     if(!indegree[i]) Push(S,i);      //入度为0进栈--O(n)
    count = 0;        //输出顶点计数
     while (!StackEmpty(S)) {
            pop(S,i);printf(i,G.vertices[i].data); ++count;
            for (p =G.vertices[i].firstarc; p; p =p->nextarc) {
                k = p->adjvex;              //对i号顶点的每个邻接点的入度减1
                if (!(--indegree[k])) Push(S,k);     } //for                    --O(e)
            if(count<G.vexnum)  return ERROR;        else   return OK;
}    // 算法时间复杂度T(n) = O(n +e)
```

DS/SS

有向无环图的拓扑排序算法





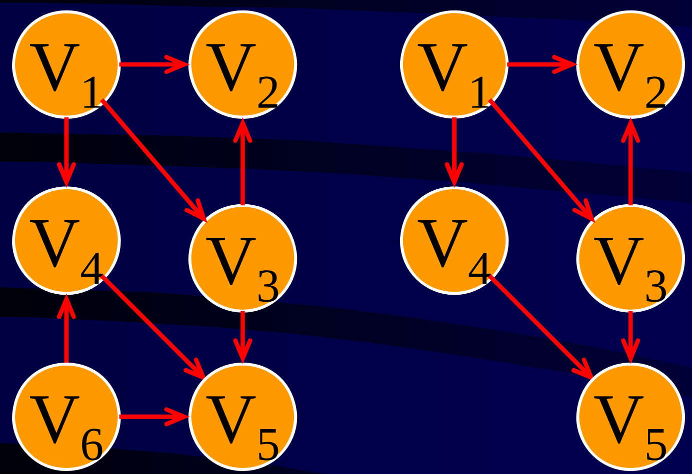
0	2	1	2	3	0
0	1	2	3	4	5

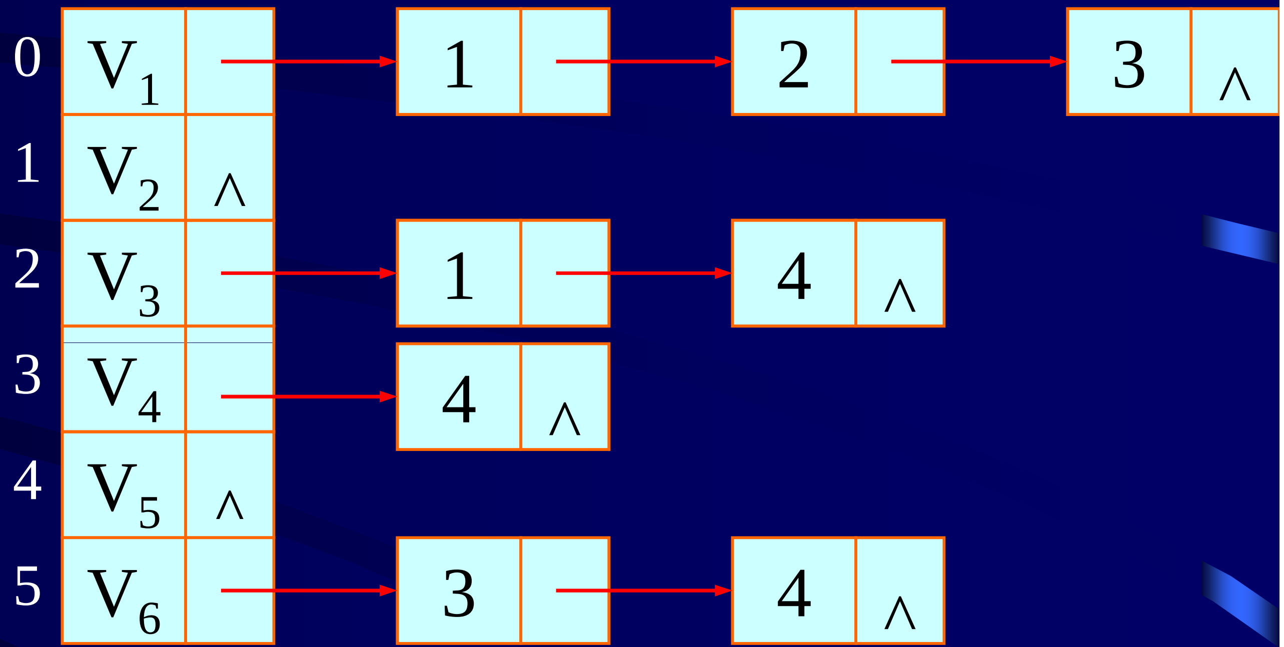
制作：李青山

7.4图的应用（续）

DS/SS

有向无环图的拓扑排序算法

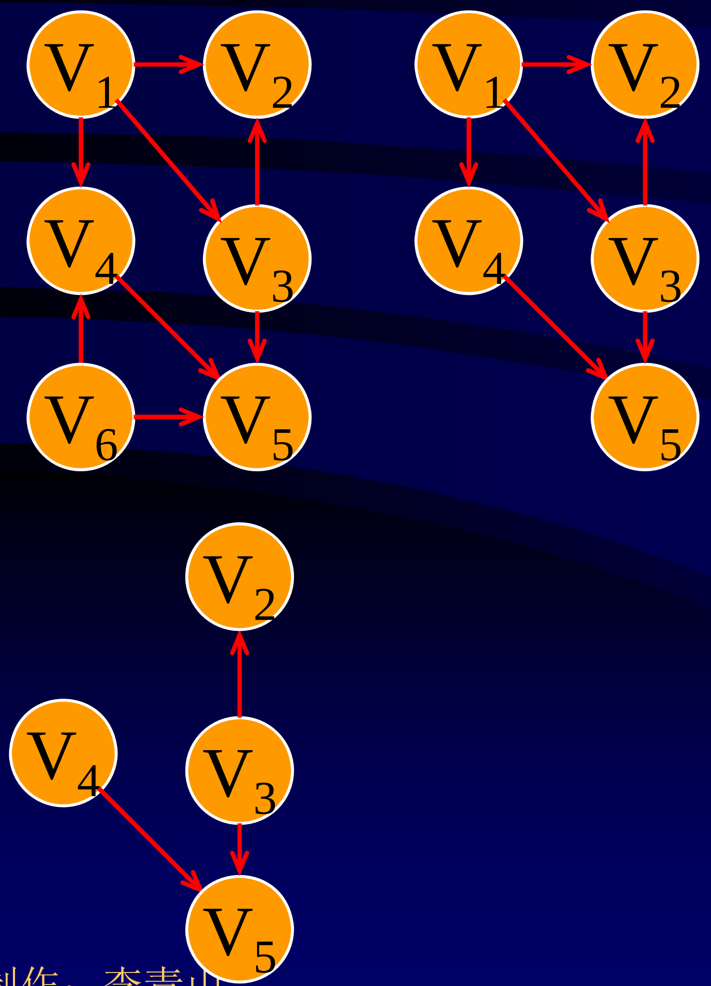


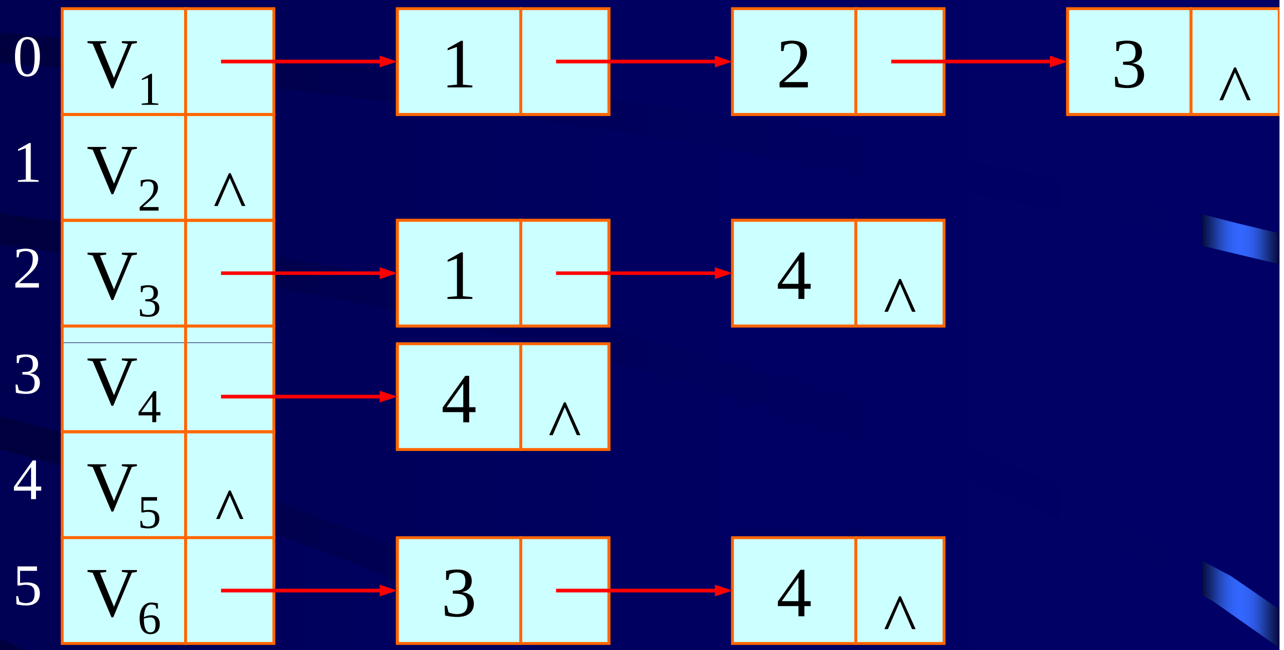


