ACM / ICPC程序设计

并查集

作者：屠添翼

Disjoint Sets（并查集）

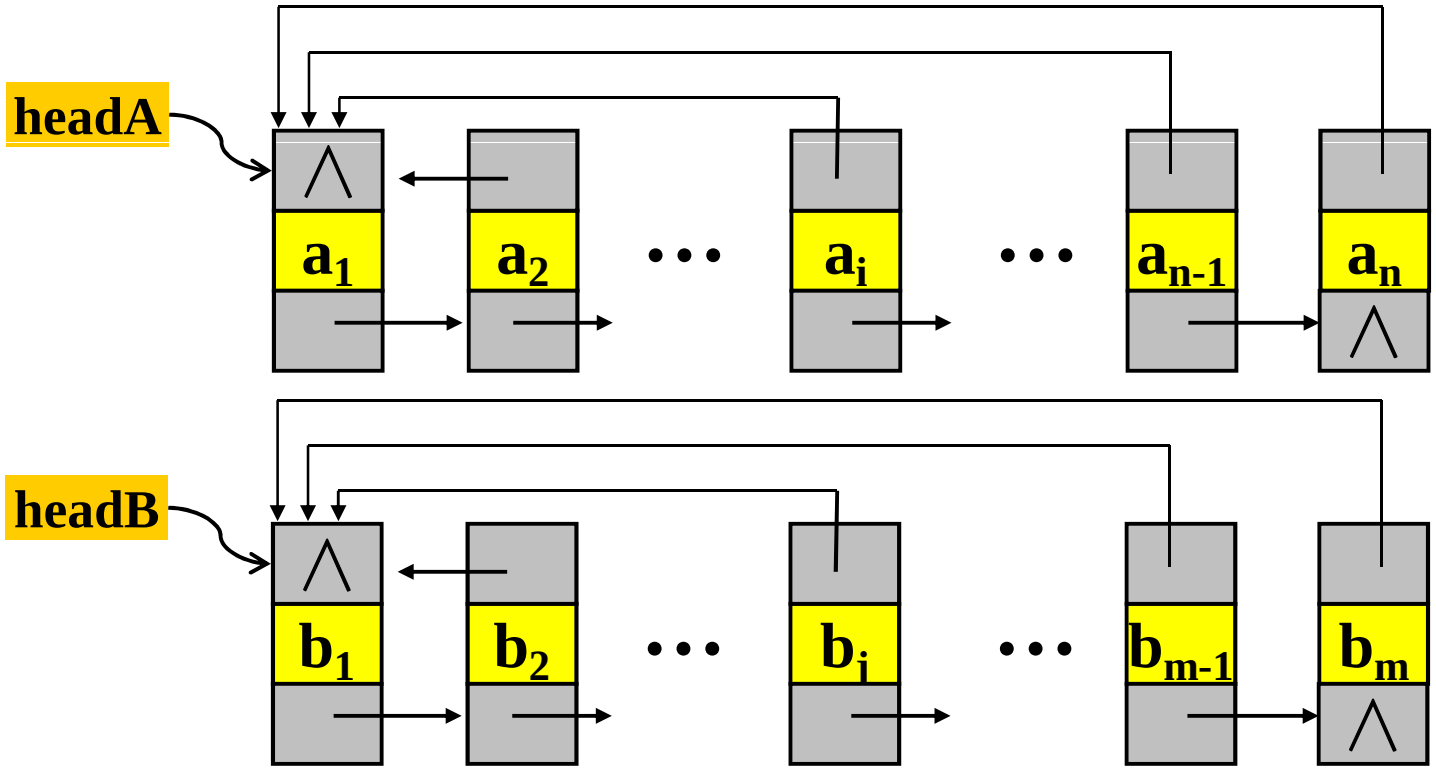
- 基本概念
  - 如果给出各个元素之间的联系，要求将这些元素分成几个集合，每个集合中的元素直接或间接有联系。在这类问题中主要涉及的是对集合的合并和查找，因此将这种集合称为并查集。
  - 链结构的并查集
  - 树结构的并查集

链结构的并查集

- 表中的每个元素结点设两个指针：一个指向同一集合中的下一个元素；另一个指向表首元素。

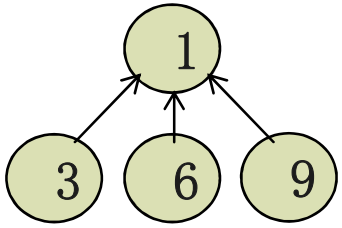
  例如，集合{a_1，a_2，…，a_i，…，a_n}

  集合{b_1，b_2，…，b_i，…，b_n}

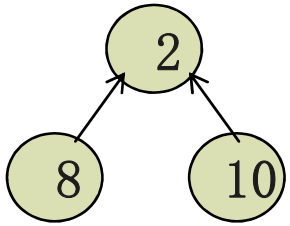


树结构的并查集

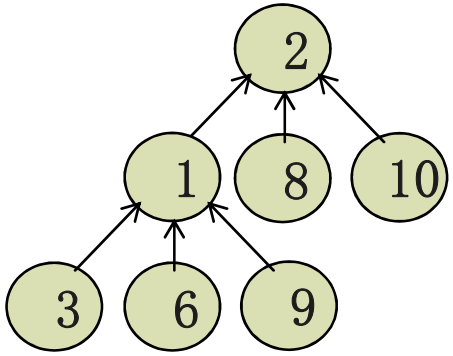
- 采用树结构支持并查集的计算能够满足我们的要求。并查集与一般的树结构不同，每个顶点记录的不是它的子结点，而是将它的父结点记录下来。



S1 = {1,3,6,9}



S2 = {2,8,10}

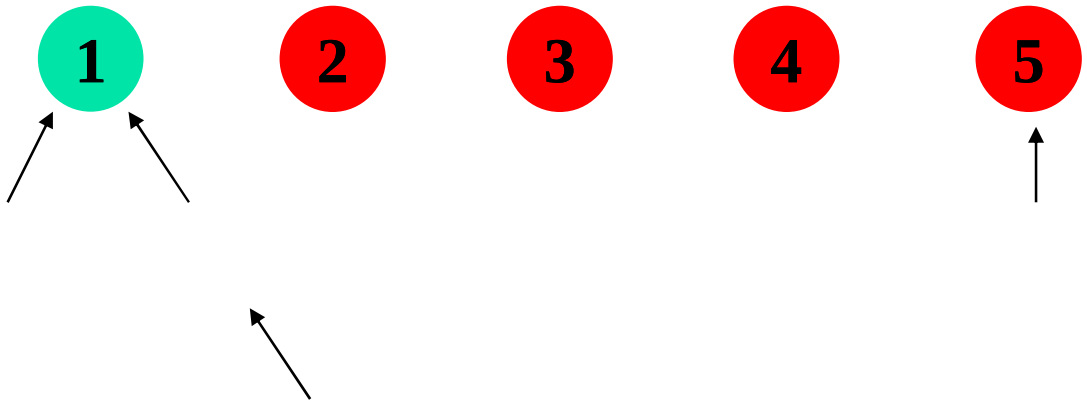


S3 = s1 U s2

并查集的主要操作

- 1－合并两个不相交集合
- 2－判断两个元素是否属于同一个集合
- 3－路径压缩

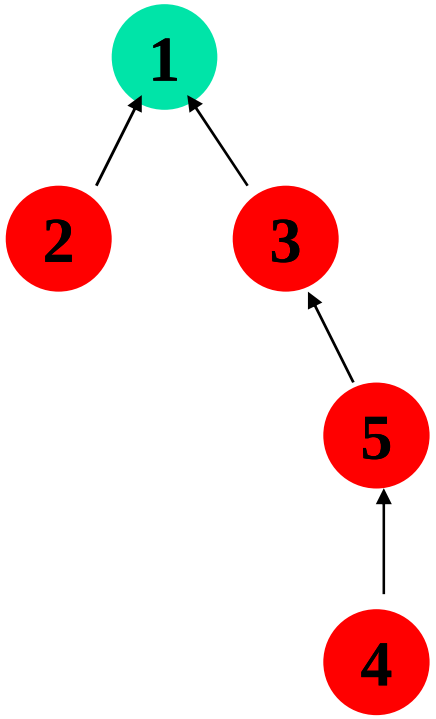
元素的合并图示



- 合并1和2
- 合并1和3
- 合并5和4
- 合并5和3

判断元素是否属于同一集合

- 用father[i]表示元素i的父亲结点，如刚才那个图所示。



faher[1]=1

faher[2]=1

faher[3]=1

faher[4]=5

faher[5]=3

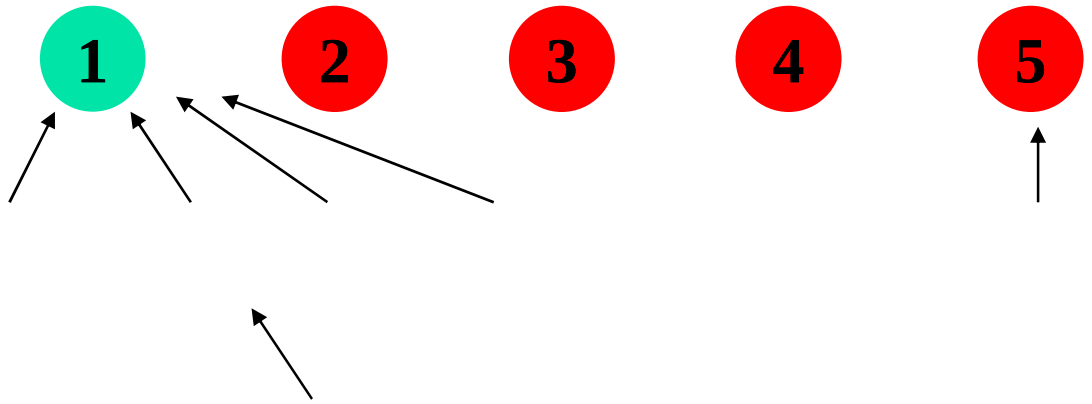
判断元素是否属于同一集合

- 由此用某个元素所在树的根结点表示该元素所在的集合。
- 判断两个元素时候属于同一个集合的时候，只需要判断他们所在树的根结点是否一样即可。
- 也就是说，当我们合并两个集合的时候，只需要在两个根结点之间连边即可。

路径压缩

- 上述的做法是指通过找寻元素的父亲结点一层层向上找到最顶的元素来确定在哪个集合。当这课树是链的时候，可见判断两个元素是否属于同一集合需要O(N)的时间，于是需要进行路径压缩。
- 路径压缩实际上是在找完根结点之后，在递归回来的时候顺便把路径上元素的父亲指针都指向根结点。

路径压缩示意图



- 由此我们得到了一个复杂度只是O(1)的算法

程序清单

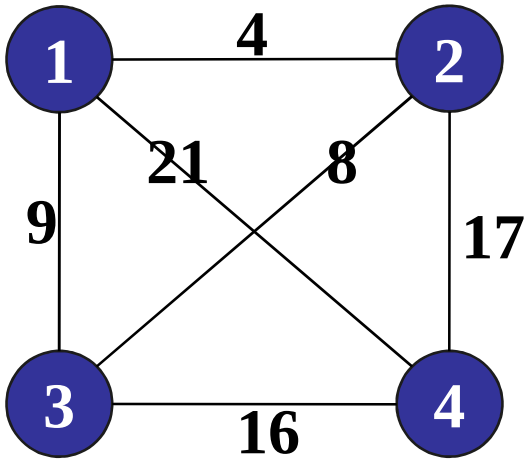
```
function getfather(v:integer):integer;
  begin
    if (father[v]=v) then
      getfather:=v
    else
      begin
        father[v]:=getfather(father[v]);
        getfather:=father[v];
      end;
  end;
```

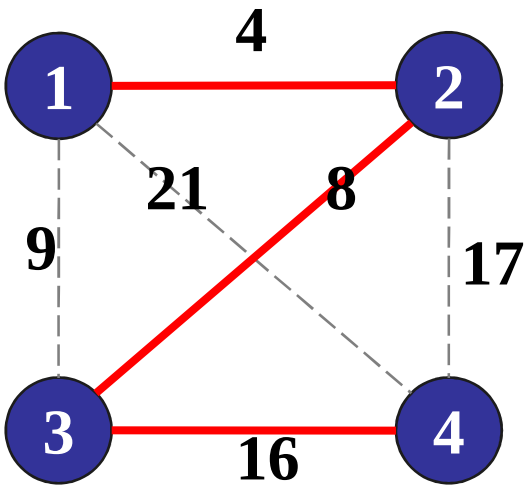
程序清单

```
function judge(x,y:integer):boolean;
 var fx,fy : integer;
  begin
    fx := getfaher(x);
    fy := gefather(y);
    If fx=fy then judge := exit(true)
                else judge := false;
   father[fx] := fy;{合并两个集合}
  end;
```

【POJ1258】Agri-Net

Sample Input
4
0 4 9 21
4 0 8 17
9 8 0 16
21 17 16 0



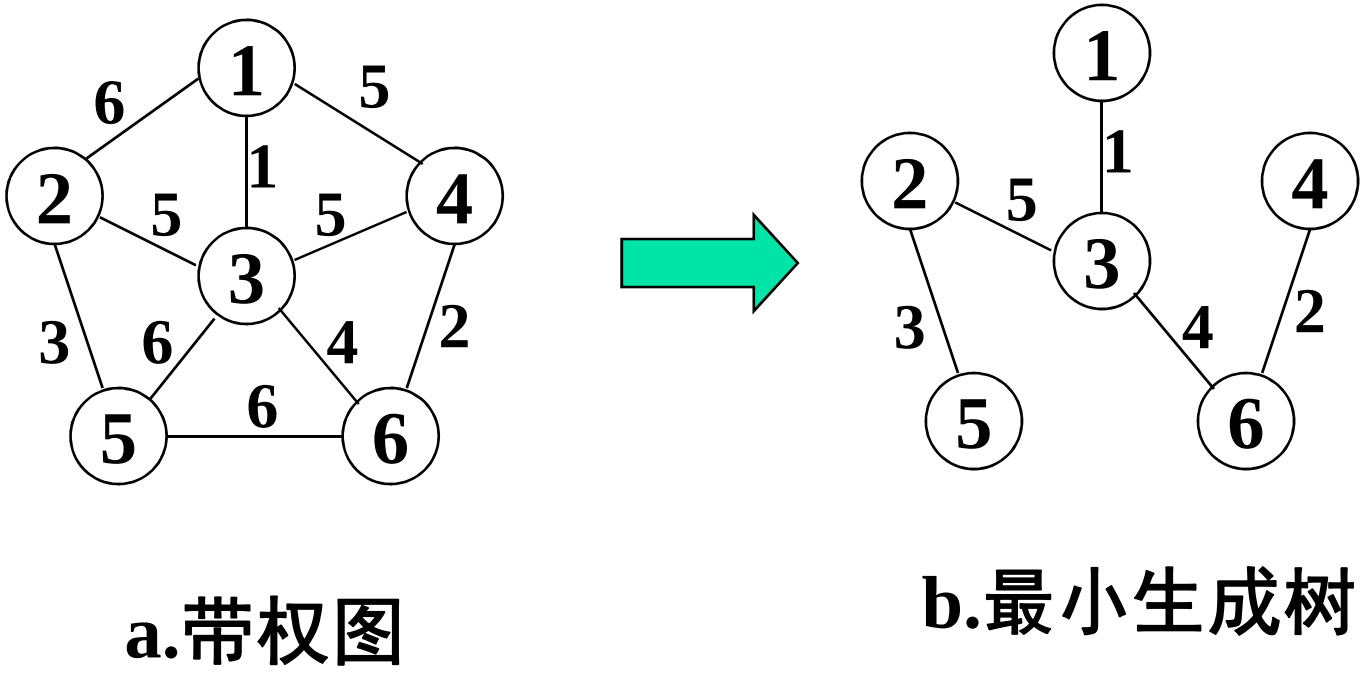


Sample Output
28

最小生成树

- 设G =(V,E)是无向连通带权图，即一个网络。E中每条边(v,w)的权为c[v][w]。如果G的子图G’是一棵包含G的所有顶点的树，则称G’为G的生成树。生成树上各边权的总和称为该生成树的耗费。在G的所有生成树中，耗费最小的生成树称为G的最小生成树。
- 网络的最小生成树在实际中有广泛应用。例如，在设计通信网络时，用图的顶点表示城市，用边(v,w)的权c[v][w]表示建立城市v和城市w之间的通信线路所需的费用，则最小生成树就给出了建立通信网络的最经济的方案。

最小生成树示例



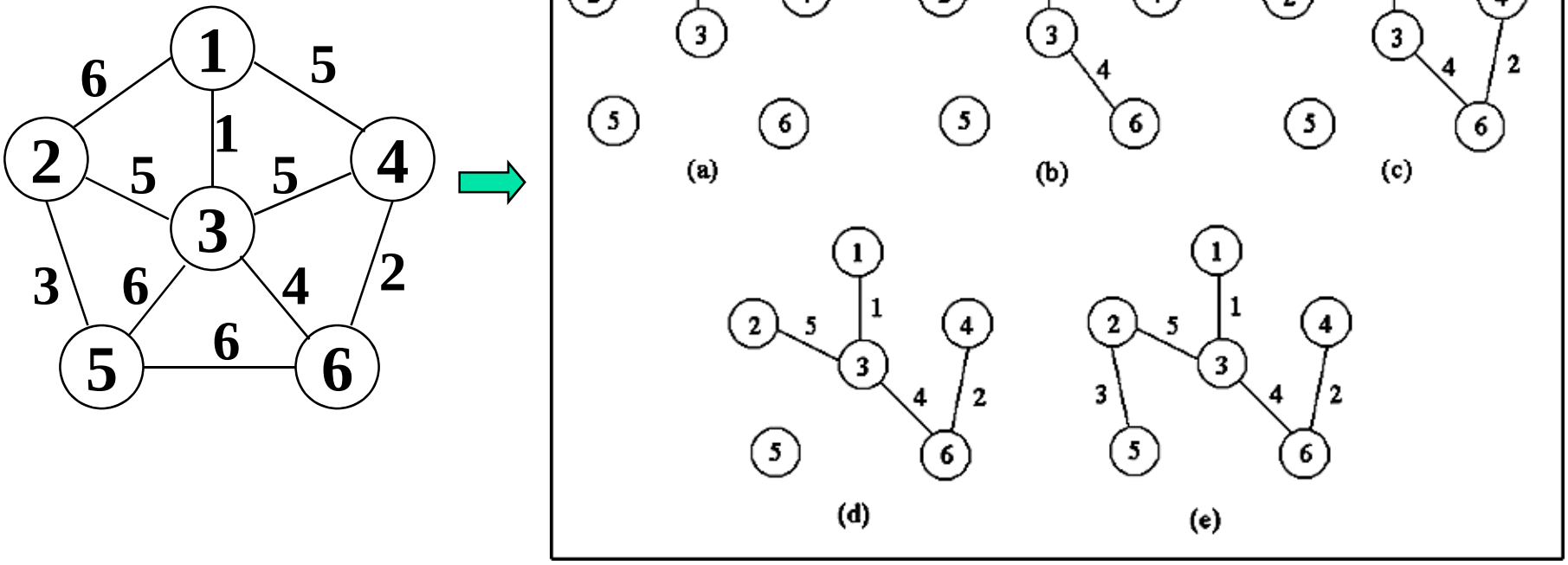
a.带权图

b.最小生成树

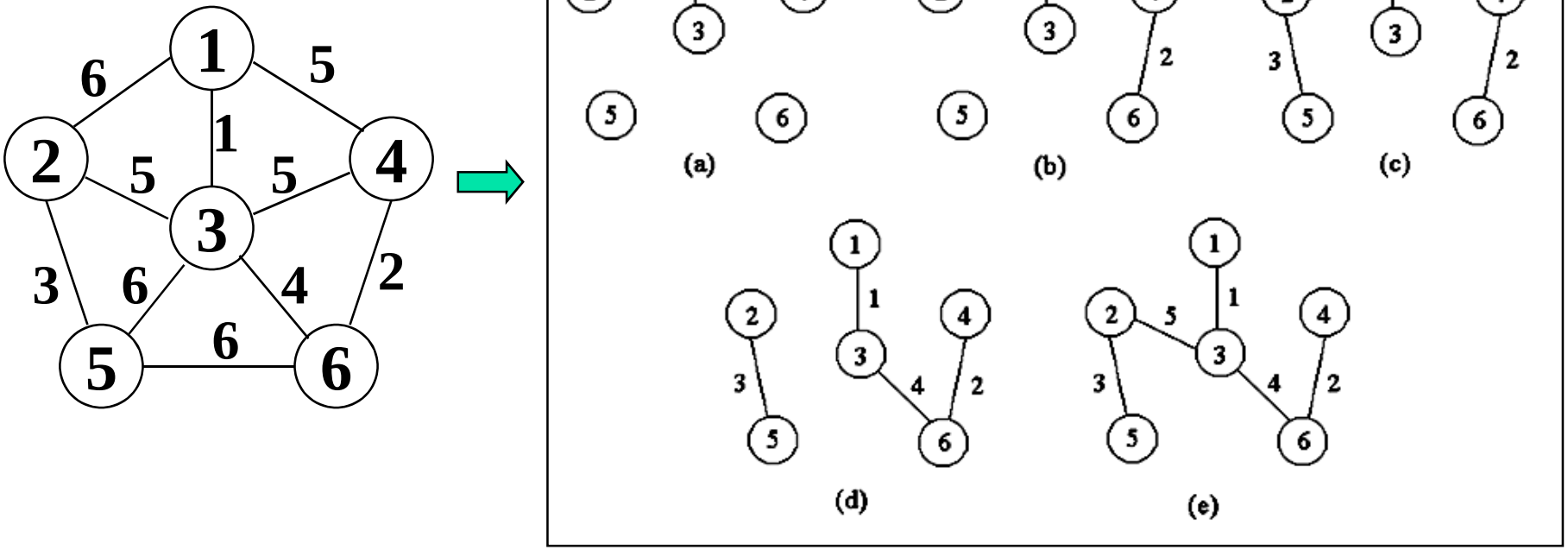
最小生成树算法

- Prim算法
- Kruskal算法

Prim 算法图解



Kruskal算法图解



Kruskal算法基本思想

- 首先将G的n个顶点看成n个孤立的连通分支。
- 将所有的边按权从小到大排序。
- 然后从第一条边开始，依边权递增的顺序查看每一条边，并按下述方法连接2个不同的连通分支：
  - 当查看到第k条边(v,w)时，如果端点v和w分别是当前2个不同的连通分支T1和T2中的顶点时，就用边(v,w)将T1和T2连接成一个连通分支，然后继续查看第k+1条边；
  - 如果端点v和w在当前的同一个连通分支中，就直接再查看第k+1条边。这个过程一直进行到只剩下一个连通分支时为止。

【POJ1258】Agri-Net 解题思路

- Kruskal算法+并查集