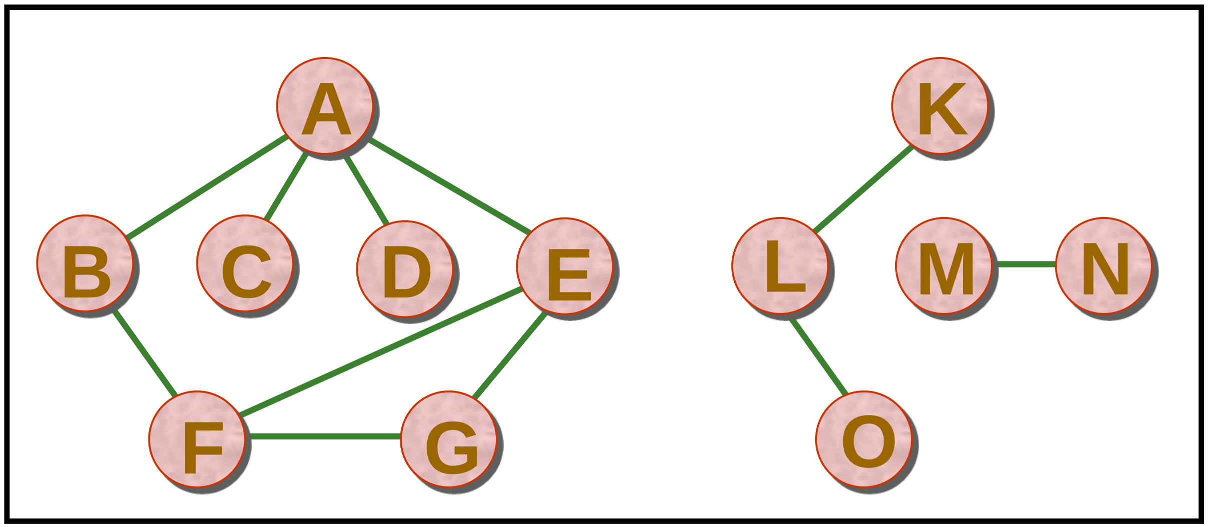
第七章 图

7.4 图的连通

一. 无向图的连通性

- 如果无向图中，存在不连通的顶点，则该图称为非连通图



7.4 图的连通

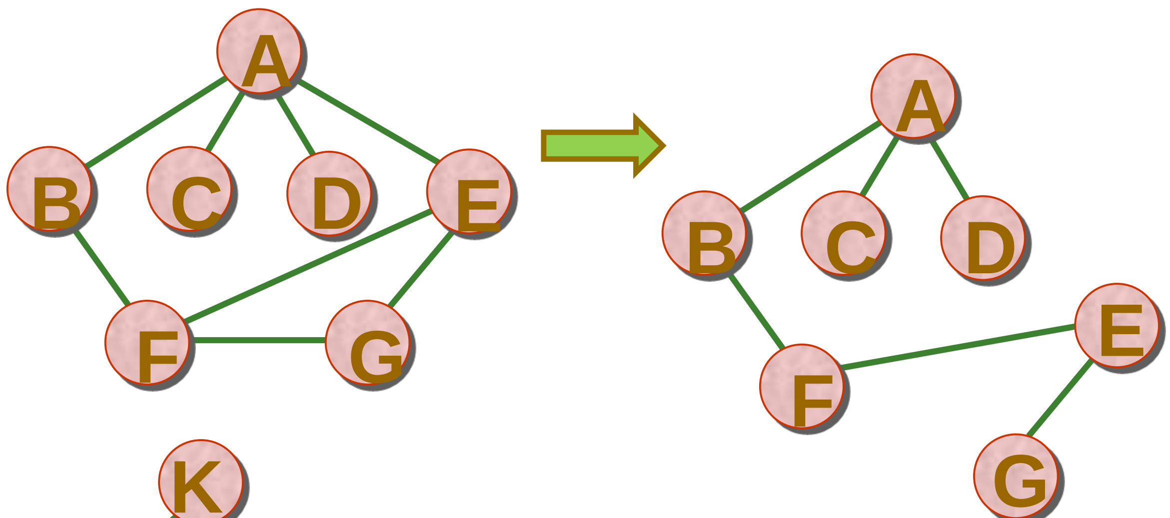
一．无向图的连通性

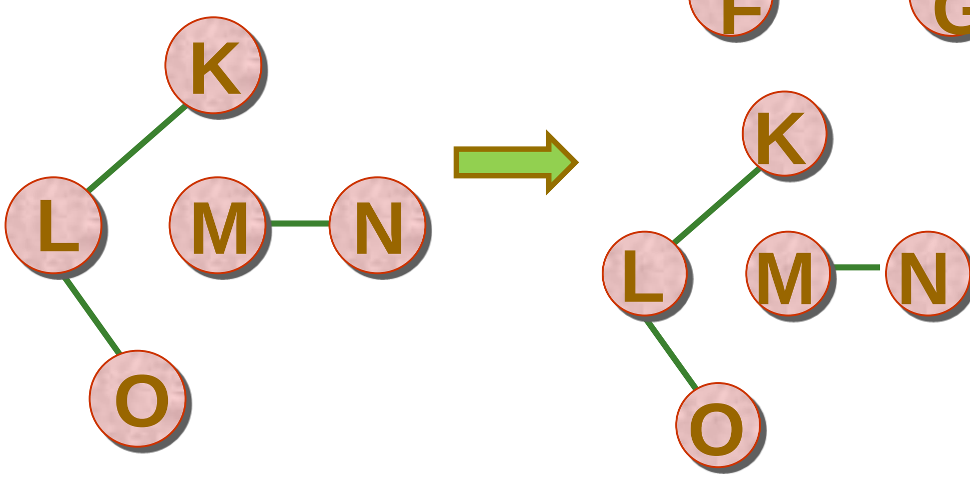
- 非连通图的极大连通子图叫做连通分量
- 若从无向图的每一个连通分量中的一个顶点出发进行DFS或BFS遍历，可求得无向图的所有连通分量的生成树(DFS或BFS生成树)
- 所有连通分量的生成树组成了非连通图的生成森林

7.4 图的连通

一. 无向图的生成树

- 无向图由DFS遍历，求得连通分量称为DFS生成树
  - 下图的三棵DFS生成树组成一个生成森林
    - A-B-F-E-G-C-D
    - K-L-O
    - M-N

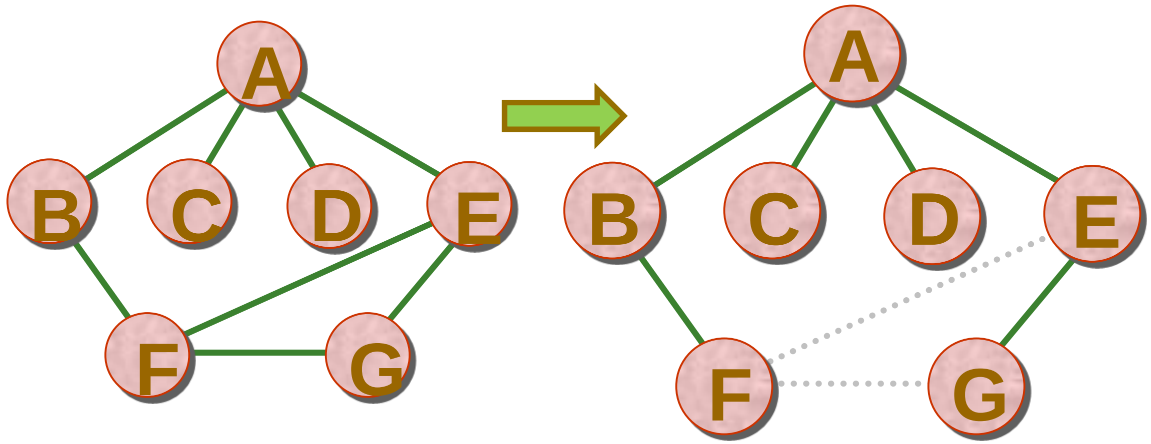


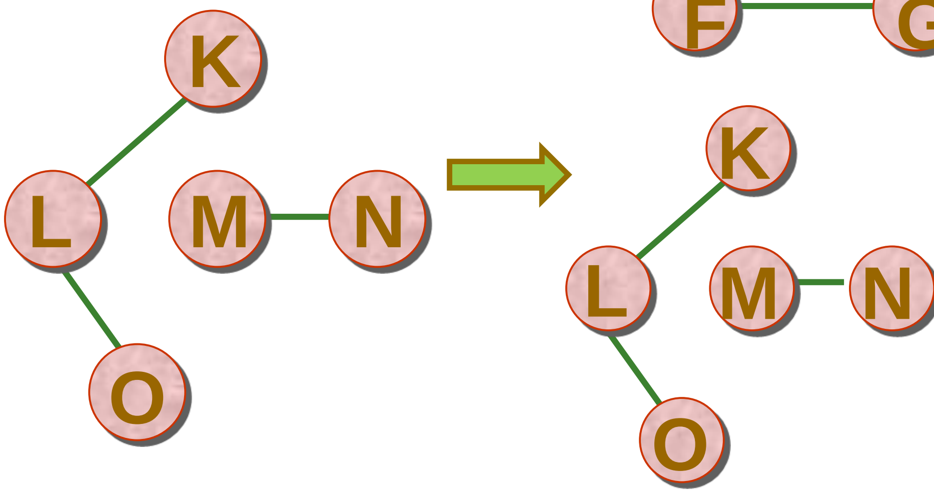


7.4 图的连通

二. 无向图的生成树

- 无向图由BFS遍历，求得连通分量称为BFS生成树
  - 下图的三棵BFS生成树组成一个生成森林
    - A-B-C-D-E-F-G
    - K-L-O
    - M-N

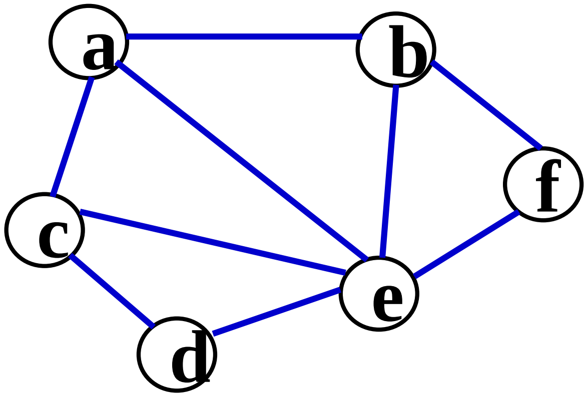




练习

一. 已知下图，从字母最小的顶点出发，求解

- 写出该图的深度优先搜索和广度优先搜索结果
- 画出下图的DFS生成树或生成森林，和BFS生成树或生成森林



7.4 图的连通

三. 最小生成树

- 如果无向图中，边上有权值，则称该无向图为无向网
- 如果无向网中的每个顶点都相通，称为无向连通网
- 最小生成树(Minimum Cost Spanning Tree)是代价最小的生成树，即该连通网的生成树上各边的权值和最小

7.4 图的连通

四. 最小生成树

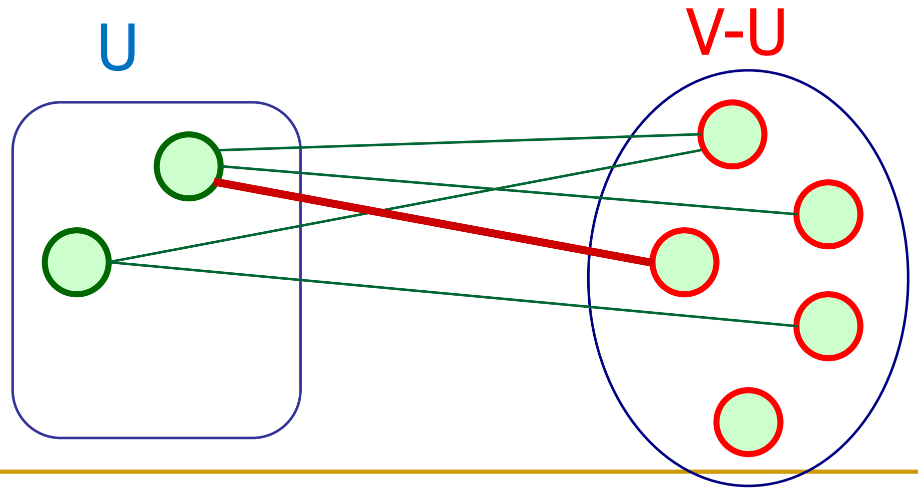
- 最小生成树准则
  - 必须使用且仅使用连通网中的n-1条边来连接网络中的n个顶点；
  - 不能使用产生回路的边；
  - 各边上的权值的总和达到最小。

7.4 图的连通

四. 最小生成树

普里姆(Prim)算法

在生成树的构造过程中，图中 n 个顶点分属两个集合：生成树的顶点集 U和尚未在生成树上的顶点集V-U，应在所有连通U中顶点和V-U中顶点的边中选取权值最小的边逐渐加入生成树的边的集合TE中，相应顶点加入U中。



7.4 图的连通

四. 最小生成树

- 最小生成树生成算法——普里姆(Prim)算法

假设N=(V, E)是连通网， TE是N上最小生成树中边的集合，U是树TE中顶点的集合

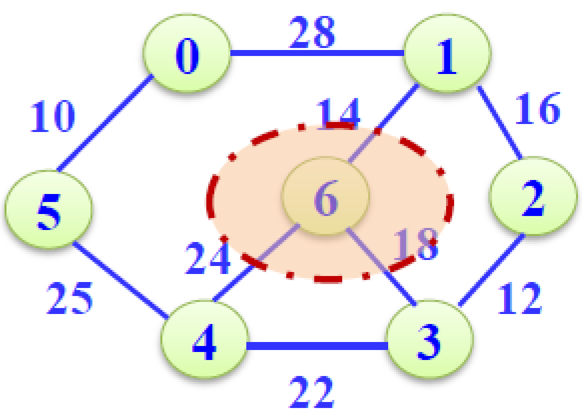
① 初始化：U={u_0}，（$u_0 \in$V)， TE={ }

② 在所有u∈U, v∈V-U的边(u, v)∈E中找一条代价最小的边(u, v)并入集合TE，同时将顶点v并入U

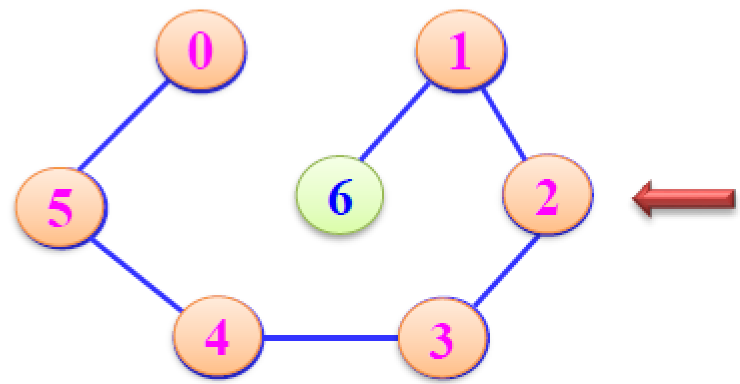
③ 重复2，直到U=V

四. 最小生成树

- 普里姆(Prim)算法举例



图G



最小生成树

U={ 0, 5 , 4 , 3, 2 , 1, 6 }

7.4 图的连通

四. 最小生成树

- 最小生成树生成算法——普里姆(Prim)算法
  - 算法中要解决的4个问题：

① 一个顶点属于哪个集合？

② 如何求U、V-U两个顶点集之间的最小边？（只求一条）

只考虑V-U中顶点*j*到U顶点集的最小边，比较来找最小边

③ 如何存储V-U中顶点*j*到U顶点集的最小边？

④ 图采用哪种存储结构更合适？

7.4 图的连通

四. 最小生成树

- 普里姆(Prim)算法实现

为直观化，以表格来表示实现过程中各变量变化和顶点加入情况。

其中变量定义如下：

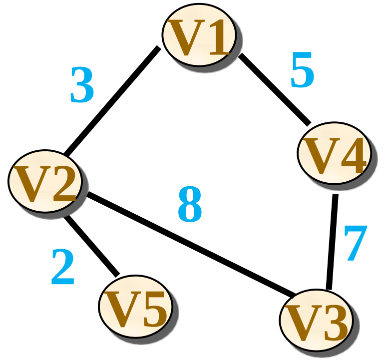
visited — 数组，表示顶点v是否已加入生成树U中，加入为1，否则为0。

dis — 数组，表示从V-U到U中各顶点的最小边权值。

adjVex — 数组，使dis取最小的U中邻接点。

minDis — dis中的最小值

minAdj — minDis对应的V-U中顶点v

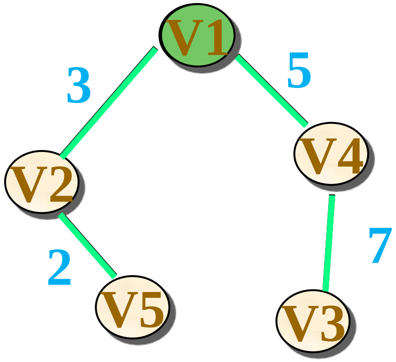


	V1	V2	V3	V4	V5	minDis	minAdj	**U**
visited dis adjvex	0 0	0 ∞	0 ∞	0 ∞	0 ∞	0	V1	V1
visited dis adjvex	1 0	0 **3** V1	0 ∞	0 5 V1	0 ∞	3	V2	V1,V2
visited dis adjvex	1 0	1 3 V1	0 8 V2	0 5 V1	0 **2** V2	2	V5	V1,V2,V5
visited dis adjvex	1 0	1 3 V1	0 8 V2	0 **5** V1	1 2 V2	5	V4	V1,V2 V5,V4
visited dis adjvex	1 0	1 3 V1	0 **7** V4	1 5 V1	1 2 V2	7	V3	V1,V2,V5, V4,V3

7.4 图的连通

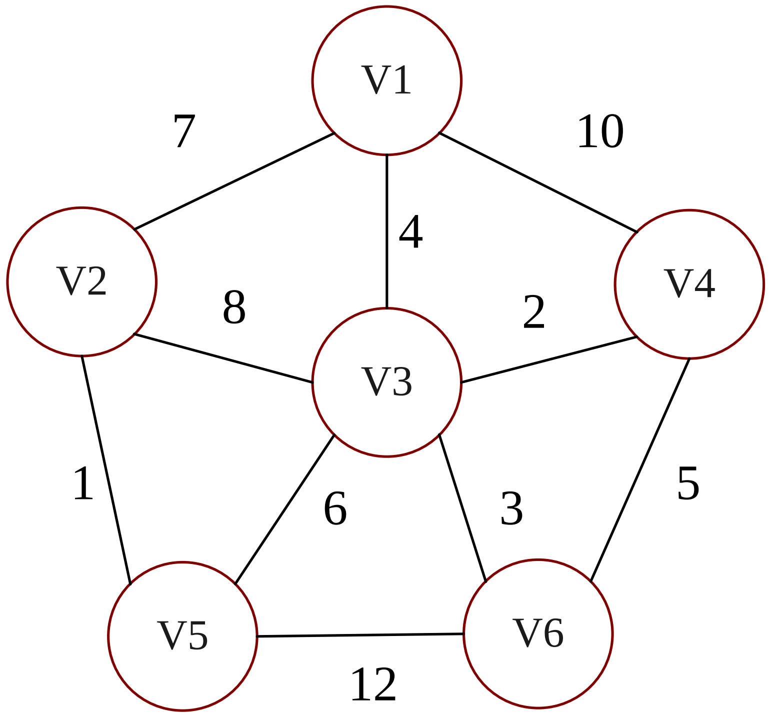
四. 最小生成树

- 普里姆(Prim)算法实现



7.4 图的连通

练习： 用Prim算法求下图的最小生成树，给出生成过程。



7.4 图的连通

四. 最小生成树

- 最小生成树生成算法——克鲁斯卡尔(Kruskal)算法
  - 按**权值的递增次序**选择合适的边来构造最小生成树的方法
  - 算法思想：

假设N=(V, E)是连通网

① 初始，有n个顶点而无边的非连通图T={V, {}}，图中每个顶点自成一个连通分量；

② 在E中按权值从小到大的顺序依次选取一条代价最小、且其两个顶点分别属于不同的连通分量的边，将其加入T中；

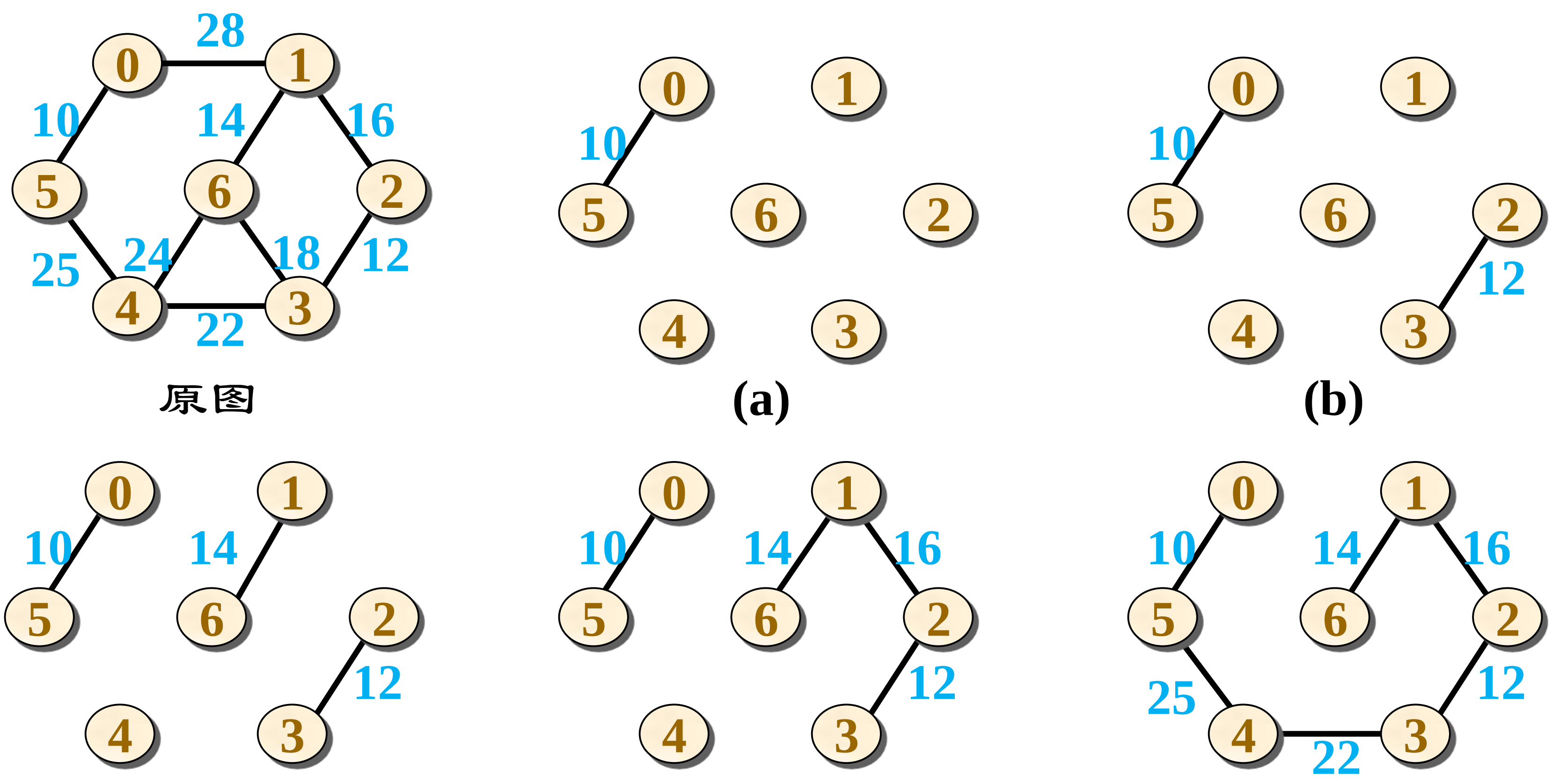
③ 重复2，直到T[illegible]有顶点都在同一连通分量上。

若该边相关联的两个顶点已在一个连通分量上，加入该边就会形成回路，生成树是没有回路的，所以得舍弃此边

7.4 图的连通

四. 最小生成树

- 克鲁斯卡尔(Kruskal)算法举例



原图 (a) (b)

(c) (d) (e) (f)

7.4 图的连通

四. 最小生成树

- 最小生成树生成算法——克鲁斯卡尔(Kruskal)算法
  - 算法设计中应解决的3个问题：

① 图采用哪种存储结构更合适？

算法是从边的权值递增次序出发构造生成树，与边的数目有关，考虑图的邻接表形式

② 边的排序问题？

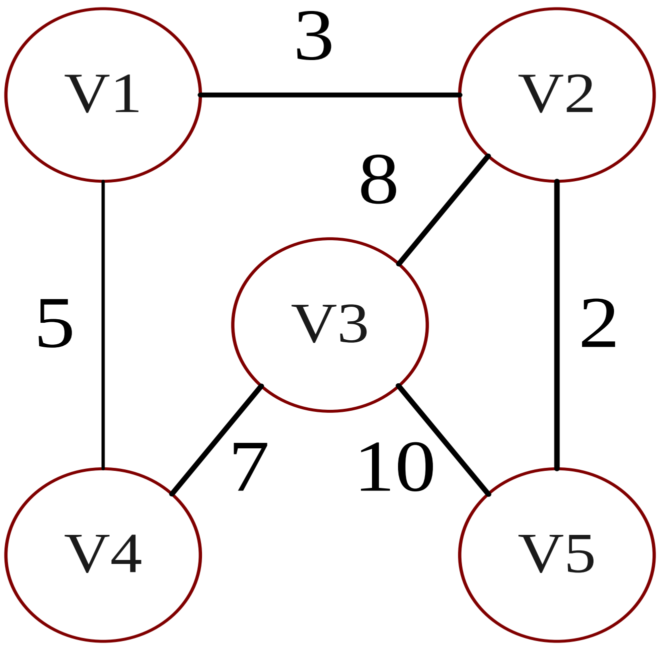
选择合适的排序算法

③ 如何解决加入一条边后是否出现回路？

采用连通分量编号或顶点集合编号

7.4 图的连通

练习： 用Kruskal算法求下图的最小生成树，给出生成过程。



7.4 图的连通

四. 最小生成树

- 两种算法的对比
  - **Prim**算法更适合求稠密网的最小生成树（稠密图的无向网更适合使用邻接矩阵形式存储），与网中的边数e无关，算法时间复杂度为 $O(n^2)$
  - **Kruskal**算法更适合求稀疏网的最小生成树（稀疏图的无向网更适合使用邻接表形式存储），算法时间复杂度为 $O(e\ \log e)$

练习

- 已知下图所示，采用普里姆(Prim)算法和克鲁斯卡尔(Kruskal)算法求最小生成树，要求写出详细求解过程.