2-9 Sorting and Selection

Hengfeng Wei

hfwei@nju.edu.cn

May 28, 2018

How to Argue?

Show that $\cdots$, Argue that $\cdots$, Explain why $\cdots$

How to Argue?

Show that $\cdots$, Argue that $\cdots$, Explain why $\cdots$

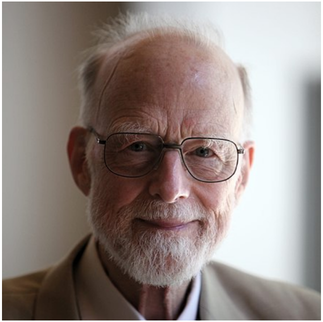
$= $ Prove that $\cdots$

Am I Alone?

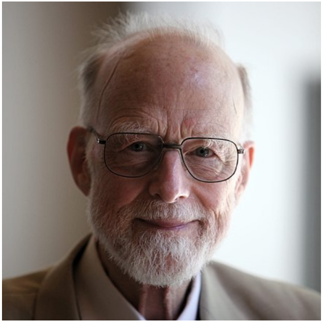
Am I Alone?

原来你也在这里

QUICKSORT (Tony Hoare, 1959/1960)

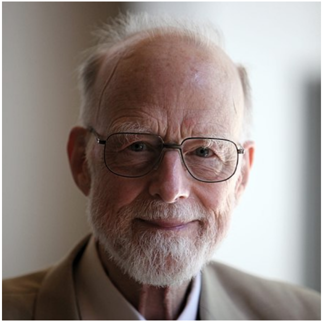


QUICKSORT (Tony Hoare, 1959/1960)



Hoare Logic: $\{P\}\,S\,\{Q\}$

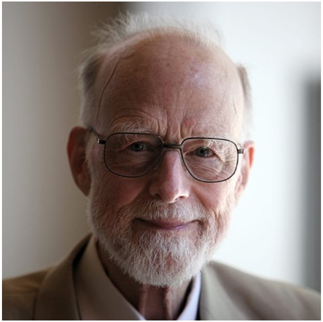
QUICKSORT (Tony Hoare, 1959/1960)



Hoare Logic: $\{P\}\, S\, \{Q\}$

`null pointer`

QUICKSORT (Tony Hoare, 1959/1960)



Hoare Logic: $\{P\}\, S\, \{Q\}$

`null pointer`

"I call it my billion-dollar mistake."

Best-Case Complexity of QUICKSORT $(7.4-2)$

Show that QUICKSORT's *best-case* running time is $\Omega(n \log n)$.

Best-Case Complexity of QUICKSORT $(7.4-2)$

Show that QUICKSORT's *best-case* running time is $\Omega(n \log n)$.

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

Best-Case Complexity of QUICKSORT $(7.4-2)$

Show that QUICKSORT's *best-case* running time is $\Omega(n \log n)$.

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

$$T(n) = \underbrace{n}_{\text{PARTITION}} \; \underbrace{\log n}_{\text{Height}} \quad \text{(Recursion Tree)}$$

Best-Case Complexity of QUICKSORT $(7.4-2)$

Show that QUICKSORT's *best-case* running time is $\Omega(n \log n)$.

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

$$T(n) = \underbrace{n}_{\text{PARTITION}} \underbrace{\log n}_{\text{Height}} \quad \text{(Recursion Tree)}$$

$$\boxed{T(n) = \min_{0 \le q \le n-1} \Big(T(q) + T(n-q-1) \Big) + \Theta(n)}$$

Best-Case Complexity of QUICKSORT $(7.4 - 2)$

Show that QUICKSORT's *best-case* running time is $\Omega(n \log n)$.

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

$$T(n) = \underbrace{n}_{\text{PARTITION}} \; \underbrace{\log n}_{\text{Height}} \quad \text{(Recursion Tree)}$$

$$\boxed{T(n) = \min_{0 \le q \le n-1} \Big(T(q) + T(n - q - 1) \Big) + \Theta(n)}$$

$$T(n) = \Omega(n \log n)$$

Best-Case Complexity of QUICKSORT $(7.4-2)$

Show that QUICKSORT's *best-case* running time is $\Omega(n \log n)$.

