02-C-7 唯一化

#数据结构邓神

唯一话应用：

1. 网络搜索的局部结果经过去重复操作，汇总为最终报告

...

代码实现

```
template <typename T>
int Vector<T>::deduplicate() { // 繁琐版本 + 错误版本
    int oldSize = _size; // 记录初始规模
    Rank i = 1; // 从 1 开始 第0个一定不会重复
    while (i < _size) { // 循环所有元素
        (find(_elem[i],0,i) < 0) ? i++ : remove(i); // 如果找到了就删除这个，没找到就往后
    }
    return oldSize - _size; // 返回删除的元素的个数
}
```

我们如何给出这个算法正确性的严格证明?

1. 不变性：在当前元素V[i]的前缀V[0,i)中各个元素彼此互异

初始 i = 1时自然成立 // 递归基

其余的一般情况：

可以用数学证明即

如果 前i个元素唯一，如果新元素e，与前面的元素都不想等，那么归入后仍然唯一

如果有相同，就删除这个元素

所以只要第K次迭代满足唯一，那么K+1次就必然唯一。

所以由数学归纳法可以得出算法正确!

2. 单调性（保证必然会结束）

随着反复的while迭代

当前元素前缀的长度单调非降，且迟早增加到_size

当前元素后缀的长度单调下降，且迟早减少到0

故算法必然结束，且至多迭代O(n)轮

唯一化算法的复杂度

每轮迭代中的find()和remove累计消耗线性时间O(n^2)，故总体时间为O(n^2)

可以进一步优化比如：

1.仿照uniquify()高效版本的思路，元素移动次数可以下降到O(n),但是元素的比较次数为O(n^2)，而且稳定性将会被破坏。

2.先对需要删除的元素做标记，然后统一删除，稳定性保持，但是需要对比更多次

3. V.sort().uniquify() : 简单明了的实现O(nlogn) // 下节