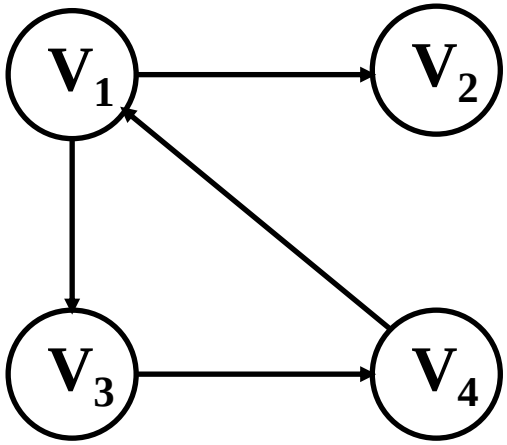
GRAPH STRUCTURE

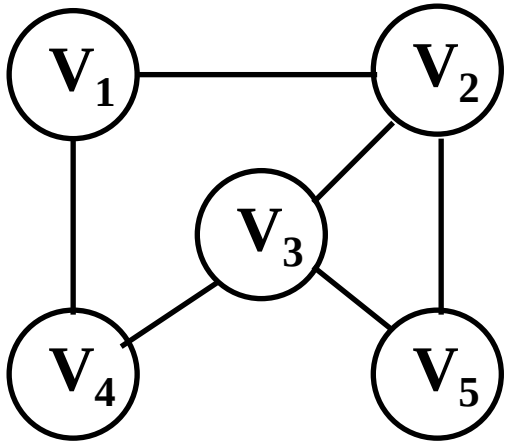
图(Graph)

- 图G=(V,E)，V是顶点集合，E是边(弧)的集合.
- 顶点的度、出度和入度的概念

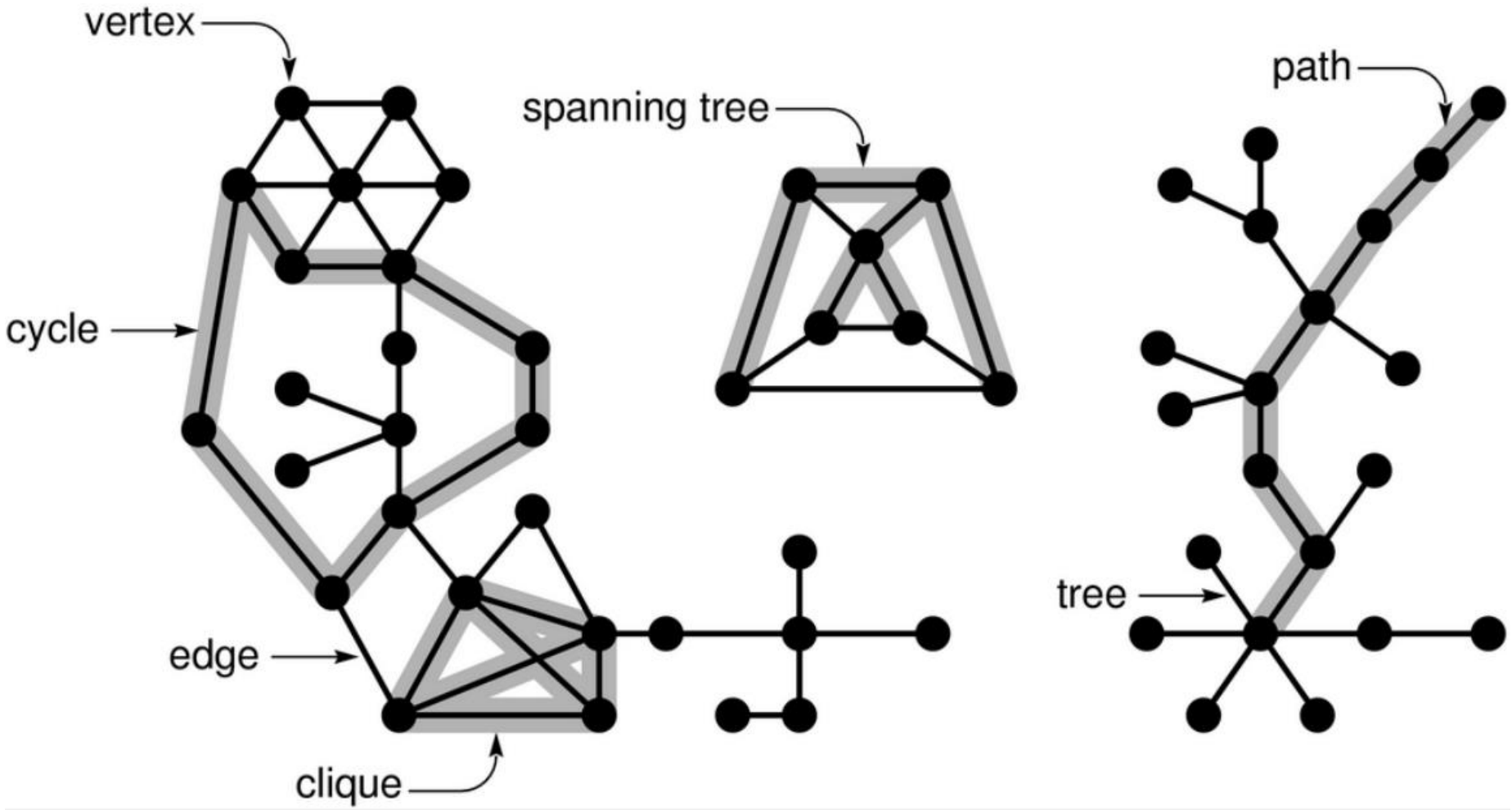
有向图 G_1



无向图 G_2



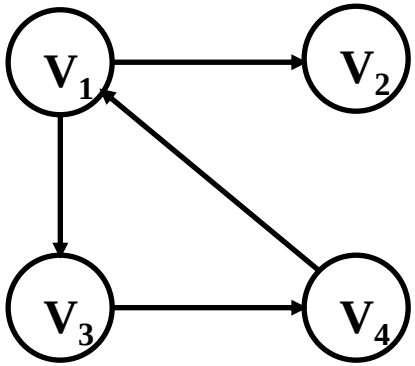
图的相关概念



图的存储

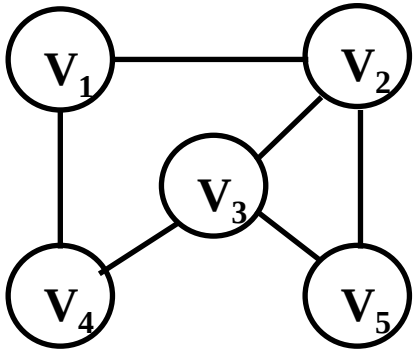
- 试想如何表达下图的信息？

有向图 G_1



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

无向图 G_2

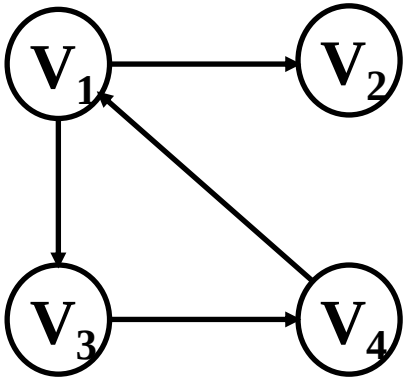


0	1	0	1	0
1	0	1	0	1
0	1	0	1	1
1	0	1	0	0
0	1	1	0	0

- 可用邻接矩阵表达顶点及其关系。
- 根据邻接矩阵，如何判断各顶点的度？

图的存储(cont.)

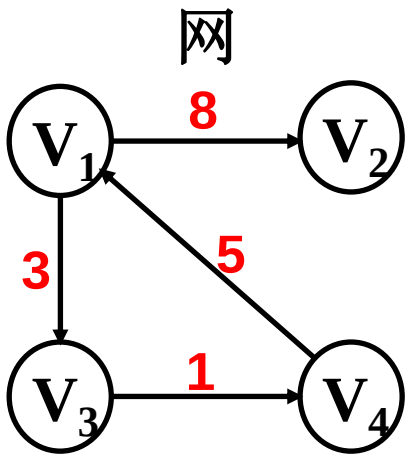
有向图 G_1



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

- 建立二维数组A[n][n]，n=|V|
- 另需存放n个顶点信息
- 此方法直观、简单，但是会有什么问题？
- 现实中的图经常对应稀疏矩阵，在这样情形下会有很大空间浪费.

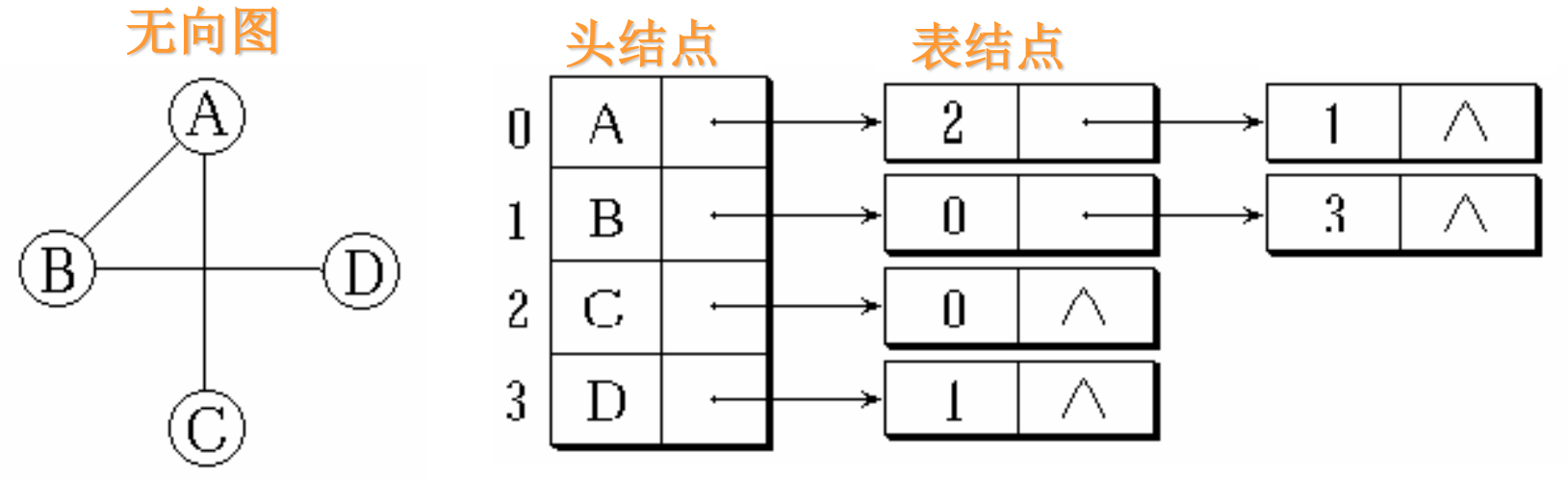
网的邻接矩阵



- 有些图的边带有权重(常用来表示成本、距离、时间等)，这样的图称为网。
- 网的邻接矩阵表达权重，没有边的顶点之间的权重默认为∞.

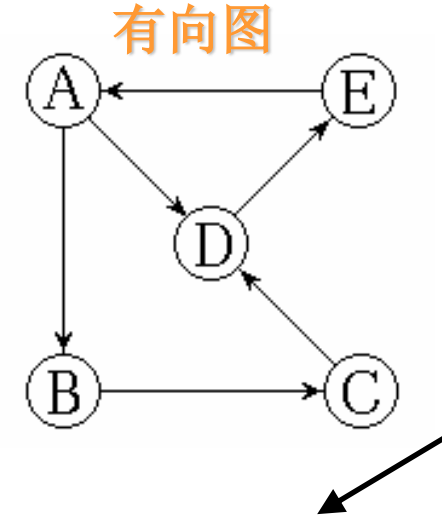
	1	2	3	4
1	∞	8	3	∞
2	∞	∞	∞	∞
3	∞	∞	∞	1
4	5	∞	∞	∞

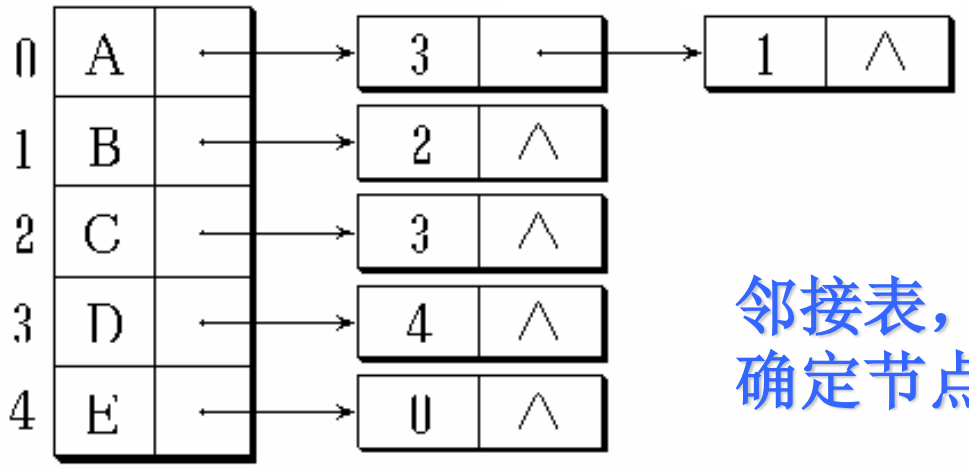
邻接表 (Adjacency List)



- 无向图的邻接表：同一个顶点发出的边链接在同一个边链表中，便于确定顶点的度
- 需要n个头结点，$2e$个表结点

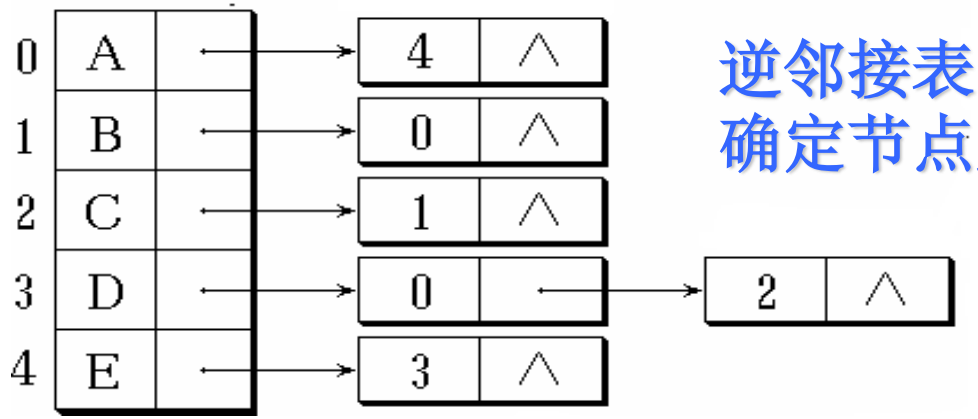
邻接表 (cont.)





