12.排序

选取

QuickSelect

12-B3

世兄的才名，弟所素知的。在世兄是数万人里头选出来最清最雅的，至于弟乃庸庸碌碌一等愚人，忝附同名，殊觉玷辱了这两个字。

邓俊辉

deng@tsinghua.edu.cn

尝试：蛮力

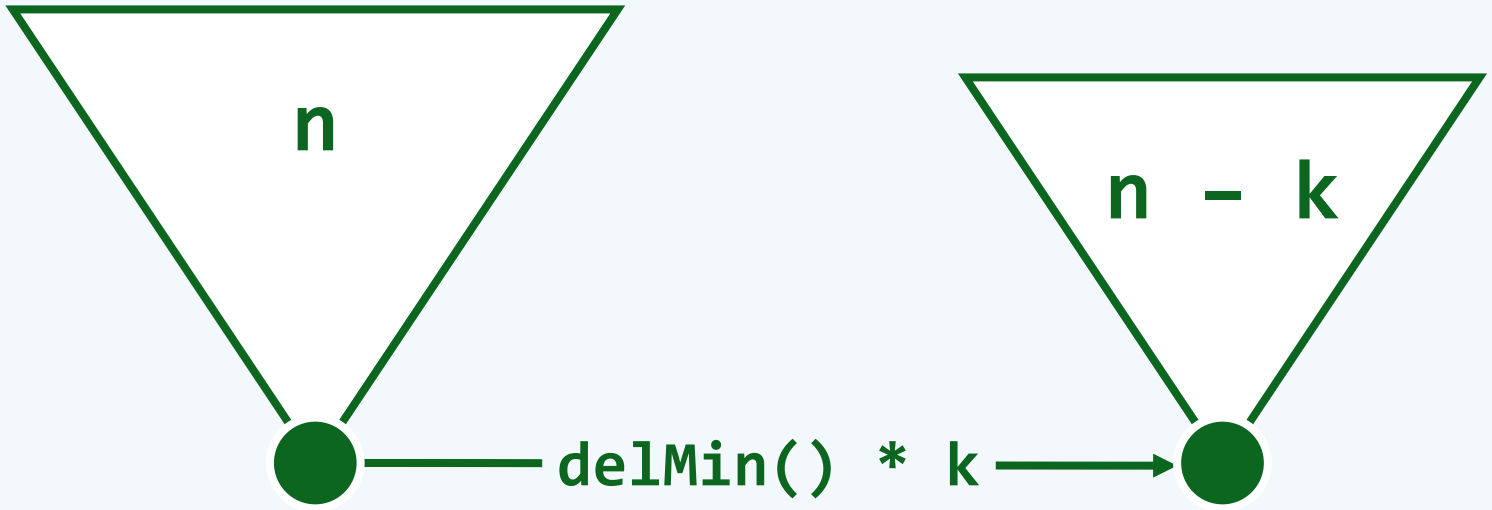
- 对A排序　　//O(nlogn)

从前向后行进k步　　//O(k) = O(n)

尝试：堆（A）

❖ 将所有元素组织为小顶堆　　//O(n)

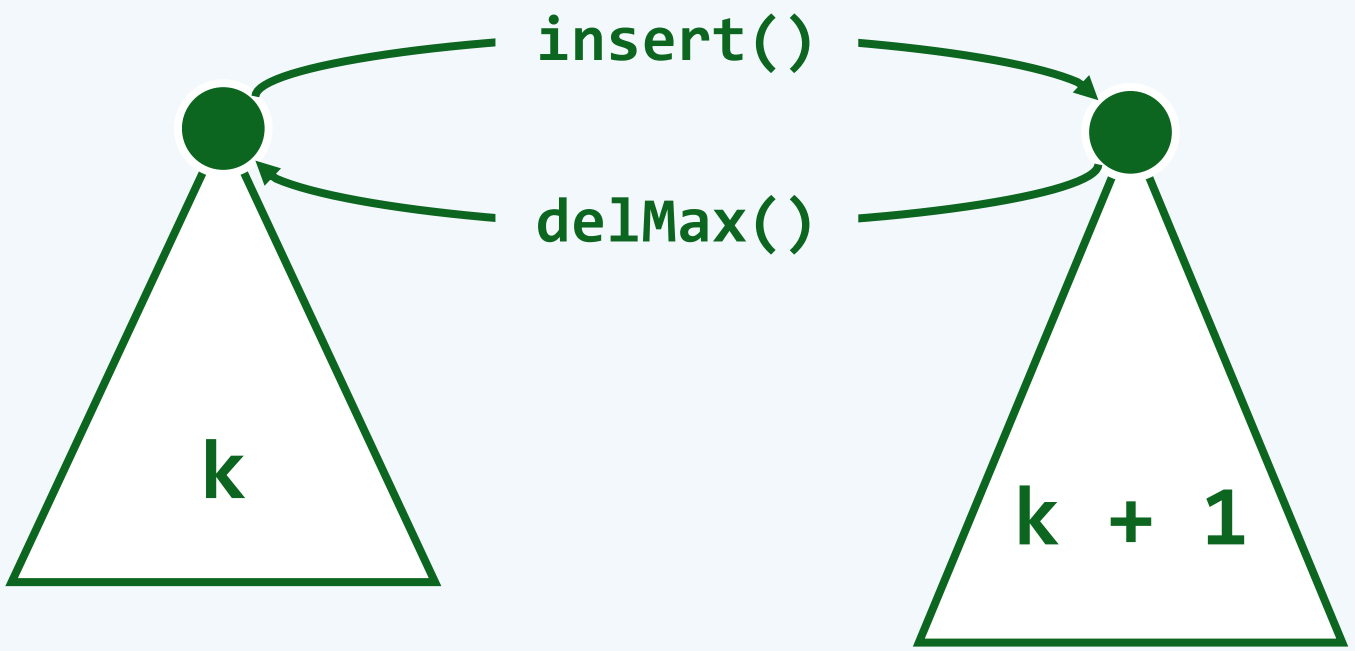
连续调用k次delMin()　　//O(klogn)



尝试：堆（B）

❖ 任选（比如前）k个元素，组织为大顶堆 //O(k)

对于剩余的n - k个元素，各调用一次insert()和delMax() //O(2*(n - k)*logk)



尝试：堆（C）

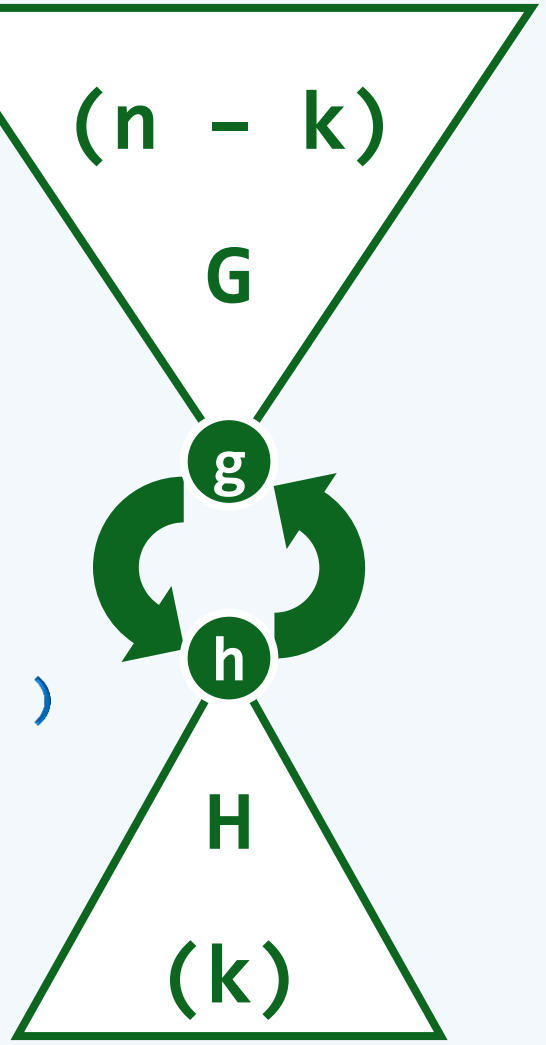
❖ H：任取k个元素，组织为大顶堆 //O(k)

G：其余n - k个元素，组织为小顶堆 //O(n - k)

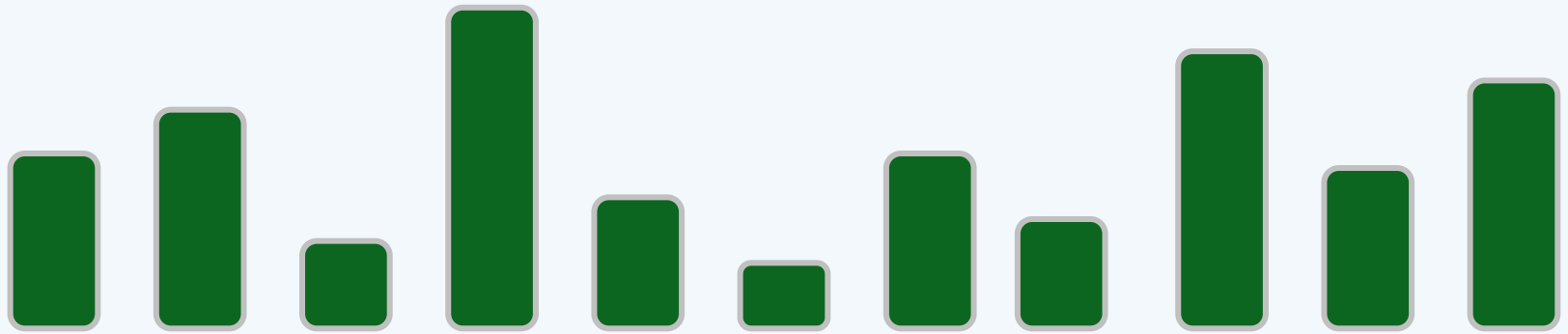
❖ 反复地： 比较h和g //O(1)

如有必要，交换之 //O(2 × (logk + log(n - k)))

直到： h ≤ g //O(min(k, n - k))



尝试：计数排序



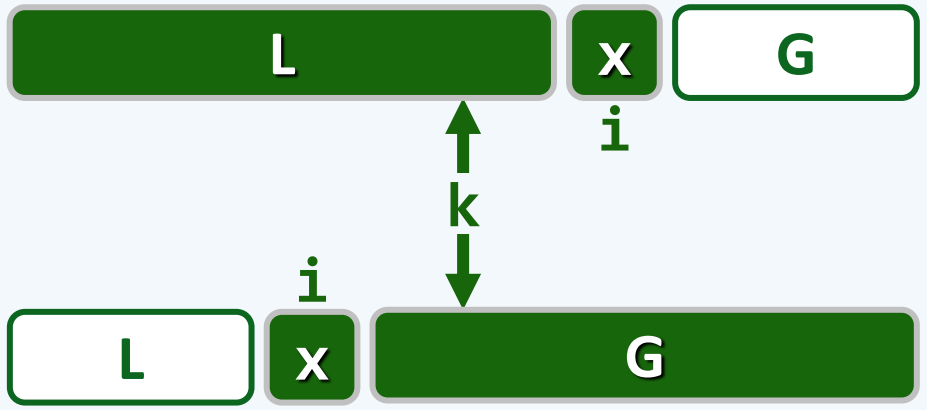
下界与最优

- 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$O(n)$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];
      while ( i < j ) { //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: 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

$$\hat{T}(n) \quad = \quad (n+1) \; + \; \frac{1}{n} \times \sum_{k=0}^{n-1} \hat{T}(max\{\, k, n-k-1 \,\}) \quad = \quad \mathcal{O}(n)$$