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

$$T(n) = \underbrace{n}_{\text{PARTITION}} \underbrace{\log n}_{\text{Height}} \quad \text{(Recursion Tree)}$$

$$\boxed{T(n) = \min_{0 \le q \le n-1} \Big(T(q) + T(n-q-1)\Big) + \Theta(n)}$$

$$T(n) = \Omega(n \log n)$$

By substitution.

Sorting Almost-Sorted Inputs (Problem $7.2-4$)

Argue that INSERTION-SORT would tend to beat QUICKSORT on *almost-sorted* inputs.

Sorting Almost-Sorted Inputs (Problem $7.2-4$)

Argue that INSERTION-SORT would tend to beat QUICKSORT on *almost-sorted* inputs.

$\#$ inversions

Sorting Almost-Sorted Inputs (Problem $7.2-4$)

Argue that INSERTION-SORT would tend to beat QUICKSORT on *almost-sorted* inputs.

$$\# \text{ inversions} = \Theta(n)$$

Sorting Almost-Sorted Inputs (Problem $7.2-4$)

Argue that $\textsc{Insertion-Sort}$ would tend to beat $\textsc{Quicksort}$ on *almost-sorted* inputs.

$$\# \text{ inversions} = \Theta(n)$$

$$\textsc{Insertion-sort} : \Theta(n)$$

Sorting Almost-Sorted Inputs (Problem $7.2-4$)

Argue that INSERTION-SORT would tend to beat QUICKSORT on *almost-sorted* inputs.

$$\#\text{ inversions} = \Theta(n)$$

$$\textsc{Insertion-sort}: \Theta(n)$$

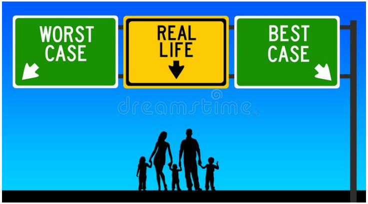
$$\textsc{Quicksort}: \Omega(n \log n)$$

Median-of-3 PARTITION (Problem $7-5$)

Argue that in the $\Omega(n \log n)$ running time of QUICKSORT, the *median-of-3* method affects only the constant factor.

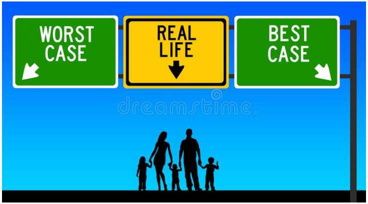
Median-of-3 PARTITION (Problem $7-5$)

Argue that in the $\Omega(n \log n)$ running time of QUICKSORT, the *median-of-3* method affects only the constant factor.



Median-of-3 PARTITION (Problem $7-5$)

Argue that in the $\Omega(n \log n)$ running time of QUICKSORT, the *median-of-3* method affects only the constant factor.



$$\boxed{T(n) = \min_{0 \le q \le n-1} \Big(T(q) + T(n-q-1) \Big) + \Theta(n)}$$

$$T(n) = \Omega(n \log n)$$

The Analysis of Quicksort Programs*

Robert Sedgewick

Received January 19, 1976

Summary. The Quicksort sorting algorithm and its best variants are presented and analyzed. Results are derived which make it possible to obtain exact formulas describing the total expected running time of particular implementations on real computers of Quicksort and an improvement called the median-of-three modification. Detailed analysis of the effect of an implementation technique called loop unwrapping is presented. The paper is intended not only to present results of direct practical utility, but also to illustrate the intriguing mathematics which arises in the complete analysis of this important algorithm.

Robert Sedgewick

The Analysis of Quicksort Programs*

Robert Sedgewick

Received January 19, 1976

Summary. The Quicksort sorting algorithm and its best variants are presented and analyzed. Results are derived which make it possible to obtain exact formulas describing the total expected running time of particular implementations on real computers of Quicksort and an improvement called the median-of-three modification. Detailed analysis of the effect of an implementation technique called loop unwrapping is presented. The paper is intended not only to present results of direct practical utility, but also to illustrate the intriguing mathematics which arises in the complete analysis of this important algorithm.