0	2	1	1	2	
0	1	2	3	4	5

7.4图的应用（续）

有向无环图的拓扑排序算法

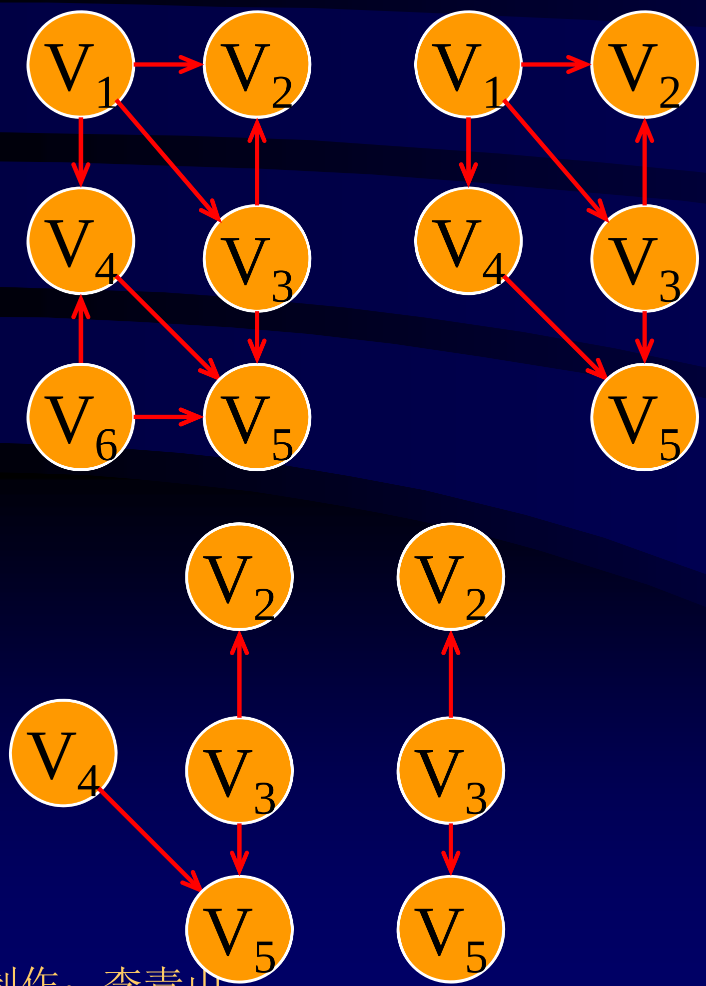


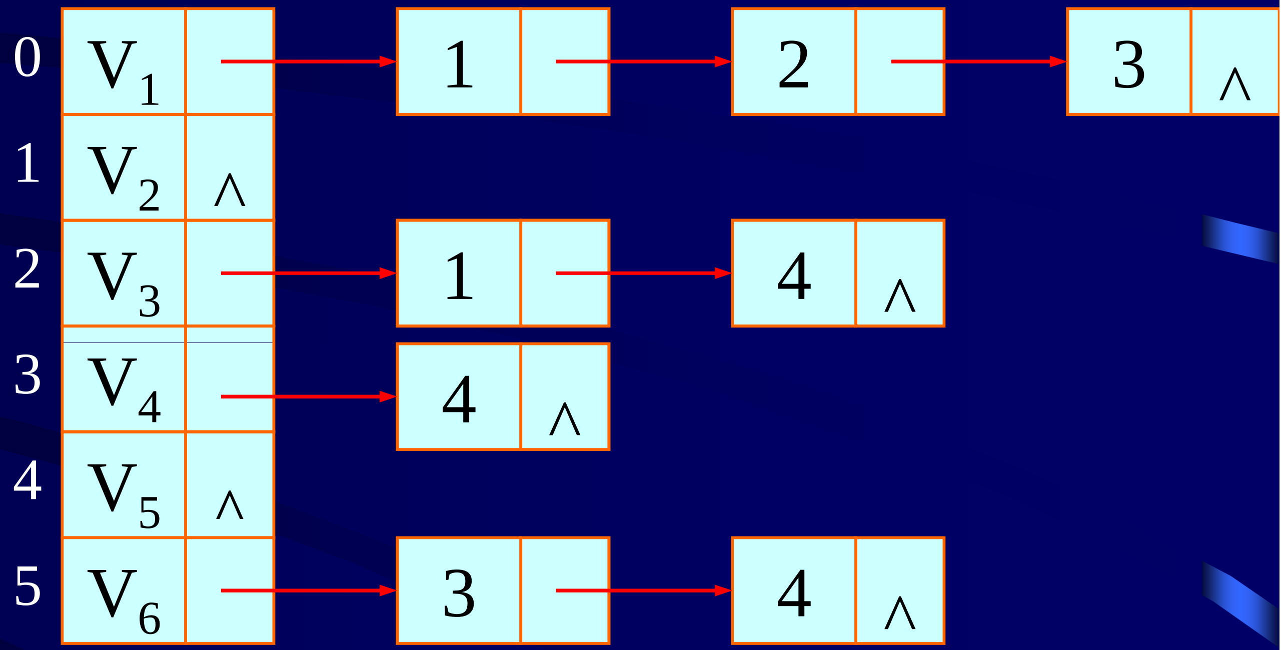


0	1	2	3	4	5
	1	0	0	2	

7.4图的应用（续）

有向无环图的拓扑排序算法

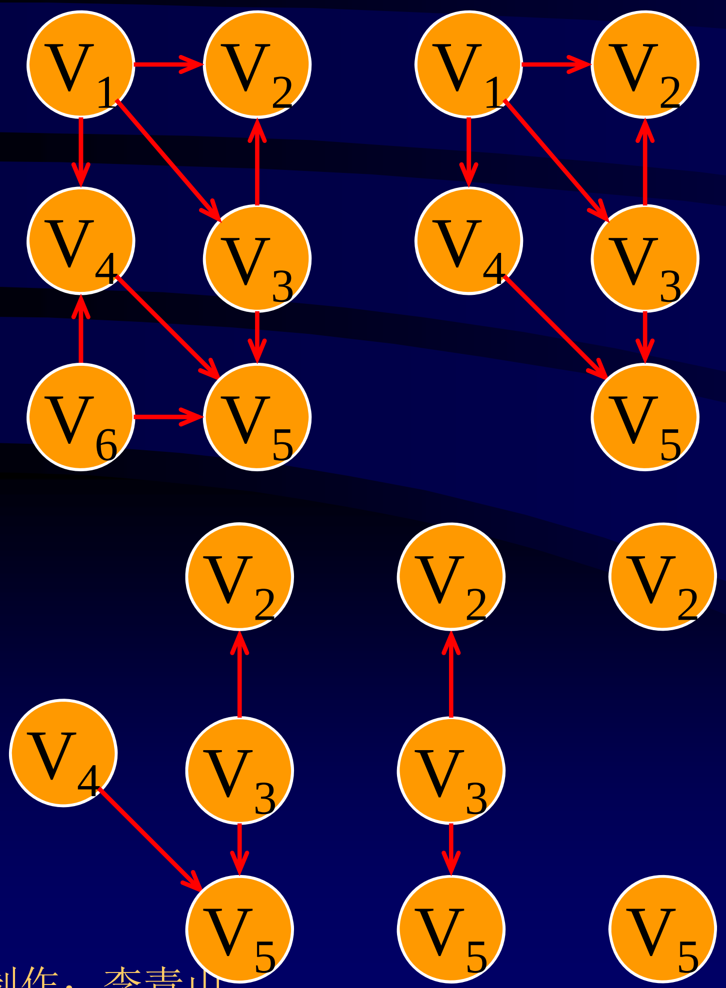


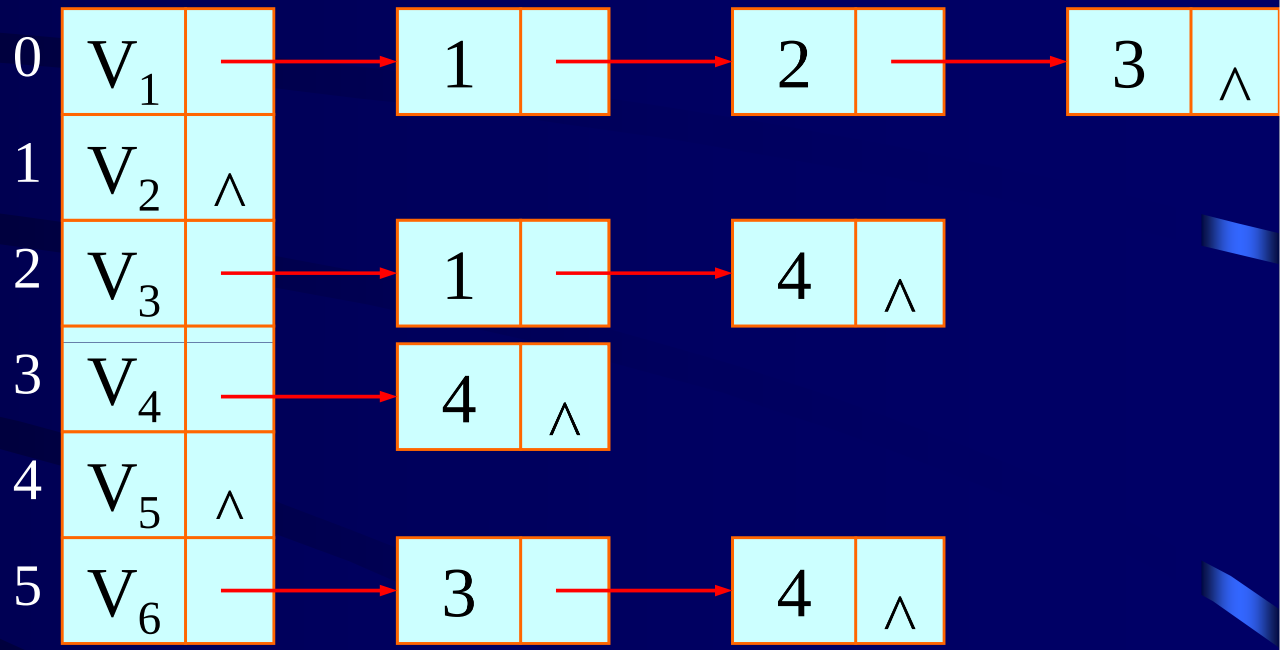


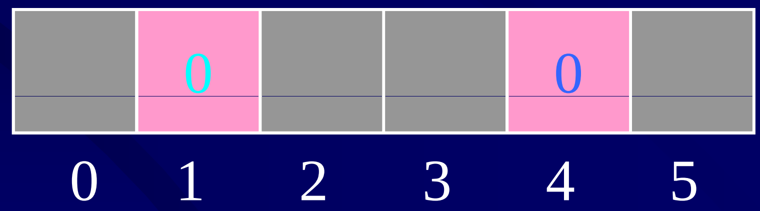
0	1	2	3	4	5
	1	0		1	

7.4图的应用（续）

有向无环图的拓扑排序算法

