2-9 Sorting and Selection

Hengfeng Wei

hfwei@nju.edu.cn

May 28, 2018

How to Argue?

How to Argue?

Show that $\cdots$

How to Argue?

Show that $\cdots$

Argue that $\cdots$

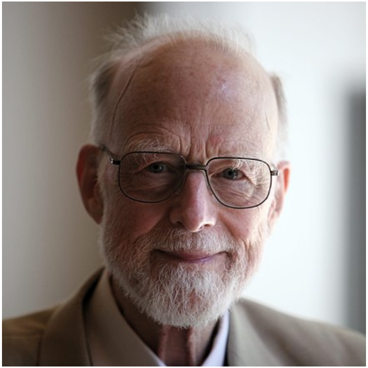
How to Argue?

Show that $\cdots$

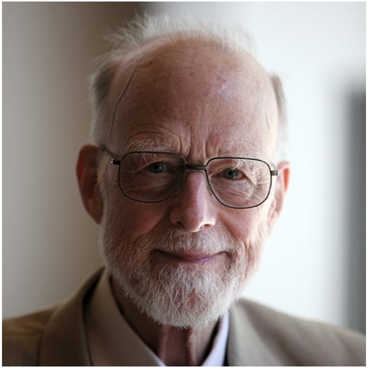
Argue that $\cdots$

$=$ Prove that $\cdots$

Quicksort Invented by Tony Hoare in 1959/1960

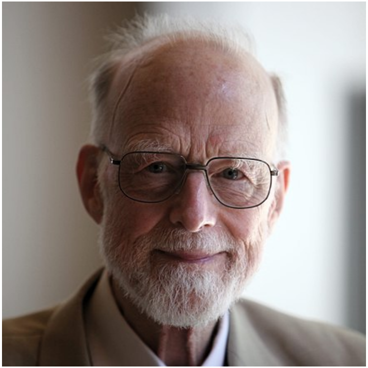


QUICKSORT Invented by Tony Hoare in 1959/1960



`null pointer`

QUICKSORT Invented by Tony Hoare in 1959/1960



`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 $\textsc{Quicksort}$ $(7.4-2)$

Show that $\textsc{Quicksort}$'s best-case running time is $\Omega(n \log n)$.

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

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

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

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

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

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

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

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

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

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

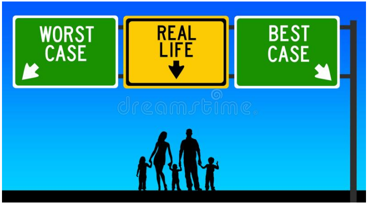
By substitution.

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.

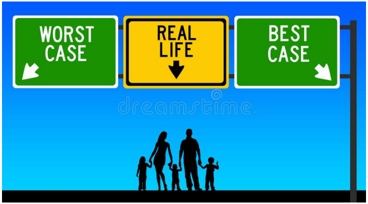
Median-of-3 PARTITION (Problem $7-5$)

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.

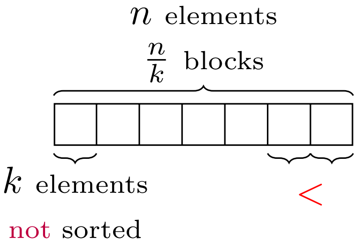


$$\boxed{T(n) = \min_{0 \le q \le n-1} \Big(T(q) + T(n-q-1) \Big) + \Theta(n)}$$

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

The $\frac{n}{k}$-sorted Problem (Problem $8.1 - 4$)

Sorts an already $\frac{n}{k}$-sorted array

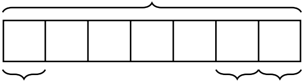


The $\frac{n}{k}$-sorted Problem (Problem $8.1-4$)

Sorts an already $\frac{n}{k}$-sorted array

n elements

$\frac{n}{k}$ blocks



k elements

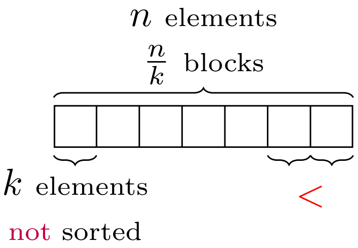
$<$

not sorted

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

The $\frac{n}{k}$-sorted Problem (Problem $8.1 - 4$)

Sorts an already $\frac{n}{k}$-sorted array



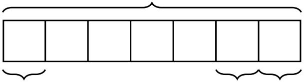
$\Omega(n \log k)$ $\qquad$ $O(n \log k)$

The $\frac{n}{k}$-sorted Problem (Problem $8.1 - 4$)

Sorts an already $\frac{n}{k}$-sorted array

n elements

$\frac{n}{k}$ blocks



k elements

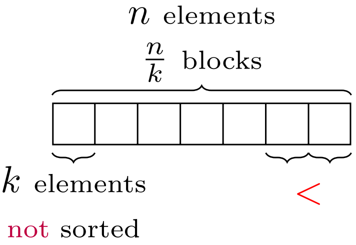
$<$

not sorted

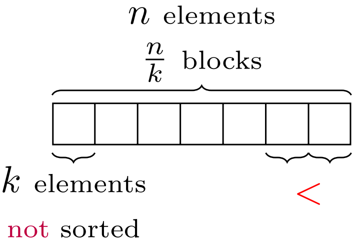
$\Omega(n \log k) \qquad O(n \log k)$

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

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

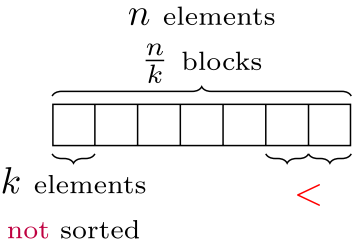


$\frac{n}{k}$-sorts an arbitrary array



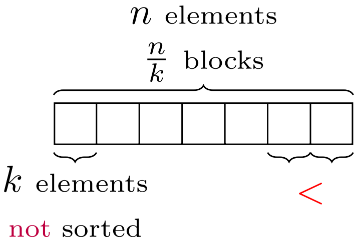
$O(?)$

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



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

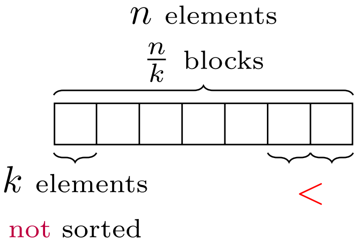
$\frac{n}{k}$-sorts an arbitrary array



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

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

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



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

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

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

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

Sorting 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.

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.

	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

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.

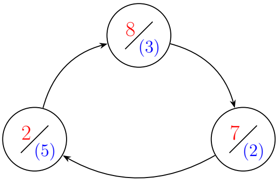
	1	2	3	4	5	6	7	8
A :	2	5	3	0	2	3	0	3

	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



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

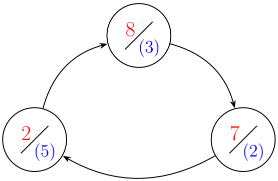
	0	1	2	3	4	5
C :	2	0	2	3	0	1
C :	2	2	4	7	7	8

	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

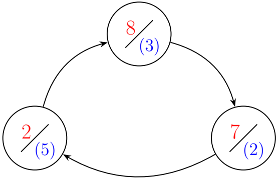
`While` $(i \geq 1)$:



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

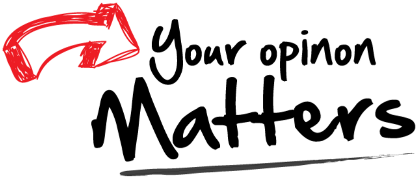
	0	1	2	3	4	5
C :	1	2	4	7	7	8

`While` $(i \geq 1)$:



Code here


Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn