12-B1

12.排序

选取

众数

邓俊辉

deng@tsinghua.edu.cn

选取与中位数

❖ 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？

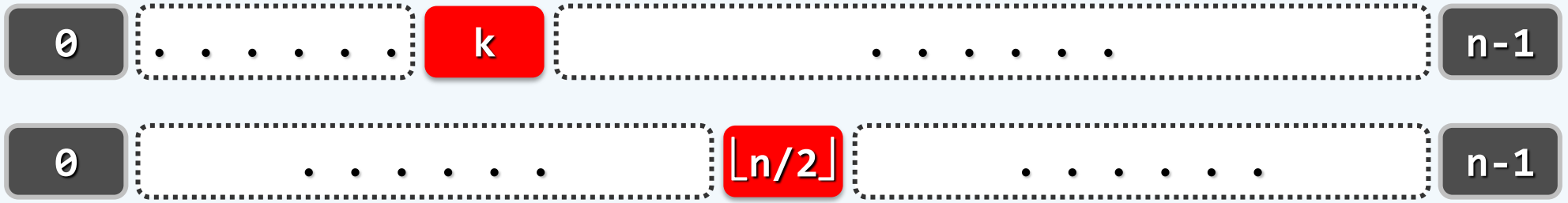
亦即，在这组元素的非降排序序列S 中，找出$S[k]$

// 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 //数值上可能有重复

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

众数

- majority　无序向量中，若有一半以上元素同为m，则称之为众数

  在{ 3, 5, 2, 3, 3 }中，众数为3；然而

  在{ 3, 5, 2, 3, 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，则亦必中位数

- 事实上　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 majEleCheck( 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 majEleCheck( A, maj = mode( A ) ); }
```

❖ 同样地　mode()算法难以兼顾时间、空间的高效

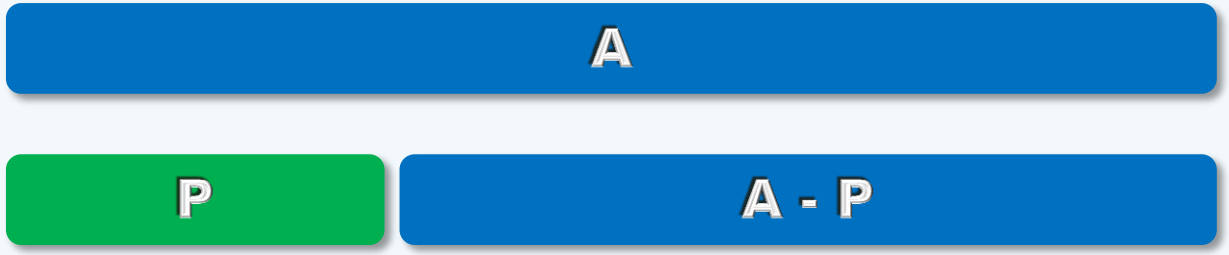
❖ 可行思路　借助更弱但计算成本更低的必要条件，选出唯一的候选者

```
template <typename T> bool majority( Vector<T> A, T & maj )
   { return majEleCheck( A, maj = majEleCandidate( 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



❖ 既然最终总要花费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

（从浓度50⁺%的盐水中渗析出50%的一部分，剩余部分的浓度仍为50⁺%）

2．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不致缩小

算法

```
❖ template <typename T> T majEleCandidate( Vector<T> A ) {
    T maj; //众数候选者
  // 线性扫描：借助计数器c，记录maj与其它元素的数量差额
    for ( int c = 0, i = 0; i < A.size(); i++ )
       if ( 0 == c ) { //每当c归零，都意味着此时的前缀P可以剪除
          maj = A[i]; c = 1; //众数候选者改为新的当前元素
       } else //否则
          maj == A[i] ? c++ : c--; //相应地更新差额计数器
    return maj; //至此，原向量的众数若存在，则只能是maj —— 尽管反之不然
  }
```

❖ 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？