2.向量

（c3）无序向量

- 去重

世雷同而炫曜兮，何毁譽之昧昧！

邓俊辉

deng@tsinghua.edu.cn

算法

❖ 应用实例：网络搜索的局部结果经过去重操作，汇总为最终报告

❖
```
template <typename T> //删除重复元素，返回被删除元素数目
int Vector<T>::deduplicate() { //繁琐版 + 错误版
   int oldSize = _size; //记录原规模
   Rank i = 1; //从_elem[1]开始
   while ( i < _size ) //自前向后逐一考查各元素_elem[i]
      find( _elem[i], 0, i ) < 0 ? //在前缀中寻找雷同者
         i++ //若无雷同则继续考查其后继
       : remove(i); //否则删除雷同者（至多一个？！）
   return oldSize - _size; //向量规模变化量，即删除元素总数
}
```

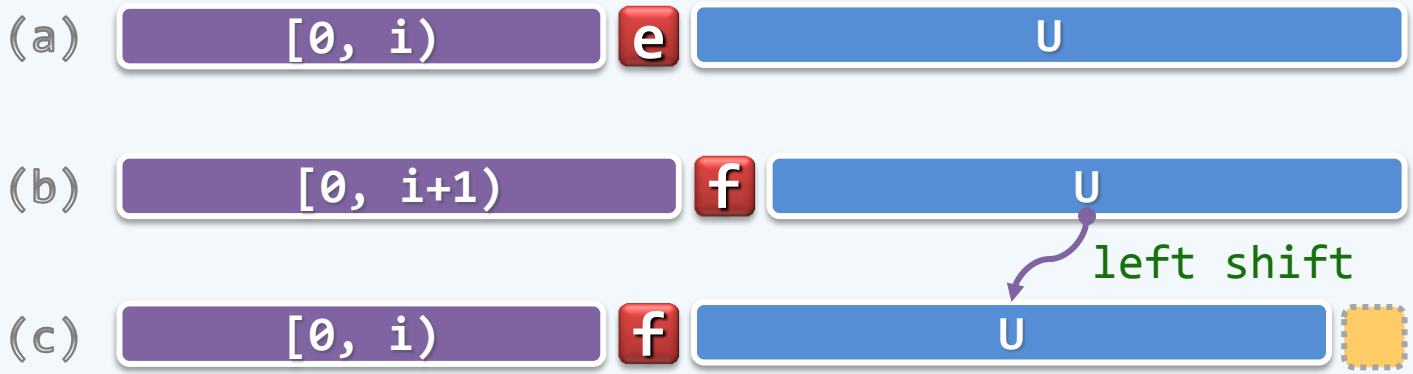
正确性

❖ 易见，凡被剔除者均为重复元素（不多）

故只需证明，算法不致遗漏重复元素（不少）

❖ 不变性：在当前元素V[i]的前缀V[0, i)中，各元素彼此互异

❖ 初始i = 1时自然成立；后续的一般情况...



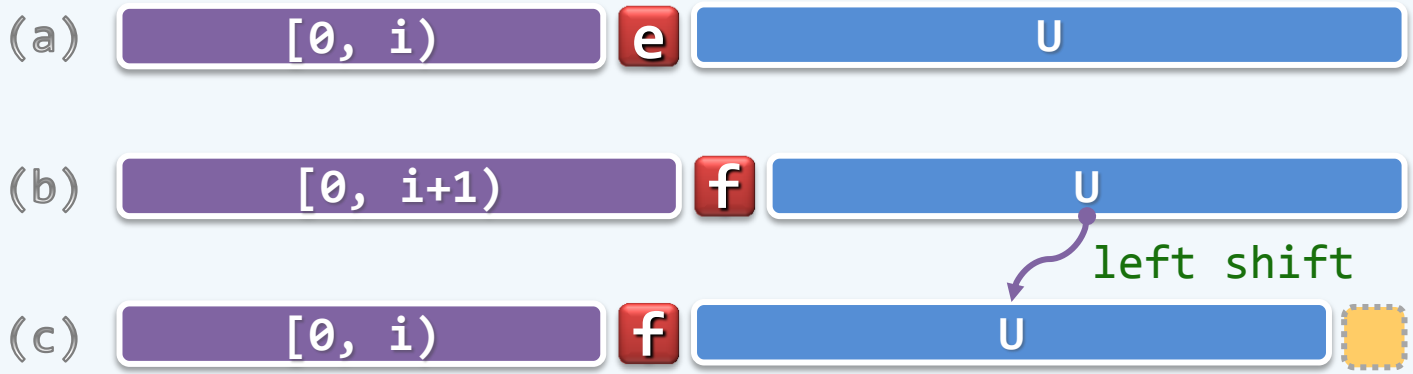
正确性

❖ **单调性**：随着反复的while迭代

1）当前元素**前缀**的长度单调非降，且迟早增至_size　//1)和2)对应

2）当前元素**后缀**的长度单调下降，且迟早减至0　//2)更易把握

故算法**必然终止**，且至多迭代$O(n)$轮



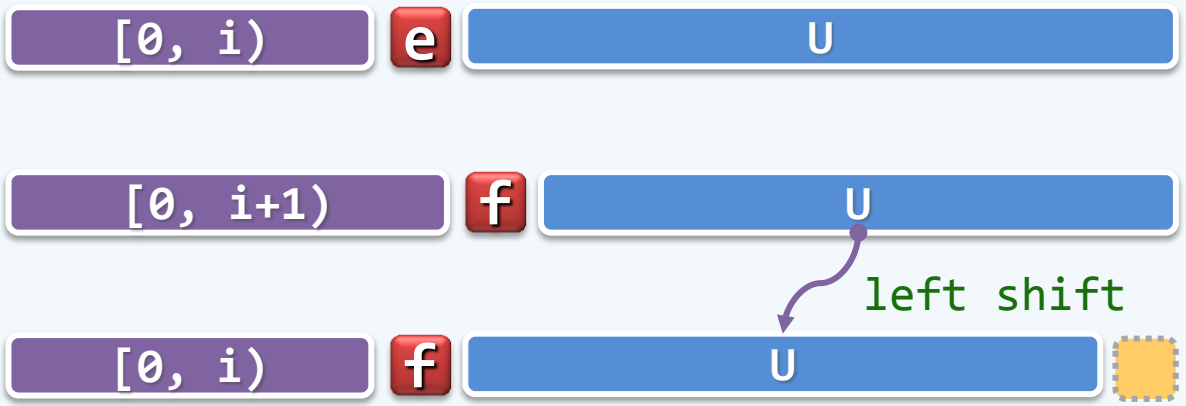
复杂度

❖ 每轮迭代中find()和remove()累计耗费线性时间，总体$O(n^2)$　//可进一步优化，比如...

- 仿照uniquify()高效版的思路

元素移动的次数可降至$O(n)$

但比较次数依然是$O(n^2)$



- 先对需删除的重复元素做标记，然后再统一删除

稳定性保持，但因查找长度更长，从而导致更多的比对操作

- V.sort().uniquify()：简明实现最优的$O(n\log n)$　//下节