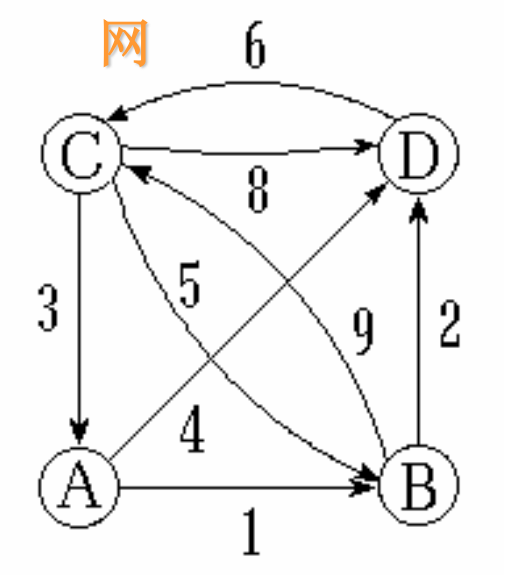
邻接表，便于确定节点出度

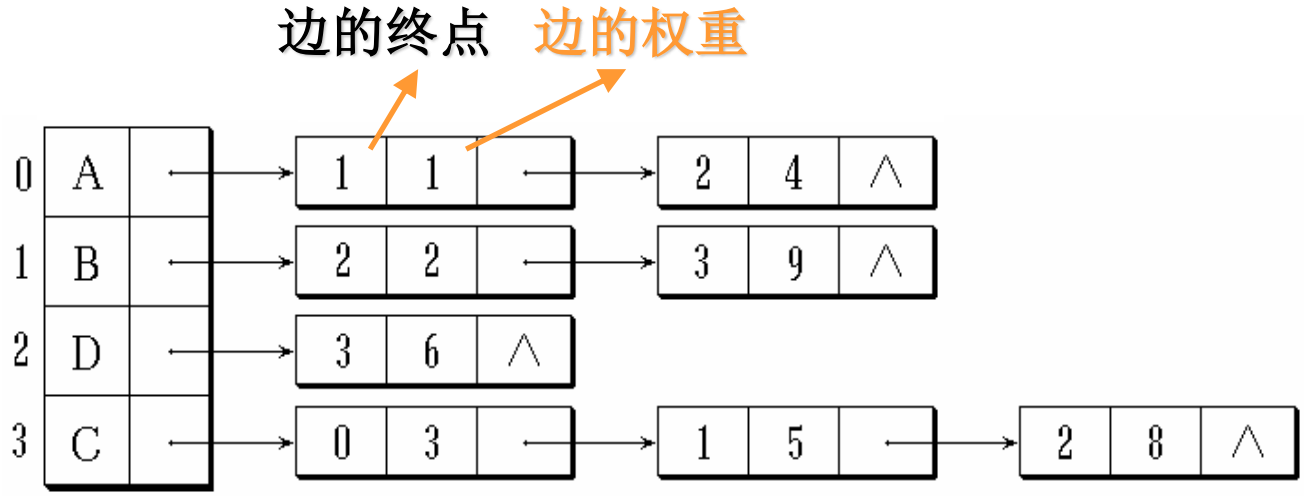
- 需n 个顶点结点，*e*个表结点



逆邻接表，便于确定节点入度

邻接表 (cont.)





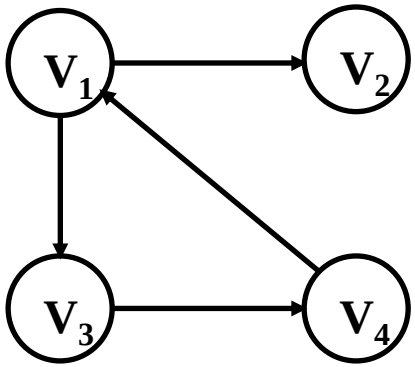
课堂练习

- **1. 请写出数组存储和邻接表的类型定义**

- **2. 请在如下方面对比数组表示法和邻接表示法**
  - **存储表示是否唯一**
  - **空间复杂度**
  - **操作a：求顶点Vi的度**
  - **操作b：判定(Vi，Vj)是否是图的一条边**
  - **操作c：通过遍历求边的数目**

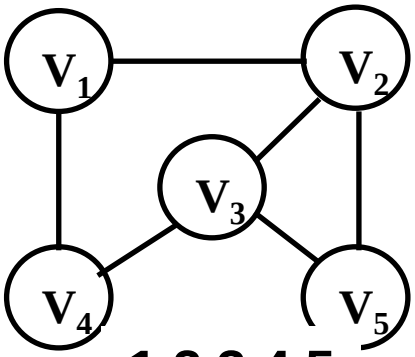
- 1. 请写出使用数组和邻接表的存储表示

有向图 G_1



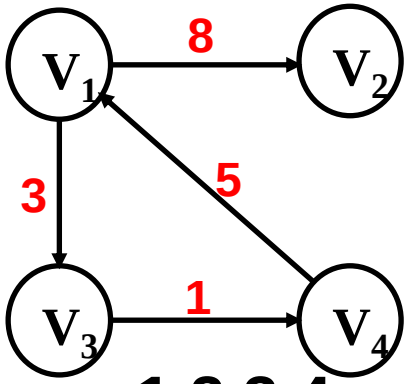
	1	2	3	4
1	0	1	1	0
2	0	0	0	0
3	0	0	0	1
4	1	0	0	0

无向图 G_2



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

网



	1	2	3	4
1	∞	8	3	∞
2	∞	∞	∞	∞
3	∞	∞	∞	1
4	5	∞	∞	∞

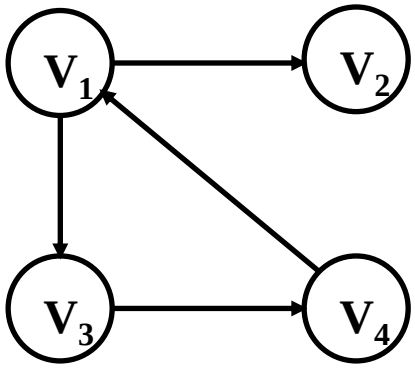
顺序存储法

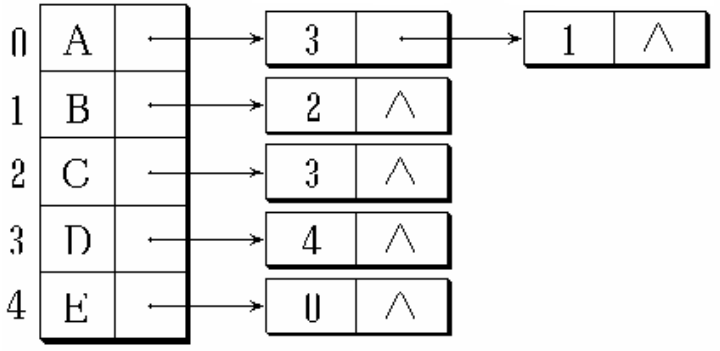
```
typedef struct Vertex {
  int VID;
  char data;
}Vertex;
```

```
typedef struct Graph {
  int AdjMatrix[VNum][VNum];//邻接矩阵
  Vertex VertexSeq[Vnum]; //顶点序列
  int GraphKind; //用1, 2, 3, 4表示有向图…
  int n, e;//顶点数和边数
}Graph;
```

- **1. 请写出使用数组和邻接表的存储表示**

有向图 G_1





邻接表

```
typedef struct LGraph {
  Vnode AdjList[VNum];//邻接表
  int GraphKind;
  int n, e;
}LGraph;
```

```
typedef struct VNode{
  VInfoType vInfo;
  ENode *firstAdj;
};
```

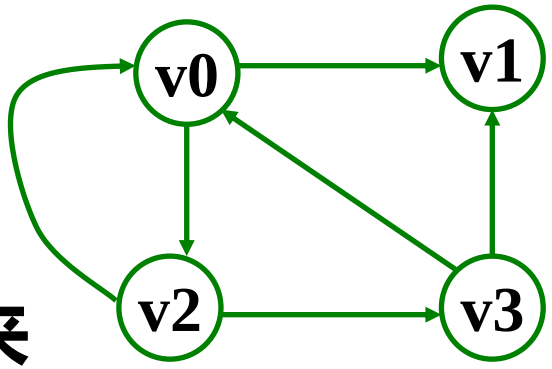
```
typedef struct ENode{
  int AdjVNode;
  ENode *nextAdj;
};
```

	数组表示法	邻接表法
存储表示结果	惟一	不惟一
空间复杂度	**$O(n^2)$** 适用于稠密图	**$O(n+e)$** 适用于稀疏图
求顶点Vi的度	无向图：第i行(或第i列)上非零元素的个数	无向图：第i个边表中的结点个数
	有向图：第i行上非零元素的个数是Vi出度，第i列上非零元素的个数是Vi的入度	有向图：第i个边表上的结点个数，求入度还需遍历各顶点的边表。逆邻接表则相反。
判定(Vi，Vj)是否是图的一条边	看矩阵中的i行j列是否为0	扫描第i个边表
求边的数目	检测整个矩阵中的非零元所耗费的时间是$O(n^2)$	对每个边表的结点个数计数所耗费的时间是$O(e+n)$

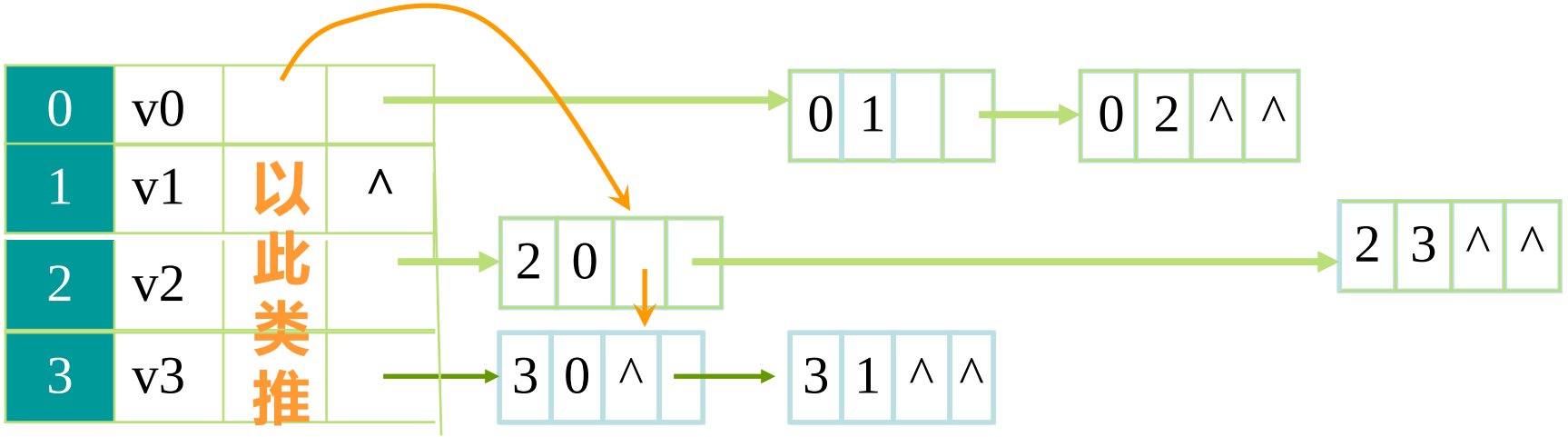
思考尝试

- 怎么把邻接表和逆邻接表相结合，同时表示出来？

有向图的十字链表



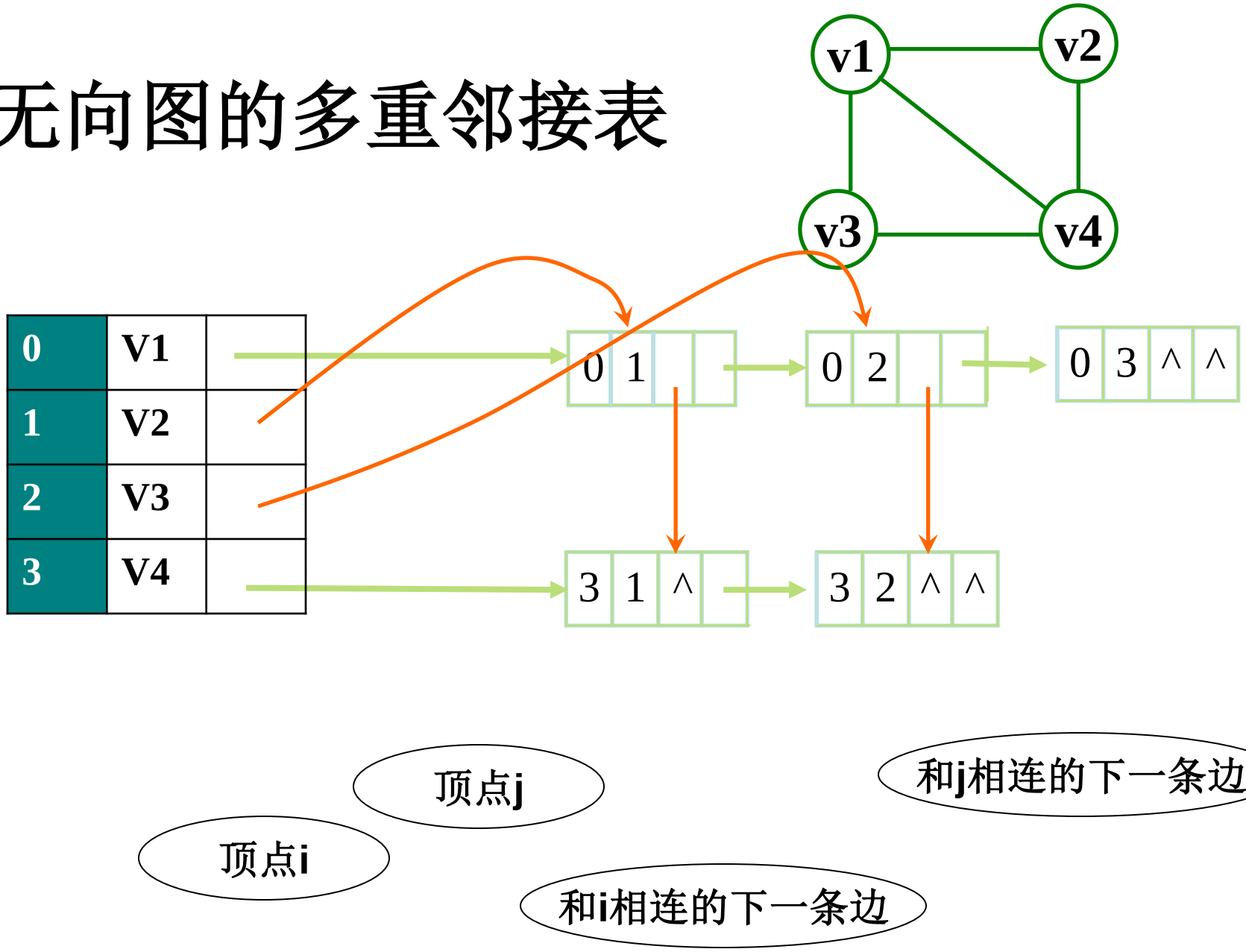
将邻接表、逆邻接表结合起来




```
typedef struct Vexnode{
  VertexType data;
  ArcBox *firstin, *firstout;
}VexNode;
```

```
Typedef struct ArcBox{
  int headvex, tailvex;
  struct ArcBox *hlink, *tlink;
  InfoType *info;
}ArcBox
```

无向图的多重邻接表

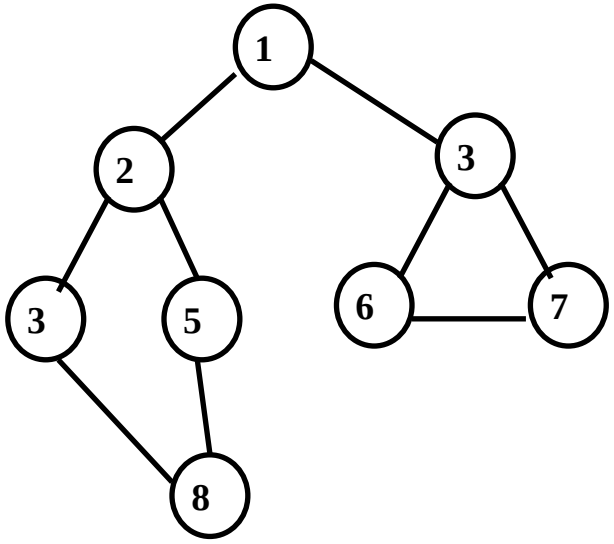

v1
v2
v3
v4
0
V1
1
V2
2
V3
3
V4
0 1
0 2
0 3 ∧ ∧
3 1 ∧
3 2 ∧ ∧
顶点j
和j相连的下一条边
顶点i
和i相连的下一条边

- 图的存储结构
- 图的遍历
- 图的连通性

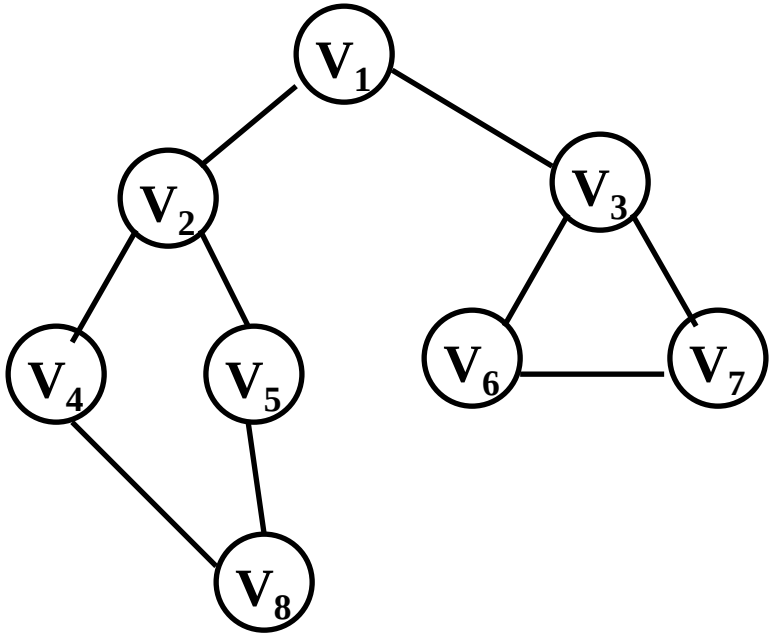
图的遍历

- 图的遍历：从图的某顶点出发，访问所有顶点，且每个顶点仅被访问一次。

- 两种遍历
  - 深度优先(类似于树的先根遍历)
  - 广度优先(类似于树的层次遍历)
  - 它们对无向图和有向图都适用



深度优先搜索 - Depth First Search



$V_1 \rightarrow V_2 \rightarrow V_3$

$V_2 \rightarrow V_1 \rightarrow V_4 \rightarrow V_5$

$V_3 \rightarrow V_1 \rightarrow V_6 \rightarrow V_7$

$V_4 \rightarrow V_2 \rightarrow V_8$

$V_5 \rightarrow V_2 \rightarrow V_8$

$V_6 \rightarrow V_3 \rightarrow V_7$

$V_7 \rightarrow V_3 \rightarrow V_6$

$V_8 \rightarrow V_4 \rightarrow V_5$

	1	2	3	4	5	6	7	8
visited	1	1	0	1	1	0	0	1

stack	V_1	V_2	V_4	V_8	V_5			

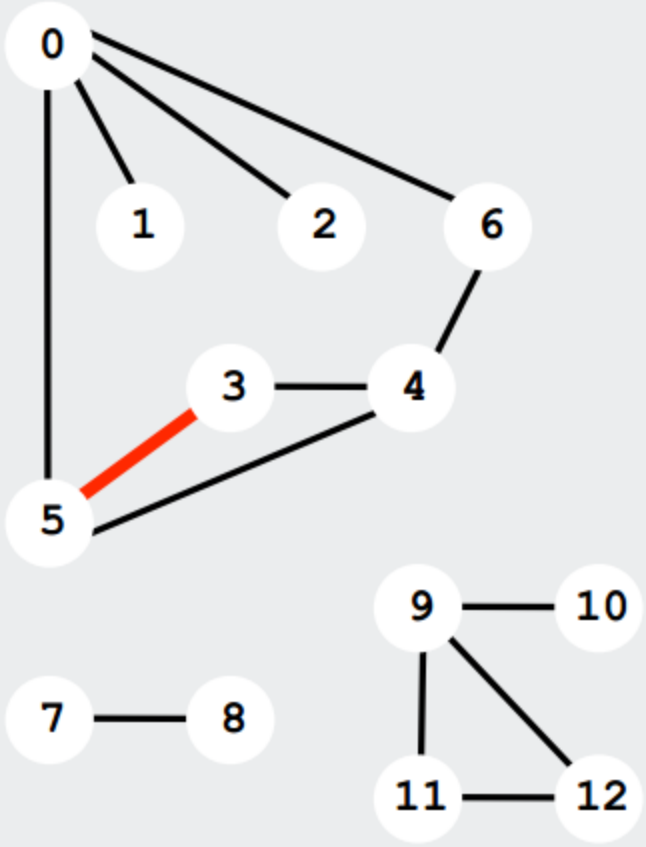
$V_1 \rightarrow V_2 \rightarrow V_4 \rightarrow V_8 \rightarrow V_5$

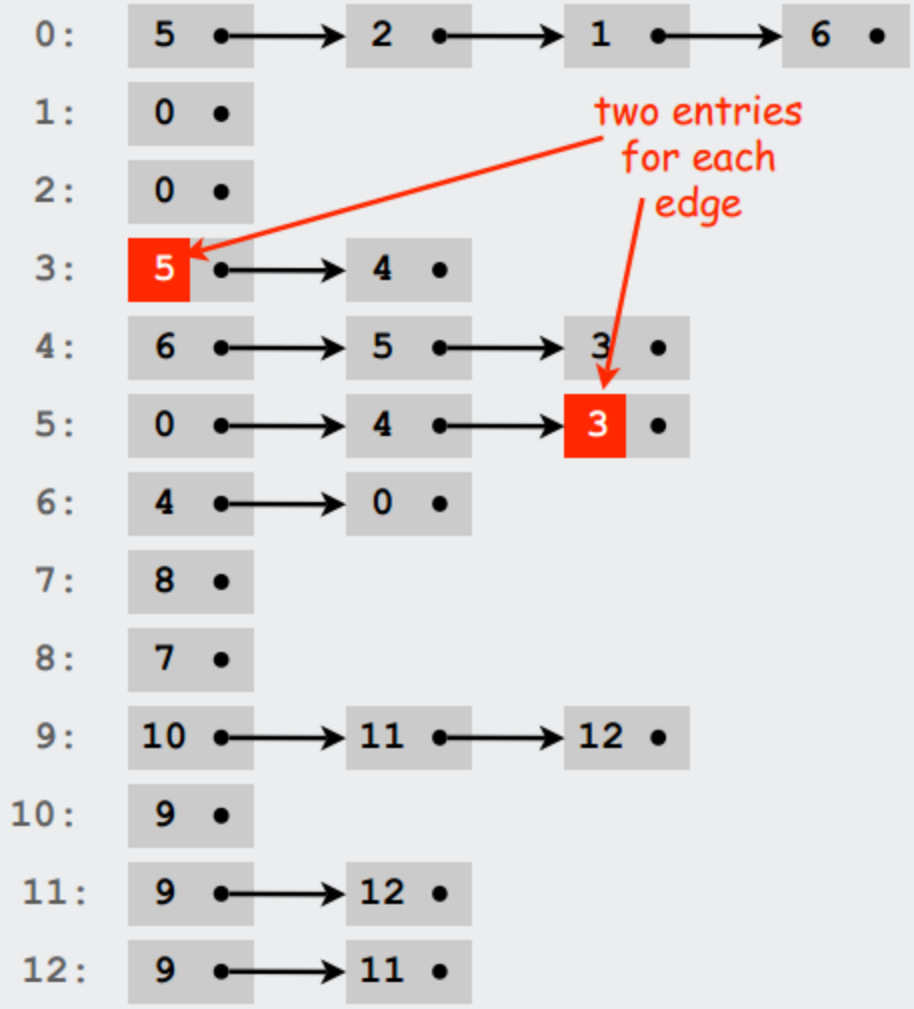
$\rightarrow V_3 \rightarrow V_6 \rightarrow V_7$

```
Boolean  visited[MAX];
Status  (* VisitFunc)(int v);

void DFSTraverse(Graph G, Status (* visit)(int  v))
//深度优先遍历图G
{  VisitFunc = visit;
    for(v=0; v<G.vexnum; ++v) visited[v] = FALSE;
    for(v=0; v<G.vexnum; ++v)
        if(!visited[v]) DFS(G,v);
}

void DFS(Graph G, int  v)
//从第v顶点出发递归的深度优先遍历图G
{ visited[v] = TRUE; VisitFunc(v);
   for(w=FirstAdjVex(G, v); w; w = NextAdjVex(G,v,w))
        if(!visited[w]) DFS(G,w);
}
```

v=0(即DFS(G,0))就可以完成遍历G，这从0开始的扫描是否多余?

图不一定连通，需要遍历每一个节点



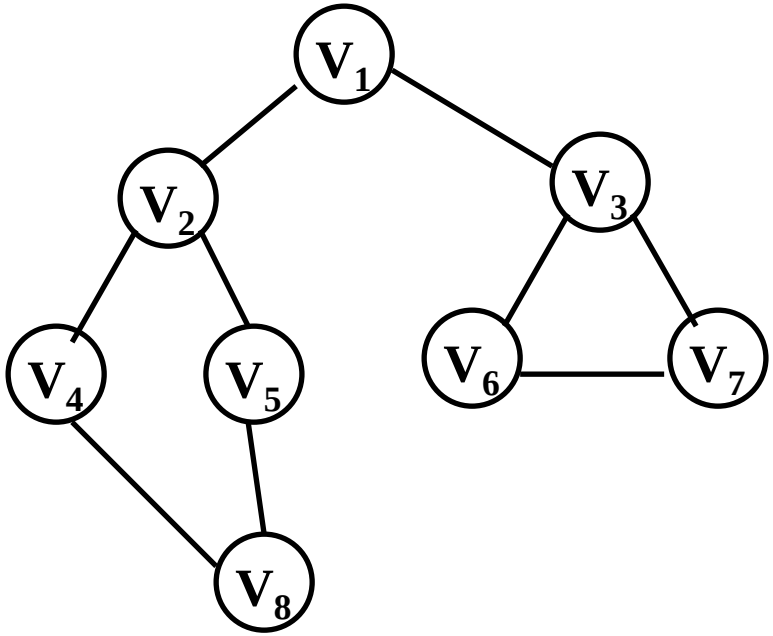


DFS算法分析

- 比较两种存储结构下的算法 (设 n 个顶点, e 条边)
  - 数组表示：查找每个顶点的邻接点要遍历每一行，遍历的时间复杂度为$O(n^2)$
  - 邻接表表示：虽然有 $2e$ 个表结点，但只需扫描 e 个结点即可完成遍历，加上访问 n个头结点的时间，遍历的时间复杂度为$O(n+e)$

- 结论：稠密图适于在邻接矩阵上进行深度遍历；稀疏图适于在邻接表上进行深度遍历。

广度优先搜索 -Breadth First Search



$V_1 \rightarrow V_2 \rightarrow V_3$

$V_2 \rightarrow V_1 \rightarrow V_4 \rightarrow V_5$

$V_3 \rightarrow V_1 \rightarrow V_6 \rightarrow V_7$

$V_4 \rightarrow V_2 \rightarrow V_8$

$V_5 \rightarrow V_2 \rightarrow V_8$

$V_6 \rightarrow V_3 \rightarrow V_7$

$V_7 \rightarrow V_3 \rightarrow V_6$

$V_8 \rightarrow V_4 \rightarrow V_5$

$V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_6 \rightarrow V_7 \rightarrow V_8$

visited

1	0	0	0	0	0	0	0

Queue

V_2	V_3						

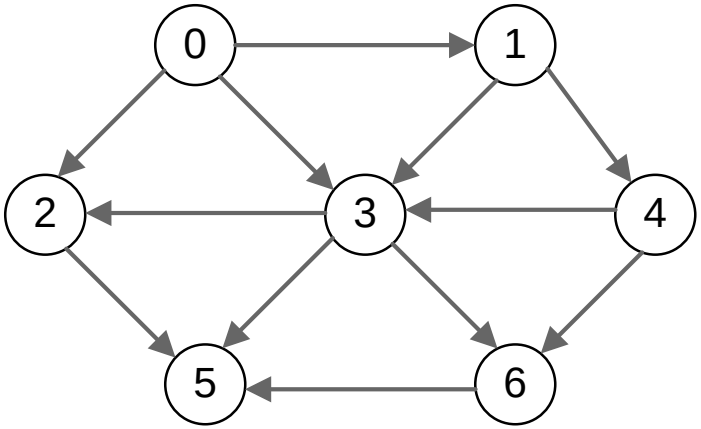
```
void BFSTraverse(Graph G, Status (* visit)(int  v)){
//广度优先遍历图G
    for(v=0; v<G.vexnum; ++v) visited[v] = FALSE;
    IntiQueque(Q);
    for(v=0; v<G.vexnum; ++v)
        if(!visited[v]) {
          visited[u] = TRUE; Visit (u);
          EnQueue(Q,v);
          while(!QueueEmpty(Q)){
              DeQueue(u);
              for(w=FirstAdjVex(G, u); w; w=NextAdjVex(G,u,w))
             if(!visited[w]) {
                visited[w]=TRUE;
                visited(w);
                EnQueue(G,w);
              }
          }
      }
}
```

BFS算法分析

- 数组表示：BFS对于每一个被访问到的顶点，都要循环检测矩阵中的整整一行（n 个元素），总的时间代价为$O(n^2)$。
- 邻接表表示：时间复杂度O(n+e).

