02-C-5 查找 & 02-C-6 单个元素的删除

#数据结构邓神

无序向量 ： T为可以判等的基本类型，或者已经重载了操作符号"==" 或者 "!="

有序向量 ： T为可比较的基本类型，或者已经重载了操作符号">"或者"<"

例如稍后会介绍 Entry 词条类

无序向量查找

从查找范围 [lo,hi) 一个一个判断等于，就是查找

```
template <typename T> // O(hi - lo) = O(n) 在命中多个元素后可以返回秩的最大值 这样可以找到 秩序最大的
Rank Vector<T>::find(T const& e,Rank lo,Rank hi) const{
    while ((lo < hi--) && (e != elem[hi]));
    return hi; // hi < lo 意味着失败，否则 hi 即命中元素的秩序 ， 是否查找成功由上层判断
}
```

输入敏感 (Input-sensitive) 最好 O(1) 最差 O(1)

删除单个元素

```
// 可以看为区间操作的特例
// 删除一个元素 [r] = [r,r+1)
T Vector<T>::remove(Rank r){ // O(n-r)
  T e = elem[r]; // 备份被删除的元素
  remove(r,r+1); // 调用区间删除删除一个元素
  return e;        // 返回被删除元素的值
}
```

为什么不通过反复调用 remove(r) 接口来实现 remove(lo,hi)

效率低下，每一次删除一个元素的复杂度都是 O(n-r) = O(n)

而循环的总体次数为 hi - lo = O(n)

所以整体的复杂度为 O(n^2)

这比原来的复杂度高了一个数量级