14-B3

排序

选取：QuickSelect

大胆猜测，小心求证

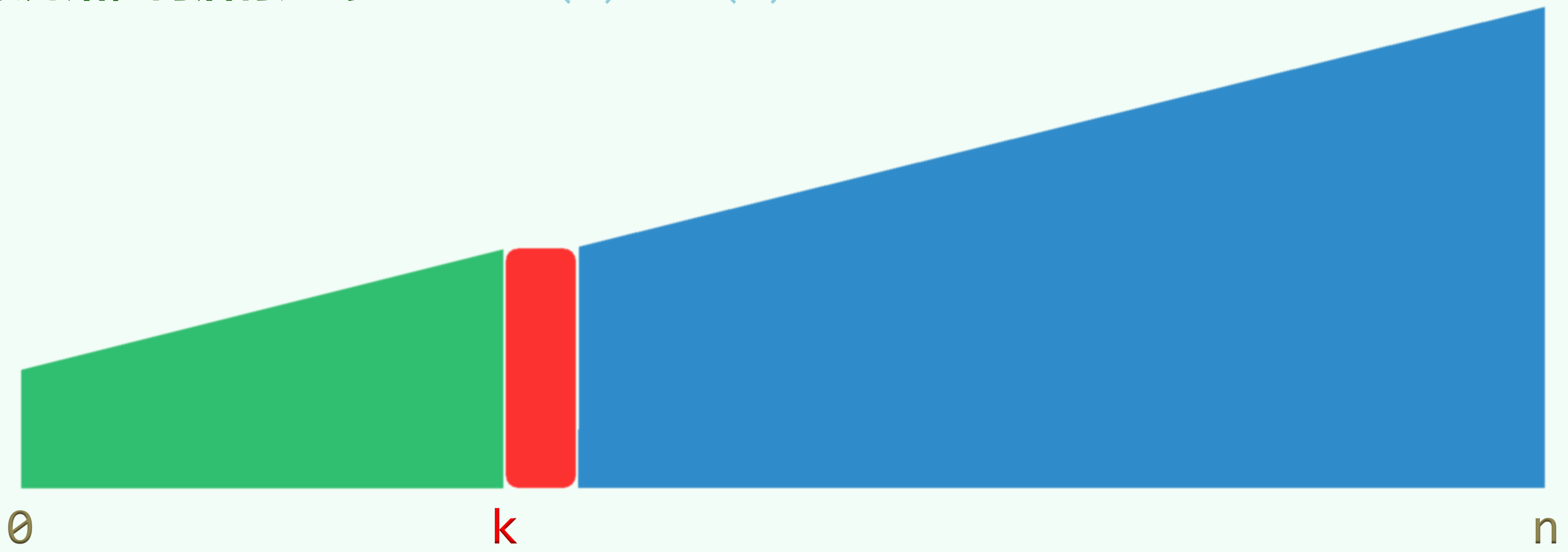
他们在一起谈了一下之后，就转过身来向我表示敬意，对此，我的老师微微一笑；此外，他们还给了我更多的荣誉，因为他们把我列入他们的行列，结果，我就是这样赫赫有名的智者中的第六位。

邓俊辉
deng@tsinghua.edu.cn

尝试：蛮力

- 对A排序　　//$\mathcal{O}$(nlogn)