$$B_N = \frac{12}{35}(N+1)(H_{N+1} - H_{M+2}) + \frac{37}{245}(N+1) - \frac{12}{7}\frac{N+1}{M+2} + 1 \quad \text{exchanges}$$

$$C_N = \frac{12}{7}(N+1)(H_{N+1} - H_{M+2}) + \frac{37}{49}(N+1) - \frac{24}{7}\frac{N+1}{M+2} + 2 \quad \text{comparisons}$$

Robert Sedgewick

The Analysis of Quicksort Programs*

Robert Sedgewick

Received January 19, 1976

Summary. The Quicksort sorting algorithm and its best variants are presented and analyzed. Results are derived which make it possible to obtain exact formulas describing the total expected running time of particular implementations on real computers of Quicksort and an improvement called the median-of-three modification. Detailed analysis of the effect of an implementation technique called loop unwrapping is presented. The paper is intended not only to present results of direct practical utility, but also to illustrate the intriguing mathematics which arises in the complete analysis of this important algorithm.

$$B_N = \frac{12}{35}(N+1)(H_{N+1} - H_{M+2}) + \frac{37}{245}(N+1) - \frac{12}{7}\frac{N+1}{M+2} + 1 \quad \text{exchanges}$$

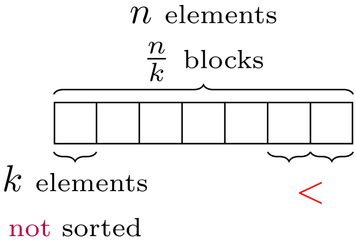
$$C_N = \frac{12}{7}(N+1)(H_{N+1} - H_{M+2}) + \frac{37}{49}(N+1) - \frac{24}{7}\frac{N+1}{M+2} + 2 \quad \text{comparisons}$$

$$B_N = (N+1)\left(\frac{1}{3}H_{N+1} - \frac{1}{3}H_{M+2} + \frac{1}{6} - \frac{1}{M+2}\right) + \frac{1}{2} \quad \text{exchanges}$$

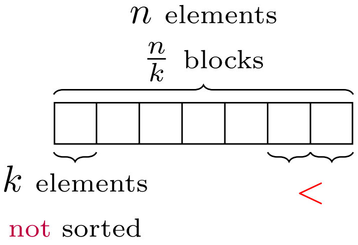
$$C_N = (N+1)(2H_{N+1} - 2H_{M+2} + 1) \quad \text{comparisons,}$$

Robert Sedgewick

Sorts a $\frac{n}{k}$-sorted Array (Problem $8.1 - 4$)

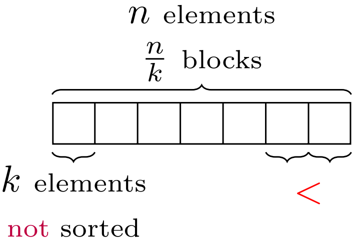


Sorts a $\frac{n}{k}$-sorted Array (Problem $8.1 - 4$)



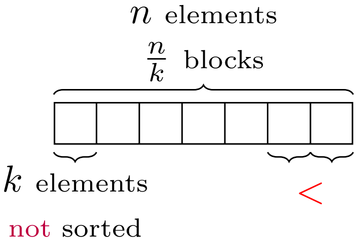
$$\Omega(n \log k)$$

Sorts a $\frac{n}{k}$-sorted Array (Problem $8.1-4$)



$\Omega(n \log k) \qquad O(n \log k)$

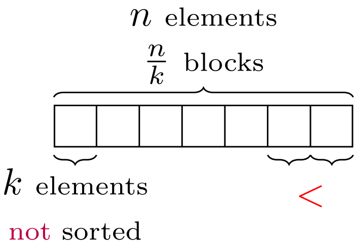
Sorts a $\frac{n}{k}$-sorted Array (Problem $8.1-4$)



$$\Omega(n \log k) \qquad O(n \log k)$$

$$\Omega : \frac{n}{k}(k \log k)$$

Sorts a $\frac{n}{k}$-sorted Array (Problem $8.1-4$)

