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 $\cdots$

不好，掉坑里了

不好，掉坑里了

“在千山万水人海相遇，喔，原来你也在这里”

入坑指南 (Coupon Collector Problem)

The Double Dixie Cup Problem

Donald J. Newman
The American Mathematical Monthly
Vol. 67, No. 1 (Jan., 1960), pp. 58-61

Published by: Taylor & Francis, Ltd. on behalf of the Mathematical Association of America
DOI: 10.2307/2308930
Stable URL: http://www.jstor.org/stable/2308930
Page Count: 4

Topics: Mathematical theorems

$$1 \to 2 \to m$$

入坑指南 (Coupon Collector Problem)

The Double Dixie Cup Problem

Donald J. Newman
The American Mathematical Monthly
Vol. 67, No. 1 (Jan., 1960), pp. 58-61

Published by: Taylor & Francis, Ltd. on behalf of the Mathematical Association of America
DOI: 10.2307/2308930
Stable URL: http://www.jstor.org/stable/2308930
Page Count: 4

Topics: Mathematical theorems

$$1 \to 2 \to m$$

The Coupon Collector's Problem

Marco Ferrante, Monica Saltalamacchia

In this note we will consider the following problem: how many coupons we have to purchase (on average) to complete a collection. This problem, which takes everybody back to his childhood when this was really "a problem", has been considered by the probabilists since the eighteenth century and nowadays it is still possible to derive some new results, probably original or at least never published. We will present some classic results, some new formulas, some alternative approaches to obtain known results and a couple of amazing expressions.

"兄弟同心，其利断金"版本

入坑指南 (Coupon Collector Problem)

The Double Dixie Cup Problem

Donald J. Newman
The American Mathematical Monthly
Vol. 67, No. 1 (Jan., 1960), pp. 58-61

Published by: Taylor & Francis, Ltd. on behalf of the Mathematical Association of America
DOI: 10.2307/2308930
Stable URL: http://www.jstor.org/stable/2308930
Page Count: 4

Topics: Mathematical theorems

The Coupon Collector's Problem

Marco Ferrante, Monica Saltalamacchia

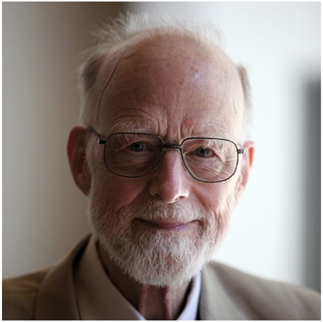
In this note we will consider the following problem: how many coupons we have to purchase (on average) to complete a collection. This problem, which takes everybody back to his childhood when this was really "a problem", has been considered by the probabilists since the eighteenth century and nowadays it is still possible to derive some new results, probably original or at least never published. We will present some classic results, some new formulas, some alternative approaches to obtain known results and a couple of amazing expressions.

"兄弟同心，其利断金"版本

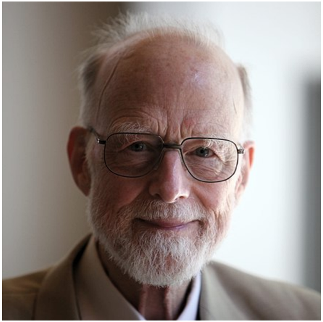
$$1 \to 2 \to m$$

$$n \log n + (m-1) n \log \log n + n C_m + o(n), \quad n \to \infty, m \text{ fixed}$$

QUICKSORT (Tony Hoare, 1959/1960)

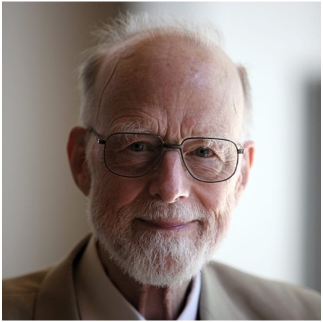


QUICKSORT (Tony Hoare, 1959/1960)



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

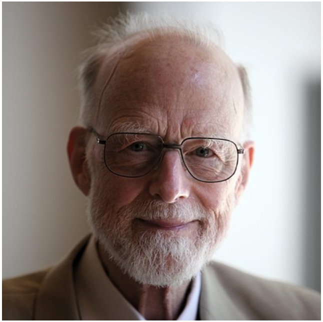
QUICKSORT (Tony Hoare, 1959/1960)



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

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 $\Omega(n \log n)$.

