CS3230 Lecture 5

NUS National University of Singapore
School of Computing

"Min-Max, Order Statistics, and Linear Time OS"

- **Lecture Topics and Readings**
  - Linear Time Sorting Algorithms [CLRS]-C8.2,8.3
  - Min, Max, and Min-Max
  - Randomized Divide-and-Conquer [CLRS]-C9
  - Order Statistics in Linear Time [CLRS]-C9

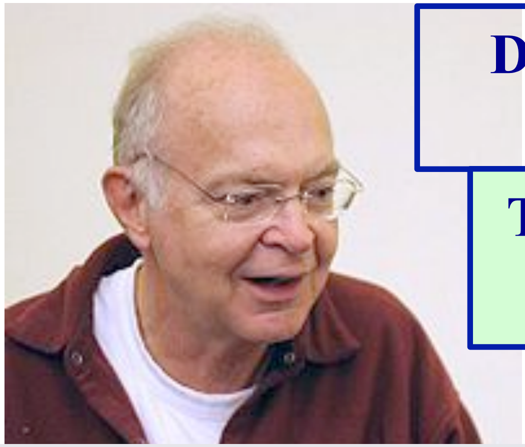
THINK!

Busting the Lower Bound
Recursive algorithms are elegant!
Balancing leads to efficient algorithms

MOE Think out of the Box

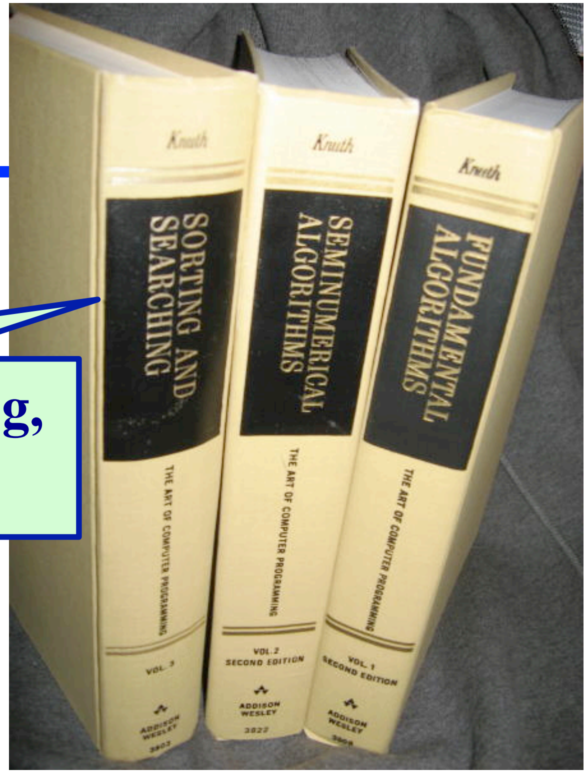
Singapore
MOE Building
Buona Vista
THINK out of the Box!

Sorting and Searching



Don Knuth, Stanford

The Art of Computing Programming, Vol 3, "*Sorting and Searching*"



Shen Chuan Qi (2011 SG IOI Team, now at Stanford) attending "Christmas Tree Lecture" given by Don Knuth, around Xmas 2013.

Topic: <u>Planar Graphs and Ternary Trees</u>

Search: Don Knuth, Christmas Tree Lectures, December 2013

Thank you.

Q & A

NUS
National University of Singapore
School of Computing

CS3230 Lecture 5(b)

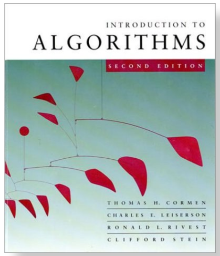
NUS National University of Singapore
School of Computing

"Lower Bound for Sorting, Linear-Time Sorting"

"Order Statistics, and Linear Time OS"

- Lecture Topics and Readings
  - Order Statistics, Min, Max, Min-Max
  - Randomized Divide-and-Conquer [CLRS]-C9
  - Order Statistics in Linear Time [CLRS]-C9

Recursive algorithms are elegant!
Balancing leads to efficient algorithms



Order statistics

Select the ith smallest of n elements (the element with ***rank*** i).

- $i = 1$: ***minimum***;
- $i = n$: ***maximum***;
- $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: ***median***.

Naive algorithm: Sort and index ith element.

Worst-case running time $= \Theta(n \lg n) + \Theta(1)$

$= \Theta(n \lg n)$,

using merge sort or heapsort (*not* quicksort).

Quick Revision:
We start with the humble Find-Max

Iterative Find-Max algorithm

FIND-MAX $A[1\,..\,n]$

1. Let $Max\text{-}sf := A[1]$;
2. **for** $k := 2$ **to** n **do**
3. **if** $A[k] > Max\text{-}sf$ **then**
4. $Max\text{-}sf := A[k]$
5. **return** $Max\text{-}sf$

If $A=[3, 1, 5, 7]$ Let $M(x, y) = \text{Max}\ \{x, y\}$

$Max\text{-}sf = 3,\ M(1,3)=3,\ M(5,3)=5,\ M(7,5)=7$

Iterative Find-Max algorithm

A-Problem:
On average, how often is line 4 executed?

Find-Max $A[1 \, . \, . \, n]$

1. Let $\mathit{Max\text{-}sf} := A[1]$;
2. **for** $k := 2$ **to** n **do**
3. **if** $A[k] > \mathit{Max\text{-}sf}$ **then**
4. $\mathit{Max\text{-}sf} := A[k]$
5. **return** $\mathit{Max\text{-}sf}$

Obviously: $T(n) = \Theta(n)$
more precisely: (n–1 comparisons)

Now, we make it Recursive

Making Find-Max recursive

FIND-MAX $A[1\,..\,n]$

1. Let $Max\text{-}sf := A[1]$;
2. **for** $k := 2$ **to** n **do**
3. **if** $A[k] > Max\text{-}sf$ **then**
4. $Max\text{-}sf := A[k]$
5. **return** $Max\text{-}sf$

Compares $A[k]$ with $\max\{ A_1, A_2, \ldots, A_{k-1} \}$

***Question:* Can we turn this into a recursive algorithm?**

Iterative Find-Max algorithm

FIND-MAX $A[1\,..\,n]$

1. Let $Max\text{-}sf := A[1]$;
2. **for** $k := 2$ **to** n **do**
3. **if** $A[k] > Max\text{-}sf$ **then**
4. $Max\text{-}sf := A[k]$
5. **return** $Max\text{-}sf$

compares $A[k]$ with $\max\{A_1, A_2, \ldots, A_{k-1}\}$

When $k=7$, what is value of $Max\text{-}sf$?

$Max\text{-}sf = \max\{A[1], A[2], \ldots A[6]\}$

Iterative Find-Max algorithm

FIND-MAX $A[1 \,..\, n]$

1. Let $Max\text{-}sf := A[1]$;
2. **for** $k := 2$ **to** n **do**
3. **if** $A[k] > Max\text{-}sf$ **then**
4. $Max\text{-}sf := A[k]$
5. **return** $Max\text{-}sf$

compares $A[k]$ with $\max\{ A_1, A_2, \ldots, A_{k-1} \}$

When $k=n$, what is value of $Max\text{-}sf$?…

$Max\text{-}sf = \max\ \{A[1], A[2], \ldots A[n-1]\}$

Recursive Find-Max (FMR)

Recursion Schematic:

$$\boxed{\text{FMR}\{A[1..n]\}} = \max\{\boxed{\text{FMR}\{A[1..(n-1)]\}},\ A[n]\ \}$$

Find-Max-R = FMR

Find-Max-R $A[1\ ..\ n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1\ ..\ n-1]$
3. return Max $\{A[n], M1\}$

Recursive Find-Max (FMR)

Find-Max-R $A[1\,..\,n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1\,..\,n-1]$
3. return Max $\{A[n], M1\}$

$\boxed{\text{FMR}\{[3,1,5,7]\}}$

Max $\{7,$ $\boxed{\text{FMR}\{[3,1,5]\}}$ $\}$

If $A=[3, 1, 5, 7]$ }

(Finish this example yourself. Check with slides at the back.)

Find-Max and Find-Max-R

Find-Max-R does exactly
the ***same computations*** as Find-Max!

Have the same asymptotic
$\Theta(n)$ worst-case time.

Let's explore the
Recursion Schematic
some more

Recursion Schematics

Recursion Schematic:

$$\boxed{\text{FMR}\{A[1..n]\}} = \max\{\boxed{\text{FMR}\{A[1..(n-1)]\}},\ A[n]\ \}$$

$(n-1)$ 1

Extreme imbalance

Recursion Schematics

Recursion Schematic:

$$\boxed{\mathrm{FMR}\{A[1..n]\}} = \max\{\boxed{\mathrm{FMR}\{A[1..(n-1)]\}},\ A[n]\ \}$$

$(n-1)$ 1

Extreme imbalance

Balanced Recursion Schematic:

$$\mathrm{BFM}\{A[1..n]\}$$

$$= \max\{\ \mathrm{BFM}\{A[1..\ n/2]\}\ ,\ \mathrm{BFM}\{A[n/2+1..\ n]\}\ \}$$

Balanced Recursive Find-Max

BFM $A[1\ ..\ n]$

1. **if** $n = 1$, done.
2. $M1$:= BFM $A[\ 1\ ..\ \lceil n/2 \rceil\ ]$
   $M2$:= BFM $A[\ \lceil n/2 \rceil + 1\ ..\ n\ ]$.
3. **return** max $\{M2, M2\}$

Don't this remind you of

Merge-Sort ?

Recall: Merge sort

MERGE-SORT $A[1 \,..\, n]$

1. If $n = 1$, done.
2. MERGE-SORT $A[\,1 \,..\, \lceil n/2 \rceil\,]$
   MERGE-SORT $A[\,\lceil n/2 \rceil + 1 \,..\, n\,]$
3. "***Merge***" the 2 sorted lists.

Balanced Recursive Find-Max

BFM $A[1\ ..\ n]$

1. **if** $n = 1$, done.
2. $M1 := \text{BFM}\ A[\ 1\ ..\ \lceil n/2 \rceil\ ]$
   $M2 := \text{BFM}\ A[\ \lceil n/2 \rceil + 1\ ..\ n\ ]$.
3. **return** max $\{M1, M2\}$

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(1) \text{ if } n > 1. \end{cases}$$

Hon Wai Leong, NUS

Balanced Recursive Find-Max

BFM $A[1\,..\,n]$

1. **if** $n = 1$, done.
2. $M1 :=$ BFM $A[\,1\,..\,\lceil n/2 \rceil\,]$
   $M2 :=$ BFM $A[\,\lceil n/2 \rceil + 1\,..\,n\,]$.
3. **return** max $\{M1, M2\}$

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(1) \text{ if } n > 1. \end{cases}$$

BFM: $a = 2$, $b = 2 \;\Rightarrow\; n^{\log_b a} = n^{\log_2 2} = n$

$f(n) = O(n^{1-\varepsilon})$ for $\varepsilon = 0.5 \;\Rightarrow$ CASE 1: $T(n) = O(n)$.

Hon Wai Leong, NUS

Recursion tree

Solve $T(n) = 2T(n/2) + 1$.

Hon Wai Leong, NUS

Recursion tree

Solve $T(n) = 2T(n/2) + 1$.

$$T(n)$$

Hon Wai Leong, NUS

Recursion tree

Solve $T(n) = 2T(n/2) + 1$.

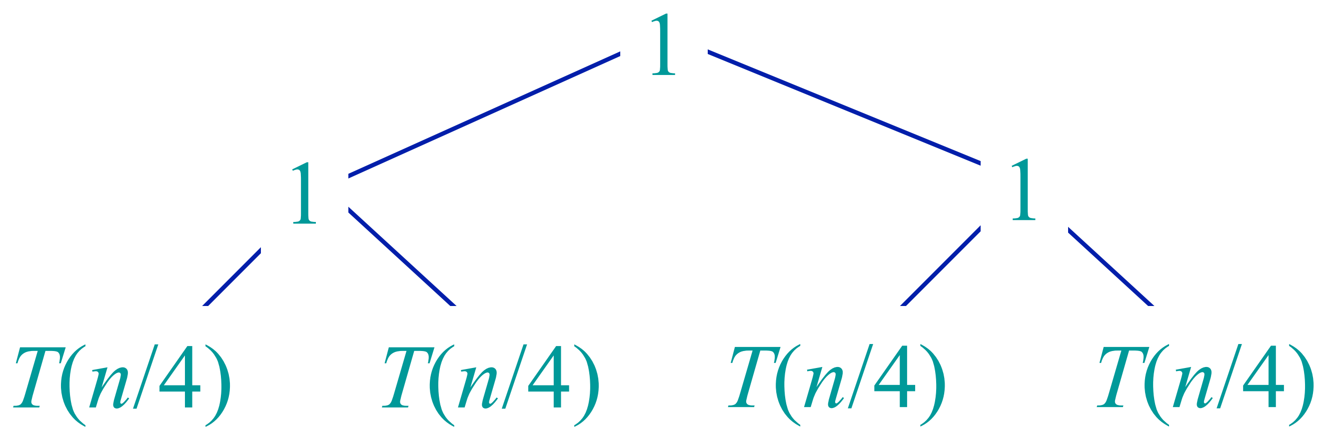
$$1$$

$$T(n/2) \qquad\qquad T(n/2)$$

Hon Wai Leong, NUS

Recursion tree

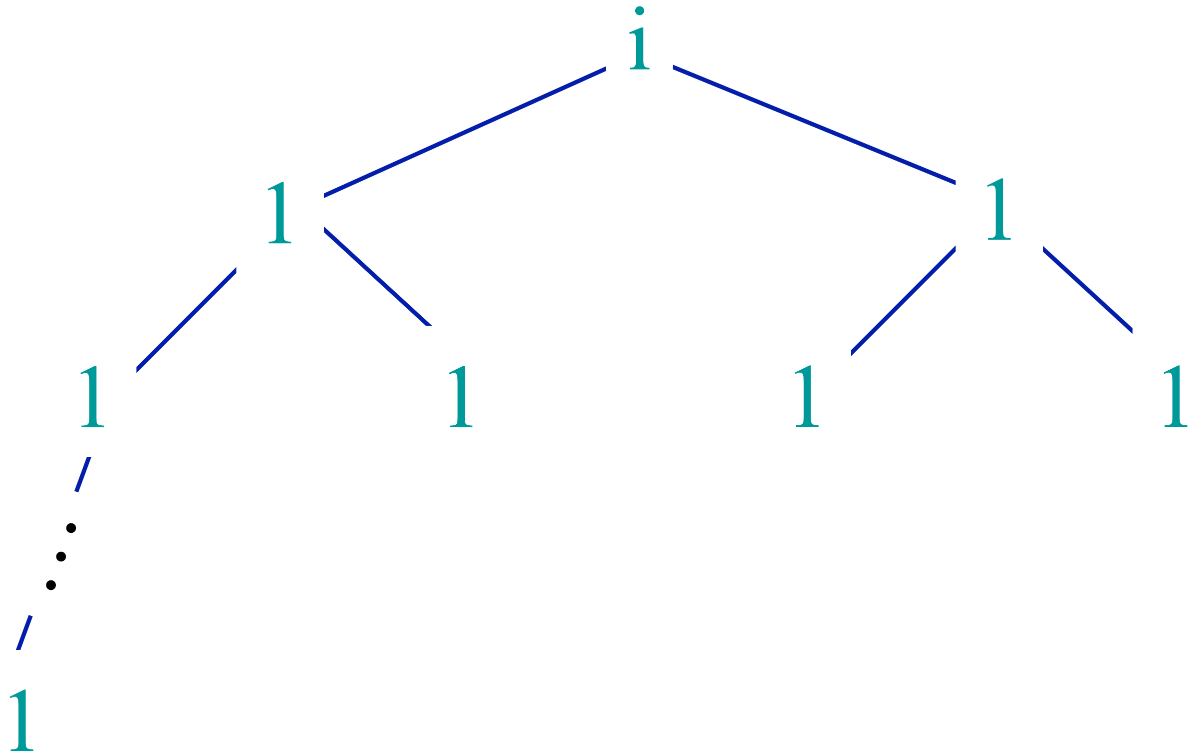
Solve $T(n) = 2T(n/2) + 1$.



Hon Wai Leong, NUS

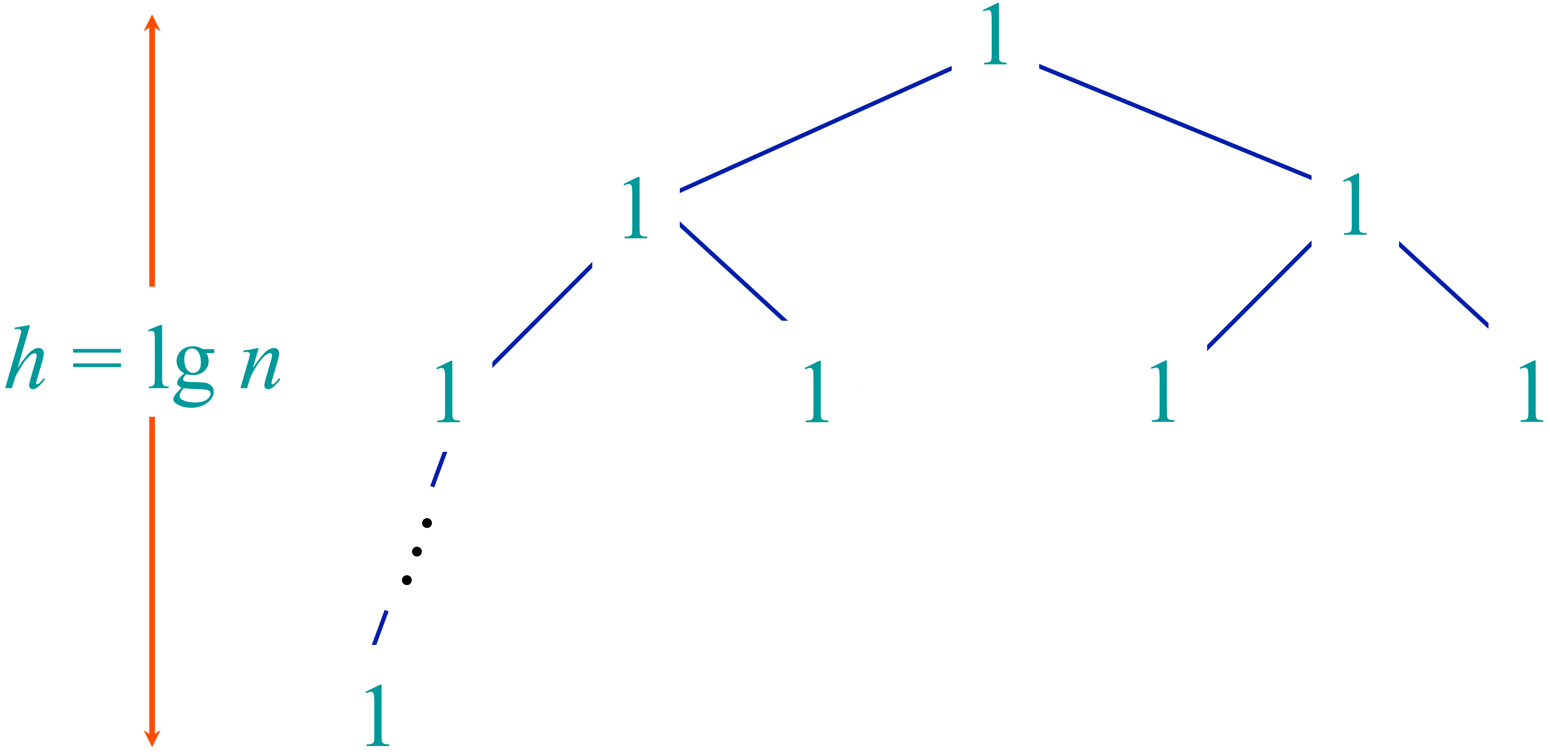
Recursion tree

Solve $T(n) = 2T(n/2) + 1$.



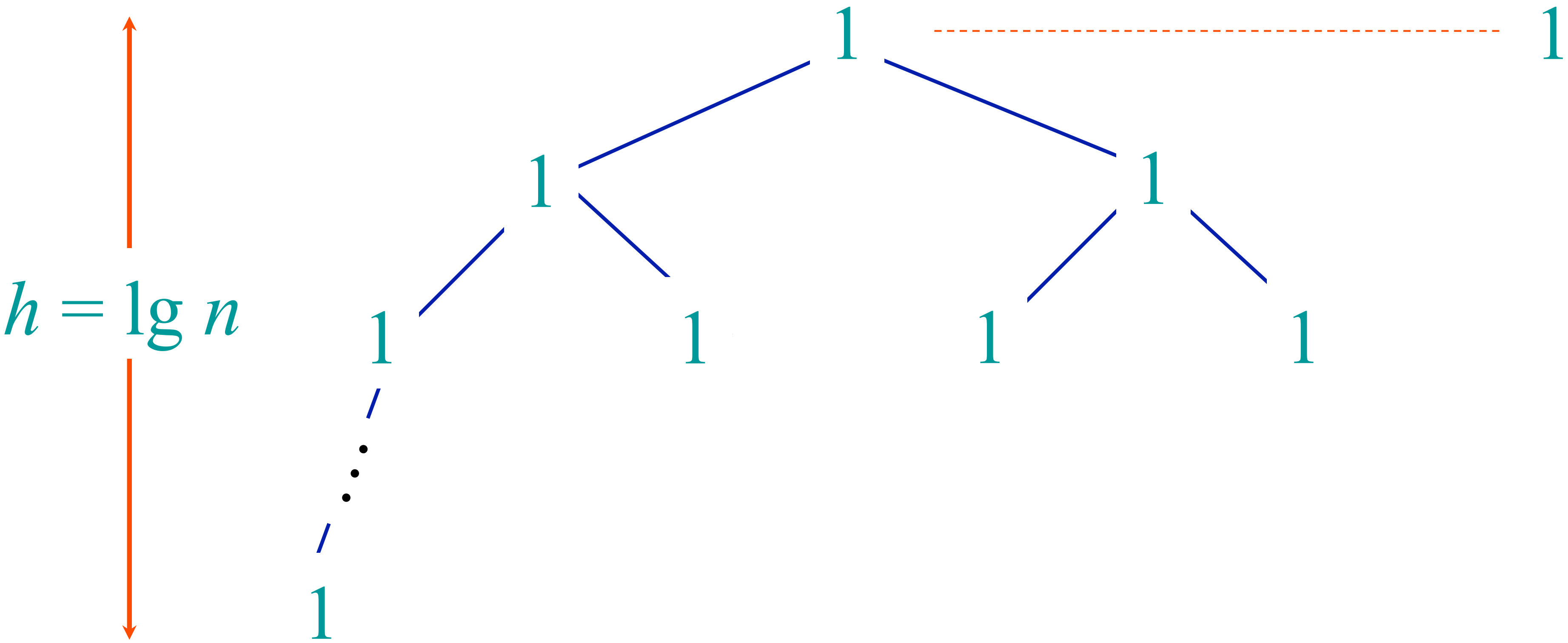
Recursion tree

Solve $T(n) = 2T(n/2) + 1$.



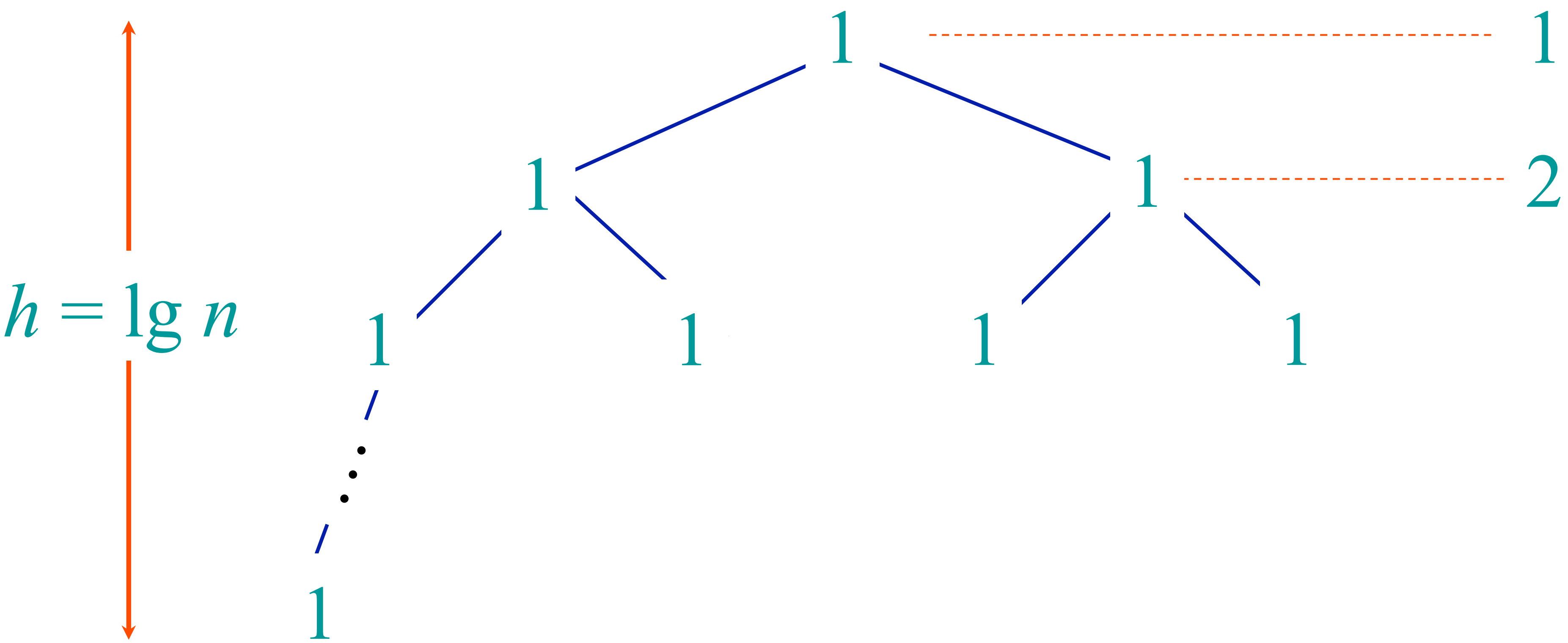
Recursion tree

Solve $T(n) = 2T(n/2) + 1$.



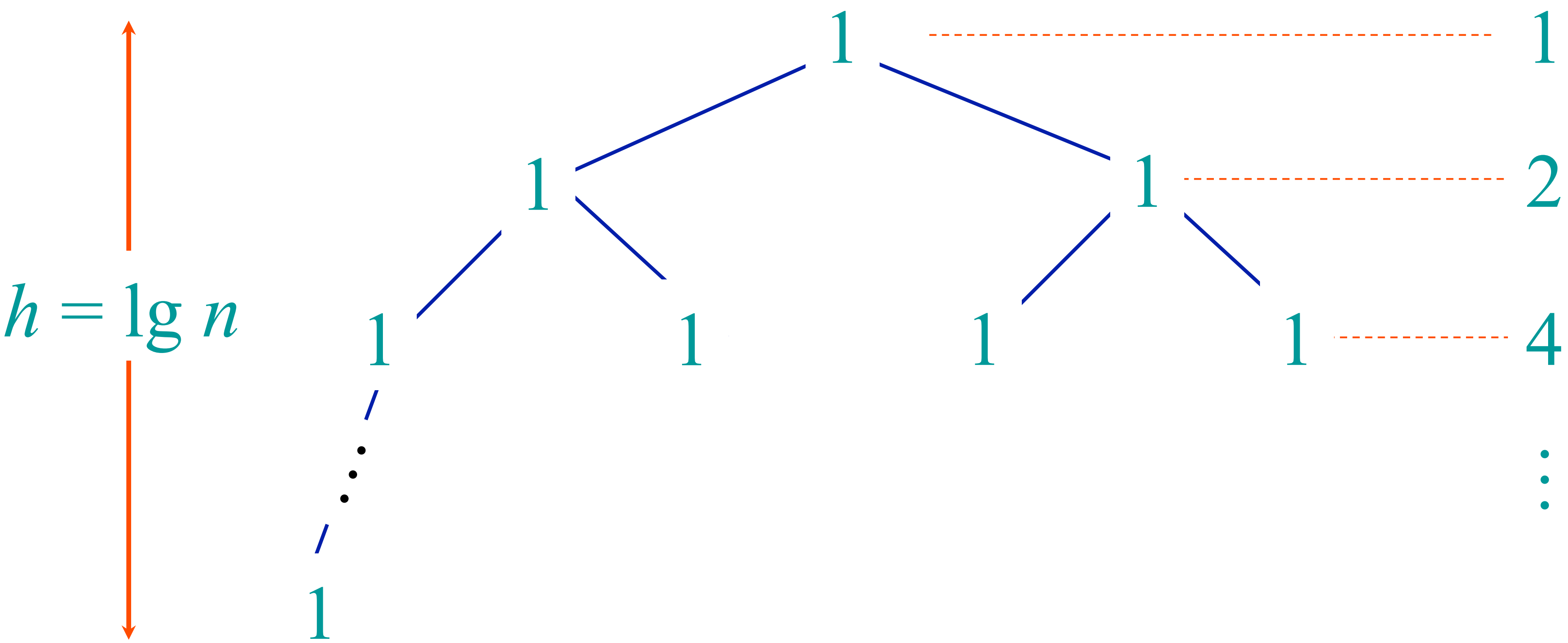
Recursion tree

Solve $T(n) = 2T(n/2) + 1$.



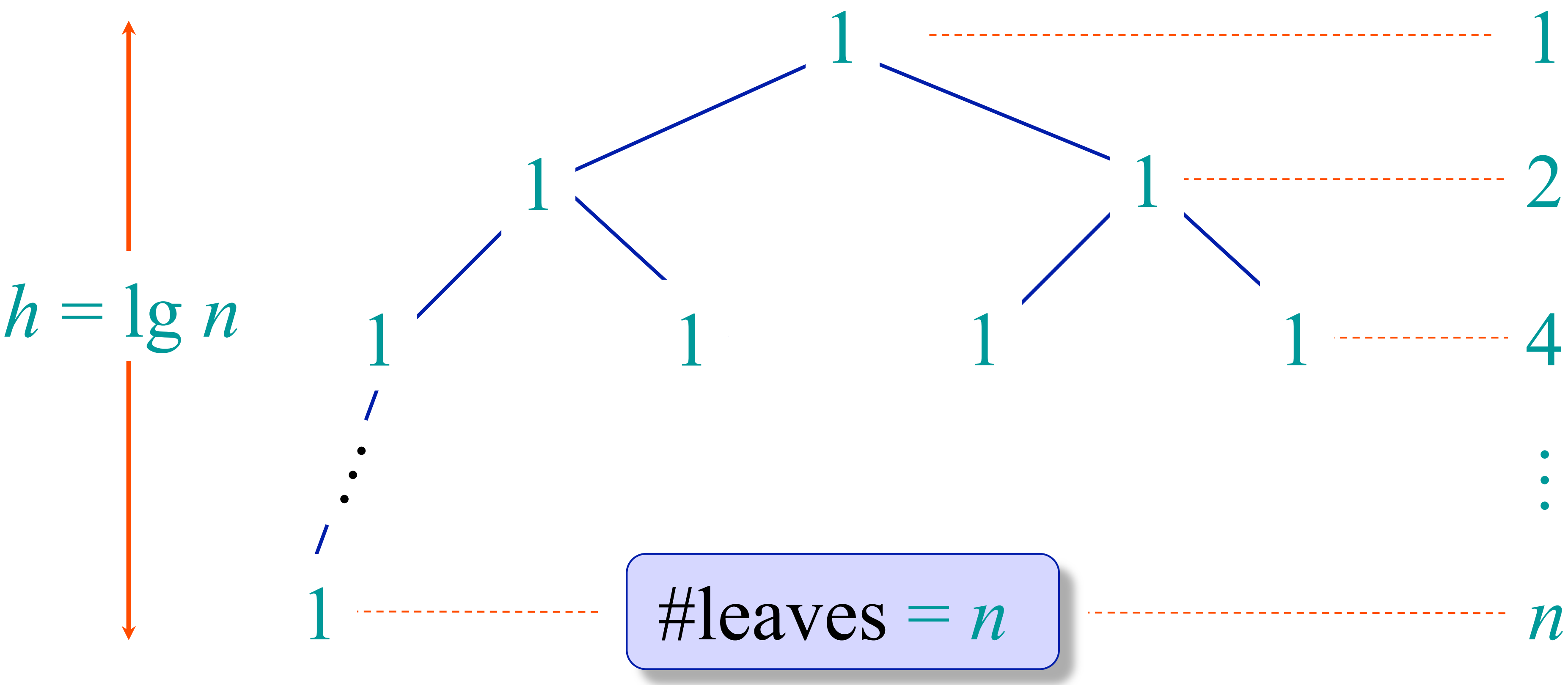
Recursion tree

Solve $T(n) = 2T(n/2) + 1$.



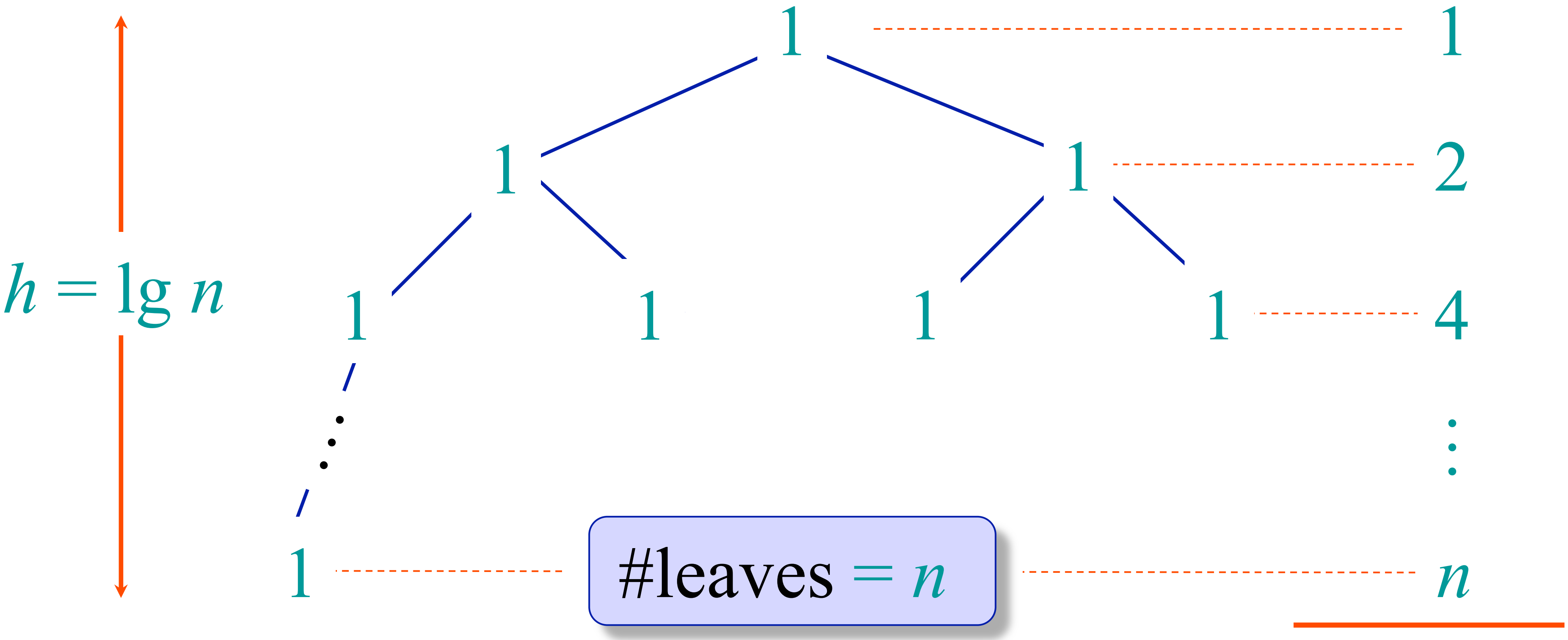
Recursion tree

Solve $T(n) = 2T(n/2) + 1$.



Recursion tree

Solve $T(n) = 2T(n/2) + 1$.



Total $= \Theta(n)$

How to do the sum?

Recall

$$\sum_{k=0}^{\lg n} 2^k = 1 + 2 + 2^2 + \cdots + 2^h \le 2n$$

Or equivalently,

$$\sum_{k=0}^{\lg n} n/2^k = \left(n + n/2 + n/2^2 + \cdots + n/2^{\lg n}\right)$$

$$= n\left(1 + \frac{1}{2} + \frac{1}{2^2} + \ldots + \frac{1}{2^{\lg n}}\right) \le 2n$$

Have you seen this before?

Isn't this the game in HW1-S1(b)?

Modification of HW1-S1(b)

Algorithm (from HW1-S1(b))

1. Each student k stand up, given number A[k]
2. Pair up with someone standing, the one with *smaller number* sits down
3. Go back to Step 2

What's the similarity?

What the difference?

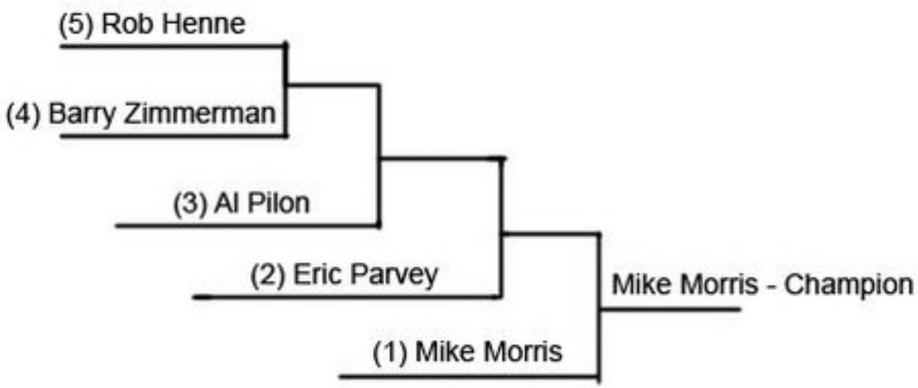
Algorithms is A&E…

Recursion Schematic 1:

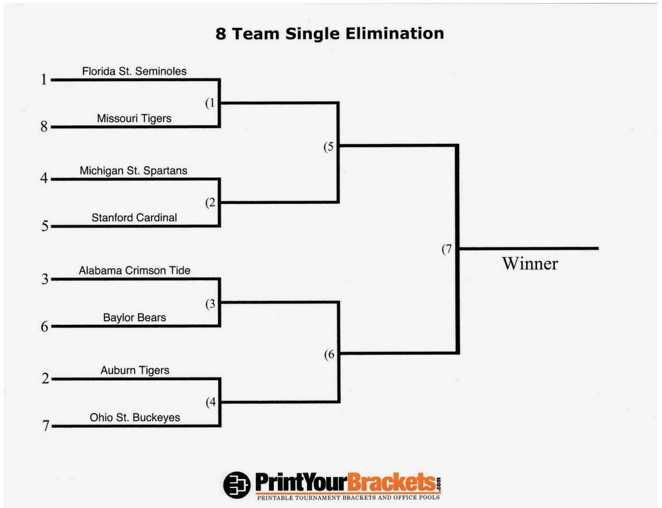
$$\mathrm{FMR}\{A[1..n]\} = \mathrm{Max}\{\ \mathrm{FMR}\{A[1..(n-1)]\}\ ,\ A[n]\ \}$$

Recursion Schematic 2:

$$\mathrm{BFM}\{A[1..n]\} = \mathrm{Max}\{\ \mathrm{BFM}\{A[1..\ n/2]\},\ \mathrm{BFM}\{A[n/2+1..\ n]\}\ \}$$

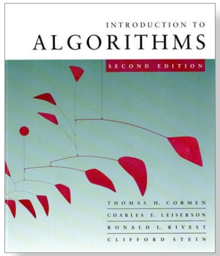


Bowling Classic Step-Ladder



Knock-out Tournament

http://legacy.wday.com/event/image/id/3122/headline/Bowling%20Classic%20Stepladder/



How about finding Max-and-Min ?

Tutorial Question (See T4).

Thank you.

Q & A

Recursive Find-Max (FMR)

Find-Max-R $A[1\,..\,n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1\,..\,n-1]$
3. return Max $\{A[n], M1\}$

$\boxed{\text{FMR}\{[3,1,5,7]\}}$

Max $\{7,$ $\boxed{\text{FMR}\{[3,1,5]\}}$ $\}$

Max $\{5,$ $\boxed{\text{FMR}\{[3,1]\}}$ $\}$

If A=[3, 1, 5, 7] }

Hon Wai Leong, NUS

Recursive Find-Max (FMR)

Find-Max-R $A[1\,..\,n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1\,..\,n-1]$
3. return Max $\{A[n], M1\}$

If A=[3, 1, 5, 7] }

$\boxed{\text{FMR}\{[3,1,5,7]\}}$

Max$\{7,$ $\boxed{\text{FMR}\{[3,1,5]\}}$ $\}$

Max$\{5,$ $\boxed{\text{FMR}\{[3,1]\}}$ $\}$

Max$\{1,$ $\boxed{\text{FMR}\{[3]\}}$ $\}$

Hon Wai Leong, NUS

Recursive Find-Max (FMR)

Find-Max-R $A[1 \,..\, n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1 \,..\, n-1]$
3. return Max $\{A[n], M1\}$

If $A=[3, 1, 5, 7]$ }

$\boxed{\text{FMR}\{[3,1,5,7]\}}$

Max $\{7,$ $\boxed{\text{FMR}\{[3,1,5]\}}$ $\}$

Max $\{5,$ $\boxed{\text{FMR}\{[3,1]\}}$ $\}$

Max $\{1,$ $\boxed{\text{FMR}\{[3]\}}$ $\}$

Recursive Find-Max (FMR)

Find-Max-R $A[1 \,..\, n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1 \,..\, n{-}1]$
3. return Max $\{A[n], M1\}$

If A=[3, 1, 5, 7] }

FMR{[3,1,5,7]}

Max{7, FMR{[3,1,5]} }

Max{5, FMR{[3,1]} }

Max{1, 3 } = 3

Recursive Find-Max (FMR)

Find-Max-R $A[1\ ..\ n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1\ ..\ n-1]$
3. return Max $\{A[n], M1\}$

If A=[3, 1, 5, 7] }

$\boxed{\text{FMR}\{[3,1,5,7]\}}$

Max$\{7,$ $\boxed{\text{FMR}\{[3,1,5]\}}$ $\}$

Max$\{5,$ $\boxed{3}$ $\}$ $\boxed{= \mathbf{5}}$

Max$\{1,$ $\boxed{3}$ $\}$ $\boxed{= \mathbf{3}}$

Recursive Find-Max (FMR)

Find-Max-R $A[1\,..\,n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1\,..\,n-1]$
3. return Max $\{A[n], M1\}$

FMR{[3,1,5,7]}

If A=[3, 1, 5, 7] }

Max{7, $\boxed{5}$ } $\boxed{= \mathbf{7}}$

Max{5, $\boxed{3}$ } $\boxed{= \mathbf{5}}$

Max{1, $\boxed{3}$ } $\boxed{= \mathbf{3}}$

Recursive Find-Max (FMR)

Find-Max-R $A[1 \,..\, n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1 \,..\, n{-}1]$
3. return Max $\{A[n], M1\}$

If A=[3, 1, 5, 7] }

$\boxed{7}$

Max$\{7, \boxed{5}\}$ $\boxed{= 7}$

Max$\{5, \boxed{3}\}$ $\boxed{= 5}$

Max$\{1, \boxed{3}\}$ $\boxed{= 3}$

Recursive Find-Max (FMR)

Find-Max-R $A[1 \mathinner{..} n]$

1. If $n = 1$, return $A[1]$
2. $M1$:= Find-Max-R $A[1 \mathinner{..} n{-}1]$
3. return Max $\{A[n], M1\}$

If A=[3, 1, 5, 7] }

$\boxed{7}$

Max $\{7, \boxed{5}\}$ $\boxed{= \mathbf{7}}$

Max $\{5, \boxed{3}\}$ $\boxed{= \mathbf{5}}$

Max $\{1, \boxed{3}\}$ $\boxed{= \mathbf{3}}$

FMR does exactly the ***same computations*** as Find-Max!

CS3230 Lecture 5

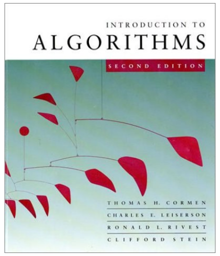
NUS National University of Singapore
School of Computing

"Linear-Time Sorting"

"Order Statistics, and Linear Time OS"

- **Lecture Topics and Readings**
  - Order Statistics, Max, Min-Max [CLRS]-C9.1
  - Randomized Divide-and-Conquer [CLRS]-C9.2
  - Order Statistics in Linear Time [CLRS]-C9.3

Recursive algorithms are elegant!
Balancing leads to efficient algorithms



Order statistics

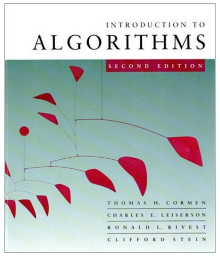
Select the ith smallest of n elements (the element with ***rank*** i).

- $i = 1$: ***minimum***;
- $i = n$: ***maximum***;
- $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: ***median***.

Naive algorithm: Sort and index ith element.

Worst-case running time $= \Theta(n \lg n) + \Theta(1)$
$= \Theta(n \lg n)$,

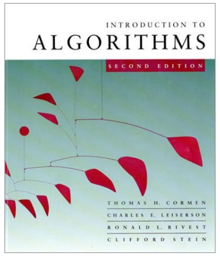
using merge sort or heapsort (*not* quicksort).



Randomized Divide-and-Conquer Algorithm

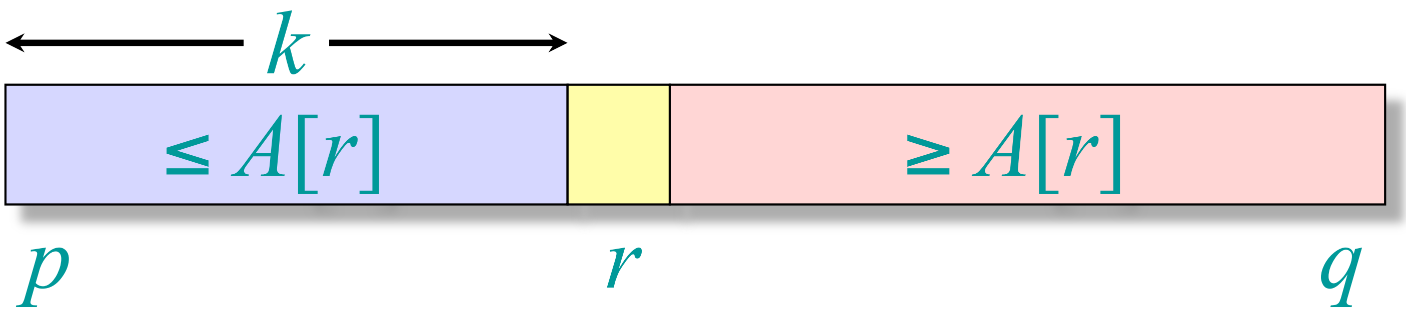
Modified from Quicksort

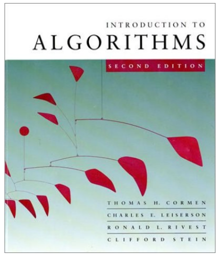
Also by **C. A. R. (Tony) Hoare**, who invented Quicksort.



Randomized divide-and-conquer algorithm

RAND-SELECT(A, p, q, i) ▷ ith smallest of $A[p\,..\,q]$
  if $p = q$ **then return** $A[p]$
  $r \leftarrow$ RAND-PARTITION(A, p, q)
  $k \leftarrow r - p + 1$ ▷ $k = \text{rank}(A[r])$
  if $i = k$ **then return** $A[r]$
  if $i < k$
    then return RAND-SELECT$(A, p, r - 1, i)$
    else return RAND-SELECT$(A, r + 1, q, i - k)$





Example

Select the $i = 7$th smallest:

6	10	13	5	8	3	2	11

$i = 7$

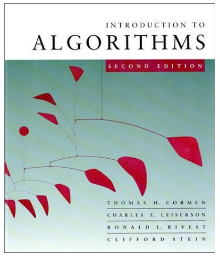
pivot

Partition:

2	5	3	6	8	13	10	11

$k = 4$

Select the $7 - 4 = 3$rd smallest recursively.



Intuition for analysis

(All our analyses today assume that all elements are distinct.)

Lucky:

$$T(n) = T(9n/10) + \Theta(n)$$
$$= \Theta(n)$$

$n^{\log_{(10/9)} 1} = n^0 = 1$

CASE 3

Unlucky:

$$T(n) = T(n-1) + \Theta(n)$$
$$= \Theta(n^2)$$

arithmetic series

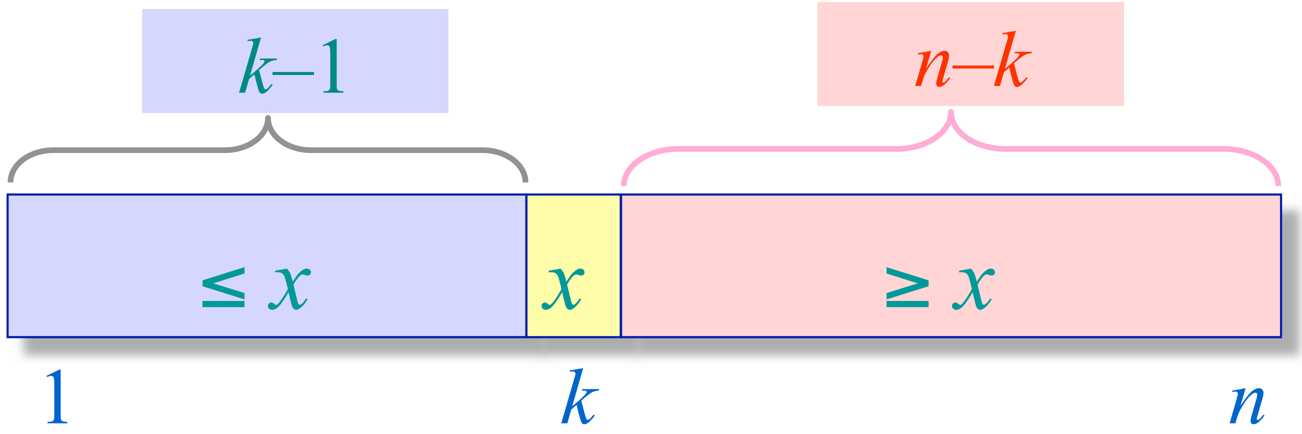
Worse than sorting!

Analysis of RAND-SELECT

Let $T(n)$ = the *expected worst-case* time taken by RAND-SELECT on input of size n.

If pivot x ends up in position k,

then $T(n) = \max\{T(k-1), T(n-k)\} + (n+1)$



Prob(pivot is at pos k) = $1/n$ *for all* k

Analysis of RAND-SELECT

Then, for expected worst-case, we have

$$T(n) = \begin{cases} \max\{T(0), T(n-1)\} + (n+1) & \text{if } 0 : n-1 \text{ split} \\ \max\{T(1), T(n-2)\} + (n+1) & \text{if } 1 : n-2 \text{ split} \\ \max\{T(2), T(n-3)\} + (n+1) & \text{if } 2 : n-3 \text{ split} \\ \vdots & \vdots \\ \max\{T(n-2), T(1)\} + (n+1) & \text{if } n-2 : 1 \text{ split} \\ \max\{T(n-1), T(0)\} + (n+1) & \text{if } n-1 : 0 \text{ split} \end{cases}$$

Prob(pivot is at pos k) = $1/n$ *for all k*

Analysis of RAND-SELECT

Then, we have the following recurrence:

$$T(n) = \sum_{k=1}^{n} \frac{1}{n} \cdot \left[\max\{T(k-1), T(n-k)\} + (n+1)\right]$$

Bigger terms appear twice.

$$T(n) \le \frac{2}{n} \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} T(k) + (n+1)$$

$$T(n) \le \frac{2}{n}\left(T(\lfloor \tfrac{n}{2} \rfloor) + T(\lfloor \tfrac{n}{2} \rfloor + 1) + \ldots + T(n-1)\right) + (n+1)$$

Substitution Method

- We will use "Substitution Method" to prove that $T(n) \leq Cn$, for some C.
- **Idea in Substitution Method:**
  1. Guess the form of the solution;
  2. Use mathematical induction to prove it and find the constants
- **Optional for CS3230 (Spring 2014)**
  - **See [CLRS]-C4.3 pp.83-87 for details**

Using the Substitution Method

$$T(n) \le \frac{2}{n} \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} T(k) + (n+1)$$

Step 1 : Guess $T(n) \le Cn$ for constant $C > 0$.

Step 2: Prove $T(n) \le Cn$ using MI, and find the constant C

Using the Substitution Method

$$T(n) \le \frac{2}{n} \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} T(k) + (n+1)$$

Prove: $T(n) \le Cn$ for constant $C > 0$.

(Use mathematical induction.)

- **Base Case:** The constant C can be chosen large enough so that $T(n) \le Cn$ for the base cases (n very small).

Later, need fact: $\sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} k \le \frac{3}{8} n^2$ (exercise).

Using the Substitution Method

- **Induction Step:**

$$T(n) \le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} Ck + (n+1)$$

Substitute inductive hypothesis.
Namely, $T(k) \le Ck$ for all $k < n$.

Using the Substitution Method

- **Induction Step:**

$$T(n) \le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} Ck + (n+1)$$

$$\le \frac{2C}{n}\left(\frac{3}{8}n^2\right) + (n+1) \qquad \text{(Use fact)}$$

Using the Substitution Method

- **Induction Step:**

$$T(n) \le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} Ck + (n+1)$$

$$\le \frac{2C}{n}\left(\frac{3}{8}n^2\right) + (n+1)$$

$$= Cn - \left(\frac{Cn}{4} - (n+1)\right)$$

Express as ***desired*** – ***residual***.

Using the Substitution Method

- **Induction Step:**

$$T(n) \le \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} Ck + (n+1)$$

$$\le \frac{2C}{n}\left(\frac{3}{8}n^2\right) + (n+1)$$

$$= Cn - \left(\frac{Cn}{4} - (n+1)\right)$$

$$\le Cn$$ (end of induction proof)

When $C=5$, then
$(Cn/4 - (n+1))$
$= (n/4 - 1) \ge 0$ for $n \ge 4$.

Choose $C=5$, $n_0 = 4$,
then residual term ≥ 0 for $n > n_0$.

Hon Wai Leong, NUS

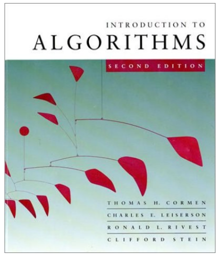
Summary of randomized order-statistic selection

- Works fast: linear expected time.
- Excellent algorithm in practice.
- But, the worst case is ***very*** bad: $\Theta(n^2)$.

Q. Is there an algorithm that runs in linear time in the worst case?

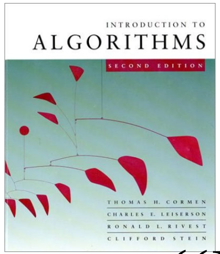
A. Yes, due to Blum, Floyd, Pratt, Rivest, and Tarjan [1973].

IDEA: Generate a good pivot recursively.



Worst-case Linear-Time Order Statistic Algorithm

M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, R. E. Tarjan,
"Time Bounds for Selection," Journal of Computer and System Sciences, (Aug 1973), **7** (4): 448–461. doi:10.1016/S0022-0000(73)80033-9



Why is CS3230 FUN?

- "Meet" many CS celebrities

(1972)

(1974)

(1978)

(1980)

(1982)

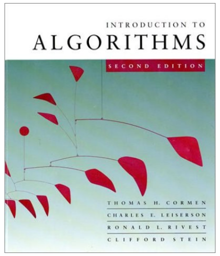
(1985)

(1986)

(1986)

(1995)

(2002)



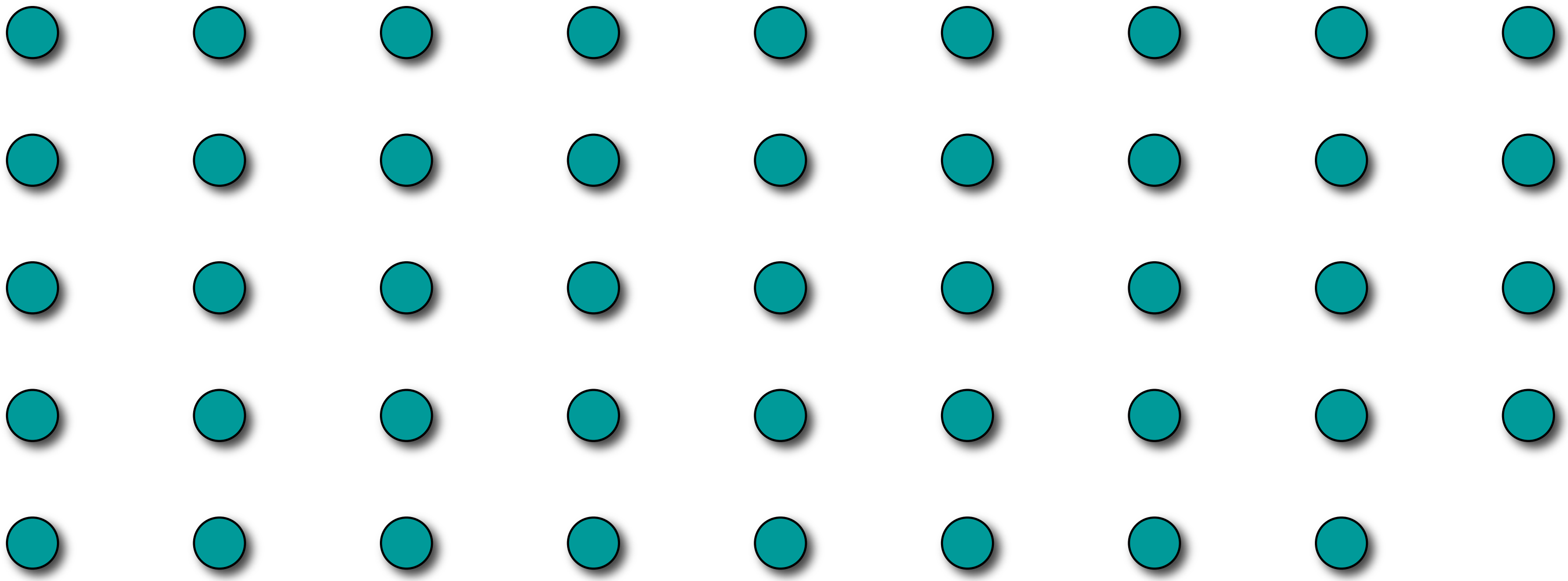
Worst-case linear-time order statistics

SELECT(i, n)

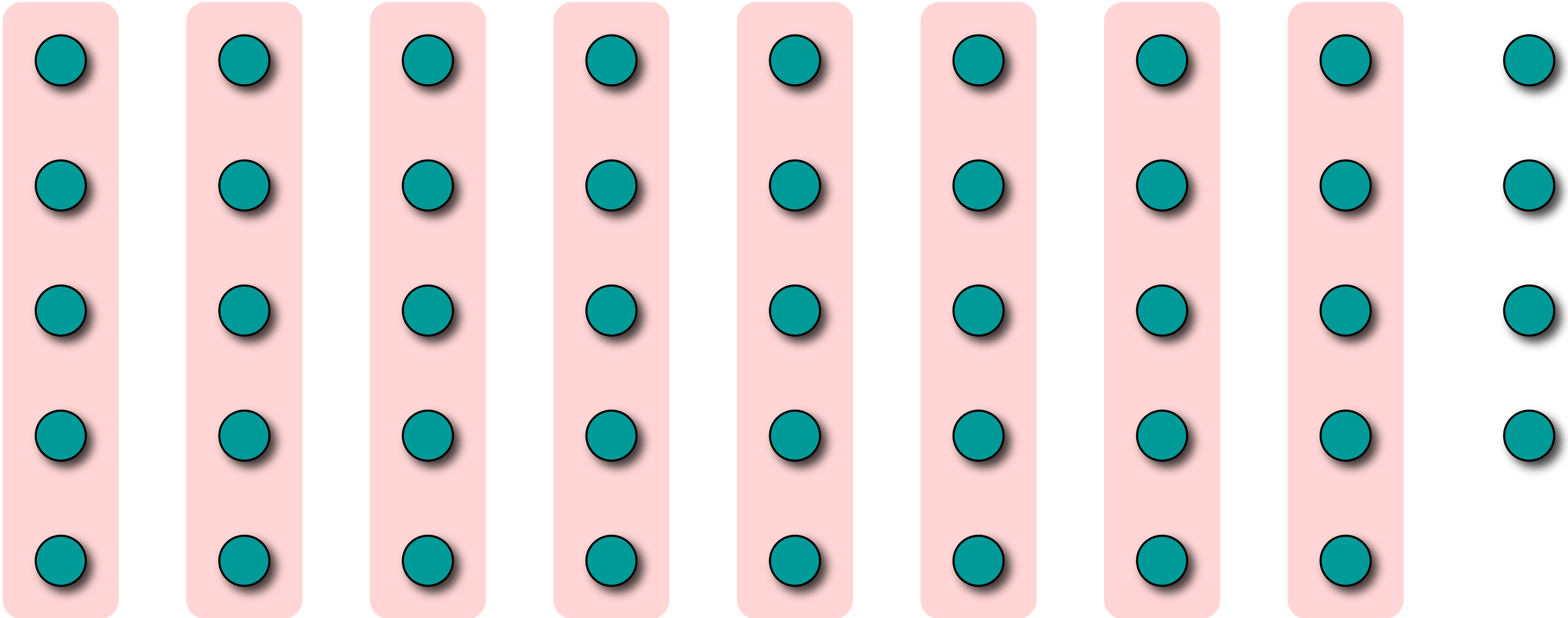
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
3. Partition around the pivot x. Let $k = \text{rank}(x)$.
4. **if** $i = k$ **then return** x
   elseif $i < k$
   then recursively SELECT the ith smallest element in the lower part
   else recursively SELECT the $(i–k)$th smallest element in the upper part

Same as RAND-SELECT

Choosing the pivot

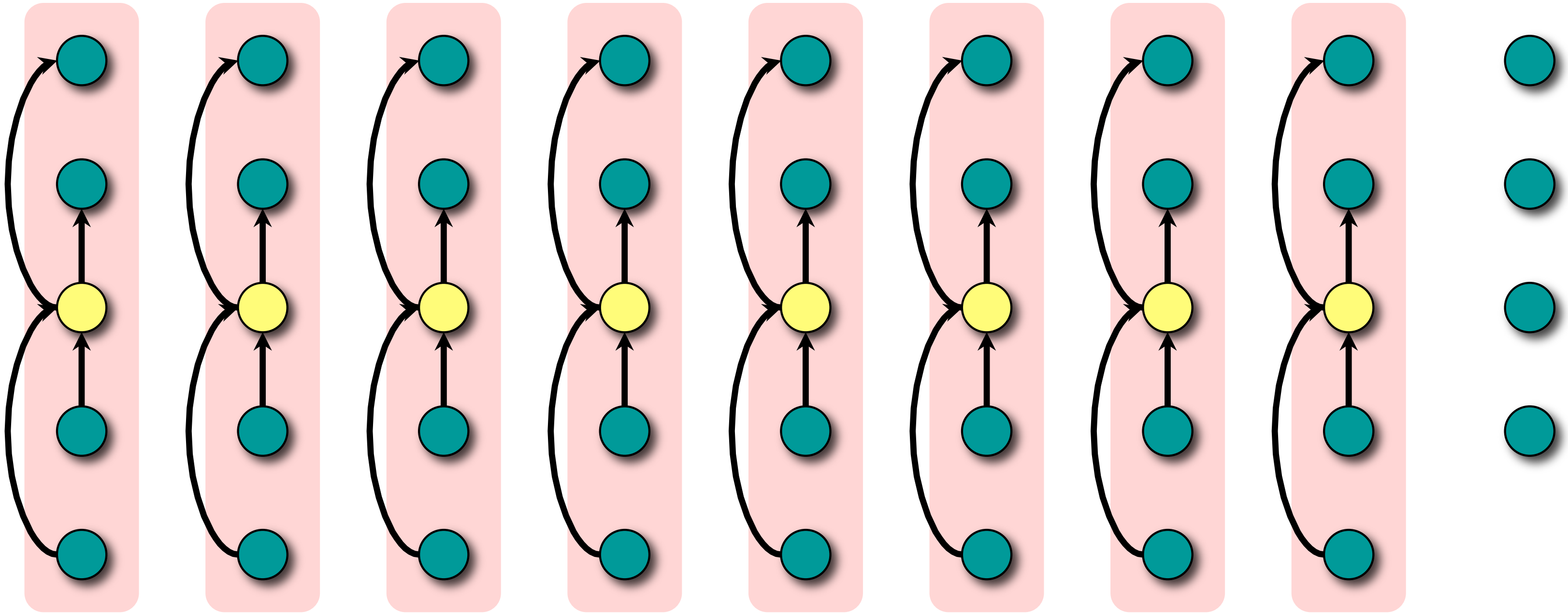


Choosing the pivot

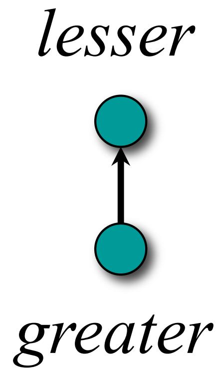


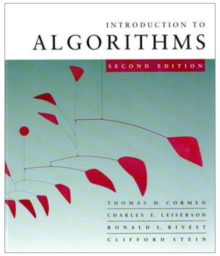
1. Divide the n elements into groups of 5.

Choosing the pivot

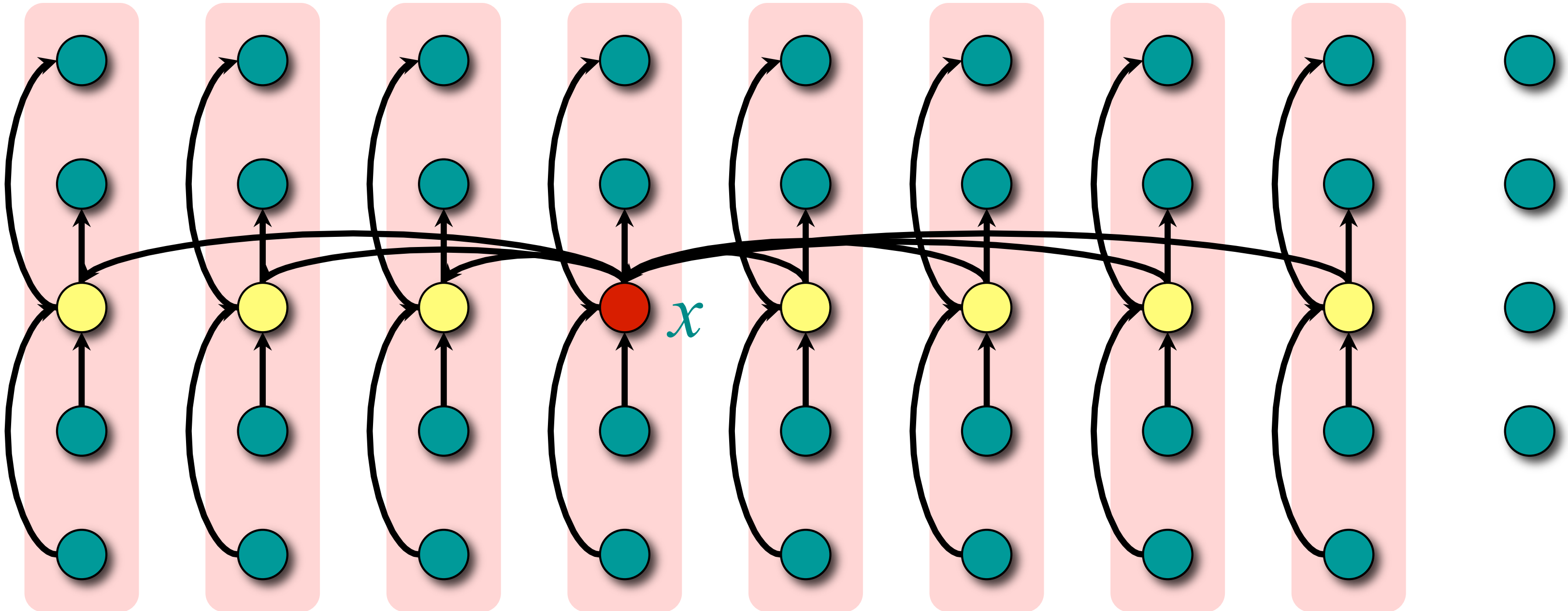


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

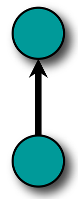


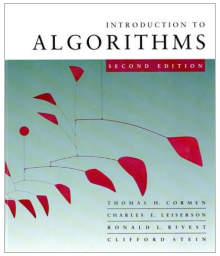


Choosing the pivot

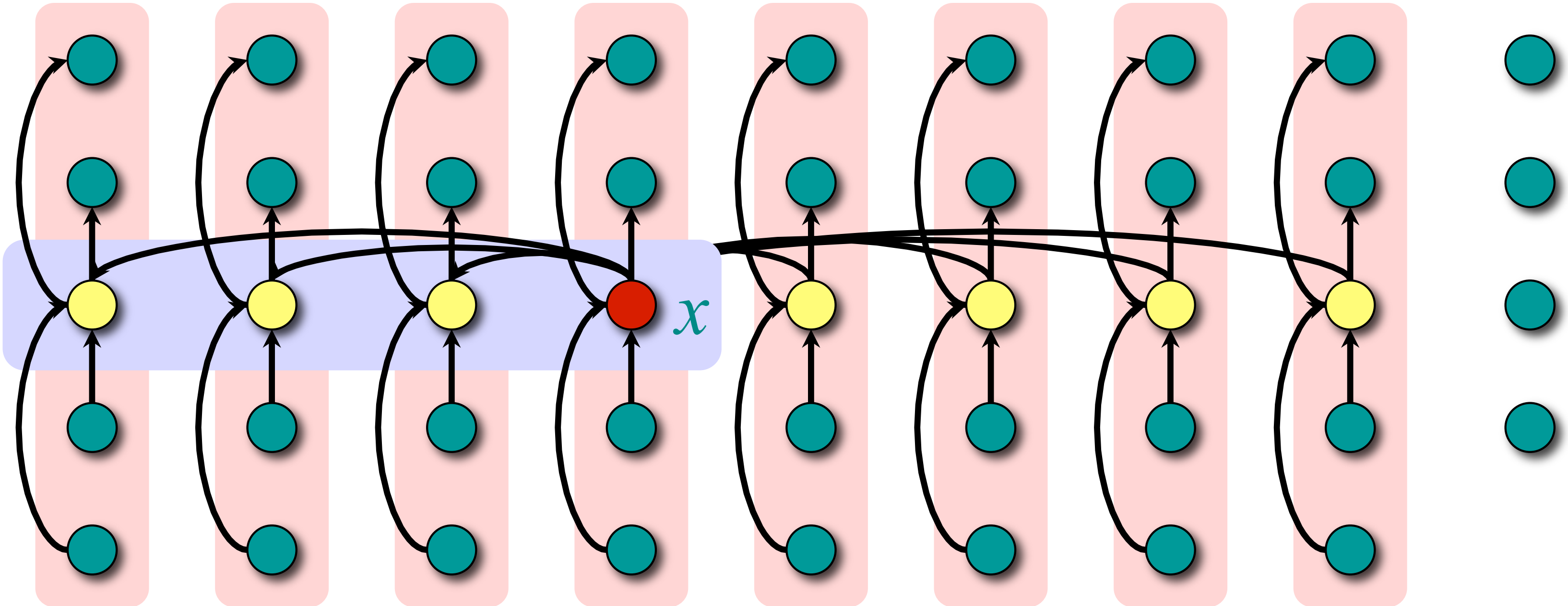


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

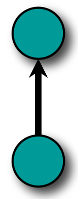




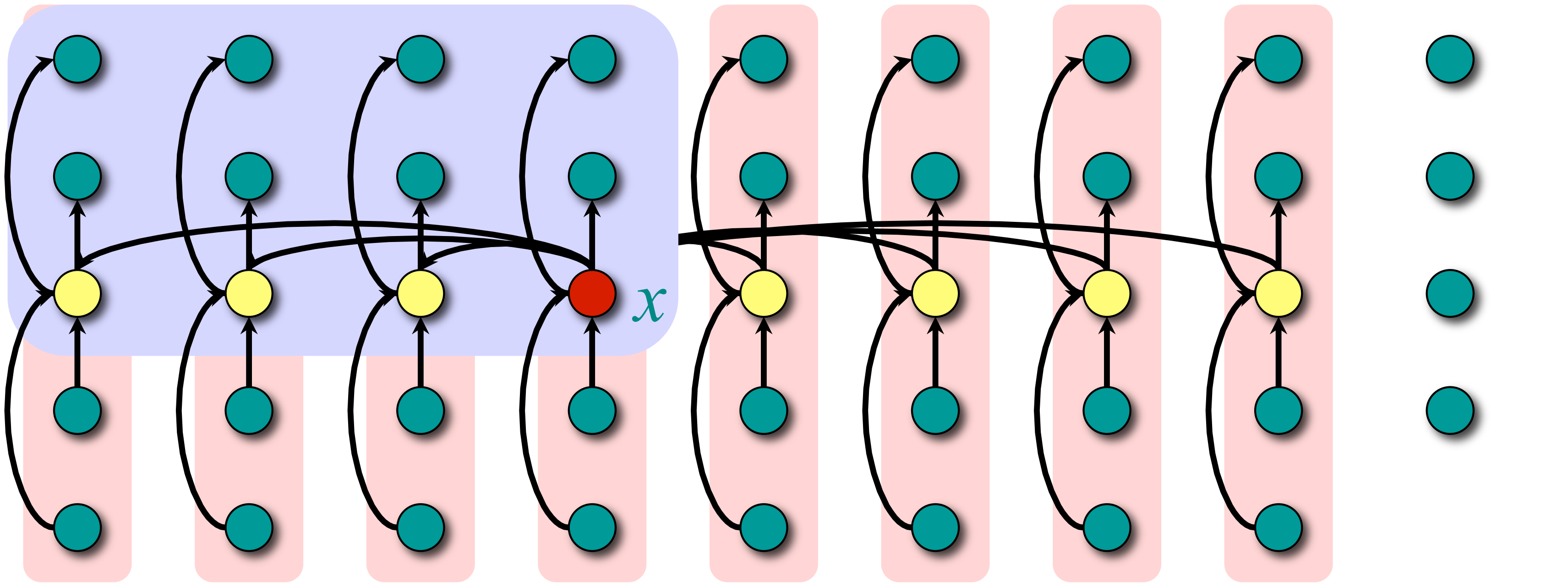
Analysis



At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

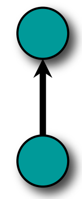


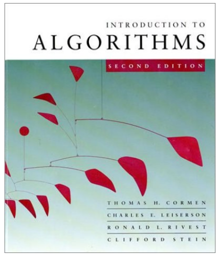
Analysis (Assume all elements are distinct.)



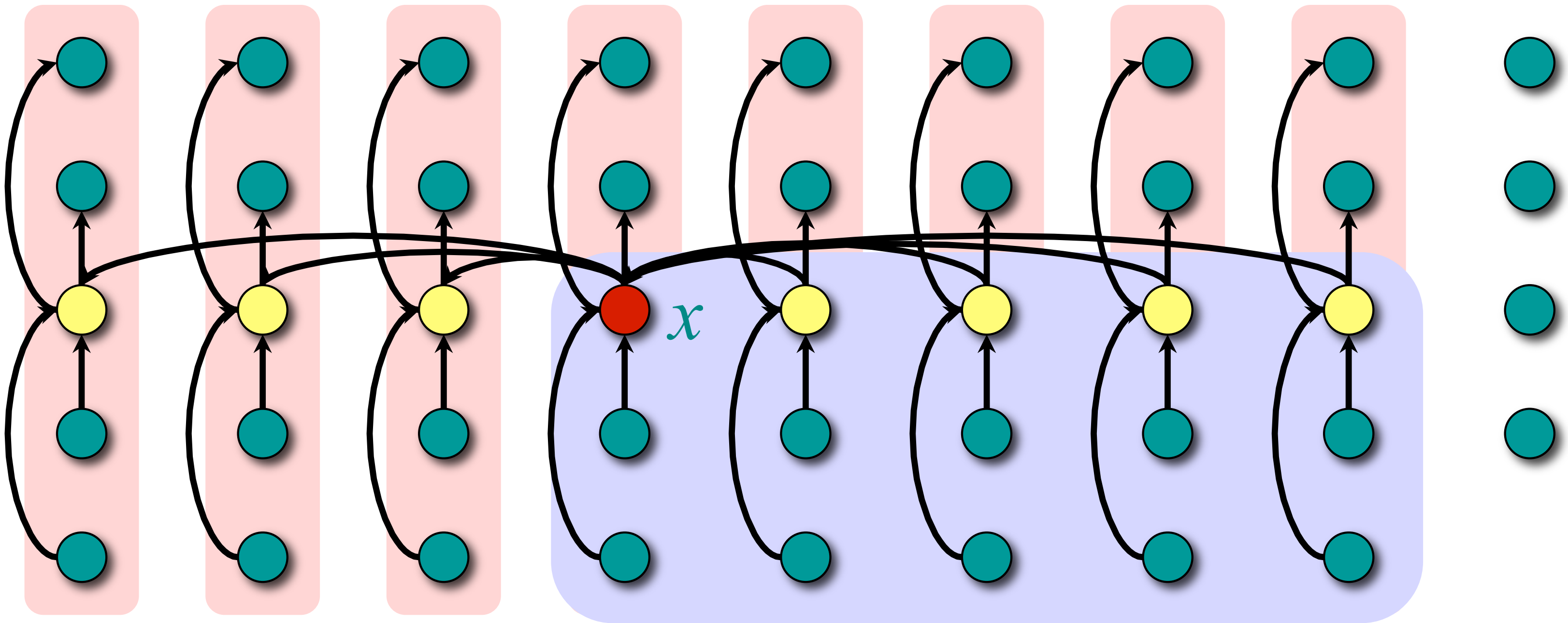
At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- Therefore, at least $3 \lfloor n/10 \rfloor$ elements are $\leq x$.



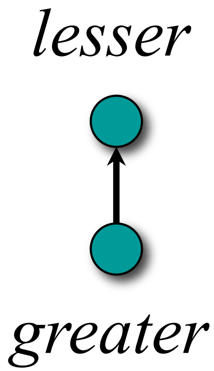


Analysis (Assume all elements are distinct.)



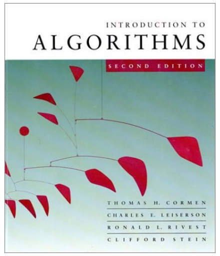
At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- Therefore, at least $3 \lfloor n/10 \rfloor$ elements are $\leq x$.
- Similarly, at least $3 \lfloor n/10 \rfloor$ elements are $\geq x$.



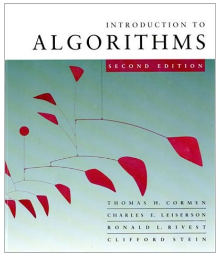
Minor simplification

- For $n \geq 50$, we have $3\lfloor n/10 \rfloor \geq n/4$.
- Therefore, for $n \geq 50$ the recursive call to SELECT in Step 4 is executed recursively on $\leq 3n/4$ elements.
- Thus, the recurrence for running time can assume that Step 4 takes time $T(3n/4)$ in the worst case.
- For $n < 50$, we know that the worst-case time is $T(n) = \Theta(1)$.



Developing the recurrence

Cost	Step
$T(n)$	SELECT(i, n)
$\Theta(n)$	1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
$T(n/5)$	2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
$\Theta(n)$	3. Partition around the pivot x. Let $k = \mathrm{rank}(x)$.
$T(3n/4)$	4. **if** $i = k$ **then return** x **elseif** $i < k$ **then** recursively SELECT the ith smallest element in the lower part **else** recursively SELECT the $(i–k)$th smallest element in the upper part



Solving the recurrence

$$T(n) = T\left(\frac{1}{5}n\right) + T\left(\frac{3}{4}n\right) + \Theta(n)$$

Substitution:

$T(n) \le cn$

$$\begin{aligned} T(n) &\le \frac{1}{5}cn + \frac{3}{4}cn + \Theta(n) \\ &= \frac{19}{20}cn + \Theta(n) \\ &= cn - \left(\frac{1}{20}cn - \Theta(n)\right) \\ &\le cn \quad , \end{aligned}$$

if c is chosen large enough to handle both the $\Theta(n)$ and the initial conditions.

Conclusions

- Since the work at each level of recursion is a constant fraction (19/20) smaller, the work per level is a geometric series dominated by the linear work at the root.
- In practice, this algorithm runs slowly, because the constant in front of n is large.
- The randomized algorithm is far more practical.

Exercise: *Why not divide into groups of 3?*