6.6 动态等价关系与并查集

动态等价关系

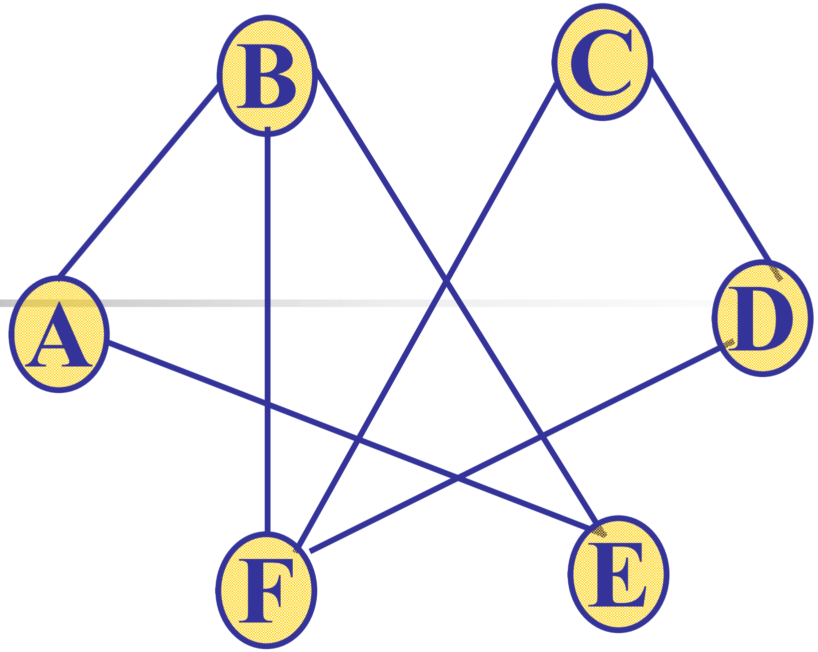
- 在求解实际应用问题时常会遇到等价类问题。
- 从数学上看，等价类是一个对象(或成员)的集合，在此集合中所有对象应满足等价关系。
- 若用符号R表示集合上的等价关系，那么对于该集合中的任意对象x, y, z，下列性质成立：
  - 自反性：x R x 。
  - 对称性：若 x R y, 则 y R x。
  - 传递性：若 x R y且 y R z, 则 x R z。
- 因此，等价关系是集合上的一个自反、对称、传递的关系。≡

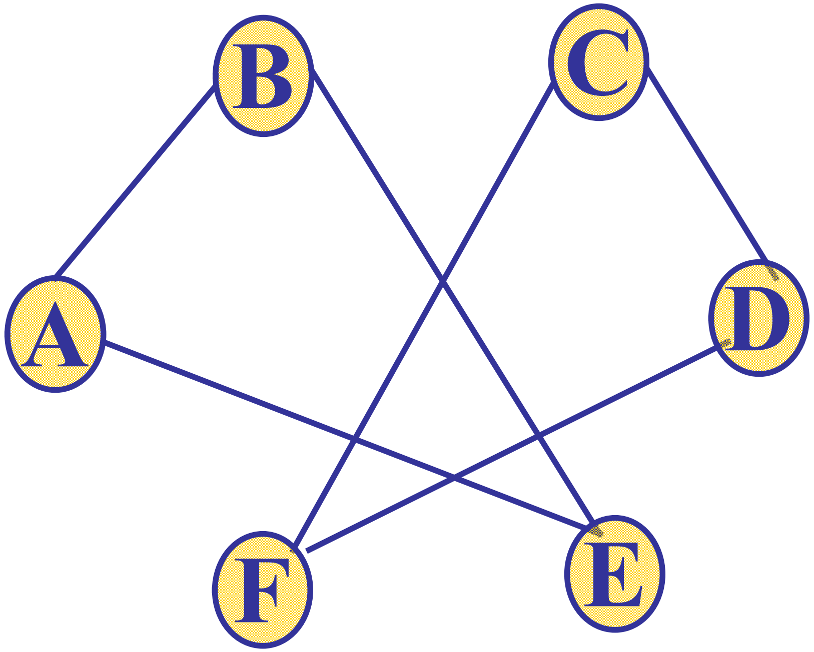
"相等"(=)就是一种等价关系，它满足上述的三个特性。
图中顶点的连通也是一种等价关系

动态等价关系

- 设R是集合S上的等价关系，对任何$x \in S$，令$[x]_R=\{y \mid y \in S \wedge xRy\}$。
- 则$[x]_R \subseteq S$，称为由$x \in S$生成的一个R等价类。
- 一个集合 S 中的所有对象可以通过等价关系划分为若干个互不相交的子集$S_1, S_2, S_3, \cdots$，它们的并就是 S。这些子集即为等价类

位于一个连通图（分量）中的顶点——一个等价类





V={A,B,C,D,E,F}

π={S1,S2}

S1={A,B,E}

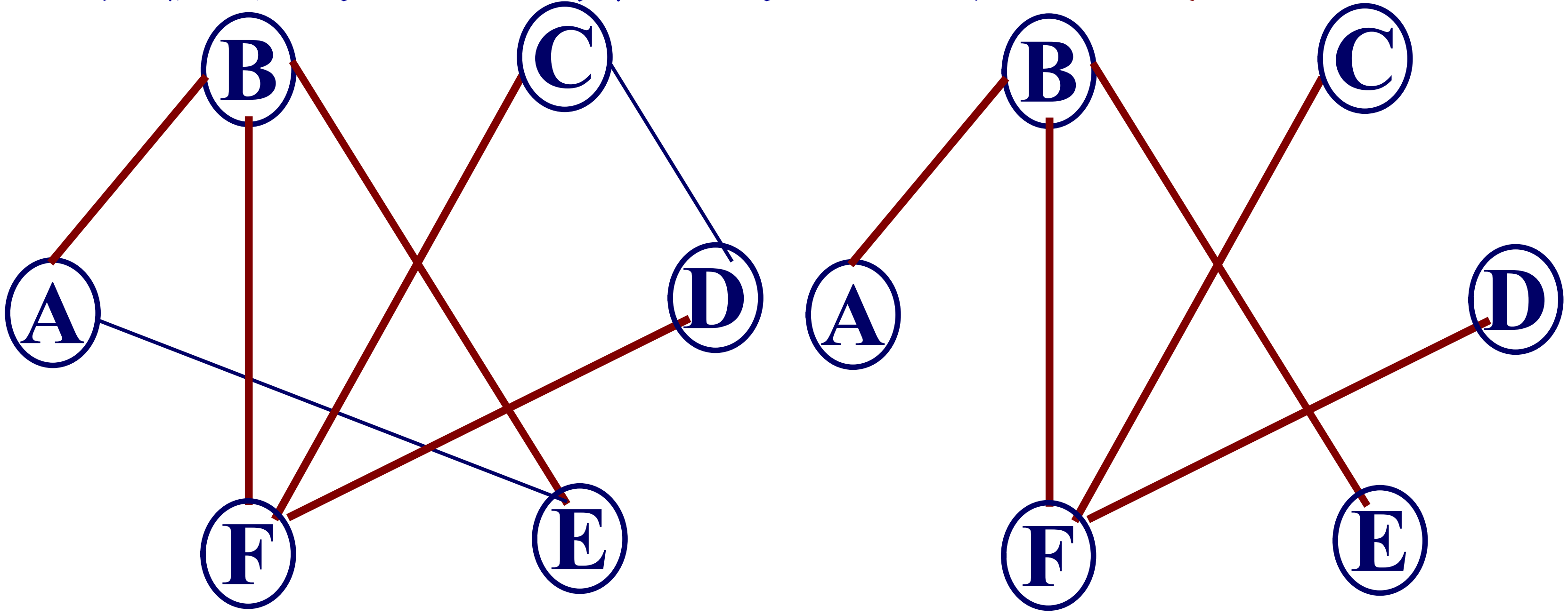
S2={C,D,F}

确定等价类的方法

- 假设：集合 S 中含有n个元素，m个形如(x,y)(x,y$\in$S)的等价偶对。求 S的划分。
  - 集合S中每个元素各自形成一个只含单个成员的子集--$S_1, S_2, S_3, \cdots S_n$。
  - 依次读入m个偶对(x,y)：若$x\in S_i$，$y\in S_j$，$S_i\neq S_j$，则S_i并入S_j,将S_i置空。
- 主要操作：
  - **IS x ≡ y----find(*x*)**： 查找x，y所属的等价类，判断是否为同一个等价类
  - **MAKE ----union(*t*,*u*)**： 将2个等价类合并为一个等价类

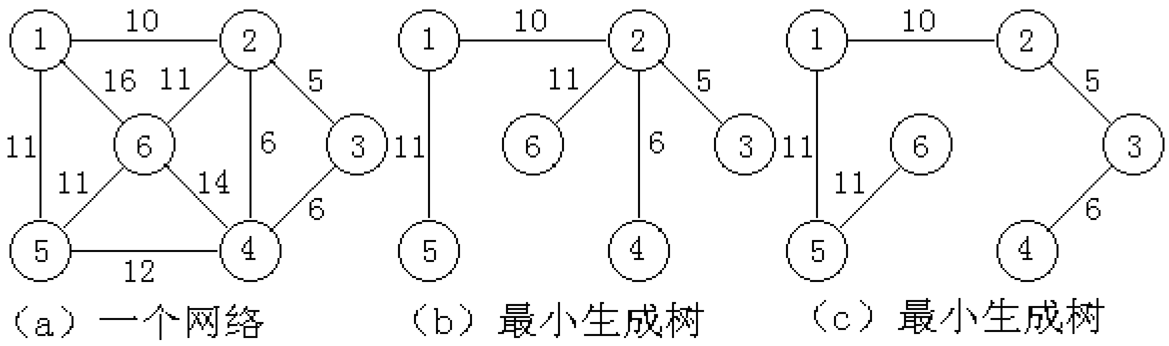
等价类的应用—最小生成树

- 假设一个连通图有 *n* 个顶点和 *e* 条边，其中 *n*-1 条边和 *n* 个顶点构成一个极小连通子图，称该极小连通子图为此连通图的**生成树**。

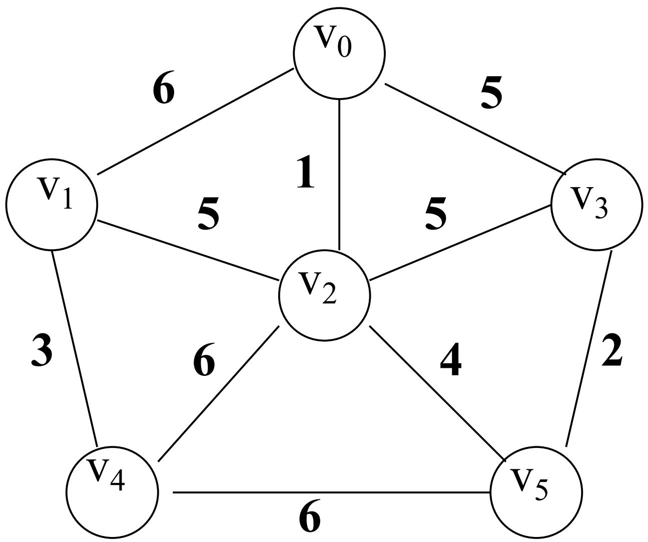


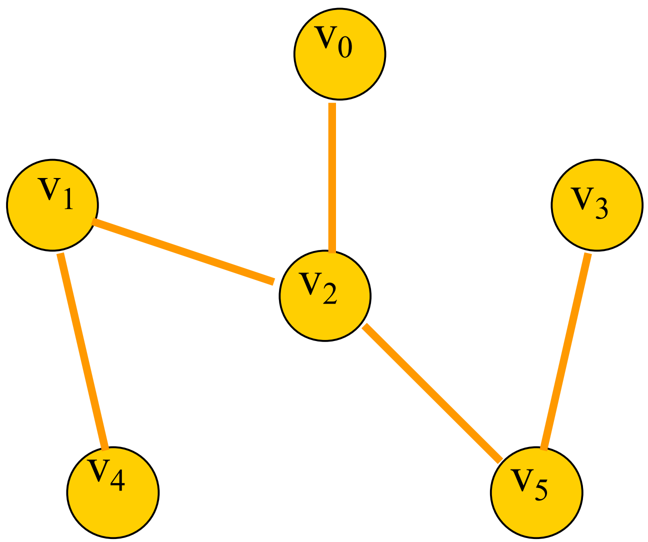
等价类的应用—最小生成树

- 最小生成树：带权图的生成树上的各边权值之和称为这棵树的代价。最小代价生成树是各边权值的总和最小的生成树。



最小生成树--Kruskal算法





一个连通分量——一个连通子图——一个等价类——即：若所加边的2个顶点在同一个等价类，加进去后产生回路

Kruskal算法的基本步骤

- 设G＝(V,E), T为G的最小生成树，初态T＝(V,{})
- n个顶点m条边的图，考察G的边集E中的各条边：
  - 从m条边中选最小的，m-1次比较，判断加进去是否产生回路
  - 从m-1条边中选最小的，m-2次比较，判断加进去是否产生回路
  - ……
- ✓ 最坏情况：选边m(m-1)/2次比较　改为排序—O(mlogm)
- 加一条边判断是否有回路：深度优先搜索$O(n+m_T)=O(n)$ (因为$m_T<n$)→最坏情况O(mn)
- Kruskal算法最坏情况O(mn)+O(m(m-1)/2)----

  完全图$m=n^2$

  说明：m_T为生成树中的边数

可否改进？

若所加边的2个顶点在同一个连通分量----加进去后产生回路

等价类的表示

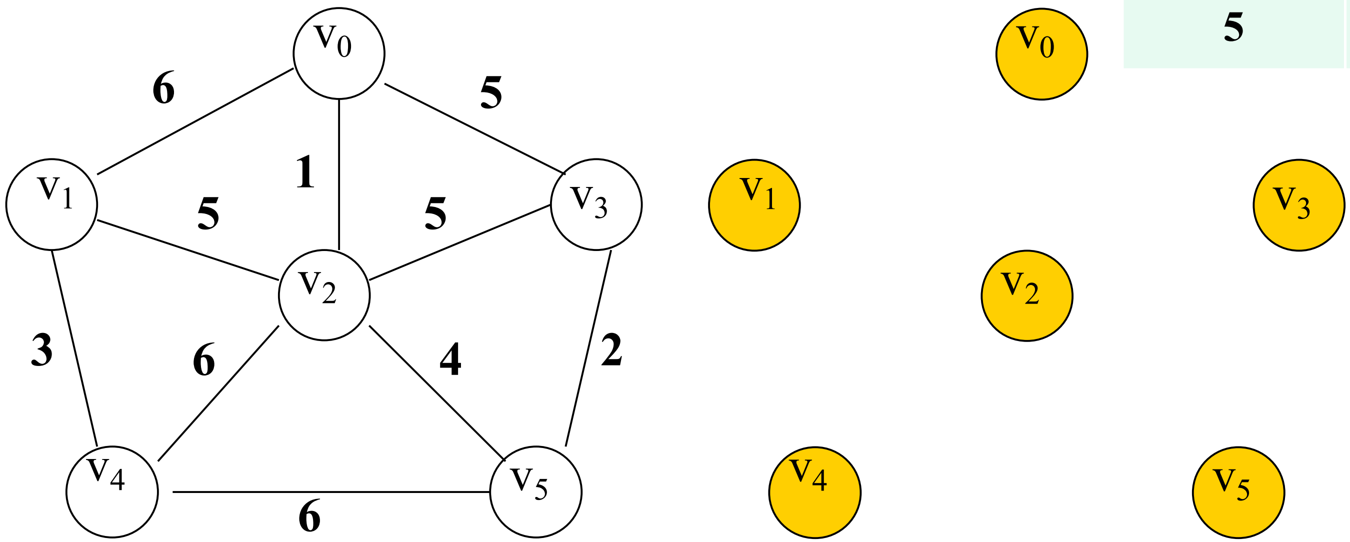
- 数组
- 链表
- 树

等价类的表示——数组

- 数组记录每个数据元素所属等价类的编号
- Find(x)—O（1）
- Union（x,y）--O(n):修改其中一个等价类中所有数据元素的等价类编号
- n个顶点m条边kruscal算法构造最小生成树:

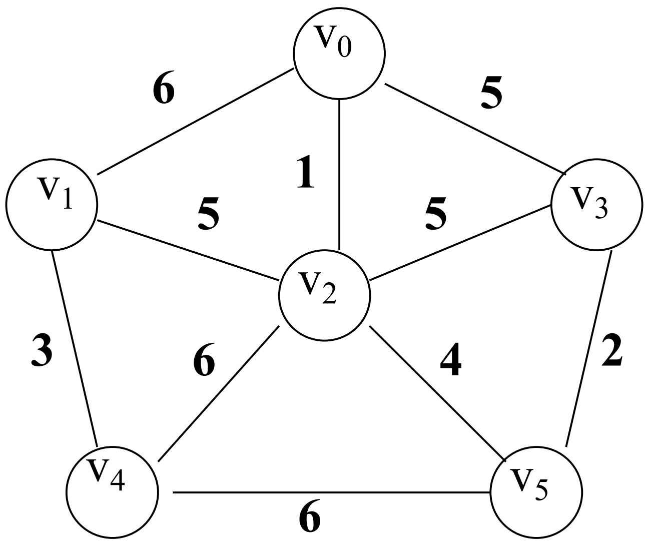
  $O(2m)+O(mlogm)+O(n^2)$

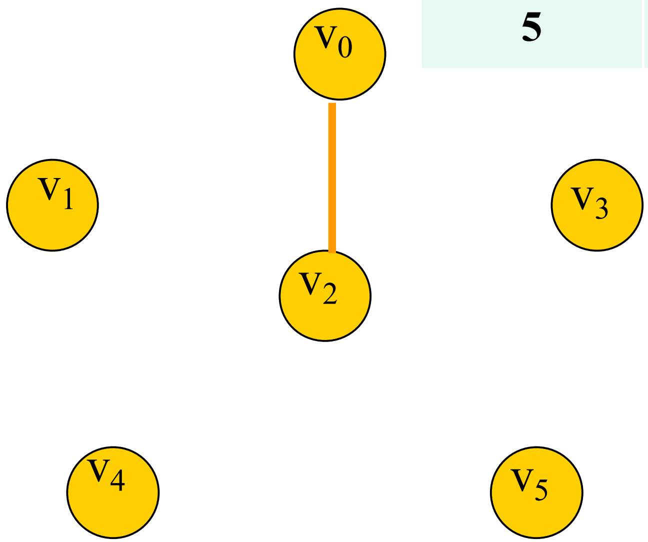
data	label
0	0
1	1
2	2
3	3
4	4
5	5



(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

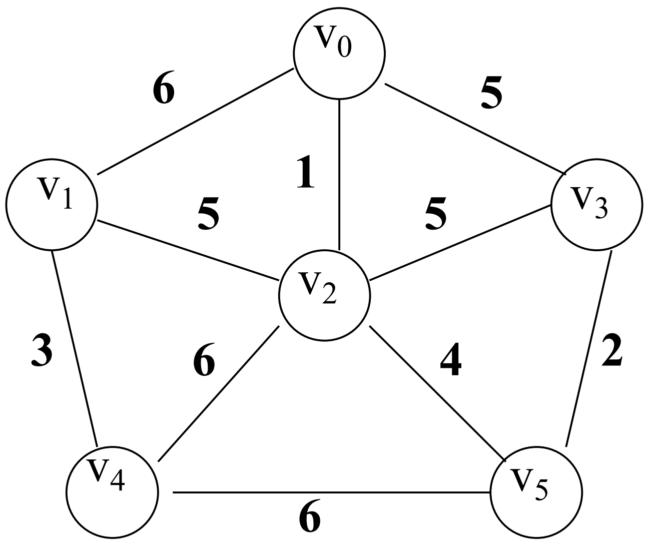
data	label
0	0
1	1
2	2
3	3
4	4
5	5

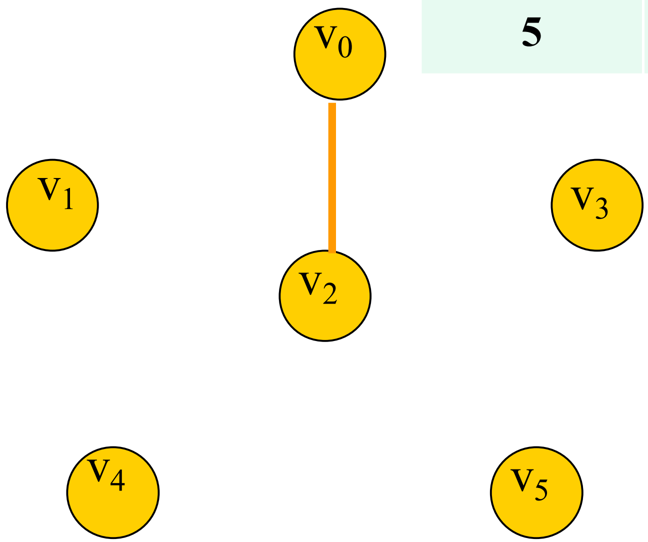




(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

data	label
0	0
1	1
2	0
3	3
4	4
5	5

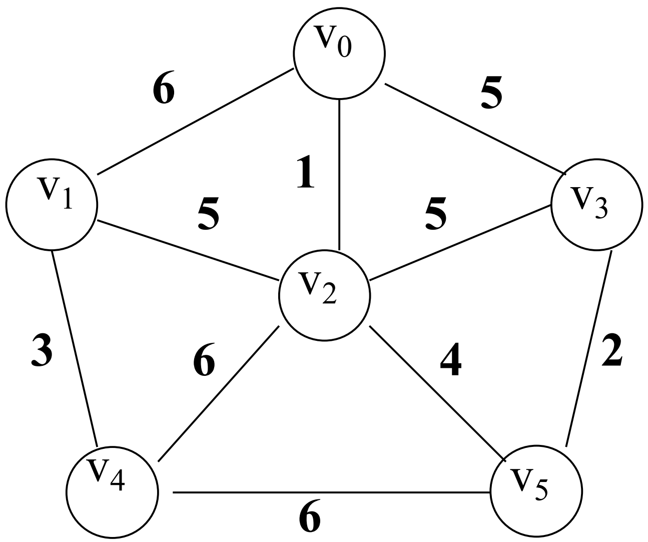


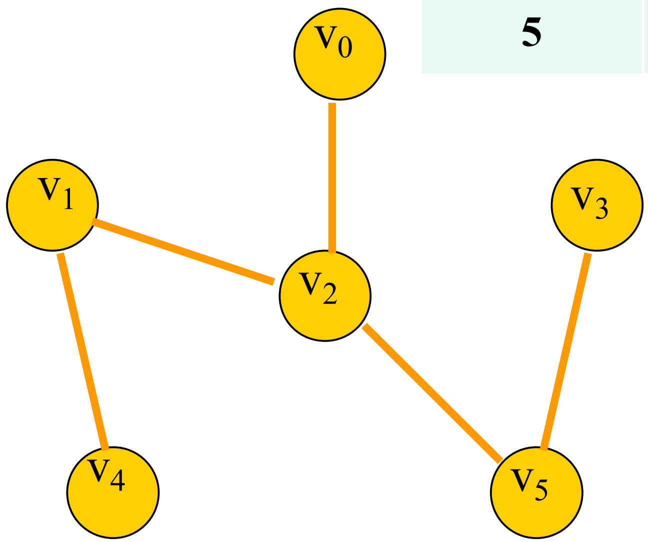


(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

n-1次MAKE操作-----O(n^2)

data	label
0	1
1	1
2	1
3	1
4	1
5	1





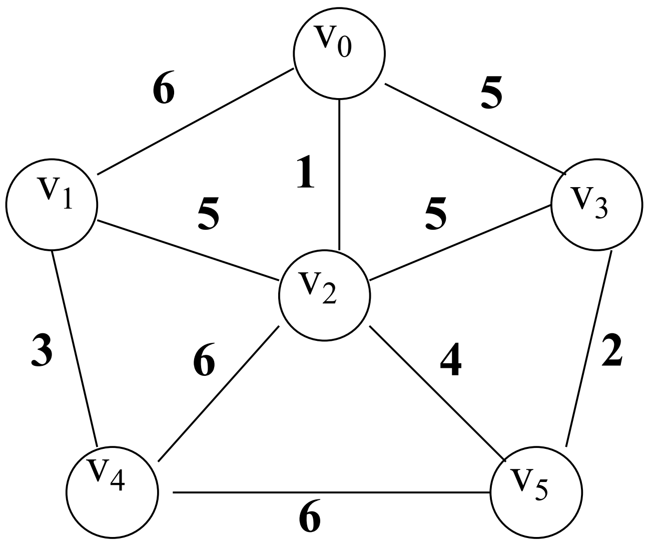
(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

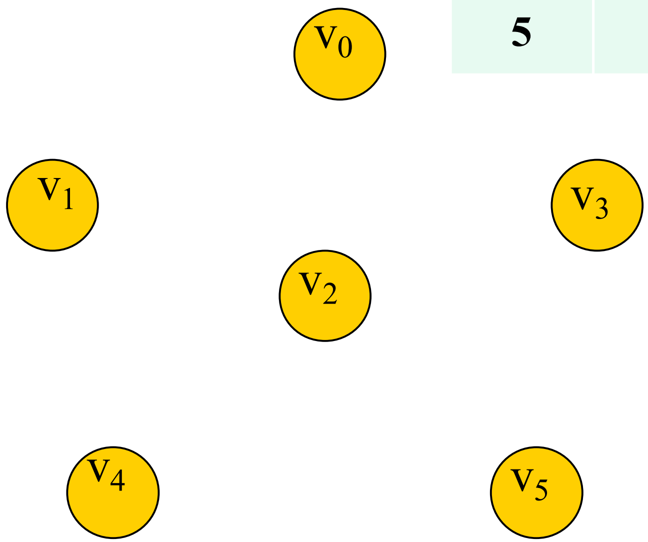
n个顶点m条边kruscal算法构造最小生成树：　O(2m)+O(mlogm)+O(n^2)

O(n(n-1))

等价类的表示——链表

data	head	last	next
0	0	0	-1
1	1	1	-1
2	2	2	-1
3	3	3	-1
4	4	4	-1
5	5	5	-1

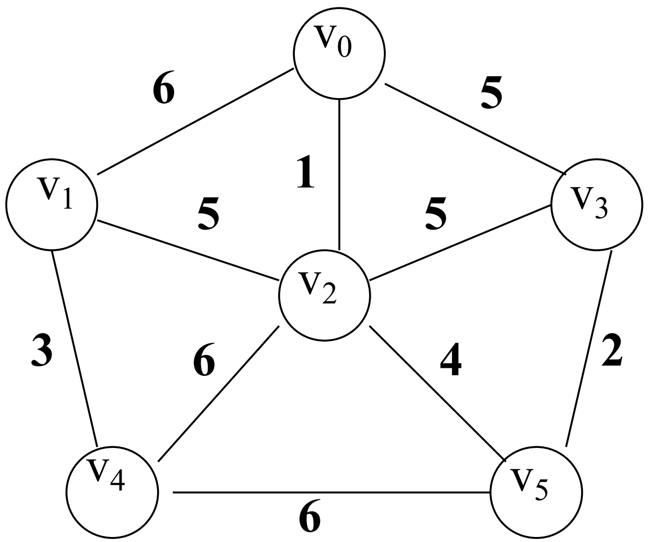


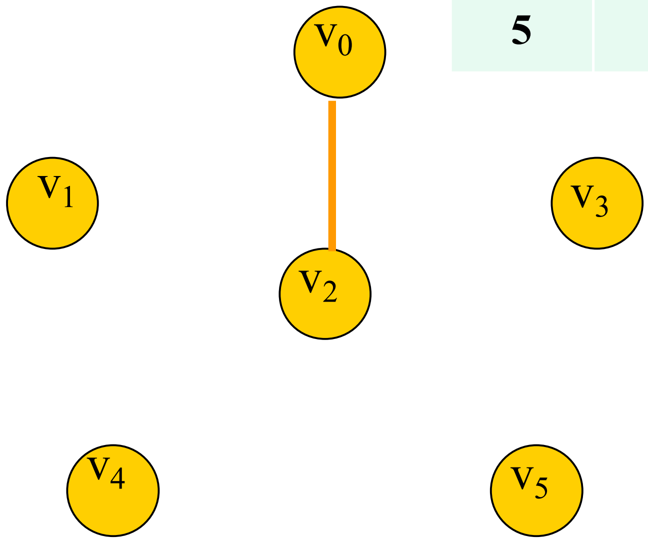


(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

等价类的表示——链表

data	head	last	next
0	0	0	-1
1	1	1	-1
2	2	2	-1
3	3	3	-1
4	4	4	-1
5	5	5	-1

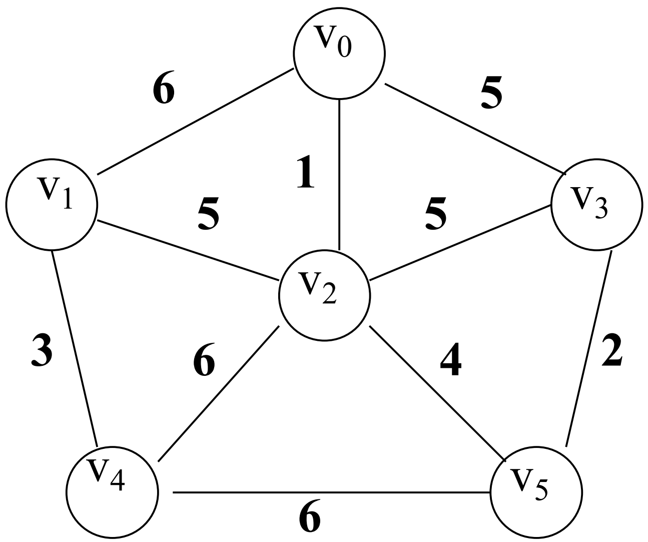


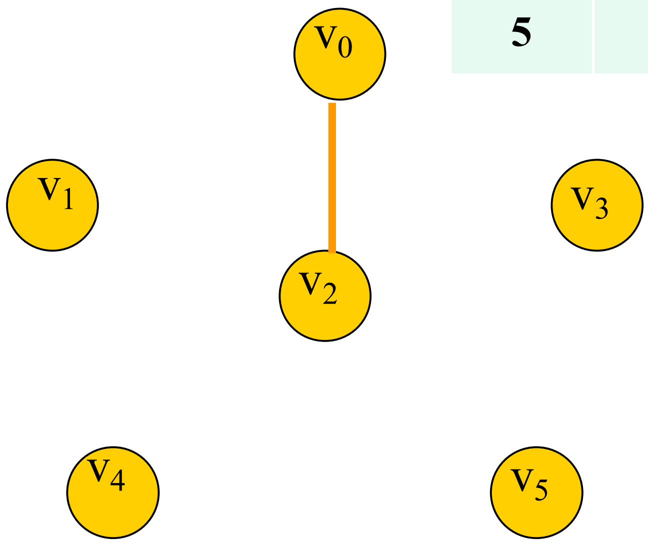


(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

等价类的表示——链表

data	head	last	next
0	0	2	2
1	1	1	-1
2	0	2	-1
3	3	3	-1
4	4	4	-1
5	5	5	-1





(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

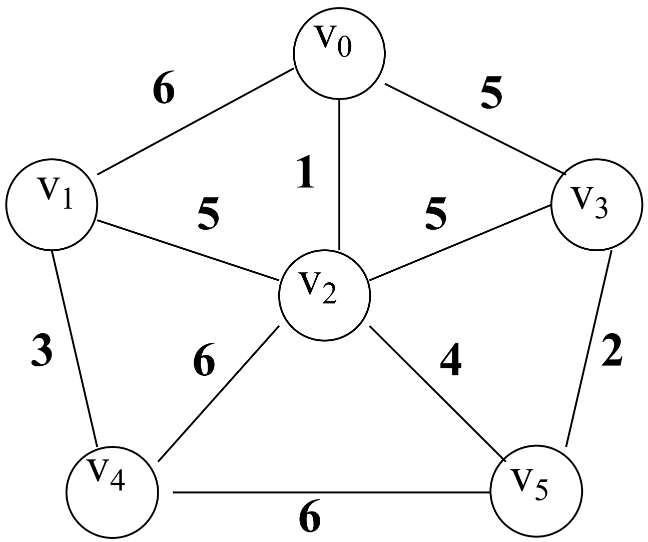
小集合合并到大集合，合并后集合体积至少为以前的2倍，每个顶点至多移动*logn*次

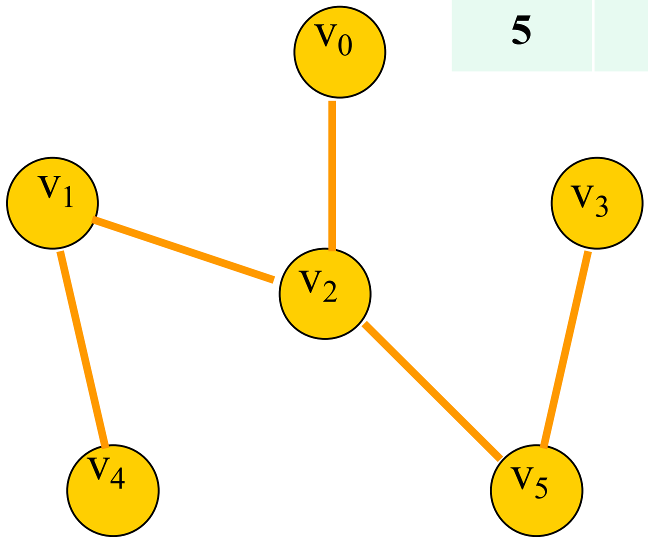
$n \geqslant 2^k$

等价类的表示——链表

最小生成树-----O(m*logm*)+O(2*m*)+O(*nlogn*)

data	head	last	next
0	0	4	2
1	0	4	4
2	0	2	3
3	0	5	5
4	0	4	-1
5	0	5	1





(v0,v2),(v3,v5),(v1,v4),(v2,v5),(v2,v3),(v0,v3),(v1,v2),(v0,v1),(v2,v4),(v4,v5)

小集合合并到大集合--改进为O(*nlogn*)