02-D-01 有序向量-有序性 & 02-D-02 唯一化低效版本 & 02-D-03 复杂度（低效版本）

#数据结构邓神

有序性和甄别

与起泡排序的算法理解类似

有序/无序序列中，任意/总有一对相邻元素顺序/逆序

相邻逆序对的总数可以判断向量的逆序程度

```
template <template T> int Vector<T>::disordered() const {
    int n = 0; // 计数器
    for (int i = 0; i < _size; ++i) { // 循环所有元素
        n += (_elem[i-1] > _elem[i]); //逆序对级数
    }
    return n;
}
```

无序向量经过预处理后转换为有序向量后，相关算法多可优化

唯一化 低效版本

观察在有序向量中，重复的元素必然相互紧邻构成一个区间

因此每个区间只要要保留一个元素就可以

那么我们是否可以循环所有元素，当找到一个区间，就留下一个一个元素

```
template <typename T> int Vector<T>::uniquify(){
    int oldSize= _size;
    int i = 0;
    while (i < _size -1){
        // 如果相同就删除后面的元素，否则就往下走
```

```
        (_elem[i] == _elem[i+1]) ? remove(i+1) : i++;
    }
    return oldSize - _size;
}// 其中 _size 变化由 remove函数隐式的完成
```

复杂度

运行时间主要取决于while循环，次数总计 _size -1 = n -1

最坏的情况下：每次都需要调用remove(),耗费时间O(1)~O(n)

累计 O(n^2)，总体复杂度竟然与无序向量的deduplocate()相同

这是完全无法接受的，因为本身排序就是需要时间的