5.3.1集合的表示及查找

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个集合
- 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？

可以用树结构表示集合，树的每个结点代表一个集合元素

双亲表示法：孩子指向双亲

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

Parent的值是该结点的父亲结点的下标

集合运算

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 */
}
```

集合的并运算

- 分别找到　和　两个元素所在集合树的根结点
- 如果他们不同根，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;
}
```

为了改善合并以后的查找性能，可以采用小的集合并到相对大的集合中（修改Union函数）

可以把根结点的Parent设置为负数的绝对值代表树中的元素个数

> 如何让并查树更矮？

> 路径压缩

```c
#define MAXN 1000                  /* 集合最大元素个数 */
typedef int ElementType;           /* 默认元素可以用非负整数表示 */
typedef int SetName;               /* 默认用根结点的下标作为集合名称 */
typedef ElementType SetType[MAXN]; /* 假设集合元素下标从0开始 */

void Union( SetType S, SetName Root1, SetName Root2 )
{ /* 这里默认Root1和Root2是不同集合的根结点 */
    /* 保证小集合并入大集合 */
    if ( S[Root2] < S[Root1] ) { /* 如果集合2比较大 */
        S[Root2] += S[Root1];     /* 集合1并入集合2  */
        S[Root1] = Root2;
    }
    else {                        /* 如果集合1比较大 */
        S[Root1] += S[Root2];     /* 集合2并入集合1  */
        S[Root2] = Root1;
    }
}
```