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

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

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

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

By substitution. (Section 7.4.1, CLRS)

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 $\textsc{Insertion-Sort}$ would tend to beat $\textsc{Quicksort}$ 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 INSERTION-SORT would tend to beat 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.

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

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

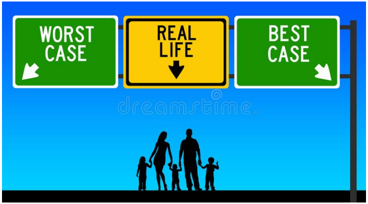
$$\textsc{Quicksort} : \Omega(n \log n)$$

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.

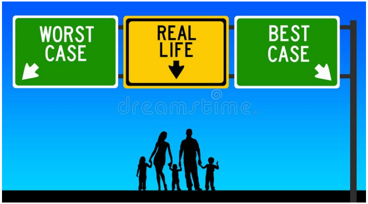
Median-of-3 PARTITION (Problem $7-5$)

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.

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

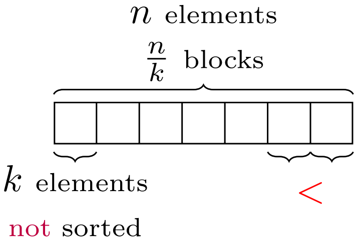
$$C_N = \frac{12}{7}(N+1)(H_{N+1} - H_{M+2}) + \frac{37}{49}(N+1) - \frac{24}{7}\frac{N+1}{M+2} + 2 \quad \text{comparisons}$$

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

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

Robert Sedgewick

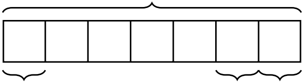
Sorts a $\frac{n}{k}$-sorted Array (Problem $8.1-4$)



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

n elements

$\frac{n}{k}$ blocks



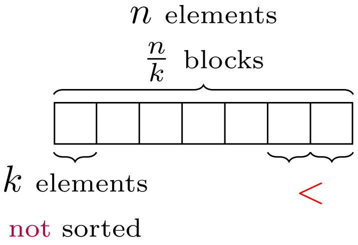
k elements

$<$

not sorted

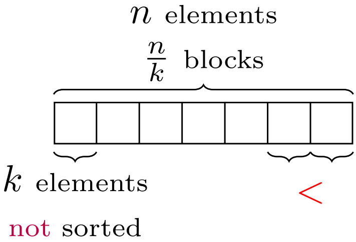
$\Omega(n \log k)$

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



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

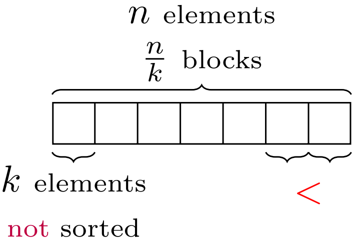
Sorts a $\frac{n}{k}$-sorted Array (Problem $8.1-4$)



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

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

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

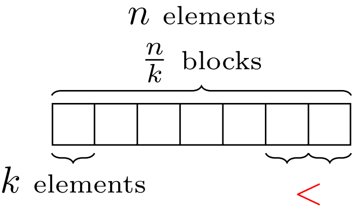


$\Omega(n \log k) \qquad O(n \log k)$

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

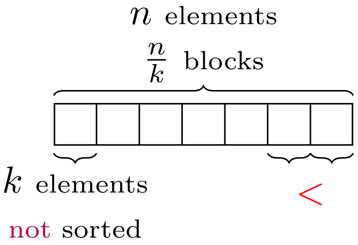
$$(k!)^{\frac{n}{k}} \leq L \leq 2^H$$

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



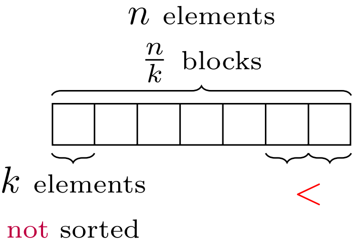
not sorted

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



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

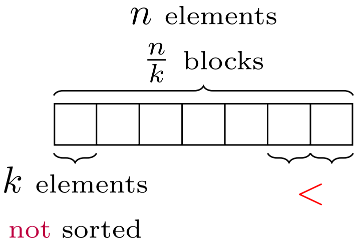
$\frac{n}{k}$-sorts an arbitrary array



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

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

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



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

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

$$O(n \log k)$$

$$O(n \log k)$$

反馈: 这是什么意思? 我们并没学过多变量的渐近符号的定义。

$$O(n \log k)$$

反馈: 这是什么意思? 我们并没学过多变量的渐近符号的定义。

Problem $3.1 - 8$

When n and m go to ∞ independently at different rates:

$$O(g(n,m)) = \{f(n,m) \mid \exists c > 0, \exists n_0 > 0, \exists m_0 > 0 : \\ \forall n \geq n_0 \vee m \geq m_0, 0 \leq f(n,m) \leq cg(n,m)\}$$

$$O(n \log k)$$

反馈: 这是什么意思? 我们并没学过多变量的渐近符号的定义。

Problem $3.1-8$

When n and m go to ∞ independently at different rates:

$$O(g(n,m)) = \{f(n,m) \mid \exists c > 0, \exists n_0 > 0, \exists m_0 > 0 : \\ \forall n \geq n_0 \lor m \geq m_0, 0 \leq f(n,m) \leq cg(n,m)\}$$

$$k = O(1) \qquad k = \Theta(n); \qquad n \to \infty$$

$$\Theta(1 + \alpha), \quad \alpha = \frac{n}{m}$$

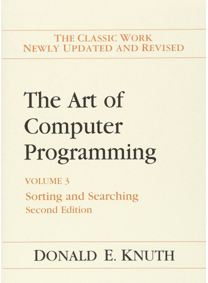
$$\Theta(1+\alpha), \quad \alpha = \frac{n}{m}$$

Two ways of understanding

$$\Theta(1+\alpha), \quad \alpha = \frac{n}{m}$$

Two ways of understanding

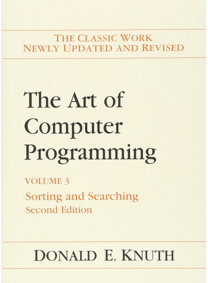
$1 + \alpha$



$$\Theta(1+\alpha), \quad \alpha = \frac{n}{m}$$

Two ways of understanding

$1 + \alpha$



$\Theta(1+\alpha)$

$$n \to \infty, m \to \infty$$

$$n = f(m), m \to \infty$$

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

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

n-ary $\qquad d = 3$

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

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

Any other costs?

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

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

n-ary $\qquad d = 3$

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

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

Any other costs?

$$3n \cdot T(\square \div n)$$

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.

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.

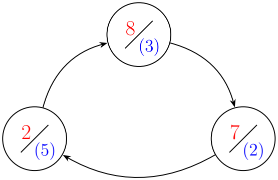
	1	2	3	4	5	6	7	8
A :	2	5	3	0	2	3	0	3

	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



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

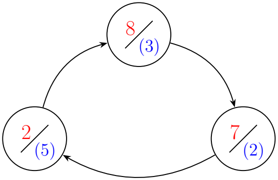
A:

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

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



Assigning elements according to C

	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

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

Assigning elements according to C

$$
A: \quad
\begin{array}{|c|c|c|c|c|c|c|c|}
\multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} \\
\hline
2 & 5 & 3 & 0 & 2 & \color{blue}{3} & 0 & \color{red}{3} \\
\hline
\end{array}
$$

$$
C: \quad
\begin{array}{|c|c|c|c|c|c|}
\multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} \\
\hline
2 & 0 & 2 & 3 & 0 & 1 \\
\hline
\end{array}
$$

$$
\color{red}{A}: \quad
\begin{array}{|c|c|c|c|c|c|c|c|}
\multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6} & \multicolumn{1}{c}{7} & \multicolumn{1}{c}{8} \\
\hline
0 & 0 & 2 & 2 & 3 & \color{blue}{3} & \color{red}{3} & 5 \\
\hline
\end{array}
$$

Q : What is the possible problem?

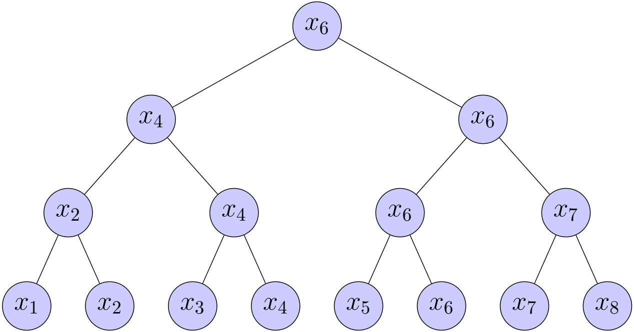
Finding the 2nd Smallest Element (Problem $9.1-1$)

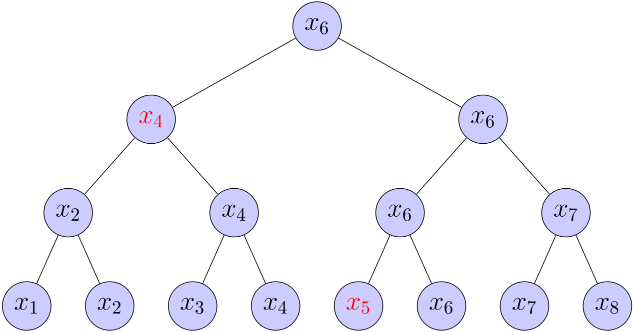
Show that the 2nd smallest of n elements can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case.

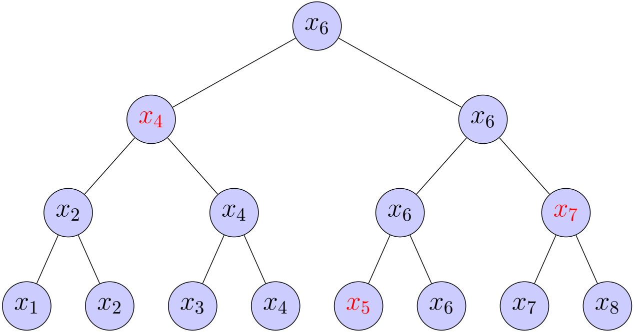
Finding the 2nd Smallest Element (Problem $9.1-1$)

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

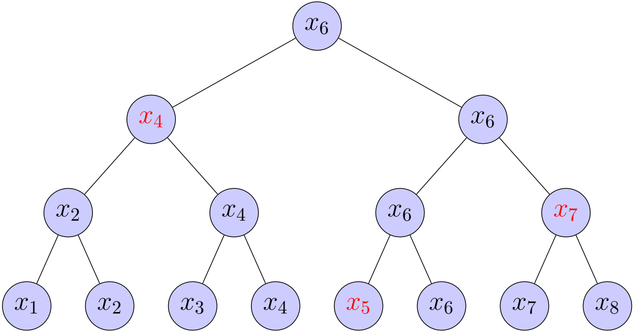
$$(n-1) + (n-1-1) = 2n-3$$


x_6
x_4
x_6
x_2
x_4
x_6
x_7
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8


x_6
x_4
x_6
x_2
x_4
x_6
x_7
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8

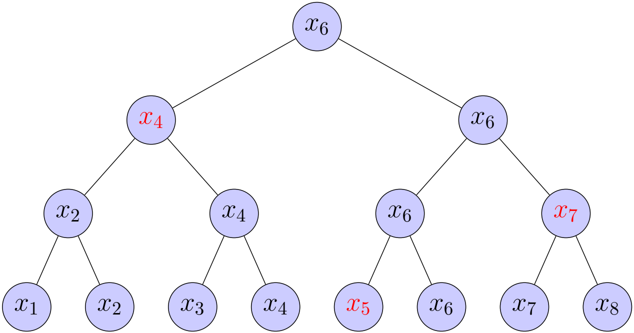


\# Potential 2nd smallest elements $\leq \lceil \log n \rceil$



\# 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 $\leq \lceil \log n \rceil$

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

Q : Can we do even better?

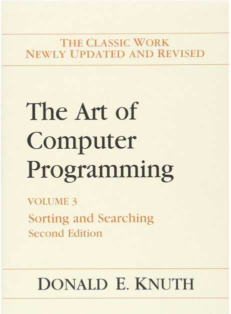
Adversary Argument

Adversary Argument

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

Adversary Argument

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

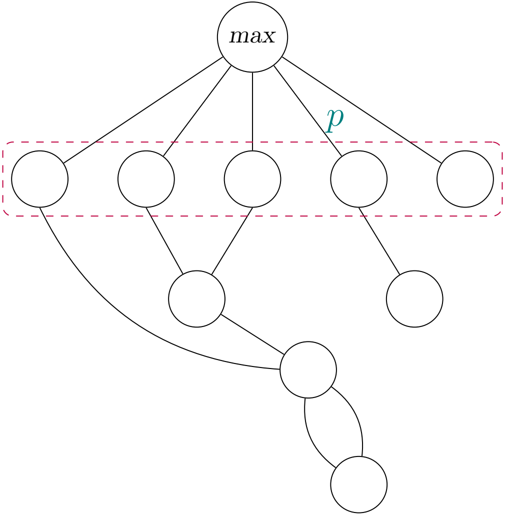


TAOCP, Vol. 3 (Page 209, Section 5.3.3)

Theorem

Any comparison-based algorithm to find secondLargest in n keys must do at least $n + \lceil \log n \rceil - 2$ comparisons in the worst case.

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



$$p \geq \lceil \log n \rceil$$

$$p \geq \lceil \log n \rceil$$

Adversary $\mathcal{A}$:

$x > y$

$x < y$

Algorithm $\mathbb{A}$:

$\textsc{Compare}(x, y)$

$$p \geq \lceil \log n \rceil$$

Adversary $\mathcal{A}$:

$x > y$

$x < y$

Algorithm $\mathbb{A}$:

$\textsc{Compare}(x, y)$

$$\exists \mathcal{A}\ \forall \mathbb{A} \left(\geq \lceil \log n \rceil \text{ distinct keys lost directly to } \textit{max} \text{ in } \mathbb{A} \right)$$

$$\forall x : w(x) = 1$$

$\textsc{Compare}(x, y)$

$\textsc{Case}$	$\textsc{Reply}$	$\textsc{(Local) Action}$
$w(x) > w(y)$	$x > y$	$w'(x) \leftarrow w(x) + w(y); w'(y) \leftarrow 0$
$w(x) = w(y) > 0$	$x > y$	$w'(x) \leftarrow w(x) + w(y); w'(y) \leftarrow 0$
$w(x) = w(y) = 0$	No Conflict	$\textsc{Nop}$
$w(y) > w(x)$	$y > x$	$w'(y) \leftarrow w(x) + w(y); w'(x) \leftarrow 0$

$$\forall x : w(x) = 1$$

$\textsc{Compare}(x, y)$

Case	Reply	(Local) Action
$w(x) > w(y)$	$x > y$	$w'(x) \leftarrow w(x) + w(y); w'(y) \leftarrow 0$
$w(x) = w(y) > 0$	$x > y$	$w'(x) \leftarrow w(x) + w(y); w'(y) \leftarrow 0$
$w(x) = w(y) = 0$	No Conflict	Nop
$w(y) > w(x)$	$y > x$	$w'(y) \leftarrow w(x) + w(y); w'(x) \leftarrow 0$

赢者通吃; 败者永不得翻身

$$\sum_x w(x) = n$$

$$\sum_x w(x) = n$$

$$w(x) = 0 \iff x \text{ has lost}$$

$$\sum_x w(x) = n$$

$$w(x) = 0 \iff x \text{ has lost}$$

$$w(x) > 0 \iff x \text{ has not yet lost}$$

$$\sum_{x} w(x) = n$$

$$w(x) = 0 \iff x \text{ has lost}$$

$$w(x) > 0 \iff x \text{ has not yet lost}$$

Theorem

For any algorithm $\mathbb{A}$ that finds secondLargest, when it stops,

$$\exists! x : w(x) > 0.$$

$$\sum_x w(x) = n$$

$$w(x) = 0 \iff x \text{ has lost}$$

$$w(x) > 0 \iff x \text{ has not yet lost}$$

Theorem

For any algorithm $\mathbb{A}$ that finds secondLargest, when it stops,

$$\exists! x : w(x) > 0.$$

$$\boxed{t = \textit{max}, \quad w(t) = n}$$

Theorem

t *has directly won against* $\geq \lceil \log n \rceil$ *distinct keys.*

Theorem

t *has directly won against* $\geq \lceil \log n \rceil$ *distinct keys.*

Proof.

K : $\#$ of comparisons t wins against previously undefeated keys

Theorem

t has directly won against $\geq \lceil \log n \rceil$ distinct keys.

Proof.

K : # of comparisons t wins against previously undefeated keys

$$\boxed{n = w_K}$$

Theorem

t has directly won against $\geq \lceil \log n \rceil$ distinct keys.

Proof.

K : # of comparisons t wins against previously undefeated keys

$$\boxed{n = w_K}$$

w_k : $w(t)$ after t winning against k previously undefeated keys

Theorem

t has directly won against $\geq \lceil \log n \rceil$ distinct keys.

Proof.

K : # of comparisons t wins against previously undefeated keys

$$\boxed{n = w_K}$$

w_k : $w(t)$ after t winning against k previously undefeated keys

$$\boxed{w_k \leq 2w_{k-1}}$$

Theorem

t has directly won against $\geq \lceil \log n \rceil$ distinct keys.

Proof.

K : # of comparisons t wins against previously undefeated keys

$$\boxed{n = w_K}$$

w_k : $w(t)$ after t winning against k previously undefeated keys

$$\boxed{w_k \leq 2w_{k-1}}$$

$$n = w_K \leq 2^K w_0 = 2^K$$

Theorem

t has directly won against $\geq \lceil \log n \rceil$ distinct keys.

Proof.

K : # of comparisons t wins against previously undefeated keys

$$\boxed{n = w_K}$$

w_k : $w(t)$ after t winning against k previously undefeated keys

$$\boxed{w_k \leq 2w_{k-1}}$$

$$n = w_K \leq 2^K w_0 = 2^K \implies K \geq \log n$$

Theorem

t has directly won against $\geq \lceil \log n \rceil$ distinct keys.

Proof.

K : # of comparisons t wins against previously undefeated keys

$$\boxed{n = w_K}$$

w_k : $w(t)$ after t winning against k previously undefeated keys

$$\boxed{w_k \leq 2w_{k-1}}$$

$$\boxed{n = w_K \leq 2^K w_0 = 2^K \implies K \geq \log n \implies K \geq \lceil \log n \rceil}$$

□

max&min (Problem 9.1-2)

Prove the lower bound of $\lceil 3n/2 \rceil - 2$ comparisons in the worst case to find both the *max* and the *min* of n numbers.

max&min (Problem 9.1-2)

Prove the lower bound of $\lceil 3n/2 \rceil - 2$ comparisons in the worst case to find both the *max* and the *min* of n numbers.

$$\mathbb{A} : \lceil 3n/2 \rceil - 2$$

max&min (Problem 9.1-2)

Prove the lower bound of $\lceil 3n/2 \rceil - 2$ comparisons in the worst case to find both the *max* and the *min* of n numbers.

$$\mathbb{A} : \lceil 3n/2 \rceil - 2$$

Adversary $\mathcal{A}$:

$$n/2 + (n-2)$$

Algorithm $\mathbb{A}$:

$$2n - 2$$

units of Win/Lose info.

max&min (Problem 9.1-2)

Prove the lower bound of $\lceil 3n/2 \rceil - 2$ comparisons in the worst case to find both the *max* and the *min* of n numbers.

$$\mathbb{A} : \lceil 3n/2 \rceil - 2$$

Adversary $\mathcal{A}$:

$$n/2 + (n-2)$$

Algorithm $\mathbb{A}$:

$$2n-2$$

units of $\textsc{Win}/\textsc{Lose}$ info.

$$2n - 2 = 2 \times (n/2) + 1 \times (n-2)$$

"01" Pattern

To check whether an array $A[1 \cdots n]$ of n bits contains the "01" pattern.
Do we have to check every bit?

“01” Pattern

To check whether an array $A[1 \cdots n]$ of n bits contains the “01” pattern. Do we have to check every bit?

No, if n is odd;

Yes, if n is even.

1	2	3	4	5	6	7	8	9

First checking $A[2, 4, \ldots, n-1]$

1	2	3	4	5	6	7	8	9

First checking $A[2, 4, \ldots, n-1]$

	0		0		0		0	

	1		1		1		1	

			0				1	

	1		1		0		0	

Adversary $\mathcal{A}$:

$0/1$

Algorithm $\mathbb{A}$:

$\textsc{Check}(i)$

Adversary $\mathcal{A}$:

0/1

Algorithm $\mathbb{A}$:

$\textsc{Check}(i)$

1	1	1	1	1	1							0	0	0	0

l $\underbrace{\qquad\qquad\qquad\qquad}_{\text{even numbers}}$ r

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 $\qquad k \leq n$

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

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

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

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2}$$ (By rank)

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

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

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2} \quad \text{(By rank)}$$

$$S = \{800, 6, 900, 50, 7\}, \quad k = 2 \implies \{6, 7\} \quad \text{(By distance)}$$

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

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

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2} \quad \text{(By rank)}$$

$$S = \{800, 6, 900, 50, 7\}, \quad k = 2 \implies \{6, 7\} \quad \text{(By distance)}$$

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

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

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

$$\frac{n}{2} - \frac{k}{2} \sim \frac{n}{2} + \frac{k}{2}$$ (By rank)

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

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

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


TREASURE HUNT
on foot

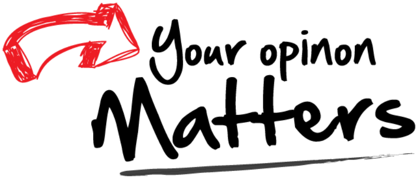

TIPS

The Coupon Collector's Problem

Marco Ferrante, Monica Saltalamacchia

In this note we will consider the following problem: how many coupons we have to purchase (on average) to complete a collection. This problem, which takes everybody back to his childhood when this was really "a problem", has been considered by the probabilists since the eighteenth century and nowadays it is still possible to derive some new results, probably original or at least never published. We will present some classic results, some new formulas, some alternative approaches to obtain known results and a couple of amazing expressions.


Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn