5.3 集合及运算

- 并查集问题中集合存储如何实现？

例子：有10台电脑{1, 2, 3, . . . , 9, 10}，已知下列电脑之间已经实现了连接：

1和2、2和4、3和5、4和7、5和8、6和9、6和10

问：2和7之间，5和9之间是否是连通的？

解决思路：

（1）将10台电脑看成10个集合{1}, {2}, {3}, . . . , {9}, {10}

（2）已知一种连接“x和y”，就将x和y对应的集合合并；

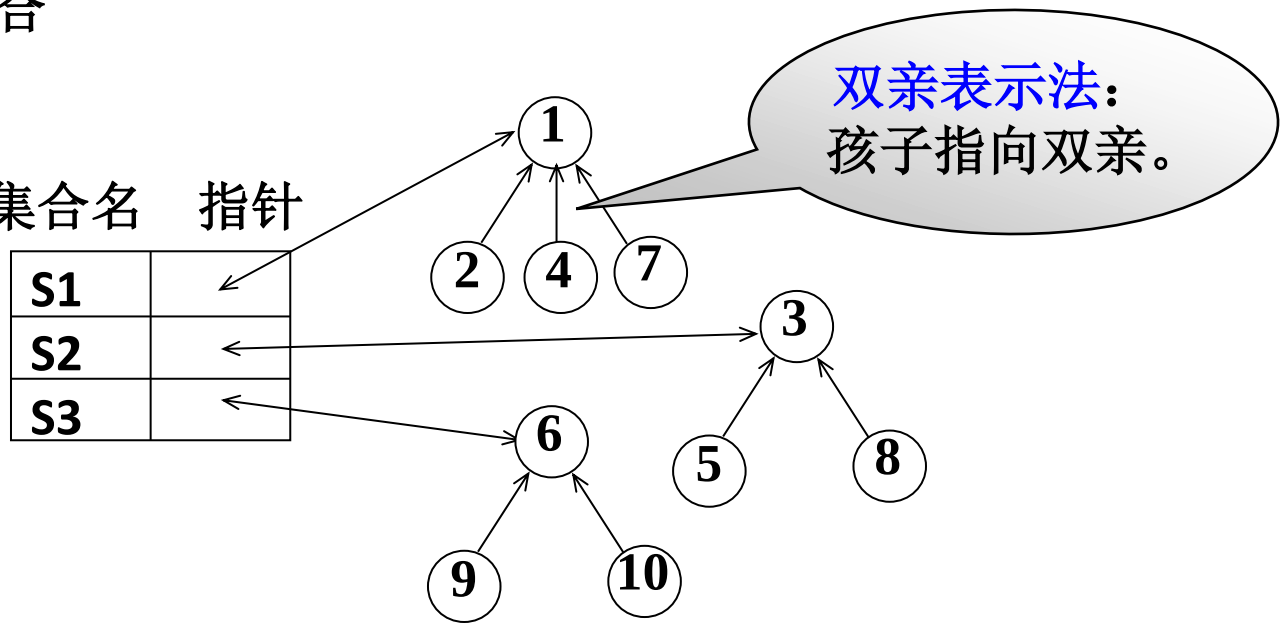
（3）查询“x和y是否是连通的”就是判别x和y是否属于同一集合。

集合的表示

- 集合运算：交、并、补、差，判定一个元素是否属于某一集合
- 并查集：集合并、查某元素属于什么集合
- 并查集问题中集合存储如何实现？　　用树根来代表这个集合
  - 可以用树结构表示集合，树的每个结点代表一个集合元素

例如，有三个整数集合

S1={1,2,4,7}
S2={3,5,8}
S3={6,9,10}



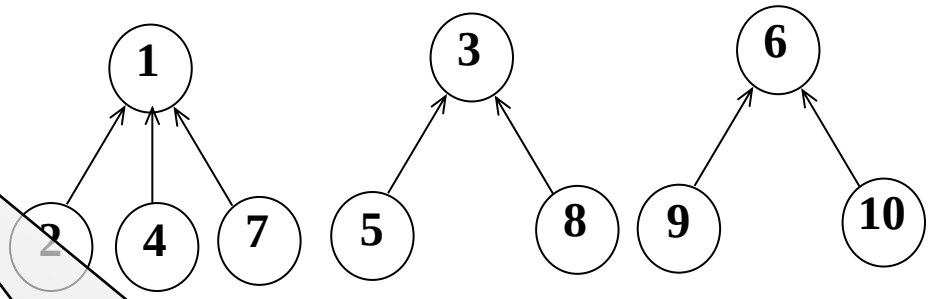
采用数组存储形式

下标	Data	Parent
0	1	-1
1	2	0
2	3	-1
3	4	0
4	5	2
5	6	-1
6	7	0
7	8	2
8	9	5
9	10	5

数组中每个元素的类型描述为：

```
typedef struct {
    ElementType Data;
    int Parent;
} SetType;
```

parent是一个下标,指向父亲的位置



负数表示根结点；
非负数表示双亲结点的下标。

集合运算

（1）查找某个元素所在的集合（用根结点表示）

```
int Find( SetType S[ ], ElementType X )
{   /* 在数组S中查找值为X的元素所属的集合 */
    /* MaxSize是全局变量，为数组S的最大长度 */
    int i;
    for ( i=0; i < MaxSize && S[i].Data != X; i++) ;
    if( i >= MaxSize ) return -1; /* 未找到X，返回-1 */
    for( ; S[i].Parent >= 0; i = S[i].Parent ) ;
    return i; /* 找到X所属集合，返回树根结点在数组S中的下标 */
}
```

（2） 集合的并运算

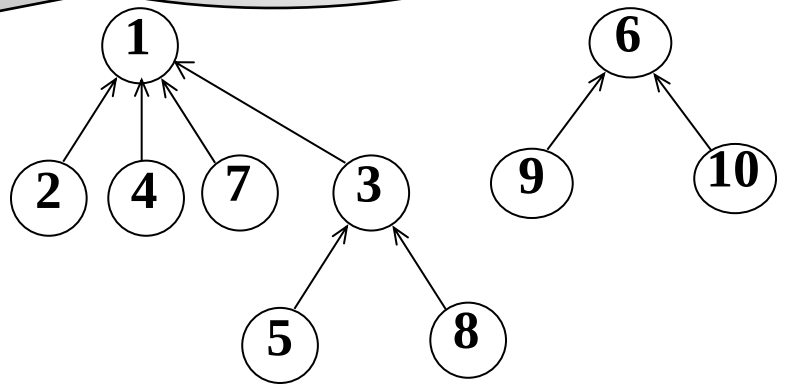
- 分别找到X1和X2两个元素所在集合树的根结点
- 如果它们不同根，则将其中一个根结点的父结点指针设置成另一个根结点的数组下标。

```
void Union( SetType S[ ], ElementType X1, ElementType X2 )
{
    int Root1, Root2;
    Root1 = Find(S, X1);
    Root2 = Find(S, X2);
    if ( Root1 != Root2 ) S[Root2].Parent = Root1;
}
```

当x1和x2不属于同一子集时，才需要合并

下标	data	Parent
0	1	-1
1	2	0
2	3	0
3	4	0
4	5	2
5	6	-1
6	7	0
7	8	2
8	9	5
9	10	5

改成-7。表示集合有7个元素



改成-3。表示集合有3个元素

为了改善合并以后的查找性能，可以采用小的集合合并到相对大的集合中。（修改Union函数, 判别根的绝对值大小）

这样树的高度就不会增加, 还是大的那颗树的高度.