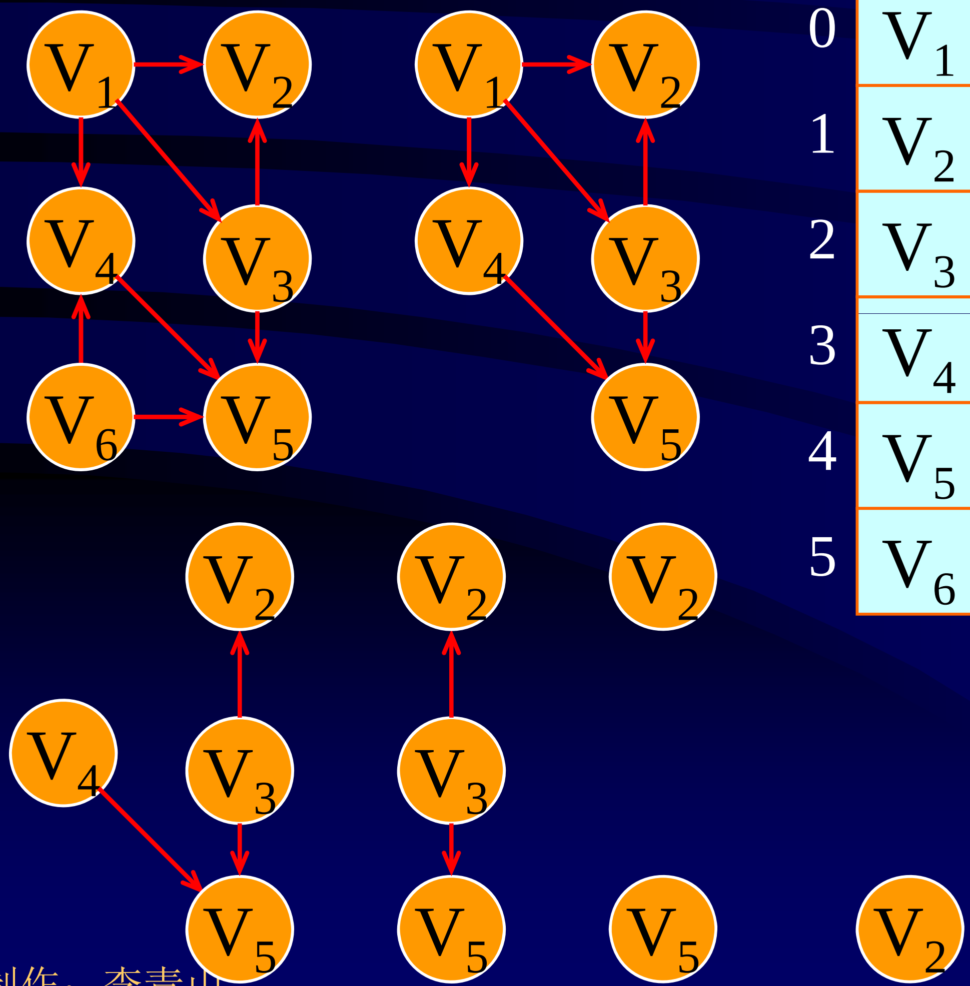


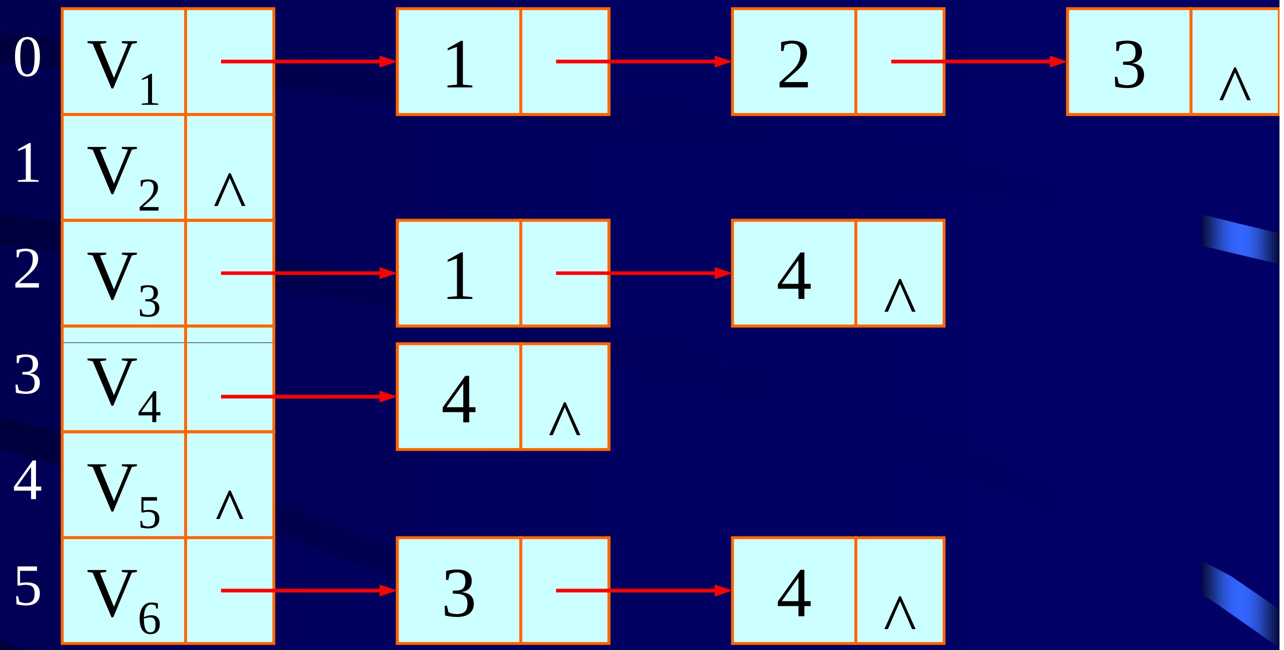


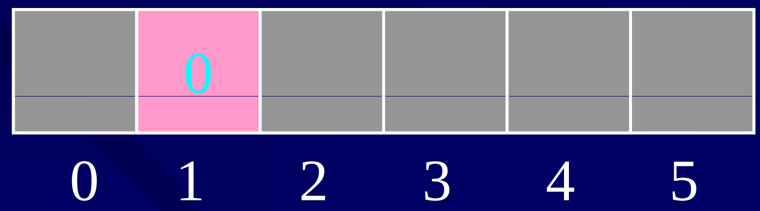


7.4图的应用（续）

有向无环图的拓扑排序算法







7.4图的应用（续）

有向无环图的拓扑排序算法