作业练习

- 1、请写出如下有向图的邻接矩阵，基于该矩阵进行图的深度优先遍历；
- 2、建立如下有向图的邻接表，进行图的广度优先遍历.

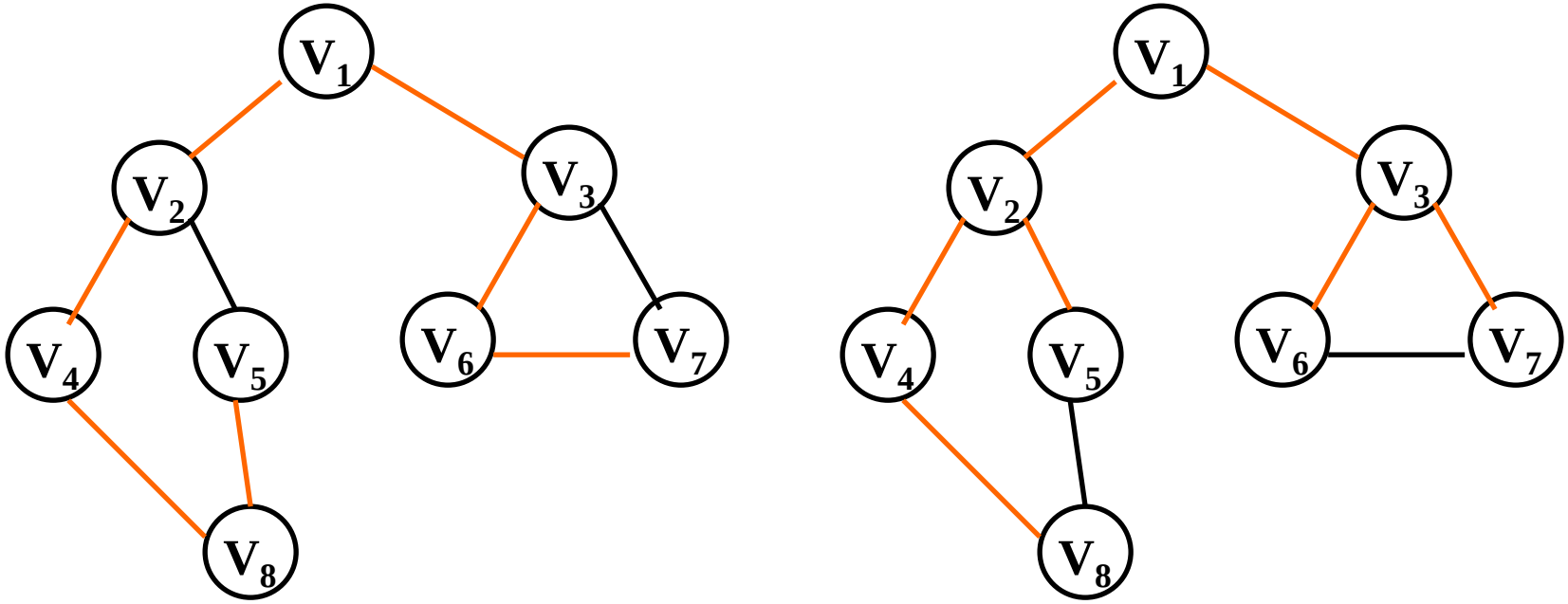


- 图的存储结构
- 图的遍历
- 图的连通性

图的连通性在计算机网、通信网和电力网等方面有着重要的应用。

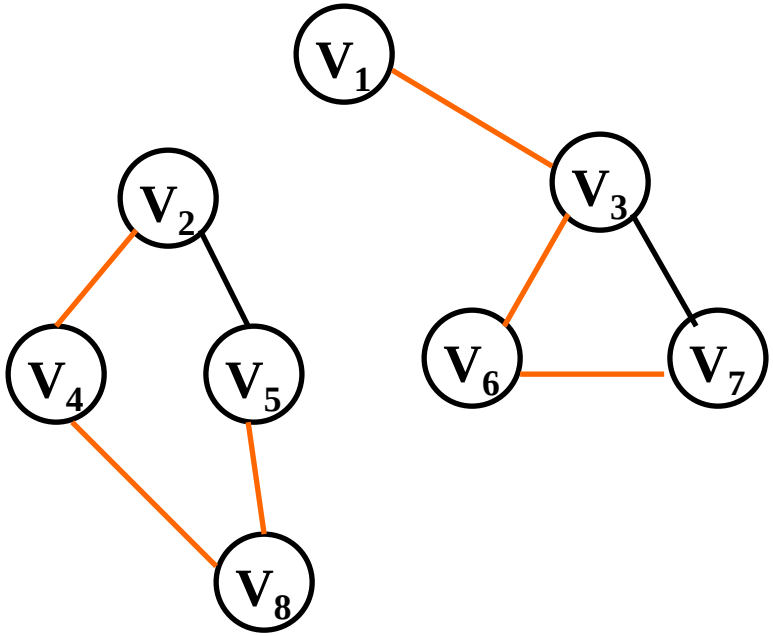
生成树(Spanning tree)

- 深度优先生成树　　vs.　　广度优先生成树



连通图的生成树是它的极小连通子图，有*n*个顶点和*n*-1条边。

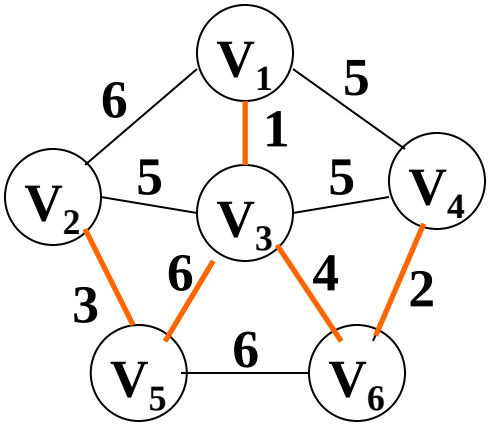
非连通图的连通分量



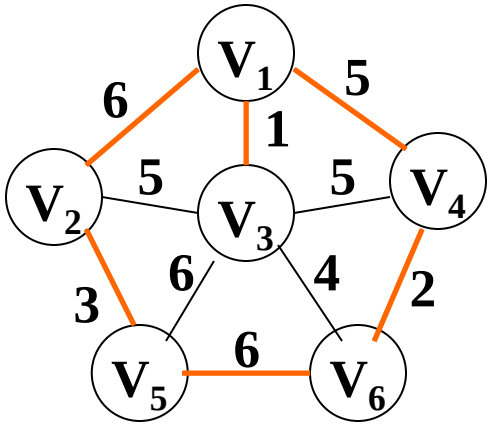
- 对于非连通图则遍历生成森林
- 左图是深度优先遍历生成森林

最小生成树

- 很多现实问题可以抽象成网。比如，在n个城市之间建立通信网，要求总成本最低。
- 上述问题是求连通网的最小生成树问题，即挑选n-1条不产生回路的最短边，则总成本（生成树的各边的权重之和）达到最低。



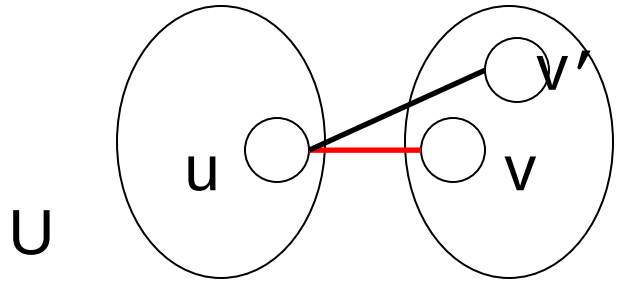
总成本为16　　优于　　总成本为23



最小生成树

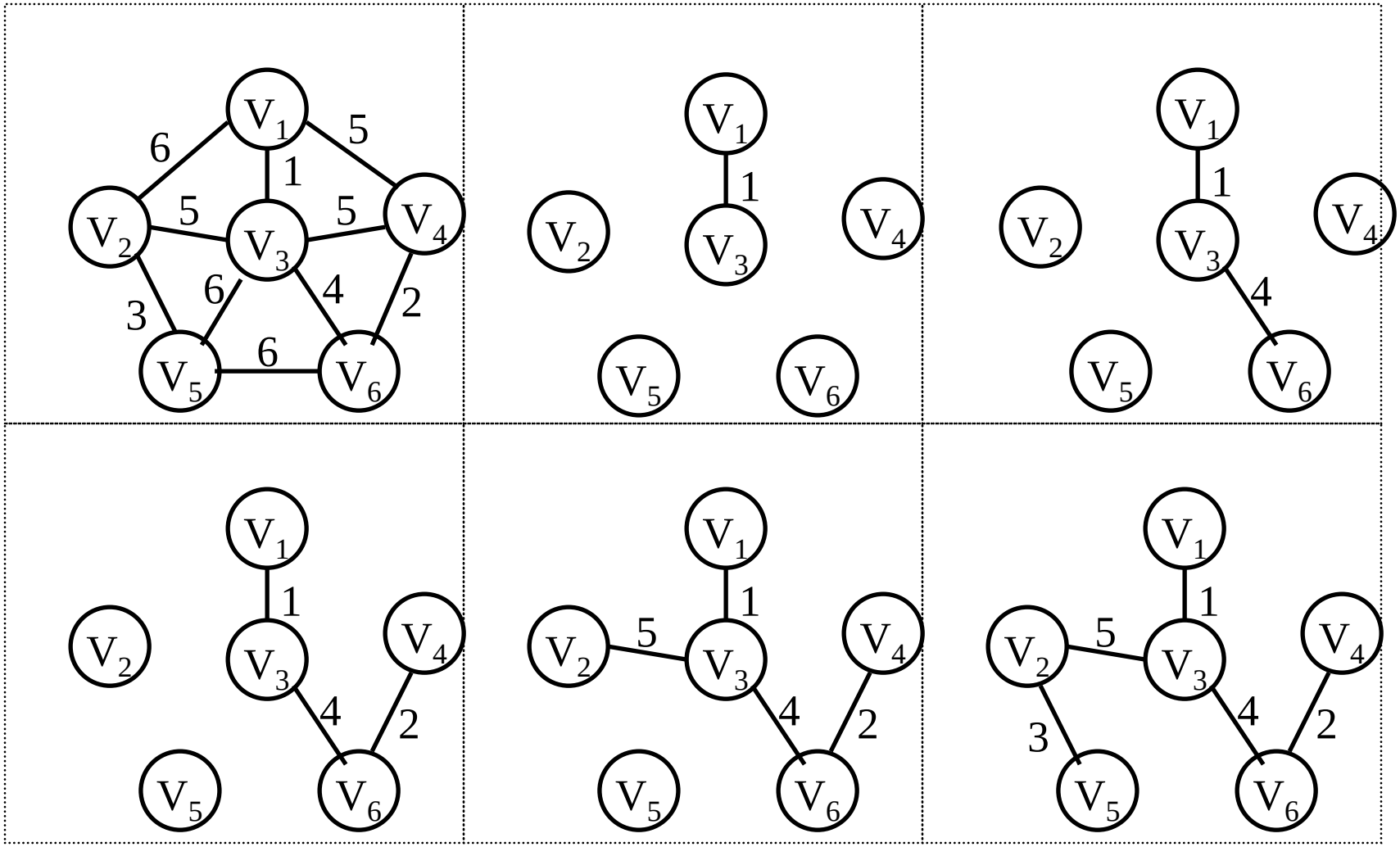
构造最小生成树有多种算法，其中多数利用了最小生成树的下列性质：

假设G=(V，E)是一个连通图，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∈U，v∈V-U，则必存在一棵包含边(u, v)的最小生成树。

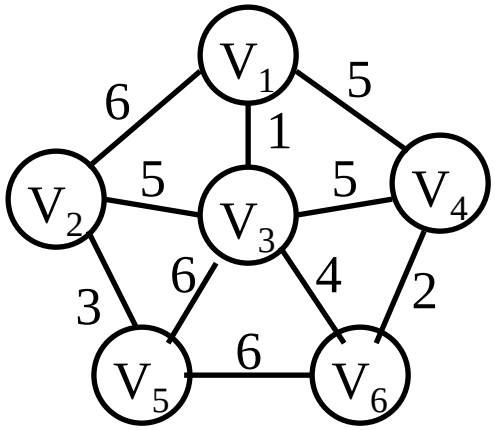


Prim算法

从顶点出发：找连通部分和未连通部分之间的最小代价的一条路

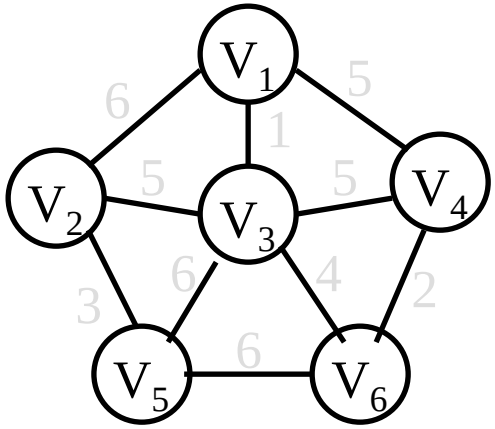


Prim算法 (cont.)



