02-D1-4 唯一化高效版本

#数据结构邓神

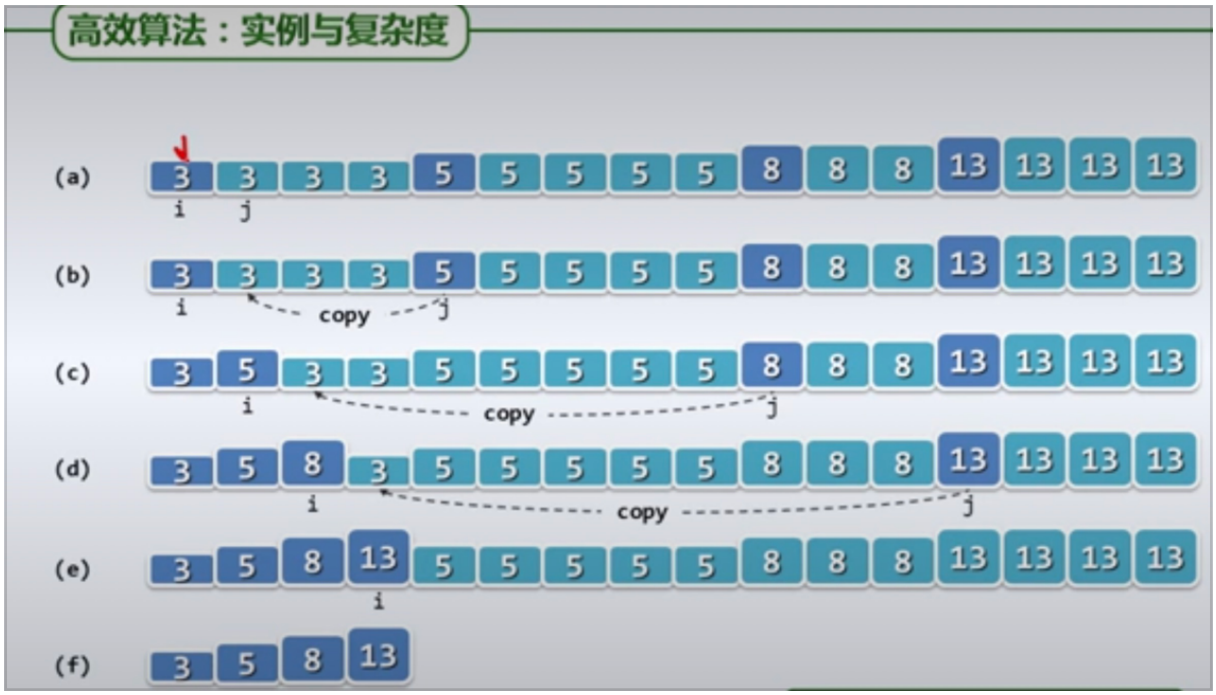
高效算法

反思：造成低效率的根源：同一个元素可能被作为被删除元素的后继被多次向前移动

如果能以重复区间为单位，成批量的删除雷同元素，性能必将改进

```
template <typename T> int Vector<T>::uniquify(){
    Rank i = 0;
    Rank j = 0;
    while(++j < _size) {
        // 跳过雷同者，找到一个不雷同的
        if (_elem[i] != _elem[j]) {
            _elem[++i] = _elem[j];
        }
    }
    _size = ++i; // 直接截断多余的类同元素
    shrink();
    return j -i;
} // High Efficient
// 注意这边很智慧的将不重复的元素直接覆盖前面重复的元素，大大提高了效率
// 如果只是用 remove删除区间并不能提高效率
```

高效算法：实例和复杂度



这里很好的揭示了这个算法的高效所在

只是将不重复的元素不断前移，最后直接缩减数组位置，很精妙的算法

复杂度

时间复杂度大大降低，而且空间复杂度还没有任何上升

空间复杂度:O(n)

空间复杂度:O(1) // 原地工作！！！