

排序

选取：众数

善鈞，從眾。夫善，眾之主也。三卿為主，可謂眾矣。從之，不亦可乎？！

诚若为今立计，所当稽求既往，相度方来，掇物质而张灵明，任个人而排众数

然而，在现代文明社会里，居有其所的家庭却不到一半；在文明特别发达的大城市里，拥有住房的人只占全体居民的极小部分

邓俊辉

deng@tsinghua.edu.cn

选取 + 中位数

❖ k-selection 在任意一组可比较大小的元素中，如何由小到大，找到次序为 k 者？

亦即，在这组元素的非降排序序列 S 中，找出 $S[k]$

// Excel: large(range, rank)

❖ median 长度为 n 的有序序列 S 中，元素 $S[\lfloor n/2 \rfloor]$ 称作中位数 // 与之相等者也是

在任意一组可比较大小的元素中，如何找到中位数？ // Excel: median(range)



❖ **中位数**是k-选取的一个特例；稍后将看到，也是其中难度最大者

Majority

❖ 无序向量中，若有一半以上元素同为 m ，则称之为众数

- 在 $\{ \boxed{3}, 5, 2, \boxed{3}, \boxed{3} \}$ 中，众数为3；然而
- 在 $\{ \boxed{3}, 5, 2, \boxed{3}, \boxed{3}, 0 \}$ 中，却无众数

❖ 平凡算法 = 排序 + 扫描

但进一步地：若限制时间不超过 $O(n)$ ，附加空间不超过 $O(1)$ 呢？

❖ 必要性：众数若存在，则亦必是中位数

❖ 于是，只要能够找出中位数，即不难验证它是否众数

```
template <typename T> bool majority( Vector<T> A, T & maj )  
{ return majCheck( A, maj = median( A ) ); }
```

必要条件

❖ 然而，在高效的中位数**算法未知**之前，如何确定众数的候选呢？

❖ mode：众数若存在，则亦必**频繁数** //Excel: mode(range)

```
template <typename T> bool majority( Vector<T> A, T & maj )  
  
    { return majCheck( A, maj = mode( A ) ); }
```

❖ 同样地，mode()算法难以**兼顾**时间、空间的高效

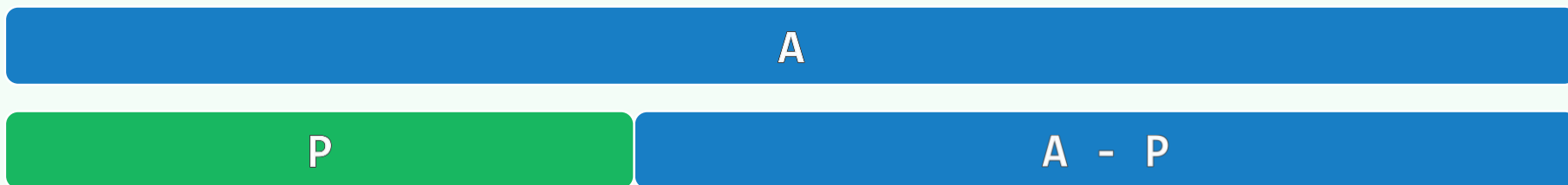
❖ 可行思路：借助**功能更弱**但计算**成本更低**的必要条件，选出唯一的候选者

```
template <typename T> bool majority( Vector<T> A, T & maj )  
  
    { return majCheck( A, maj = majCandidate( A ) ); }
```

减而治之

❖ 若在向量A的前缀P ($|P|$ 为偶数) 中, 元素 x 出现的次数恰占半数, 则

A有众数, 仅当对应的后缀 $A - P$ 有众数 m , 且 m 就是A的众数



❖ 既然最终总花费 $O(n)$ 时间做验证, 故而只需考虑A的确含有众数的两种情况:

1. 若 $x = m$, 则在排除前缀P之后, m 与其它元素在数量上的差距保持不变
2. 若 $x \neq m$, 则在排除前缀P之后, m 与其它元素在数量上的差距不致缩小

❖ 若将众数的标准从“一半以上”改作“至少一半”, 算法需做什么调整?

算法

```
template <typename T> T majCandidate( Vector<T> A ) {
```

```
    T maj;
```

```
    for ( Rank c = 0, i = 0; i < A.size(); i++ )
```

```
        if ( 0 == c ) {
```

```
            maj = A[i]; c = 1;
```



```
        } else
```

```
            maj == A[i] ? c++ : c--;
```



```
    return maj;
```

```
}
```