	6	1	5		
6		5		3	
1	5		5	6	4
5		5			2
	3	6			6
		4	2	6	

- 采用邻接矩阵，搜索最小权重的边需要$O(n^2)$;
- 优化方法
  - 采用堆(binary heap)结构和邻接表，则需要O(elogn);
  - 采用Fibonacci heap可以降至O(e+nlogn)。



	6	1	5		
6		5		3	
1	5		5	6	4
5		5			2
	3	6			6
		4	2	6	

T V1

V2,V3,V4,V5,V6

T V1,V3

V2,V4,V5,V6

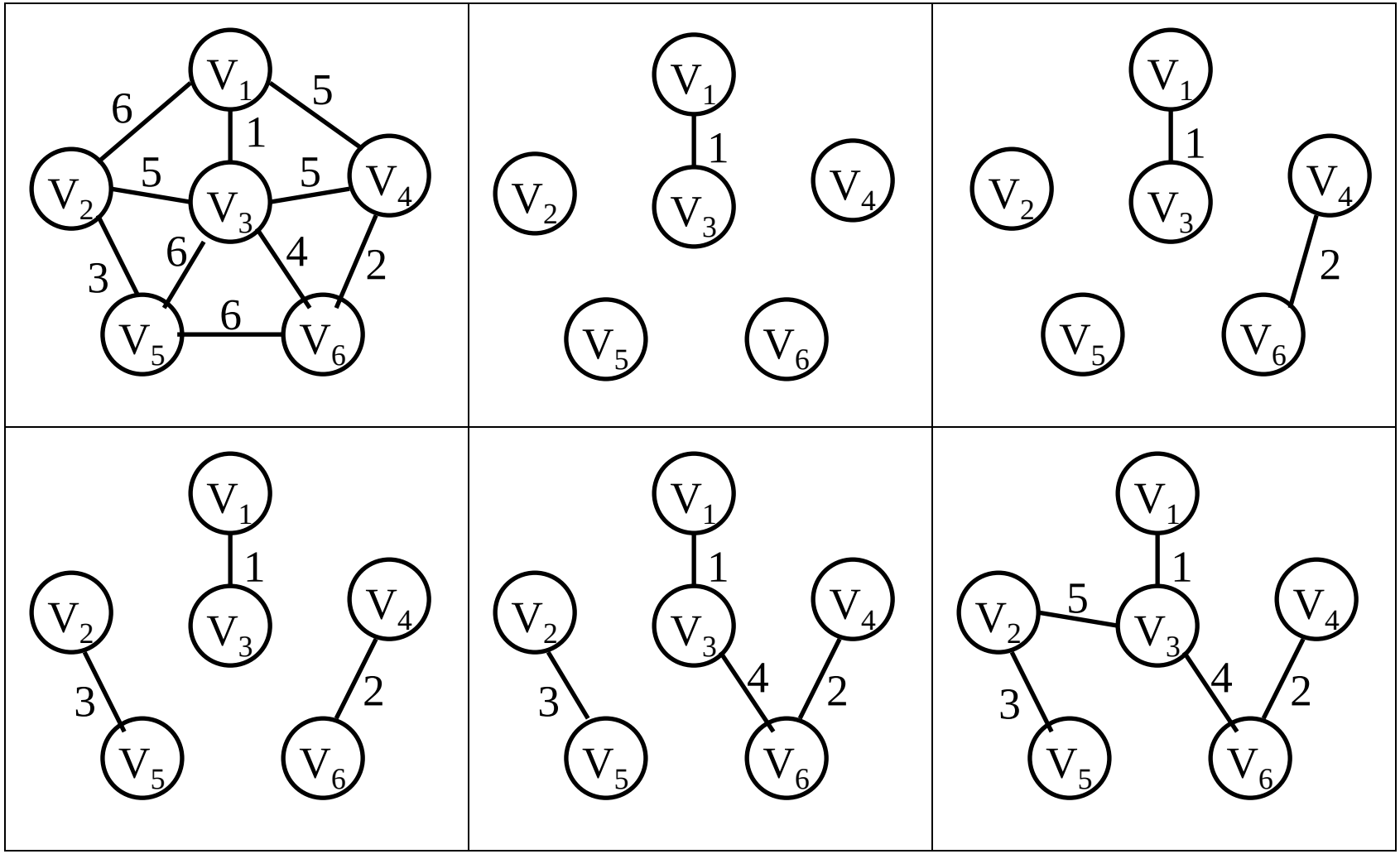
lowcost 0 6 1 5 M M
closest 1 1 1 1 1 1

lowcost 0 5 0 5 6 4
closest 1 3 1 1 3 3

不妨使用两个数组动态地维护每个点到生成树T的点最短距离信息，即可提高找最短边的效率。
Lowcost：每个顶点到生成树中的最小代价
Closest：最小代价对应边的相邻节点

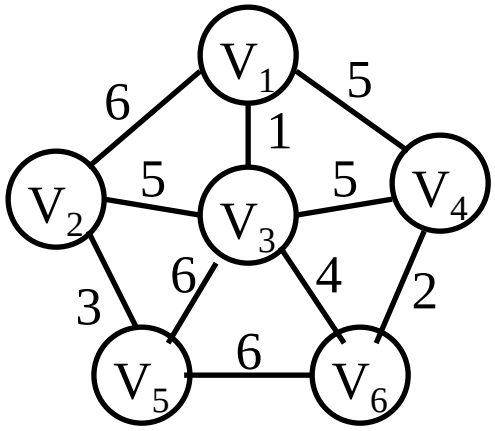
Kruskal算法

从边出发：找最小代价、且连通不同连通分量的一条路



课堂练习

- **1、请分别写出使用Prim和Kruscal算法生成最小生成树的步骤；**
- **2、请分别写出两种算法的存储结构（可以采用邻接矩阵，或邻接表），以及算法伪码；**
- **3、分析算法的复杂度。**



总结

- **Prime算法:**
  将顶点归并，每次选与树最近的点加进来，直到所有点加进来为止。

  算法复杂度$O(n^2)$，适于稠密网。

- **Kruskal算法：**
  将边归并，每次选权最小且不产生回路的边加进来，直到找到n-1条边加到树为止。

  算法复杂度O(eloge)，适于稀疏网。

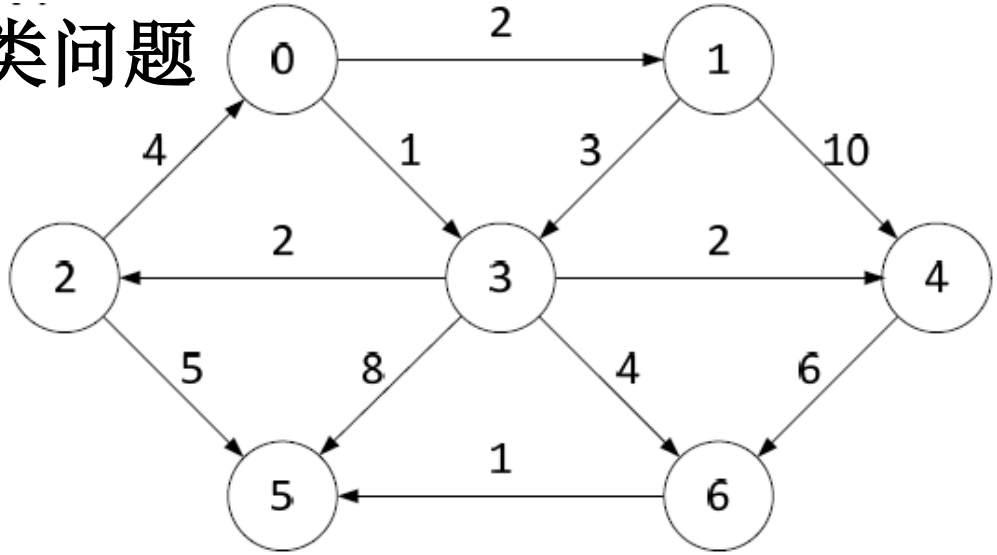
最短路径

- 许多地理问题在被抽象为图论下的网络图时，问题的核心就变成了网络图上的优化计算问题。最为常见的莫过于关于路径和顶点的优选计算问题。
- 前者最常见的是最短路径问题；后者最常见的是中心点和中位点选址问题。

最短路径的不同含义

- 纯距离意义上的最短路径
  e.g. 需要运送一批物资从城市**0**到城市**5**，选择什么样的运输路线距离最短？
- 经济、时间距离意义上的最短路径

- 以上均可抽象为同一类问题
  ——网的最短路径

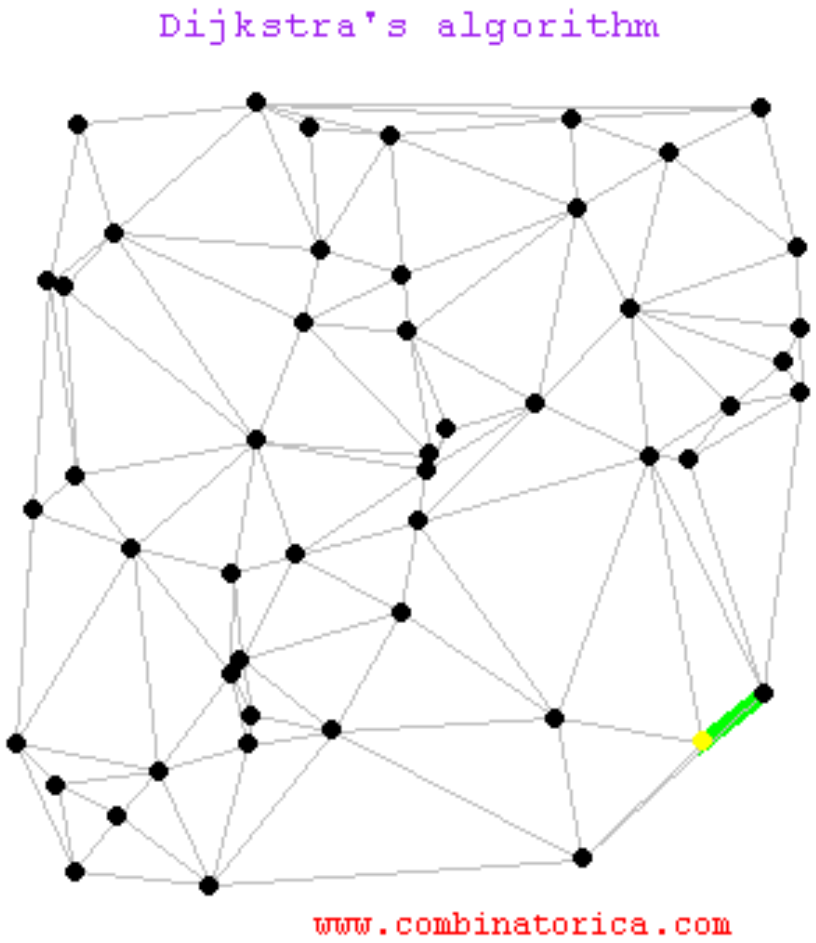


Dijkstra算法

- **E. W. Dijkstra (Netherlands), who won the 1972's ACM Turing Award, often regarded as the Nobel Prize for computing.**

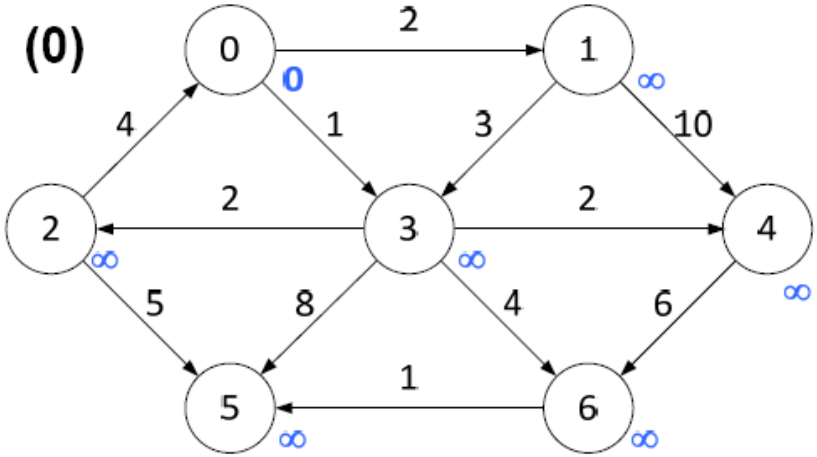
- **Dijkstra算法是典型的贪婪算法(Greedy Algorithm)——在每一步作出的选择使得当前有最好的结果。**
  - 在每一步找距离源点最近的点
  - 找到之后看其他节点是否可以通过该节点变短？如果是，则修改
  - 实现时，记录下每个节点经过哪个最短路径到达当前节点

Demo

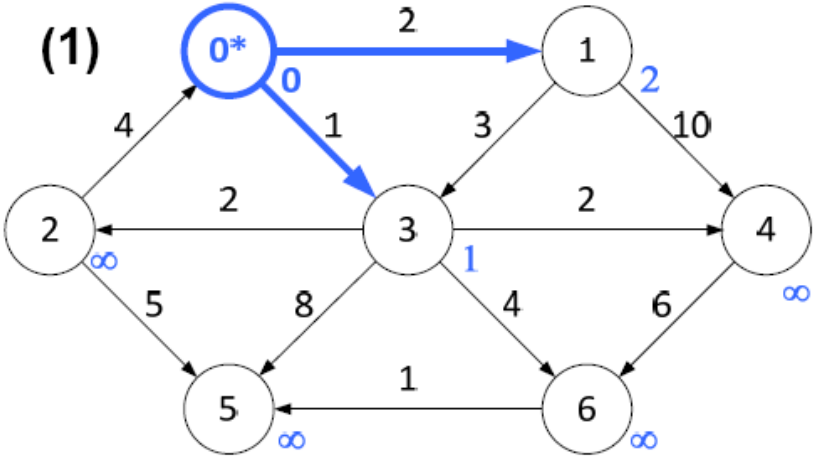


举例

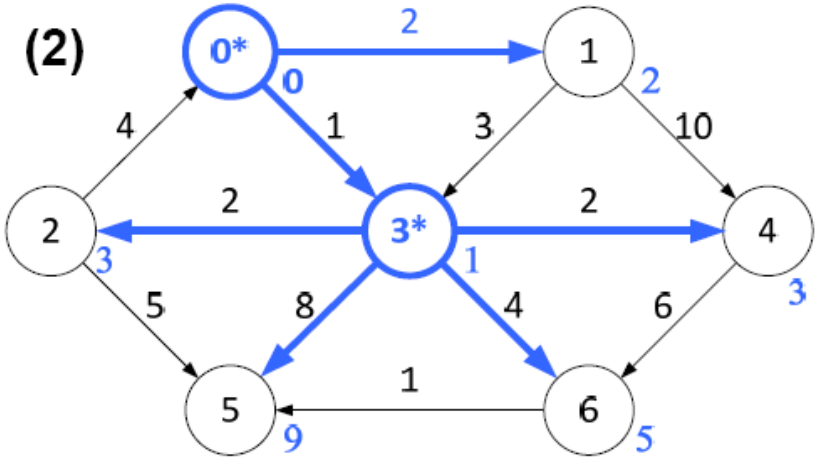
(0)


0
1
2
3
4
5
6
2
4
1
3
10
2
2
5
8
4
6
1
0
∞
∞
∞
∞
∞

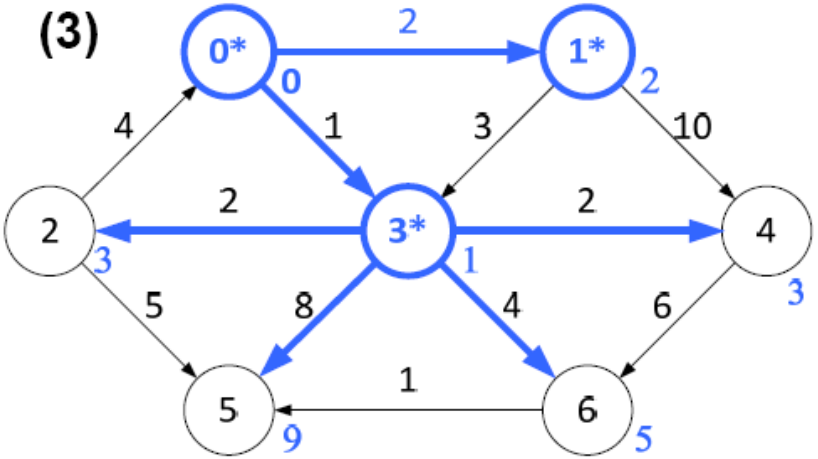
(1)


0*
1
2
3
4
5
6
2
4
1
3
10
2
2
5
8
4
6
1
0
2
∞
1
∞
∞

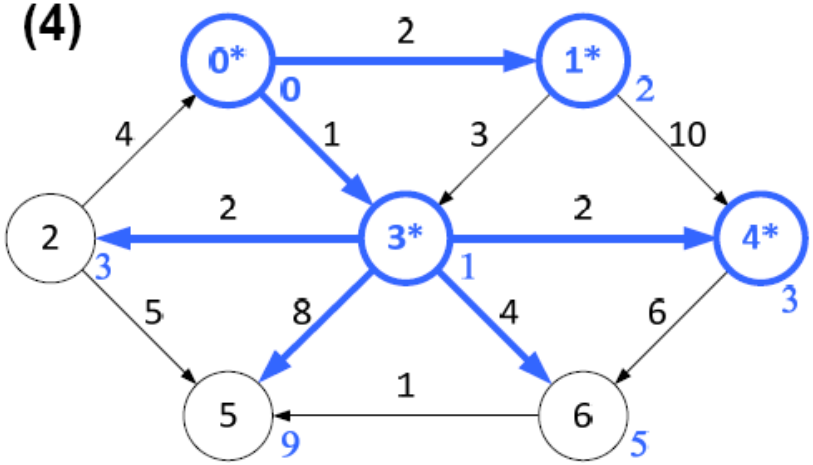
(2)


0*
1
2
3*
4
5
6
2
4
1
3
10
2
2
5
8
4
6
1
0
2
3
1
3
9
5

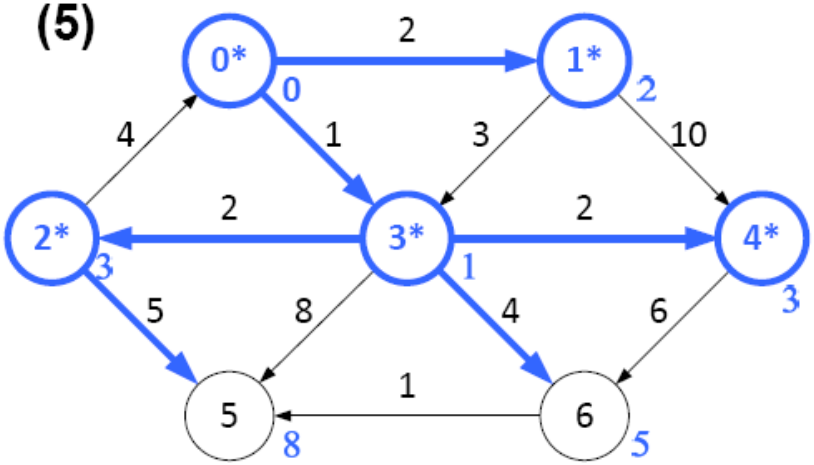
(3)


0*
1*
2
3*
4
5
6
2
4
1
3
10
2
2
5
8
4
6
1
0
2
3
1
3
9
5

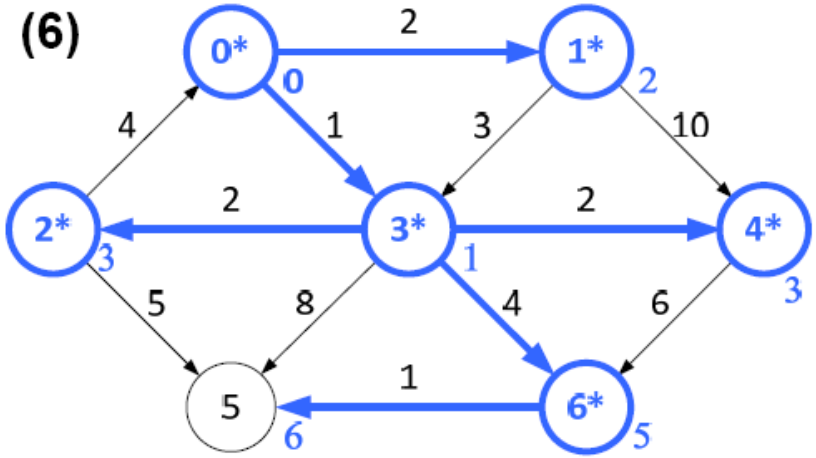
(4)


0*
1*
2
3*
4*
5
6
2
4
1
3
10
2
2
5
8
4
6
1
0
2
1
3
9
5

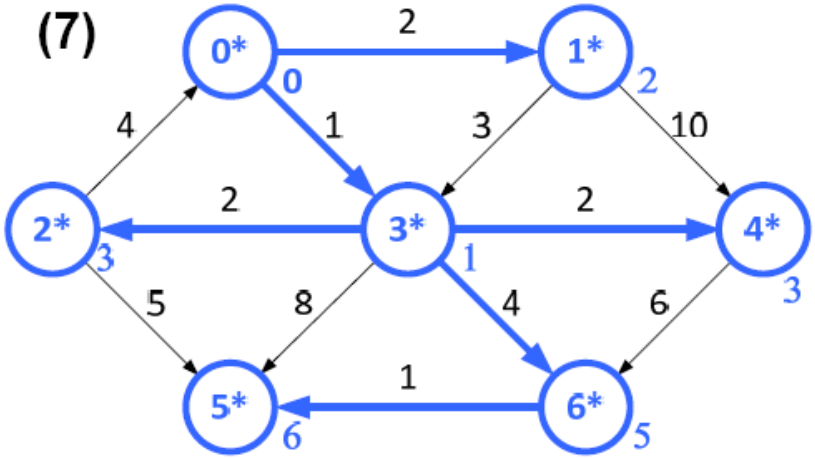
(5)


0*
1*
2*
3*
4*
5
6
2
4
1
3
10
2
2
5
8
4
6
1
0
2
1
3
8
5

(6)


0*
1*
2*
3*
4*
5
6*
2
4
1
3
10
2
2
5
8
4
6
1
0
2
1
3
6
5

(7)


0*
1*
2*
3*
4*
5*
6*
2
4
1
3
10
2
2
5
8
4
6
1
0
2
1
3
6
5

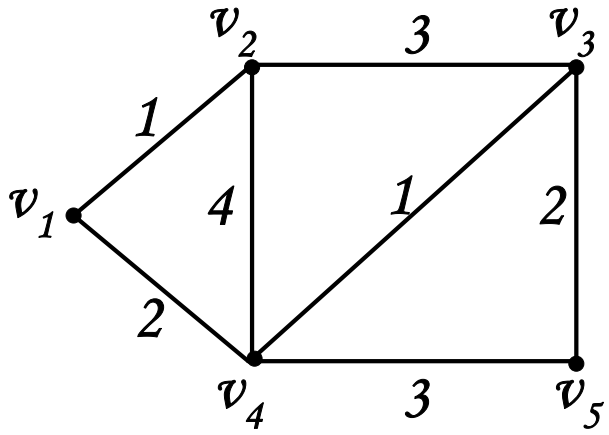
Floyd算法

- 在加权图中求任意两个顶点之间的最短路时，Floyd算法是常用的算法。这是一个动态规划的算法。
  - Robert Floyd于1962年提出该算法，其实它与1959年B. Roy和1962年Warshall提出的算法不谋而合。

Robert Floyd(1936-2001)是计算机领域著名的科学家，1978年图灵奖获得者。他出生在纽约，在1953年他17岁时便获得了liberal arts学士学位，1958年又获得了物理学学士学位。曾在Carnegie Mellon, standford任职。

Floyd算法

- 逐步尝试在原路径中加入其它顶点作为中间顶点，如果增加中间顶点后，得到的路径比原来的路径长度减少了，则以此新路径代替原路径，修改矩阵元素。
- 第一步，让所有边上加入中间顶点1，取A[i][j]与A[i][1]+A[1][j]中较小的值作A[i][j]的值，完成后得到A(1)，
- 以此类推…



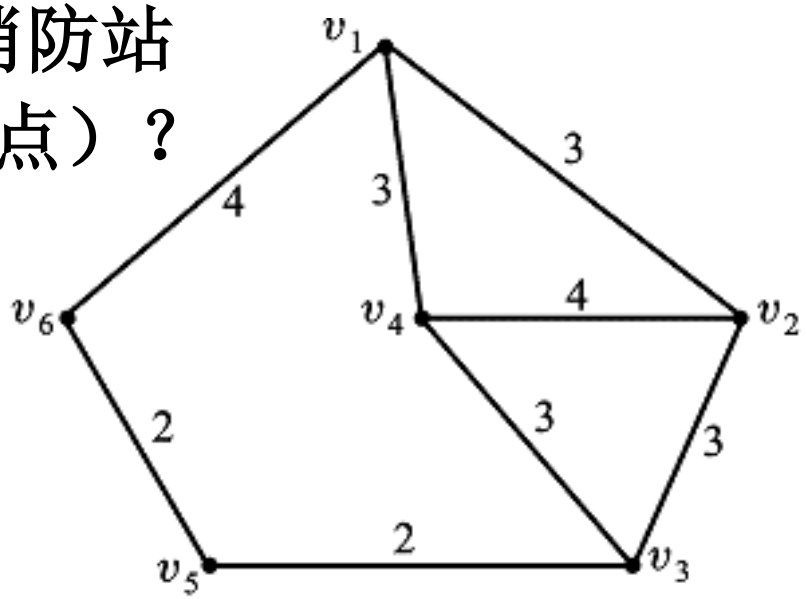
$$\mathbf{D}^0 = \left(\mathbf{w_{ij}}\right) = \begin{pmatrix} 0 & 1 & \infty & 2 & \infty \\ 1 & 0 & 3 & 4 & \infty \\ \infty & 3 & 0 & 1 & 2 \\ 2 & 4 & 1 & 0 & 3 \\ \infty & \infty & 2 & 3 & 0 \end{pmatrix}$$

$$D^1 = \begin{pmatrix} 0 & 1 & \infty & 2 & \infty \\ 1 & 0 & 3 & \underline{3} & \infty \\ \infty & 3 & 0 & 1 & 2 \\ 2 & \underline{3} & 1 & 0 & 3 \\ \infty & \infty & 2 & 3 & 0 \end{pmatrix} \quad D^2 = \begin{pmatrix} 0 & 1 & \underline{4} & 2 & \infty \\ 1 & 0 & 3 & 3 & \infty \\ \underline{4} & 3 & 0 & 1 & 2 \\ 2 & 3 & 1 & 0 & 3 \\ \infty & \infty & 2 & 3 & 0 \end{pmatrix} \quad D^3 = \begin{pmatrix} 0 & 1 & 4 & 2 & \underline{6} \\ 1 & 0 & 3 & 3 & \underline{5} \\ 4 & 3 & 0 & 1 & 2 \\ 2 & 3 & 1 & 0 & 3 \\ \underline{6} & \underline{5} & 2 & 3 & 0 \end{pmatrix}$$

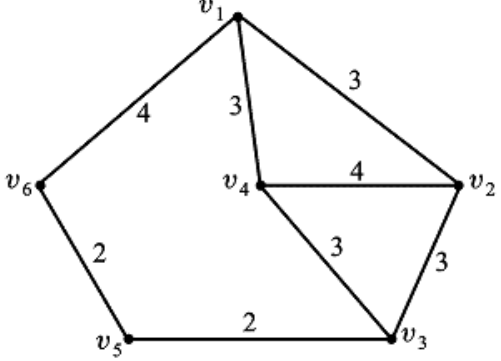
$$D^4 = \begin{pmatrix} 0 & 1 & \underline{3} & 2 & \underline{5} \\ 1 & 0 & 3 & 3 & 5 \\ \underline{3} & 3 & 0 & 1 & 2 \\ 2 & 3 & 1 & 0 & 3 \\ \underline{5} & 5 & 2 & 3 & 0 \end{pmatrix} \quad D^5 = \begin{pmatrix} 0 & 1 & 3 & 2 & 5 \\ 1 & 0 & 3 & 3 & 5 \\ 3 & 3 & 0 & 1 & 2 \\ 2 & 3 & 1 & 0 & 3 \\ 5 & 5 & 2 & 3 & 0 \end{pmatrix}$$

中心点选址举例

- 假设某县下属的6个乡镇及其之间公路联系如图所示。每一顶点代表一个乡镇；每一条边代表连接两个乡镇之间的公路，每一条边旁的数字代表该条公路的长度。现在要设立一个消防站，为全县的6个乡镇服务。试问该消防站应该设在哪一个乡镇（顶点）？



最短路径长度



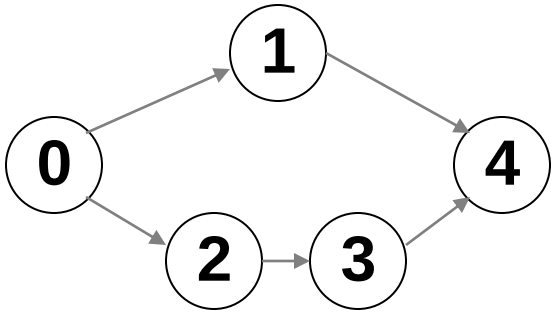
$$D=\begin{pmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} & d_{26} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} & d_{36} \\ d_{41} & d_{42} & d_{43} & d_{44} & d_{45} & d_{46} \\ d_{51} & d_{52} & d_{53} & d_{54} & d_{55} & d_{56} \\ d_{61} & d_{62} & d_{63} & d_{64} & d_{65} & d_{66} \end{pmatrix}=\begin{pmatrix} 0 & 3 & 6 & 3 & 6 & 4 \\ 3 & 0 & 3 & 4 & 5 & 7 \\ 6 & 3 & 0 & 3 & 2 & 4 \\ 3 & 4 & 3 & 0 & 5 & 7 \\ 6 & 5 & 2 & 5 & 0 & 2 \\ 4 & 7 & 4 & 7 & 2 & 0 \end{pmatrix}$$

消防站设在*v*1，*v*3，*v*5中任何一点上都是可行的参考(v1, v3, v5最坏的情况下（距离6）比v2， v4要好（距离7）):

拓扑排序 (Topological sort)

- 一个表示偏序的有向图可以用来表示一个流程图，比如施工流程图、生产路线图、数据流图。如下图所示，这样的图也成为Activity on Vertex Network (AOV网).

全序



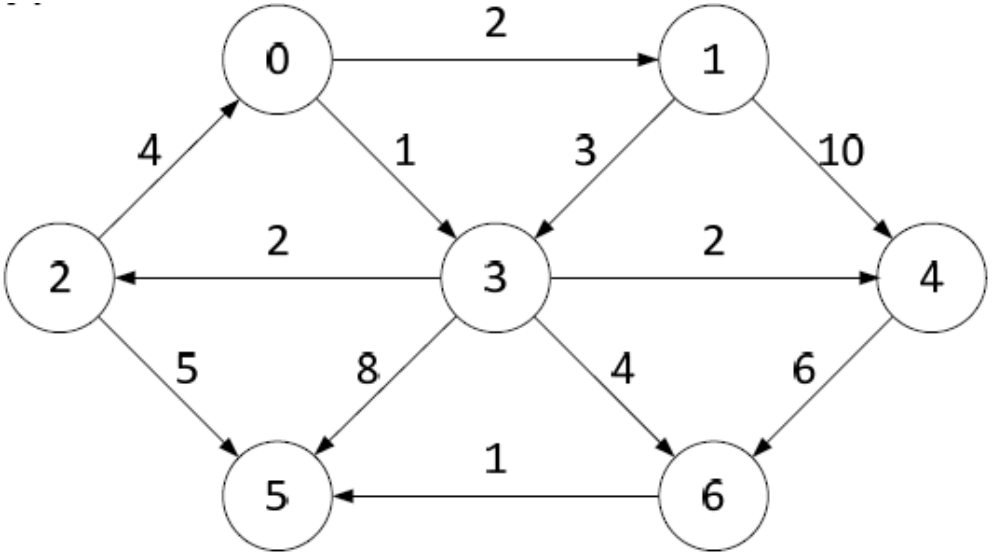
偏序

拓扑排序的结果不是唯一的

- 拓扑排序是由某个集合上的一个偏序得到该集合上的一个全序。

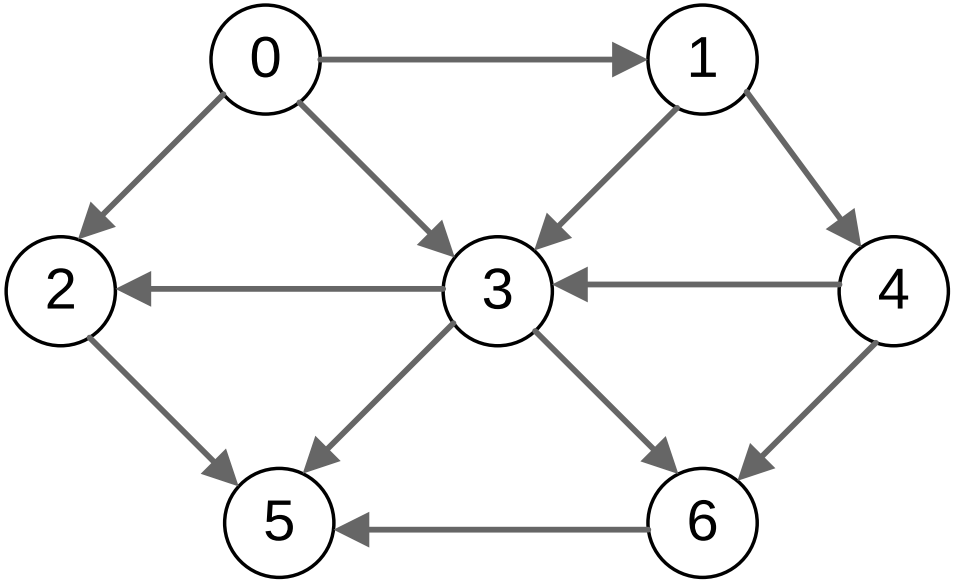
图的拓扑排序

- 对含环的有向图如何进行拓扑排序？



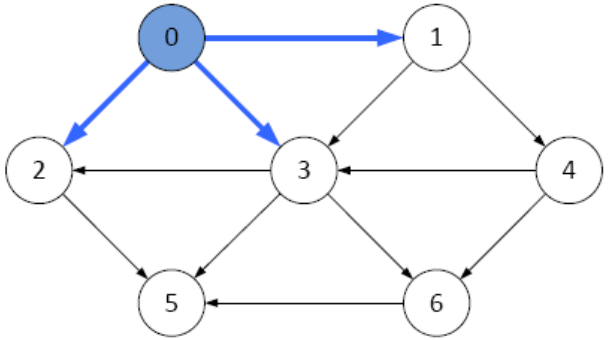
AOV网中不应该出现有向环，这意味着某项活动以自己为先决条件！

请对下列图进行拓扑排序



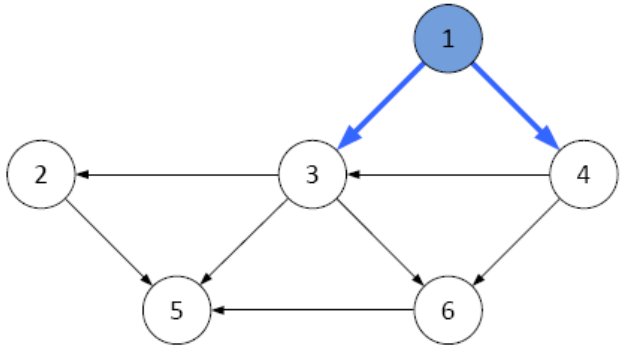
图的拓扑排序 (cont.)

V	Indeg.	TopNum
0	0	1
1	1	-1
2	2	-1
3	3	-1
4	1	-1
5	3	-1
6	2	-1



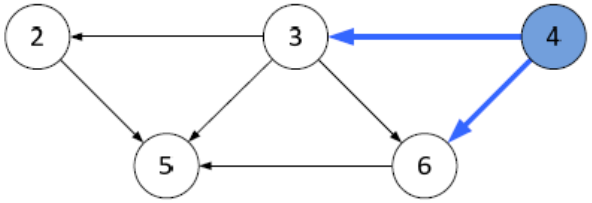
(1)

V	Indeg.	TopNum
0	0	0
1	0	1
2	1	-1
3	2	-1
4	1	-1
5	3	-1
6	2	-1



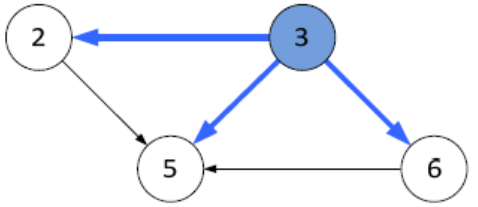
(2)

V	Indeg.	TopNum
0	0	0
1	0	1
2	1	-1
3	1	-1
4	0	2
5	3	-1
6	2	-1



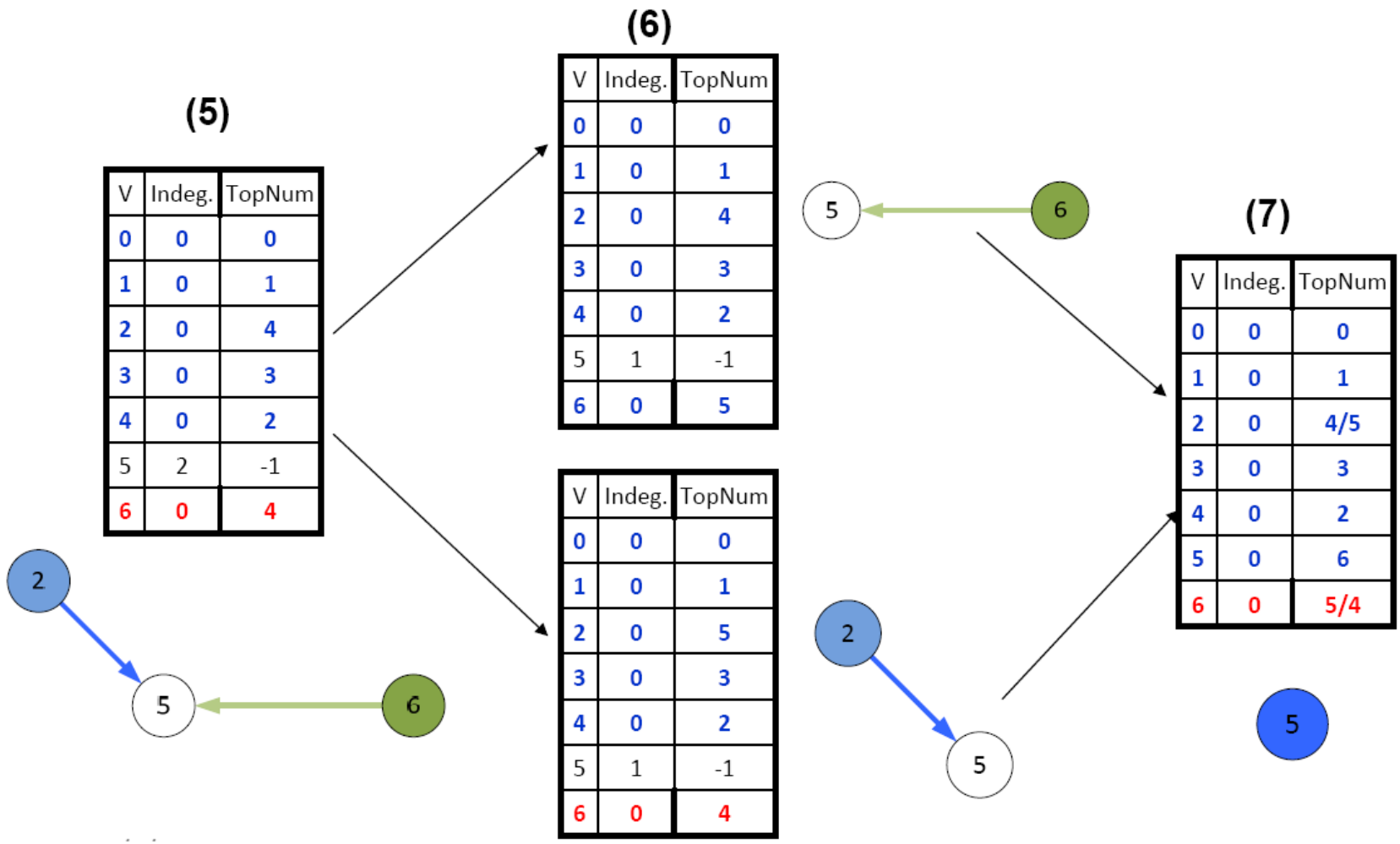
(3)

V	Indeg.	TopNum
0	0	0
1	0	1
2	1	-1
3	0	3
4	0	2
5	3	-1
6	1	-1



(4)

图的拓扑排序 (cont.)


(5)
V Indeg. TopNum
0 0 0
1 0 1
2 0 4
3 0 3
4 0 2
5 2 -1
6 0 4
2
5
6
(6)
V Indeg. TopNum
0 0 0
1 0 1
2 0 4
3 0 3
4 0 2
5 1 -1
6 0 5
V Indeg. TopNum
0 0 0
1 0 1
2 0 5
3 0 3
4 0 2
5 1 -1
6 0 4
5
6
2
5
(7)
V Indeg. TopNum
0 0 0
1 0 1
2 0 4/5
3 0 3
4 0 2
5 0 6
6 0 5/4
5

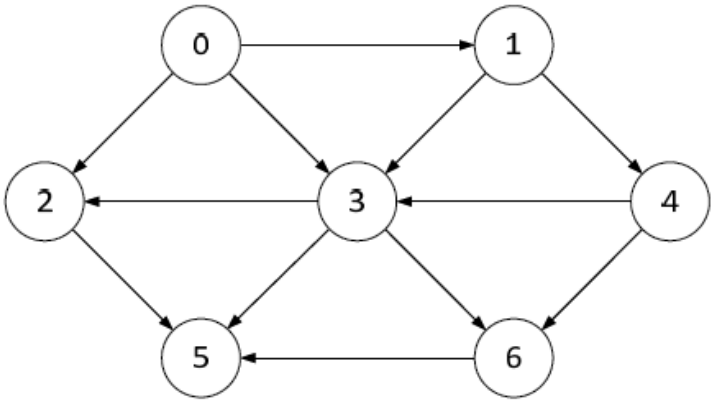
Simple Topological Sort Algorithm

```
Status Toposort(Graph G) {
//有向图G采用邻接表存储，若G无回路输出它的一个拓扑序列
  FindInDegree(G, indegree);
  InitStack(S);
  for(i = 0; Counter < G.vexnum; ++i)
    if (!indegree[i])
      push(S,i);
    count=0;
    while (!StackEmpty(S)){
      Pop(S,i); printf(I, G.vertices[i].data); ++count;
      for(p=G.vertices[i].firstarc; p; p++) {
        k=p->adjvex;
        if (!(--indegree[k])) push(S,k);
       }//for
     }//while
     if(count<G.vexnum) return ERROR;
     else return OK;
}
```

求入度的时间复杂度为O(e), 入栈出栈O(n), 总时间复杂度为O(n+e).

图的拓扑排序 (cont.)