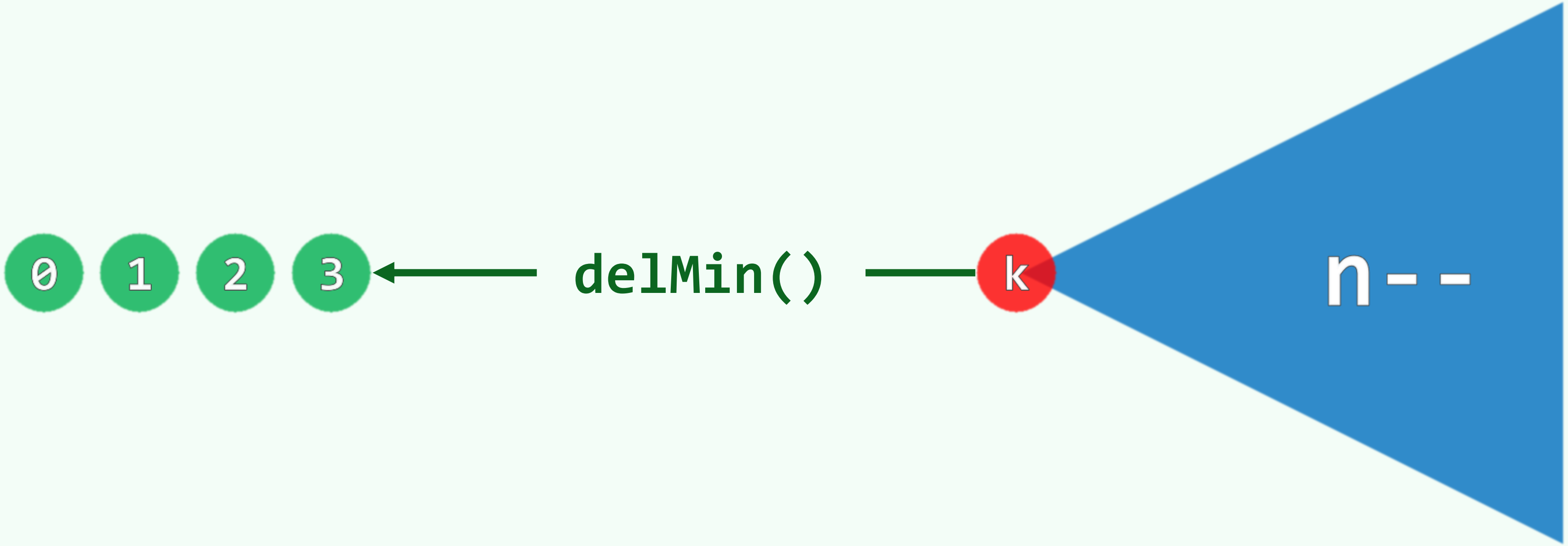
从首元素开始，向后行进k步　　//$\mathcal{O}$(k) = $\mathcal{O}$(n)



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）

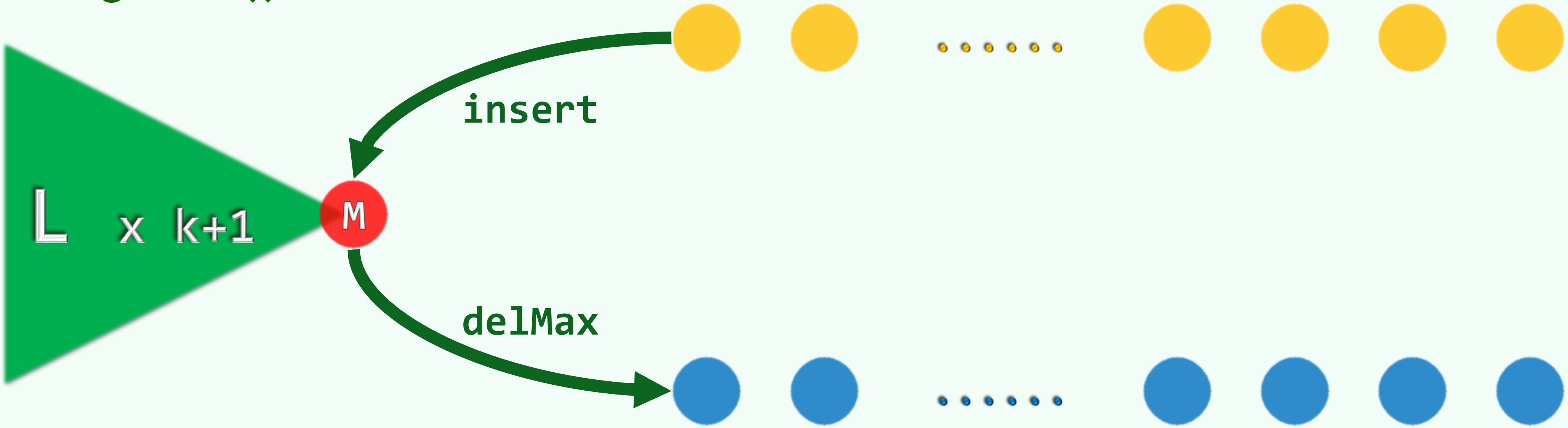
❖ `L = heapify( A[0, k] )` //任选 k+1 个元素，组织为大顶堆：$\mathcal{O}(k)$

❖ `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()
```



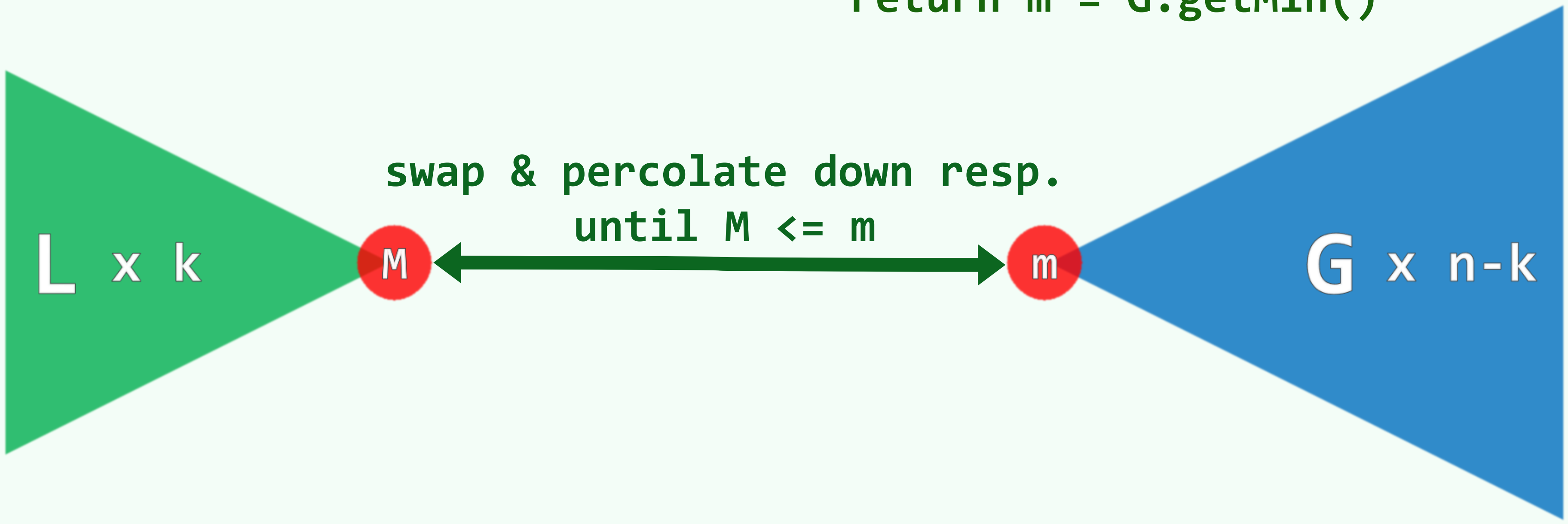
尝试：堆（C）

❖ 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()
```



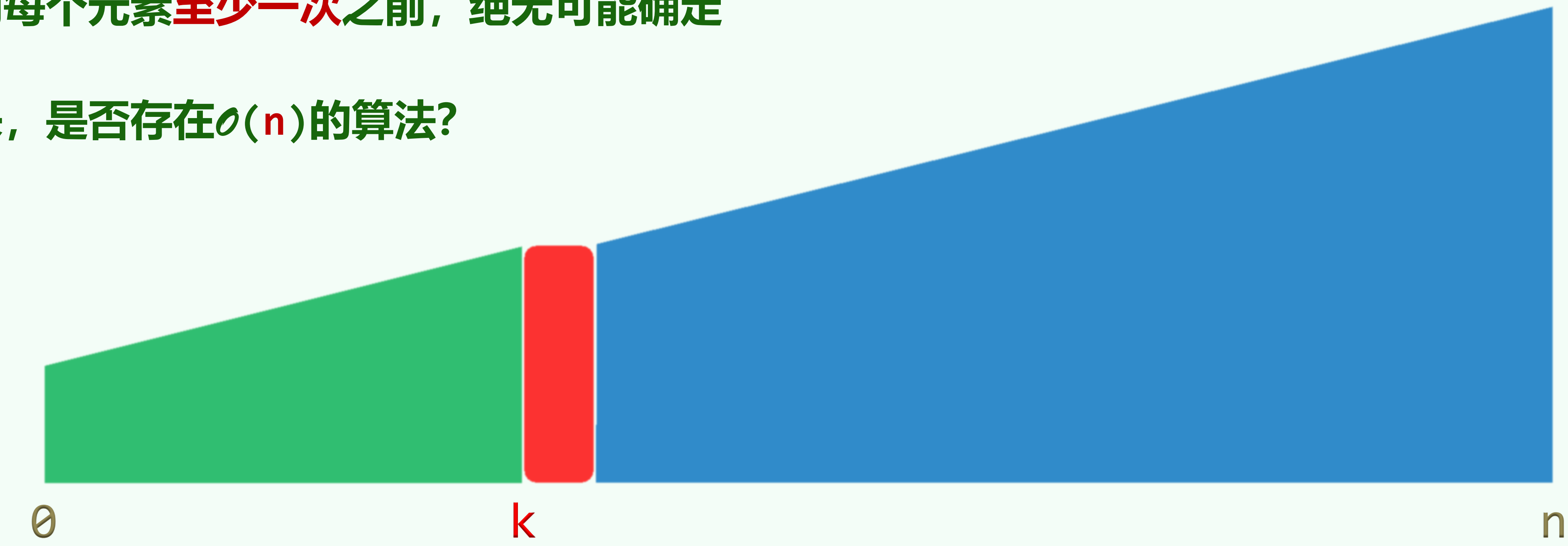
下界与最优

❖ 是否存在更快的算法？当然，最快也不至于快过$\Omega(n)$！

❖ 所谓第k小，是相对于序列整体而言，所以...

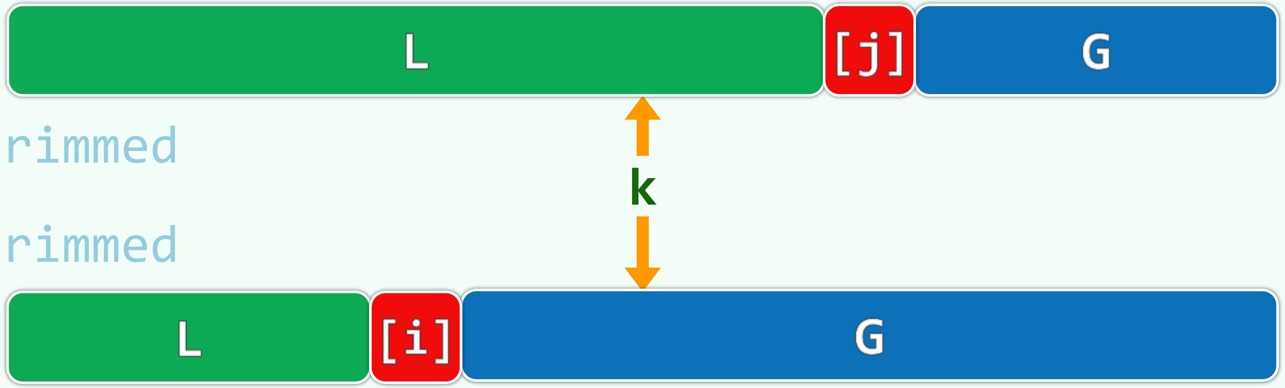
在访问每个元素至少一次之前，绝无可能确定

❖ 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(); 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)
         do j--; while ( (i < j) && (pivot <= A[j]) );   if (i < j) A[i] = A[j];
         do i++; while ( (i < j) && (A[i] <= pivot) );   if (i < j) A[j] = A[i];
      } //assert: quit with i == j or j+1
      A[j] = pivot;
      if ( k <= j ) hi = j; //suffix trimmed
      if ( i <= k ) lo = i; //prefix trimmed
   } //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：$T(n) = \mathcal{O}(n)$ ...

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

❖ 事实上，不难验证：$T(n) < 4 \cdot n$ ...

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