14-B3

排序

选取：QuickSelect

And it's ride, willie, ride,
Roll, Willie, roll.
Wherever you are a-gamblin' now,
Nobody really knows.

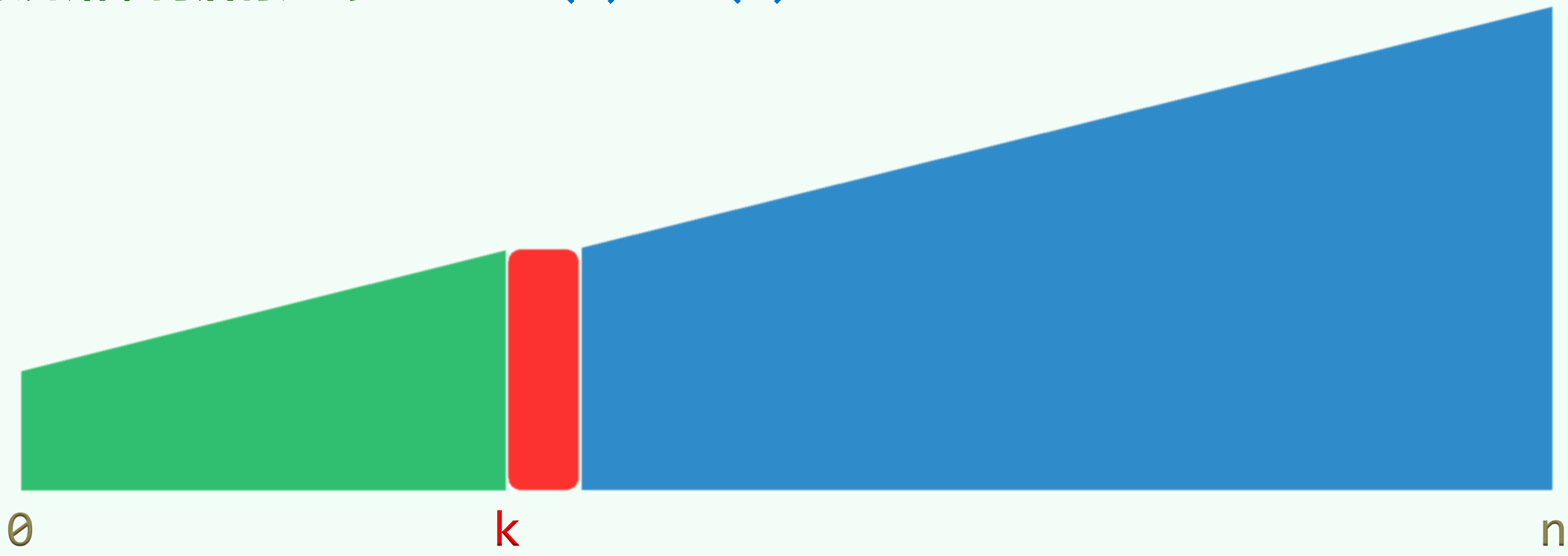
大胆猜测，小心求证

邓俊辉
deng@tsinghua.edu.cn

尝试：蛮力

❖ 对A**排序**　　//$\mathcal{O}$(nlogn)

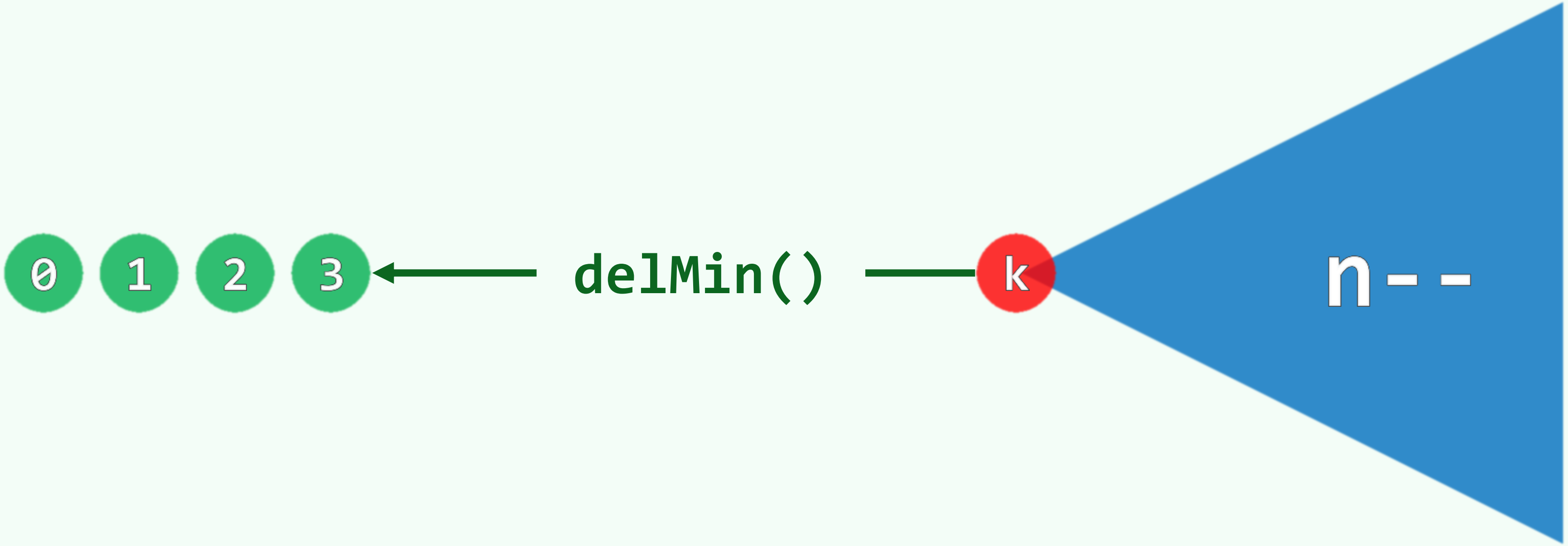
　从首元素开始，向后行进**k**步　　//$\mathcal{O}$(k) = $\mathcal{O}$(n)



尝试：堆（A）

❖ 将所有元素组织为小顶堆　　//$\mathcal{O}(n)$

连续调用k+1次delMin()　　//$\mathcal{O}(k\log n)$



尝试：堆（B）

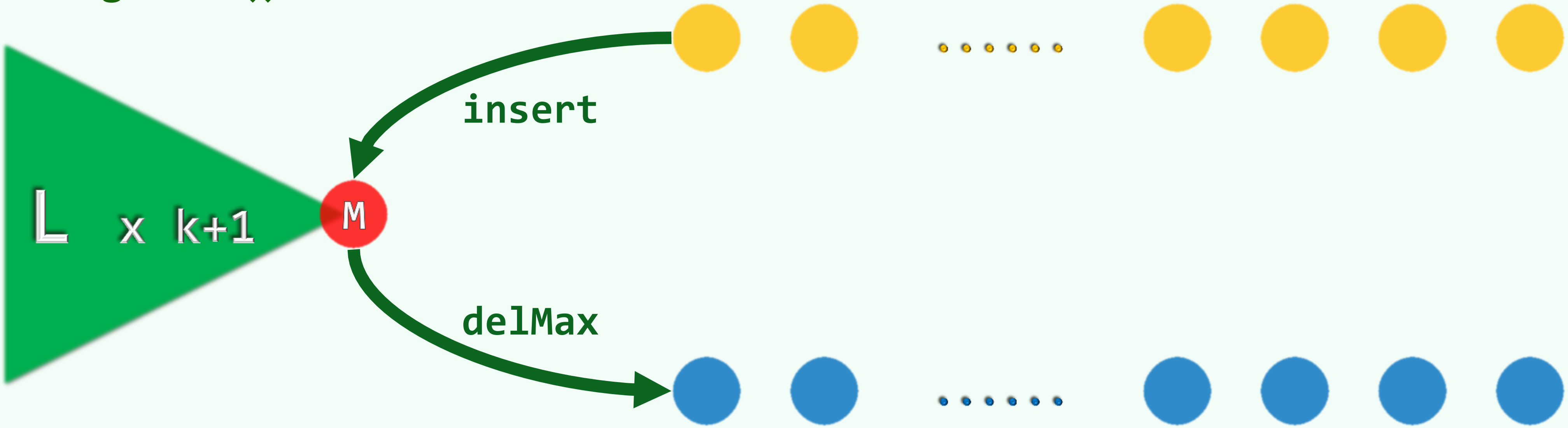
❖ L = heapify(A[0, k]) //任选 k+1 个元素，组织为**大顶堆**：$\mathcal{O}(k)$

❖ for each i in (k, n) //$\mathcal{O}(n - k)$

    L.insert(A[i]) //$\mathcal{O}(\log k)$

    L.delMax() //$\mathcal{O}(\log k)$

  return L.getMax()



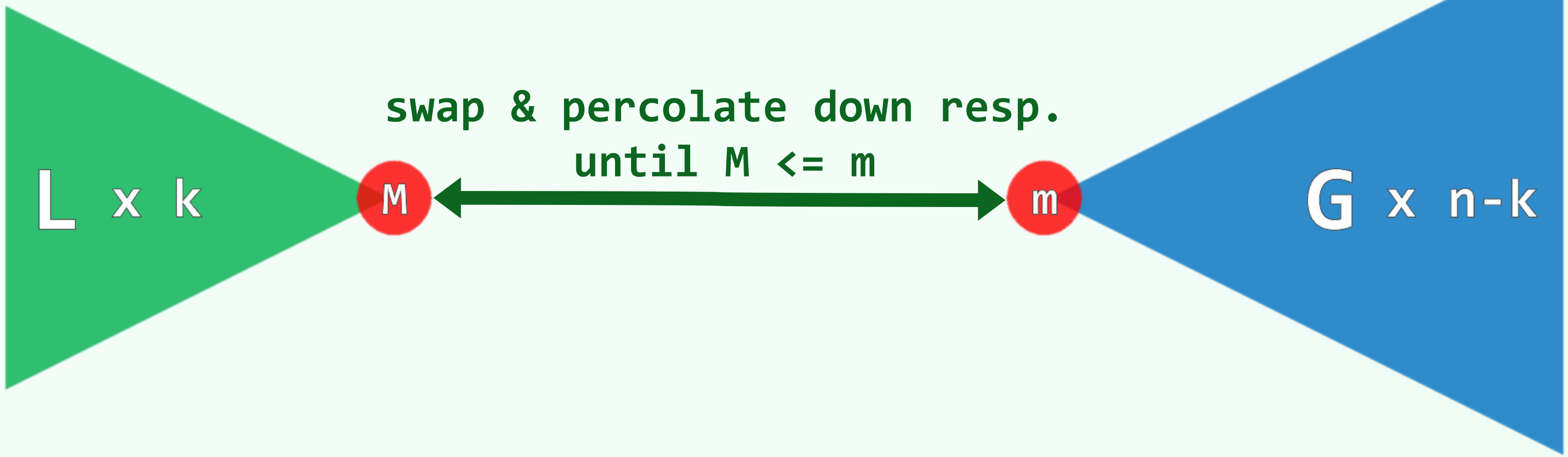
尝试：堆（C）

❖ 将输入任意划分为规模为k、n-k的子集

分别组织为大、小顶堆

//$\mathcal{O}$(k + (n-k)) = $\mathcal{O}$(n)

```
while ( m < M ) //O(min(k, n - k))
    swap( m, M )
    L.percolateDown() //O(logk)
    G.percolateDown() //O(log(n - k))
return m = G.getMin()
```



下界与最优

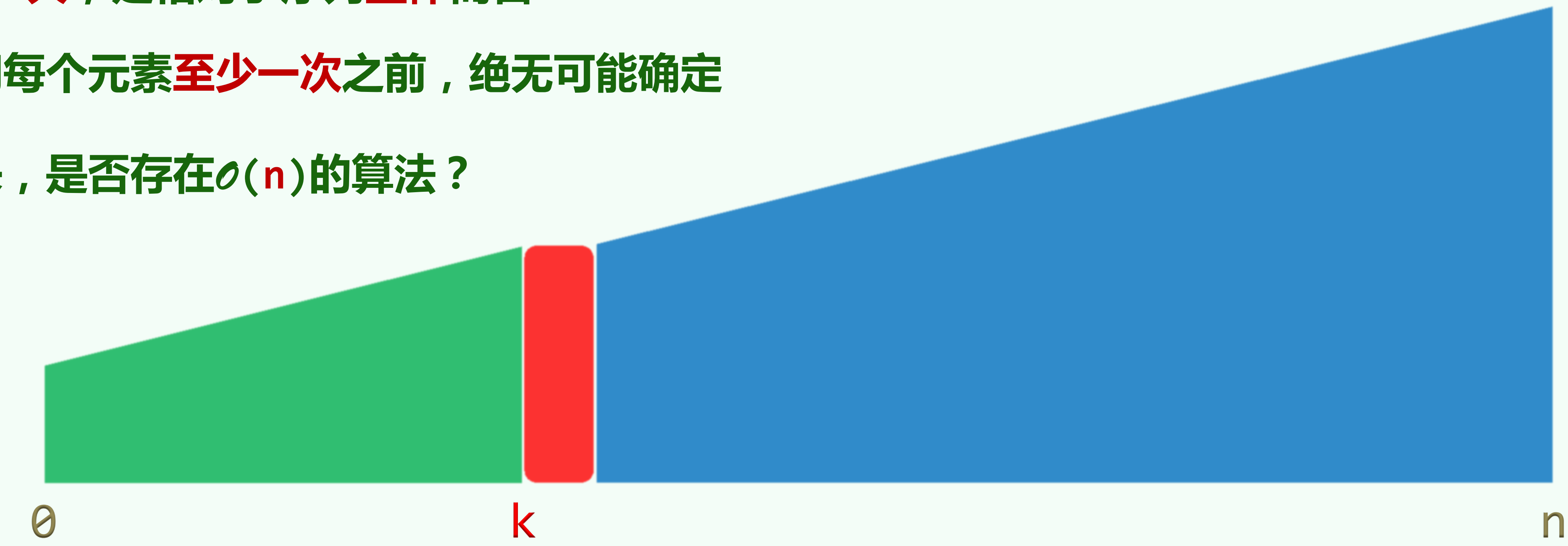
❖ 是否存在更快的算法？

❖ $\Omega(n)$！

❖ 所谓第k大，是相对于序列整体而言

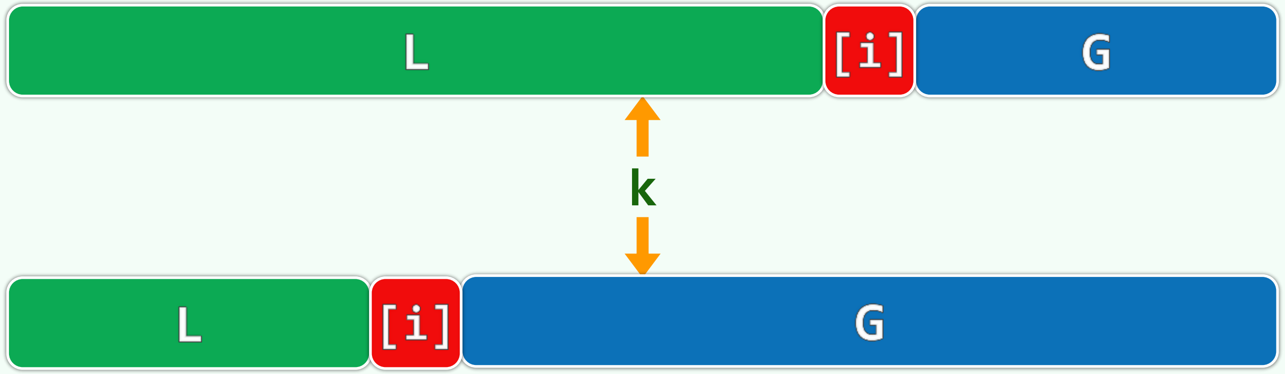
在访问每个元素至少一次之前，绝无可能确定

❖ 反过来，是否存在$\mathcal{O}(n)$的算法？



quickSelect()

```
template <typename T> void quickSelect( Vector<T> & A, Rank k ) {
   for ( Rank lo = 0, hi = A.size() - 1; lo < hi;  ) {
      Rank i = lo, j = hi; T pivot = A[lo]; //大胆猜测
      while ( i < j ) { //小心求证：O(hi - lo + 1) = O(n)
         while ( i < j  &&  pivot <= A[j] ) j--;  A[i] = A[j];
         while ( i < j  &&  A[i] <= pivot ) i++;  A[j] = A[i];
      } //assert: quit with i == j
      A[i] = pivot;
      if ( k <= i ) hi = i - 1;
      if ( i <= k ) lo = i + 1;
   } //A[k] is now a pivot
}
```



期望性能

❖ 记期望的比较次数为 $T(n)$

$$T(1) = 0,\ T(2) = 1,\ \ldots$$

❖ 可以证明：$T(n) = \mathcal{O}(n)$ ...

$$T(n) \ = \ (n-1) \ + \ \frac{1}{n} \times \sum_{k=0}^{n-1} \max\{T(k), T(n-k-1)\} \ \leq \ (n-1) \ + \ \frac{2}{n} \times \sum_{k=n/2}^{n-1} T(k)$$

❖ 事实上，不难验证：$T(n) \ < \ 4 \cdot n$ ...

$$T(n) \ \leq \ (n-1) \ + \ \frac{2}{n} \times \sum_{k=n/2}^{n-1} 4k \ \leq \ (n-1) + 3n \ < \ 4n$$