- 可以考虑使用队列



(0)

V	Indeg.	TopSort	
V0	0	-1	← Head
V1	1	-1	
V2	2	-1	
V3	3	-1	
V4	1	-1	
V5	3	-1	
V6	2	-1	
			← Rear

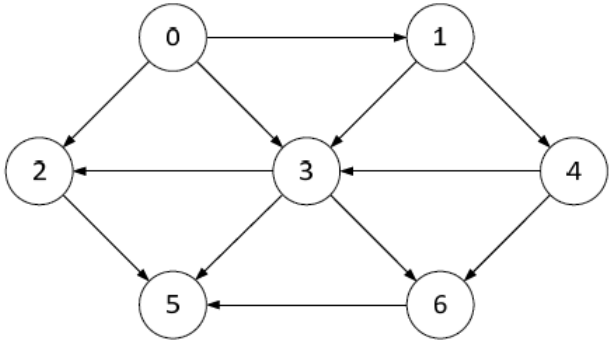
(1)

V	Indeg.	TopSort	
V0	0	V0	← Head ← Rear
V1	1	-1	
V2	2	-1	
V3	3	-1	
V4	1	-1	
V5	3	-1	
V6	2	-1	

(2)

V	Indeg.	TopSort	
V0	0	V0	
V1	0	V1	← Head ← Rear
V2	1	-1	
V3	2	-1	
V4	1	-1	
V5	3	-1	
V6	2	-1	

图的拓扑排序 (cont.)



(3)

V	Indeg.	TopSort
V0	0	V0
V1	0	V1
V2	1	V4
V3	1	-1
V4	0	-1
V5	3	-1
V6	2	-1

Head
Rear

(4)

V	Indeg.	TopSort
V0	0	V0
V1	0	V1
V2	1	V4
V3	0	V3
V4	0	-1
V5	3	-1
V6	1	-1

Head
Rear

V	Indeg.	TopSort
V0	0	V0
V1	0	V1
V2	0	V4
V3	0	V3
V4	0	V2
V5	2	V6
V6	0	-1

Head
Rear

(6)

V	Indeg.	TopSort
V0	0	V0
V1	0	V1
V2	0	V4
V3	0	V3
V4	0	V2
V5	1	V6
V6	0	-1

Head
Rear

(7)

V	Indeg.	TopSort
V0	0	V0
V1	0	V1
V2	0	V4
V3	0	V3
V4	0	V2
V5	0	V6
V6	0	V5

Head
Rear

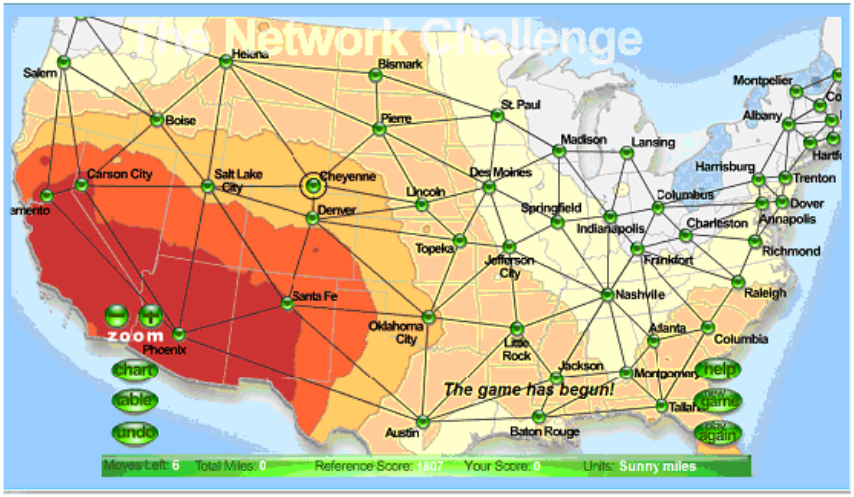
(7)

V	Indeg.	TopSort
V0	0	V0
V1	0	V1
V2	0	V4
V3	0	V3
V4	0	V2
V5	0	V6
V6	0	V5

Rear
Head

图与网络

- 用数学语言来说，网络就是一种图，一般认为它专指加权图。网络除了数学定义外，还有具体的物理含义，即网络是从某种相同类型的实际问题中抽象出来的模型，习惯上就称其为什么类型网络，如开关网络、运输网络、通信网络、计划网络等。
- 现实生活中的许多问题都可以使用图与网络的模型来刻画，比如交通路线的优化、ERP系统的工作计划制定等。



社会网络 (social network)

- 社会网络是指社会个体成员之间因为互动而形成的相对稳定的关系体系。
- 社会网络分析(SNA)是西方社会学的一个重要分支，是从上世纪30年代末在国外出现并逐步得到发展的研究社会结构的方法和技术。从30年代到60年代，在心理学、社会学、人类学以及数学、物理、统计、概率研究领域，越来越多的学者开始构建和研究社会结构概念，SNA的理论、方法和技术日益深入，成为一种重要的社会结构研究范式。

六度分隔 (Six Degrees of Separation)

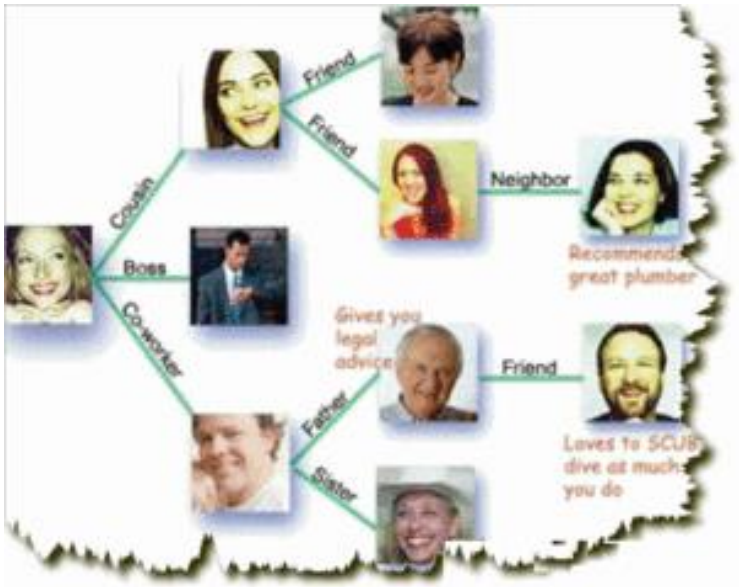
- 这个理论是1929年由匈牙利作家弗里奇斯·卡林思在短篇小说中首次提出的，他指出，世界上所有人都可以通过至多五个中间人联系起来。
- 1967年，哈佛大学社会心理学教授Stanley Milgram (1933-1984)进行了一项实验。他从Nebraska和Kansas随机选择了300多名志愿者，请他们寄一封信函给指定的一名住在波士顿的股票经纪人。结果，有六十多封信函最终到达了目标人手中，并且这些信函经过的中间人的数目平均只有5个。从此，六度空间理论受到了全世界的关注，后来还催生了电影《六度空间》。 此外，Milgram还发现了漏斗效应， 他发现大部分的传递都是由那些极少数的明星人物完成的。

150法则 (Rule of 150)

- 从欧洲发源的“赫特兄弟会”是一个自给自足的农民自发组织，他们有一个不成文的严格规定：每当聚居人数超过150人的规模，他们就把它变成两个，再各自发展。
- 150也被称为“邓巴数字(Dunbar’s number)”，该理论认为人的大脑新皮层大小有限，提供的认知能力只能使一个人维持与大约150人的稳定人际关系。
- 把人群控制在150人以下似乎是管理人群的一个最佳和最有效的方式。早期中国移动的动感地带sim卡最多保存150个手机号，早期一个MSN帐号只能对应150个联系人。
- 不管你认识多少人，或者与更多人建立了弱链接，那些强链接仍然在此次此刻符合150法则。这也符合“二八”法则，即80%的社会活动可能被150个强链接所占有。

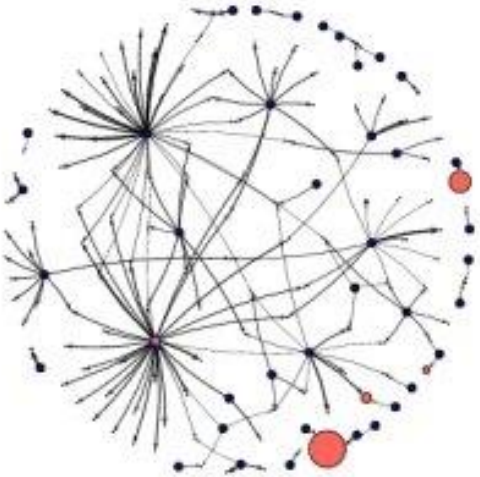
社会网络 (cont.)

- 社会网络是在成员之间传播信息、意见、创新等的介质。
  - 农夫们仿效他们的邻居而采纳新品种的稻谷**[Ryan and Gross, 1943]**
  - 在商业领域，口碑营销**(Word-of-mouth)**的力量有目共睹。

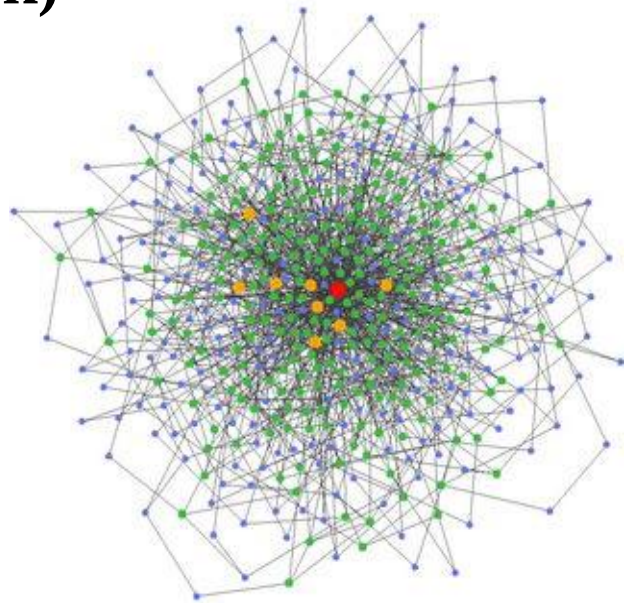


网络模型

- 热点从传统的随机网络向其它转移
  - 小世界网络(**small world network**)
    Watts, Strogatz, nature, 1998.
  - 无标度网络(**scale-free network**)
    Albert, Science, 1999.

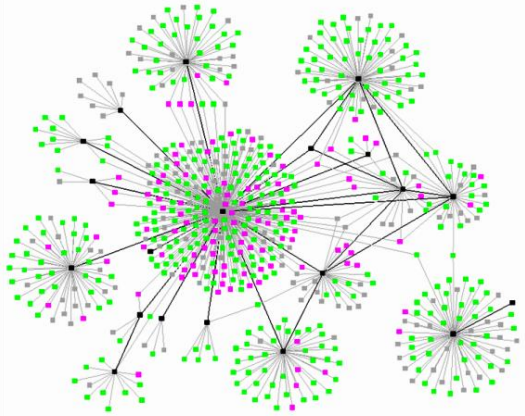


小世界网络
Small world network

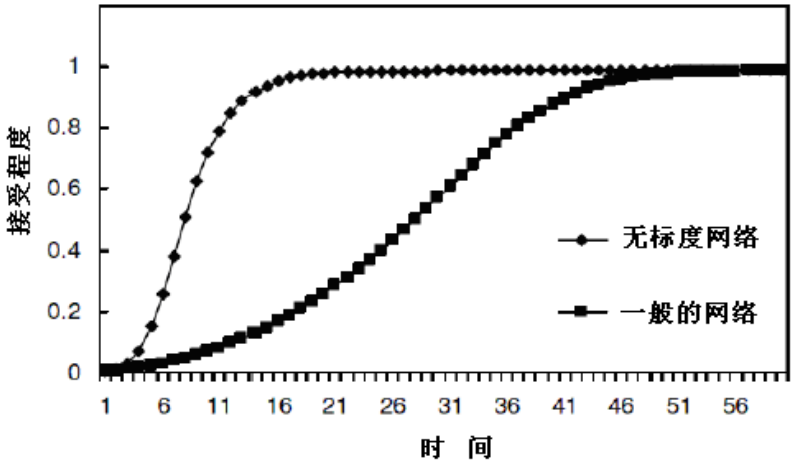


无标度网络
Scale-free network

小世界网络

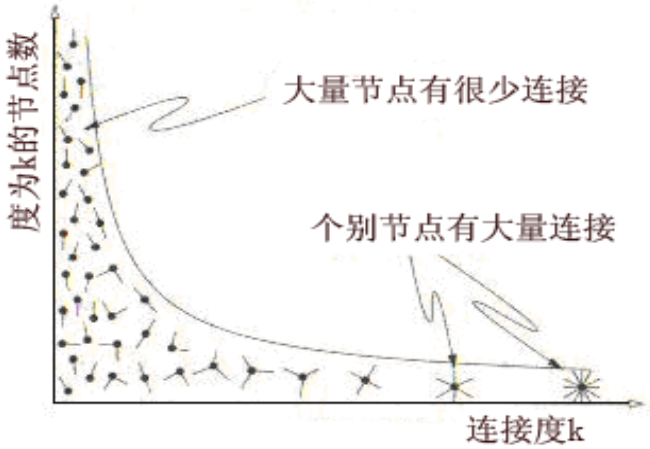


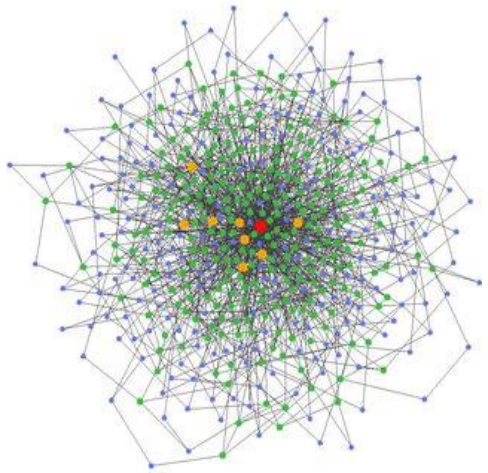
- 在小世界网络中，节点具有均质特性(homogeneous)，即具有大致相同数目的连接
- 小世界网络对于许多社会过程的动态(dynamics)研究非常重要
  - 疾病/病毒传播
  - 信息传播与公众舆情
  - 财富分配
  - 文化/价值观传播



无标度网络

- 在无标度网络中，节点不再具有同质性，节点的度服从幂律发布。任意节点与其它k个节点连接的概率$P(k)$满足$P(k) \propto k^{-r}$，通常$2<r\leq 3$。绝大多数节点有很少连接，而有极大连结度的节点数量非常之少，节点的度没有明显的长度，这也是无标度网络名称的由来。
- 无标度性已被证明是因特网的重要特性





本章作业

- 理解求解生成树和最短路径的4个算法，然后编码实现，编码中加上必要的注释，方法和变量需要指定对应的类型（例如对于变量x，指定其为字符串类型，则可以：x:str = 'hello’）。
- 四个算法中，普里姆算法和迪杰斯特拉算法必做，另外两种为选做题目。
- 截止时间为下5月26日晚23:00之前，邮件标题为：姓名+学号+图算法