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

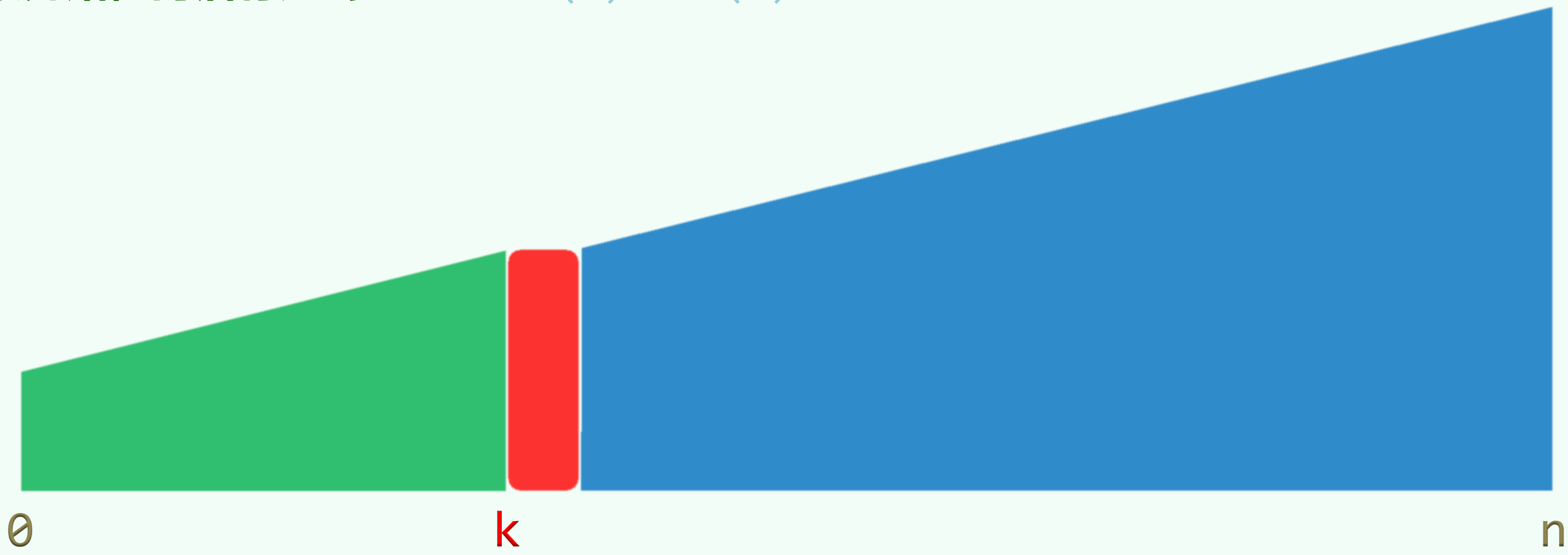
邓俊辉

deng@tsinghua.edu.cn

尝试：蛮力

❖ 对A**排序** //$\mathcal{O}(n\log n)$

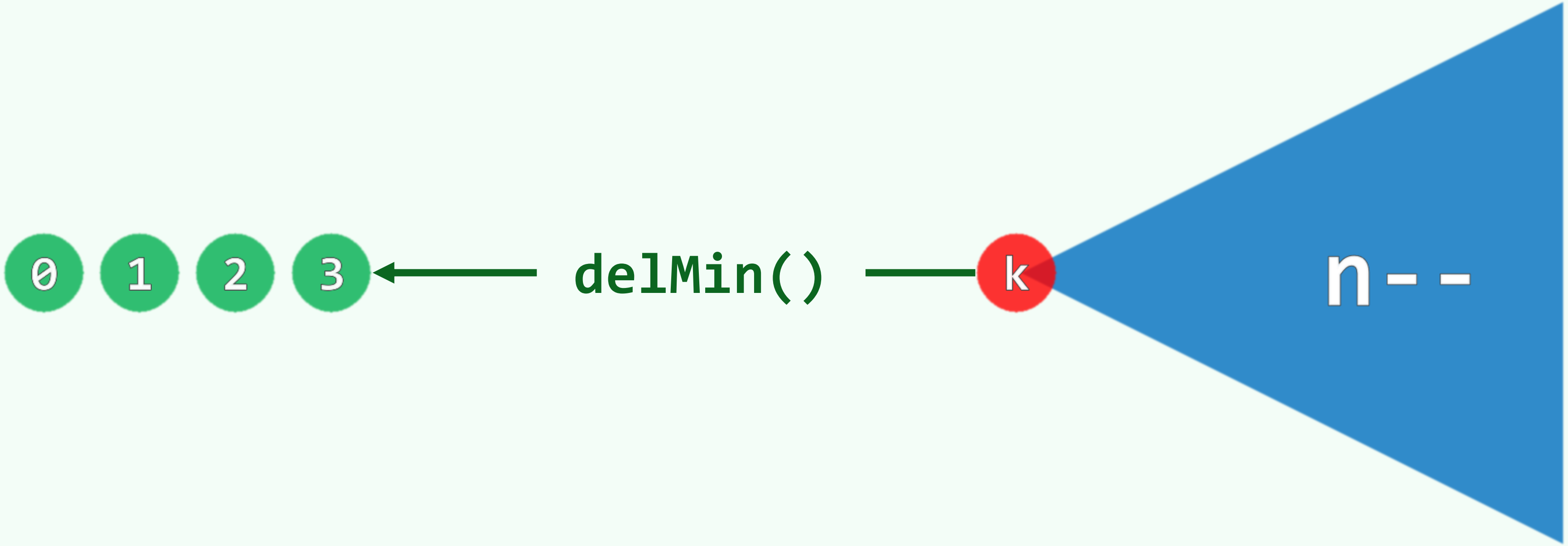
从首元素开始，向后行进k步 //$\mathcal{O}(k) = \mathcal{O}(n)$



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）

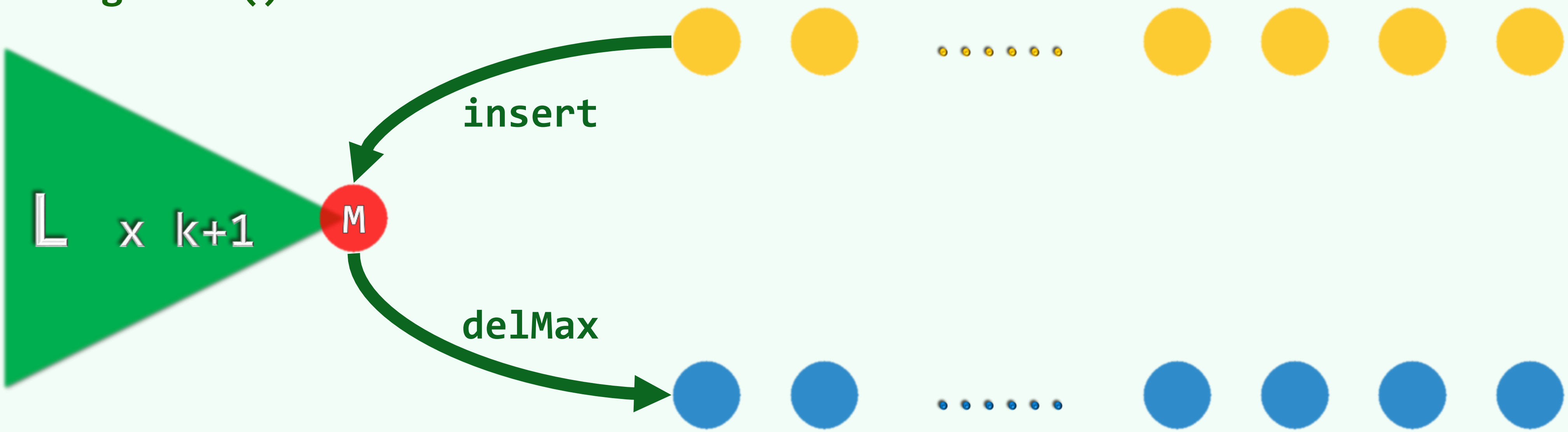
❖ L = heapify(A[0, k]) //任选 k+1 个元素，组织为大顶堆：𝒪(k)

❖ for each i in (k, n) //𝒪(n - k)

```
    L.insert( A[i] ) //O(logk)

    L.delDelMax() //O(logk)

return L.getMax()
```



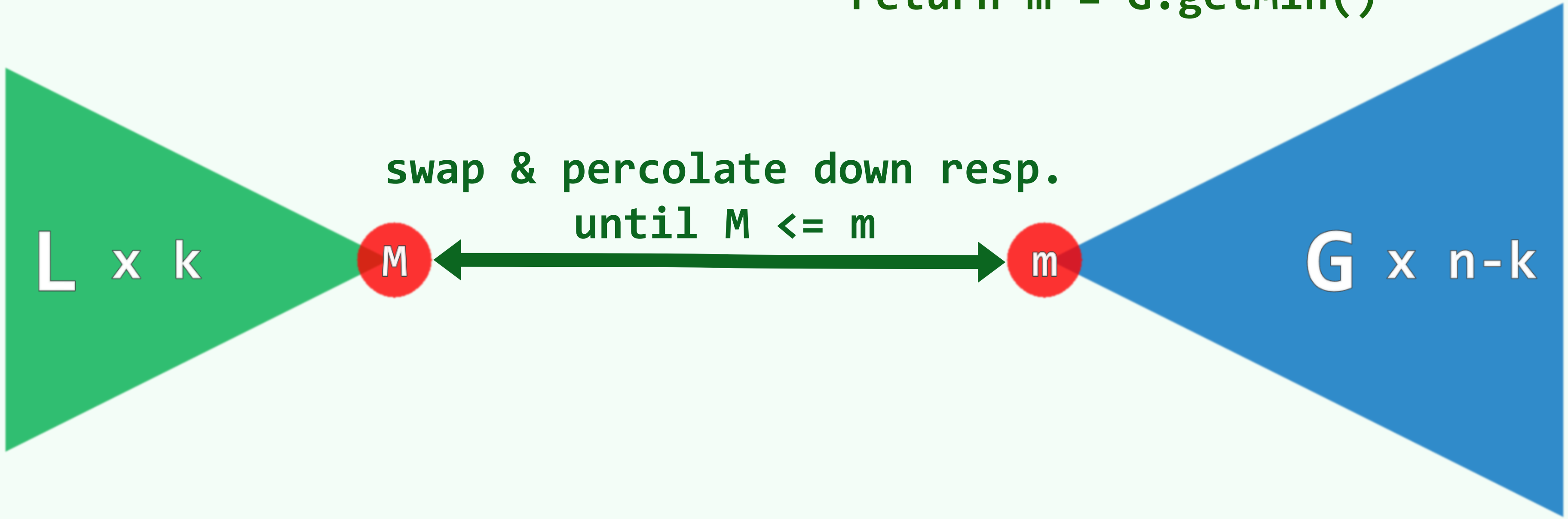
尝试：堆（C）

❖ 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

//𝒪(k + (n-k)) = 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()
```



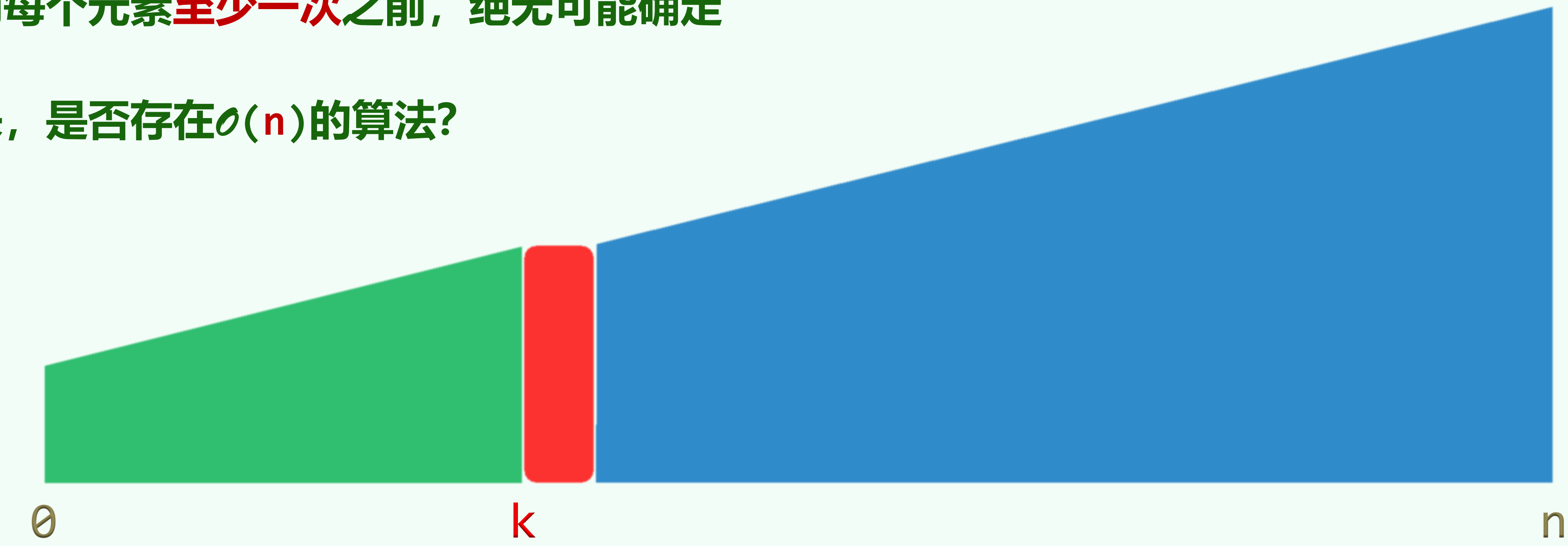
下界与最优

❖ 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...

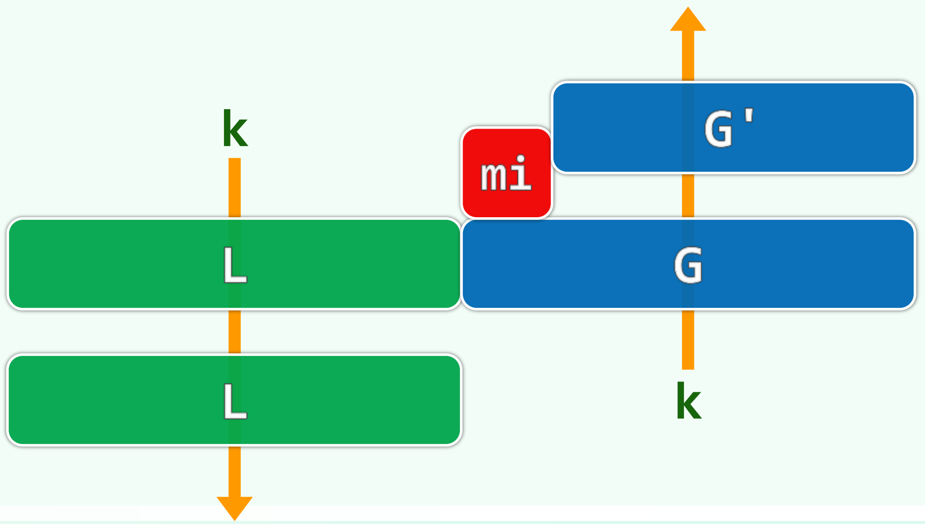
在访问每个元素**至少一次**之前，绝无可能确定

❖ 反过来，是否存在$\mathcal{O}(n)$的算法？



快速选取

```
template <typename T> Rank quickSelect( T const * A, Rank n, Rank k ) {
   Vector<Rank> R(n); for ( Rank k = 0; k < n; k++ ) R.insert(k); //使用索引向量，保持原次序
   for ( Rank lo = 0, hi = n; ;  ) { //反复做quickParititon
      swap( R[lo], R[lo + rand()%(hi-lo)] ); T pivot = A[R[lo]]; Rank mi = lo; //大胆猜测
      for ( Rank i = lo+1; i < hi; i++ ) //LGU版partition算法
         if ( A[R[i]] < pivot )
            swap( R[++mi], R[i] );
      swap( R[lo], R[mi] ); //[0,mi) < [mi] <= (mi, n)
      if ( mi < k ) lo = mi + 1; //猜小了，则剪除前缀
      else if ( k < mi ) hi = mi; //猜大了，则剪除后缀
      else return return R[mi]; //或早或迟，总能猜中
   }
}
```

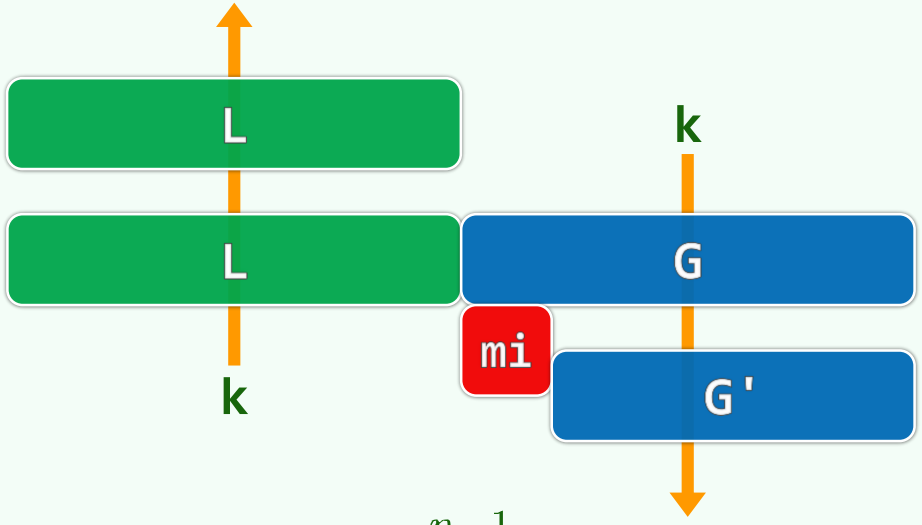


期望性能

❖ 记期望的比较次数为 $T(n)$ ，于是：

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

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



❖ 事实上，不难验证：$T(n) < 4 \cdot n = T(n) = \mathcal{O}(n)$ ...

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