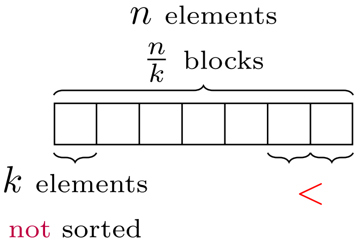


$\Omega(n \log k) \qquad O(n \log k)$

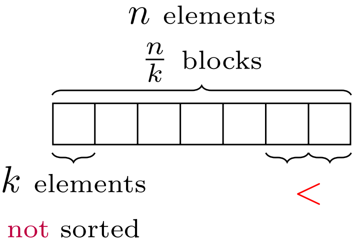
$$\Omega : \frac{n}{k}(k \log k)$$

$$(k!)^{\frac{n}{k}} \leq L \leq 2^H$$

$\frac{n}{k}$-sorts an arbitrary array

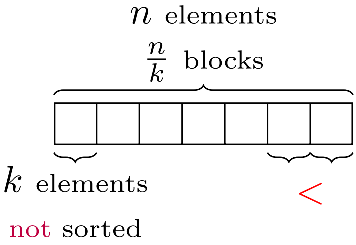


$\frac{n}{k}$-sorts an arbitrary array



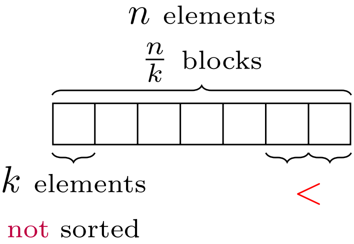
$O(?)$

$\frac{n}{k}$-sorts an arbitrary array



$O(?)$ $\Omega(?)$

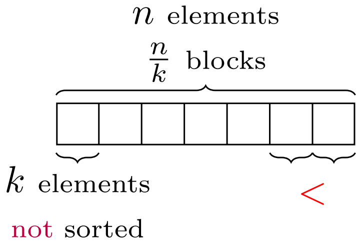
$\frac{n}{k}$-sorts an arbitrary array



$$O(?) \qquad \Omega(?)$$

$$L \geq \binom{n}{\underbrace{k, \dots, k}_{\frac{n}{k}}} = \frac{n!}{(k!)^{\frac{n}{k}}}$$

$\frac{n}{k}$-sorts an arbitrary array



$O(?) \qquad \Omega(?)$

$$L \geq \binom{n}{\underbrace{k, \ldots, k}_{\frac{n}{k}}} = \frac{n!}{(k!)^{\frac{n}{k}}} \implies \Omega(n \log(n/k))$$

Sorting $[0, n^3 - 1]$ (Problem $8.3 - 4$)

Sort n integers in $[0, n^3 - 1]$ in $O(n)$ time.

Sorting $[0, n^3 - 1]$ (Problem $8.3 - 4$)

Sort n integers in $[0, n^3 - 1]$ in $O(n)$ time.

n-ary $\qquad d = 3$

Sorting $[0, n^3 - 1]$ (Problem $8.3 - 4$)

Sort n integers in $[0, n^3 - 1]$ in $O(n)$ time.

n-ary $\qquad d = 3$

$$n = 5 : [15, 39, 20, 123, 98] = \{030, 124, 040, 443, 343\}$$

Sorting $[0, n^3 - 1]$ (Problem $8.3 - 4$)

Sort n integers in $[0, n^3 - 1]$ in $O(n)$ time.

$$n\text{-ary} \qquad d = 3$$

$$n = 5 : [15, 39, 20, 123, 98] = \{030, 124, 040, 443, 343\}$$

$$\Theta\Big(d(\underbrace{n}_{n} + \underbrace{n}_{k})\Big) = \Theta(n)$$

Sorting in Place in Linear Time (Problem $8-2$ (e))

Suppose that the n records have keys in the range $[0, k]$.
Modify $\textsc{Counting-Sort}$ to sort them in place ($O(k)$) in $O(n+k)$ time.

Sorting in Place in Linear Time (Problem $8-2$ (e))

Suppose that the n records have keys in the range $[0, k]$.
Modify COUNTING-SORT to sort them in place $(O(k))$ in $O(n+k)$ time.

	1	2	3	4	5	6	7	8
A :	2	5	3	0	2	3	0	3

	0	1	2	3	4	5
C :	2	0	2	3	0	1
C :	2	2	4	7	7	8

	1	2	3	4	5	6	7	8
B :	0	0	2	2	3	3	3	5

Sorting in Place in Linear Time (Problem $8-2$ (e))

Suppose that the n records have keys in the range $[0, k]$.
Modify COUNTING-SORT to sort them in place ($O(k)$) in $O(n+k)$ time.

A:

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C:

0	1	2	3	4	5
2	0	2	3	0	1

C:

0	1	2	3	4	5
2	2	4	7	7	8

B:

1	2	3	4	5	6	7	8
0	0	2	2	3	3	3	5

A:

1	2	3	4	5	6	7	8
2	0	3	0	2	3	3	5

C:

0	1	2	3	4	5
1	2	4	7	7	8

Sorting in Place in Linear Time (Problem $8-2$ (e))

Suppose that the n records have keys in the range $[0, k]$.
Modify COUNTING-SORT to sort them **in place** $(O(k))$ in $O(n+k)$ time.

$A:$

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

$C:$

0	1	2	3	4	5
2	0	2	3	0	1

$C:$

0	1	2	3	4	5
2	2	4	7	7	8

$B:$

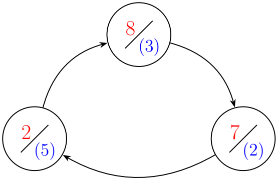
1	2	3	4	5	6	7	8
0	0	2	2	3	3	3	5

$A:$

1	2	3	4	5	6	7	8
2	0	3	0	2	3	3	5

$C:$

0	1	2	3	4	5
1	2	4	7	7	8



Sorting in Place in Linear Time (Problem $8-2\ (e)$)

Suppose that the n records have keys in the range $[0, k]$.
Modify COUNTING-SORT to sort them in place ($O(k)$) in $O(n+k)$ time.

	1	2	3	4	5	6	7	8
A :	2	5	3	0	2	3	0	3

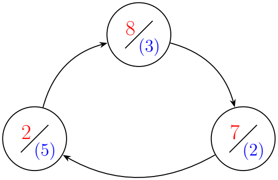
	0	1	2	3	4	5
C :	2	0	2	3	0	1
C :	2	2	4	7	7	8

	1	2	3	4	5	6	7	8
B :	0	0	2	2	3	3	3	5

	1	2	3	4	5	6	7	8
A :	2	0	3	0	2	3	3	5

	0	1	2	3	4	5
C :	1	2	4	7	7	8

`for` $(i \leftarrow n$ `to` $1)$:



Finding the 2nd Smallest Element (Problem $9.1-1$)

Show that the 2nd smallest of n elements can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case.

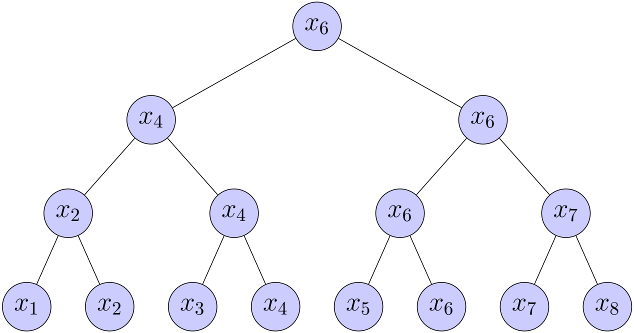
Finding the 2nd Smallest Element (Problem $9.1-1$)

Show that the 2nd smallest of n elements can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case.

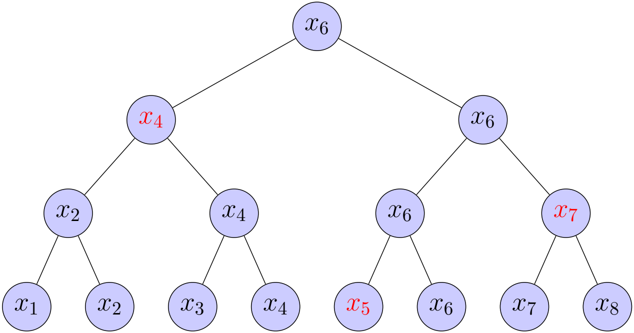
$$(n-1) + (n-1-1) = 2n-3$$

$$n + \lceil \log n \rceil - 2 = (n-1) + (\lceil \log n \rceil - 1)$$

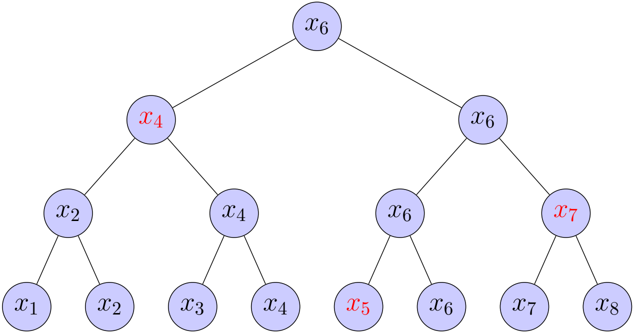
$$n + \lceil \log n \rceil - 2 = (n-1) + (\lceil \log n \rceil - 1)$$



$$n + \lceil \log n \rceil - 2 = (n-1) + (\lceil \log n \rceil - 1)$$

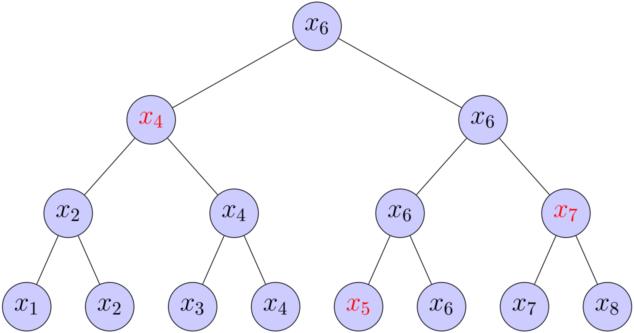


$$n + \lceil \log n \rceil - 2 = (n-1) + (\lceil \log n \rceil - 1)$$



#Potential 2nd smallest elements $\leq \lceil \log n \rceil$

$$n + \lceil \log n \rceil - 2 = (n-1) + (\lceil \log n \rceil - 1)$$



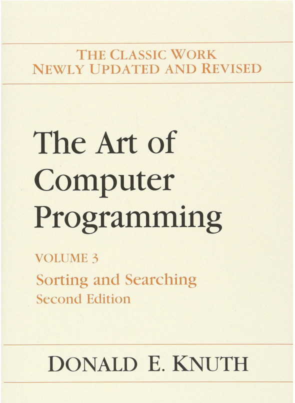
#Potential 2nd smallest elements $\leq \lceil \log n \rceil$

Q : Can we do even better?

$$\Omega = n + \lceil \log n \rceil - 2$$

$$\Omega = n + \lceil \log n \rceil - 2 = (n-1) + (\lceil \log n \rceil - 1)$$

$$\Omega = n + \lceil \log n \rceil - 2 = (n-1) + (\lceil \log n \rceil - 1)$$



TAOCP Vol 3 (Page 209, Section 5.3.3)

k Numbers Closest to the Median (Problem $9.3-7$)

$S : n$ distinct numbers $\qquad k \leq n$

k Numbers Closest to the Median (Problem $9.3-7$)

$S: n$ distinct numbers $\qquad k \le n$

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2}$$

k Numbers Closest to the Median (Problem $9.3-7$)

$S : n$ distinct numbers $\qquad k \le n$

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2} \qquad \times$$

k Numbers Closest to the Median (Problem $9.3-7$)

$S : n$ distinct numbers $\qquad k \leq n$

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2} \qquad \times$$

$$S = \{800, 6, 900, 50, 7\}, \quad k = 2 \implies \{6, 7\}$$

k Numbers Closest to the Median (Problem $9.3-7$)

$S : n$ distinct numbers $\qquad k \le n$

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2} \qquad \times$$

$$S = \{800, 6, 900, 50, 7\}, \quad k = 2 \implies \{6, 7\}$$

$$S - 50 = \{750, -44, 850, 0, -43\}$$

k Numbers Closest to the Median (Problem $9.3-7$)

$S : n$ distinct numbers $\qquad k \le n$

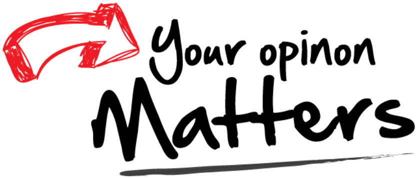
$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2} \qquad \times$$

$$S = \{800, 6, 900, 50, 7\}, \quad k = 2 \implies \{6, 7\}$$

$$S - 50 = \{750, -44, 850, 0, -43\}$$

median + subtraction + $(k+1)$-th smallest + partition + add back


Thank You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn