ADVANCED ANALYSIS OF ALGORITHMS CPS 5440

OMAR DIB

MEDIAN AND ORDER STATISTICS

ORDER STATISTICS

The $k-th$ order statistic is the $k-th$ smallest element of an array.

1	2	3	4	5	6	7	8	9	10
3	4	13	14	23	27	**41**	54	65	75

$7th$ order statistic

The lower median is the $\left\lfloor \frac{n}{2} \right\rfloor - th$ order statistic $floor()$

The upper median is the $\left\lceil \frac{n}{2} \right\rceil - th$ order statistic $ceil\ ()$

If n is odd, lower and upper median are the same

3	4	13	14	23	27	41	54	65	75

lower median *upper median*

- Selecting $i - th$ ranked item from a collection
  - First: $i = 1$
  - Last: $i = n$
- Median (s): $i = \left\lfloor \frac{n}{2} \right\rfloor, \left\lceil \frac{n}{2} \right\rceil$

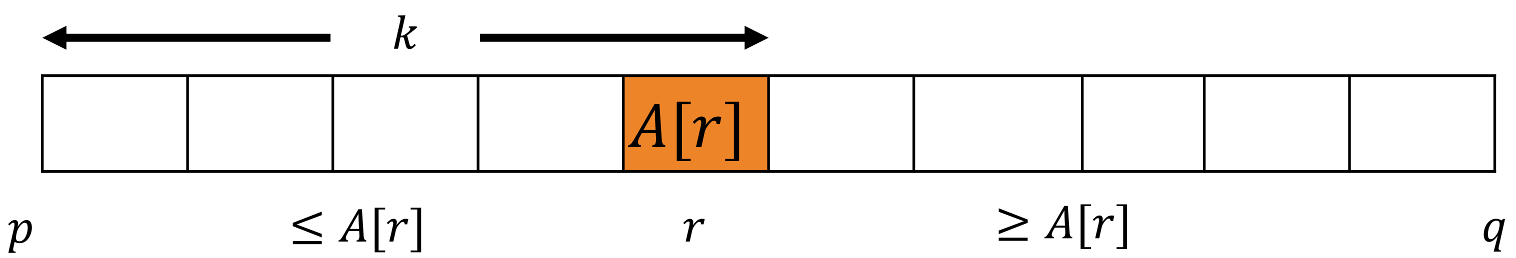
- Assume collection is unordered, otherwise trivial
  - find $i - th$ order stat = $A[i]$
- Can sort first – $\Theta(n \lg n)$, but can do better – $\Theta(n)$
- I can find $\max$ and $\min$ in $\Theta(n)$ time (obvious)
- Can we find any order statistic in linear time? (not obvious!)

- How can we modify Quicksort to obtain expected-case $\Theta(n)$?

- Pivot, partition, but recur only on one set of data. No join.

RANDOMIZED SELECTION

```
RAND-SELECT (A, p, q, i ) // i – th smallest of A[p . . q]
    if p = q then return A [ p ]
    r ← RAND-PARTITION (A, p, q)
    k ← r – p + 1       // k = rank(A[r])
    if i = k then       // i – th smallest element is found
        return A [ r ]
    if i < k then   // search on the left array
        return RAND-SELECT ( A, p, r – 1, i)
    else                // search on the right array
        return RAND-SELECT ( A, r + 1, q, i – k)
```



RAND-SELECT | EXAMPLE

- Select the $i = 6th$ smallest element

```
RAND-SELECT (A, p, q, i ) // i – th smallest of A[p . . q]
        if p = q then return A [ p ]
        r ← RAND-PARTITION (A, p, q)
        k ← r – p + 1    // k = rank(A[r])
        if i = k then       // i – th smallest element is found
                return A [ r ]
        if i < k then   // search on the left array
                return RAND-SELECT ( A, p, r – 1, i)
        else                        // search on the right array
                return RAND-SELECT ( A, r + 1, q, i – k)
```

A	6	10	13	5	8	3	2	11	$i = 6$
	$pivot$								

$partition\ (A, p = 1, q = 8, i = 6)$

A	2	5	3	6	8	13	10	11
	$p = 1$			$r = 4$				$q = 8$

$k = r - p + 1 = 4$

$i > k$ ➔ Search on the right partition ➔ Select the $(i - k) = 6 - 4 = 2 - nd$ smallest in the right

➔ $partition\ (A, r + 1, q, i - k)$

➔ $partition\ (A,\ p = 4 + 1,\ q = 8,\ i = 6 - 4)$

➔ $partition\ (A,\ p = 5,\ q = 8,\ i = 2\ )$

RAND-SELECT | EXAMPLE

- Select the $i = 2th$ smallest in the right $partition\ (A, p = 5, q = 8, i = 2\ )$

```
RAND-SELECT (A, p, q, i ) // i – th smallest of A[p . . q]
        if p = q then return A [ p ]
        r ← RAND-PARTITION (A, p, q)
        k ← r – p + 1    // k = rank(A[r])
        if i = k then      // i – th smallest element is found
                return A [ r ]
        if i < k then    // search on the left array
                return RAND-SELECT ( A, p, r – 1, i)
        else                          // search on the right array
                return RAND-SELECT ( A, r + 1, q, i – k)
```

$p = 5$ $\quad q = 8$

2	5	3	6	8	13	10	11

$p = 5$ $\quad q = 8$

2	5	3	6	8	10	11	13

$r = 7$

$k = 7 - 5 + 1 = 3$

$i < k$ ➔ Search on the left partition ➔ Select the $(i - th) = 2 - nd$ smallest in the left

➔ $partition\ (A,\ p,\ \ r - 1,\ i)$

➔ $partition\ (A,\ p = 5,\ q = 7 - 1,\ i = 2)$

➔ $partition\ (A,\ p = 5,\ q = 6,\ i = 2\ )$

RAND-SELECT | EXAMPLE

- Select the $i\ th = 2th$ smallest $partition\ (A,\ p = 5, q = 6, i = 2\ )$

```
RAND-SELECT (A, p, q, i ) // i − th smallest of A[p . . q]
        if p = q then return A [ p ]
        r ← RAND-PARTITION (A, p, q)
        k ← r − p + 1    // k = rank(A[r])
        if i = k then     // i − th smallest element is found
                return A [ r ]
        if i < k then    // search on the left array
                return RAND-SELECT ( A, p, r − 1, i)
        else                          // search on the right array
                return RAND-SELECT ( A, r + 1, q, i − k)
```

$p = 5\ \ q = 6$

2	5	3	6	8	10	11	13

$p = 5\ \ \ q = 6$

2	5	3	6	8	10	11	13

$r = 6$

$k = r - p + 1 = 2$

$i = k = 2$

if $i = k$ then   // $i - th$ smallest element is found

return A [r]

```
RAND-SELECT (A, p, q, i ) // i -th smallest of A[ p . . q]
    if p = q then return A [ p ]                              O(1)
    r ← RAND-PARTITION (A, p, q )                             O(n)
    k ← r – p + 1 //k = rank(A [ r])                          O(1)
    if i = k then   //ith smallest element is found           O(1)
        return A [ r ]
    if i < k then   // search on the left array               T(problem of smaller size)
        return RAND-SELECT ( A, p, r – 1, i )
    else            // search on the right array              T(problem of smaller size)
        return RAND-SELECT ( A, r + 1, q, i – k )
```

- If the selected pivot leads to two partitions of size $\frac{9}{10}n$ and $\frac{1}{10}n$ and the $i-th$ element is located in the $\frac{9}{10}n$ partition $then$

$$T(n) = T(\frac{9}{10}n) + O(n)$$

How to solve the recursion?

- Master Theorem?
- Substitution?
- Recursion tree?

- Master Theorem

$$T(n) = T(\frac{9}{10}n) + O(n)$$

The master method applies to recurrences of the form

$$T(n) = a\,T(n/b) + f(n)\,,$$

where $a \geq 1, b > 1,$

and f is asymptotically positive.

$$a = 1; \mathrm{b} = \frac{10}{9}; f(n) = n$$

Master theorem can be applied!

RAND-SELECT ANALYSIS

Compare $f(n)$ with $n^{log_b a}$:

Case 1:

$If\ f(n) = O(n^{log_b a - \varepsilon})$ for some constant $\varepsilon > 0.\ Then{:}\ T(n) = \Theta(n^{log_b a})$

Case 2:

$If\ f(n) = \Theta(n^{log_b a}).\ Then{:}\ T(n) = \Theta(n^{log_b a} \lg n)$

Case 3:

$If\ f(n) = \Omega(n^{log_b a + \varepsilon})$ for some constant $\varepsilon > 0.\ and\ a\, f\left(\frac{n}{b}\right) \leq c\, f(n)$ for some constant $c < 1\ \forall\, n.\ Then{:}\ T(n) = \Theta(f(n))$

RAND-SELECT ANALYSIS

$a = 1; \mathrm{b} = \frac{10}{9}; f(n) = n$

$log_b a \; = log_{\frac{10}{9}} 1 = 0$

$f(n) = n = \;\Omega\left(n^{log_b a\,+\,\varepsilon}\right) = \Omega\left(n^{0+\,\varepsilon}\right)$ for $\varepsilon = 1 > \; 0$.

and

$a\,f\left(\frac{n}{b}\right) = 1 * f\left(\frac{9\,n}{10}\right) = \frac{9\,n}{10} \leq \frac{9}{10}\;f(n)$ for $c = \frac{9}{10} < 1\,\forall\,n$.

Then:

$T(n) = \;\Theta\big(f(n)\big) = \Theta(n)$

- We will get linear time for the RandSelect algorithm even if we always look at the longest $\frac{9}{10}n$ partition

But …?

RAND-SELECT ANALYSIS

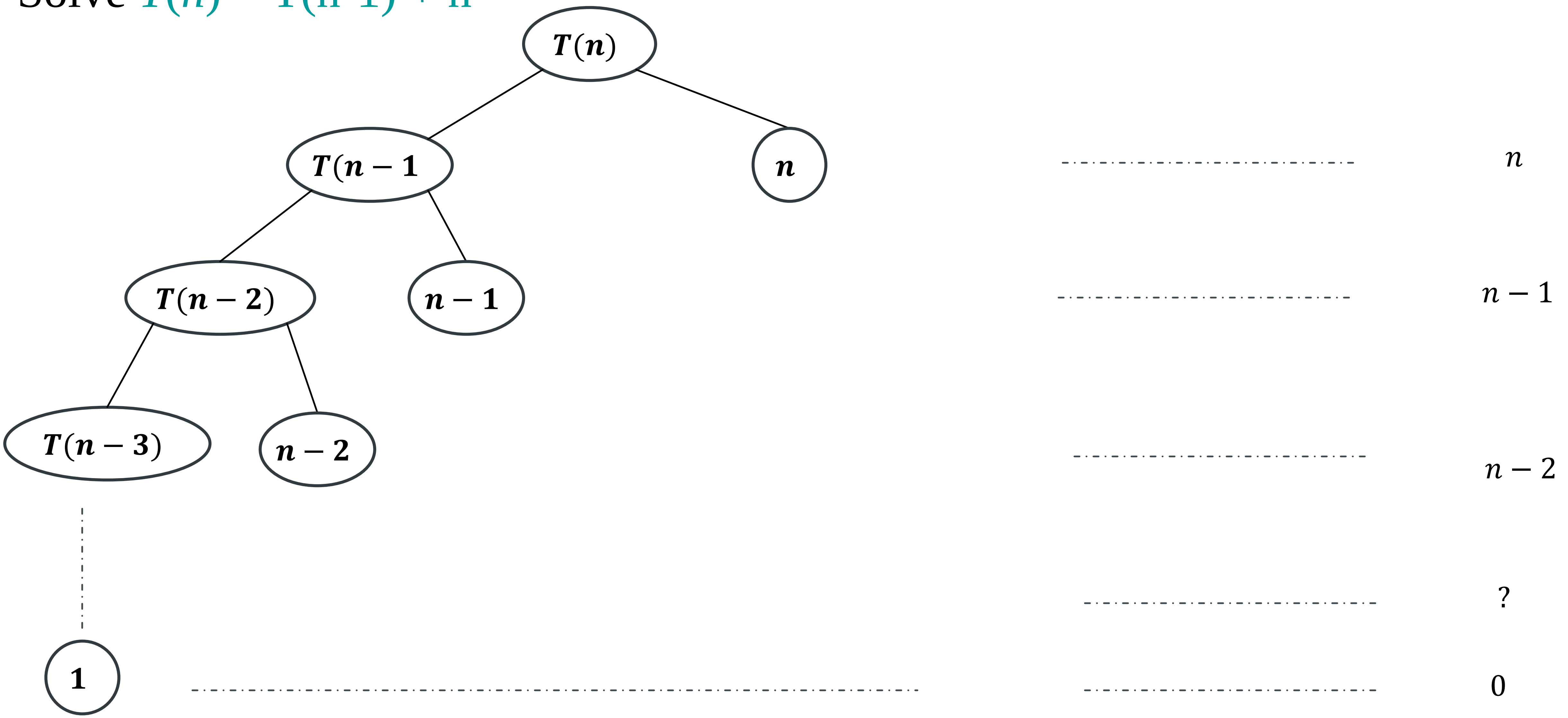
What if we select the worst pivot?
The selected pivot is always the min (or max) or the array is already sorted

- ➔ Worst case: One partition of size 0 and another of size $n-1$

- $T(n) = T(n-1) + O(n)$

Master theorem does not apply?
Recursion tree?
Substitution method?

Solve $T(n) = T(n-1) + n$



RAND-SELECT ANALYSIS

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

Arithmetic series

$$\text{Sum} = n * (\frac{a_1 + a_n}{2})$$
$$= n * (\frac{1+n}{2}) = O(n^2) = \text{Worse than sorting!}$$

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

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

Generate a good pivot recursively. This algorithm has a large constants though and therefore is not efficient in practice!!!!

MEDIAN-OF-MEDIAN ALGORITHM

MEDIAN-OF-MEDIAN ALGORITHMS

```
RAND-SELECT2(A, p, q, i ) // i − th smallest of A[ p . . q]
    if p = q then return A [ p ]
    x ← GOOD_PIVOT (A, p, q )
    r ← PARTITION_WITH_PIVOT(A, p, q, x)
    k ← r – p + 1 //k = rank(A [r])
    if i = k then     // i − th smallest element is found
        return A [ r ]
    if i < k then    // search on the left array
        return RAND-SELECT ( A, p, r – 1, i )
    else                    // search on the right array
        return RAND-SELECT ( A, r + 1, q, i – k )
```

```
RAND-SELECT (A, p, q, i ) // i − th smallest of A[p . . q]
    if p = q then return A [ p ]
    r ← RAND-PARTITION (A, p, q)
    k ← r – p + 1    // k = rank(A[r])
    if i = k then      // i − th smallest element is found
        return A [ r ]
    if i < k then    // search on the left array
        return RAND-SELECT ( A, p, r − 1, i)
    else                    // search on the right array
        return RAND-SELECT ( A, r + 1, q, i − k)
```

```
Good-Pivot(A, p, r)
    B ← array of length ⌈(r − p + 1)/5⌉
    for j ← p to r by 5
        B[1 + (j − p)/5] ← median of A[j.. min(j + 4, r)]
    return RAND-SELECT2(B, 1, B.length, ⌈B.length/2⌉)
```

EXAMPLE

[4, 8, 2, 3, 1, 5, 4, 7, 4, 7, 9, 8, 1, 3, 3, 2, 4, 1, 4, 10, 5, 1, 4, 3, 6, 6, 6, 9, 10, 4, 10, 6, 7, 9, 8], $k = 25$

4	5	9	2	5	6	10
8	4	8	4	1	6	6
2	7	1	1	4	9	7
3	4	3	4	3	10	9
1	7	3	10	6	4	8

EXAMPLE

[4, 8, 2, 3, 1, 5, 4, 7, 4, 7, 9, 8, 1, 3, 3, 2, 4, 1, 4, 10, 5, 1, 4, 3, 6, 6, 6, 9, 10, 4, 10, 6, 7, 9, 8], $k = 25$

1	4	1	1	1	4	6
2	4	3	2	3	6	7
3	5	3	4	4	6	8
4	7	8	4	5	9	9
8	7	9	10	6	10	10

EXAMPLE

[4, 8, 2, 3, 1, 5, 4, 7, 4, 7, 9, 8, 1, 3, 3, 2, 4, 1, 4, 10, 5, 1, 4, 3, 6, 6, 6, 9, 10, 4, 10, 6, 7, 9, 8], $k = 25$

1	4	1	1	1	4	6
2	4	3	2	3	6	7
3	**5**	**3**	**4**	**4**	**6**	**8**
4	7	8	4	5	9	9
8	7	9	10	6	10	10

EXAMPLE

[4, 8, 2, 3, 1, 5, 4, 7, 4, 7, 9, 8, 1, 3, 3, 2, 4, 1, 4, 10, 5, 1, 4, 3, 6, 6, 6, 9, 10, 4, 10, 6, 7, 9, 8], $k = 25$

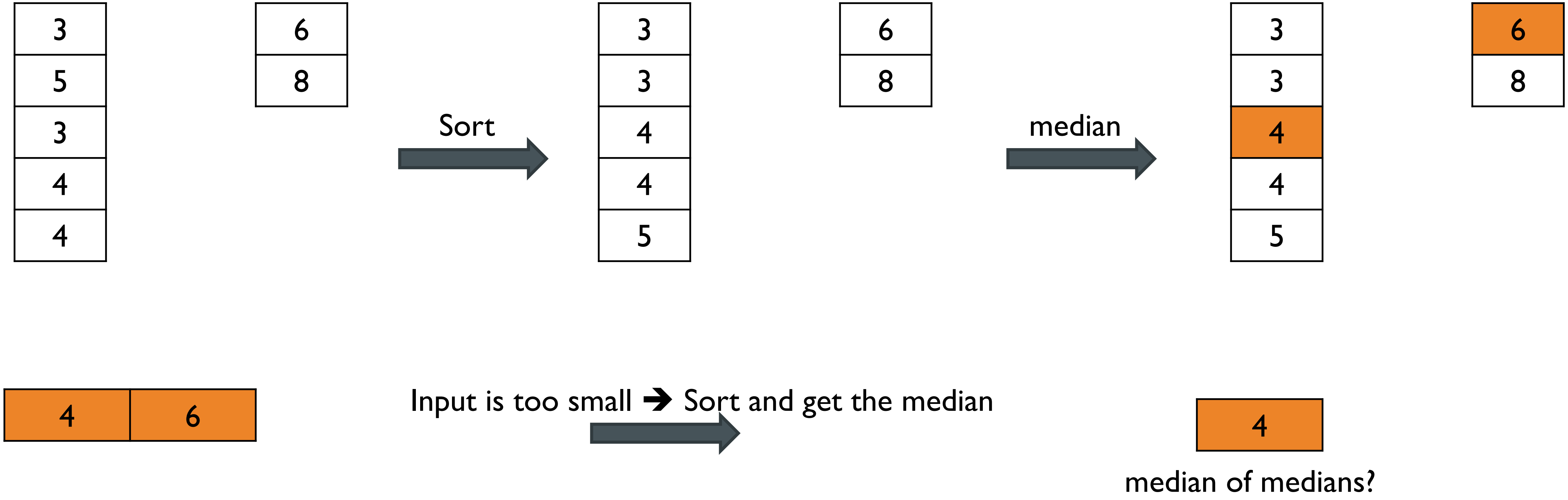
1	4	1	1	1	4	6
2	4	3	2	3	6	7
3	5	3	4	4	6	8
4	7	8	4	5	9	9
8	7	9	10	6	10	10

3	5	3	4	4	6	8

EXAMPLE

What is the median of medians?

3	5	3	4	4	6	8



EXAMPLE

[4, 8, 2, 3, 1, 5, 4, 7, 4, 7, 9, 8, 1, 3, 3, 2, 4, 1, 4, 10, 5, 1, 4, 3, 6, 6, 6, 9, 10, 4, 10, 6, 7, 9, 8], $k = 25$

$\boxed{4}$

median of medians?

Left: 17

4	2	3	1	4	4	1	3	3	2	4	1	4	1	4	3	4

≤ 4

Right: 18

8	5	7	7	9	8	10	5	6	6	6	9	10	10	6	7	9	8

> 4

$K = 25 \geq 17$ ➔ we must search on the right side.
The new index we are looking for is: $25 - 17 = 8$ ➔ 8^{th} smallest element in the Right array

EXAMPLE

8	5	7	7	9	8	10	5	6	6	6	9	10	10	6	7	9	8

$k = 8$

8	8	6	7
5	10	9	9
7	5	10	8
7	6	10	
9	6	6	

Search for a good pivot ➔ median of medians

EXAMPLE

8	5	7	7	9	8	10	5	6	6	6	9	10	10	6	7	9	8

$k = 8$

5
7
7
8
9

5
6
6
8
10

6
6
9
10
10

7
8
9

Sorting

EXAMPLE

8	5	7	7	9	8	10	5	6	6	6	9	10	10	6	7	9	8

$k = 8$

5	5	6	
7	6	6	7
7	**6**	**9**	**8**
8	8	10	9
9	10	10	

Computing medians

EXAMPLE

8	5	7	7	9	8	10	5	6	6	6	9	10	10	6	7	9	8

$k = 8$

5	5	6	
7	6	6	7
7	6	9	8
8	8	10	9
9	10	10	

7	6	9	8

What is the median of median?

EXAMPLE

8	5	7	7	9	8	10	5	6	6	6	9	10	10	6	7	9	8

$k = 8$

5	5	6	
7	6	6	7
7	6	9	8
8	8	10	9
9	10	10	

6	7	8	9

6	7	8	9

Input is too small ➔ Sort and get Arr[n/2]
Otherwise, recursively apply median of median algorithm on the Arr

EXAMPLE

8	5	7	7	9	8	10	5	6	6	6	9	10	10	6	7	9	8

$k = 8$

7

median of medians?

Left: 9

5	7	7	5	6	6	6	6	7

≤ 7

Right: 9

8	9	8	10	9	10	10	9	8

> 7

$K = 8 \leq 9$ ➔ we must search on the left side.

The new index we are looking for is: 8 ➔ 8^{th} smallest element in the left array

EXAMPLE

5	7	7	5	6	6	6	6	7

$k = 8$

5
7
7
5
6

6
6
6
7

Search for a good pivot ➔ median of medians

EXAMPLE

5	7	7	5	6	6	6	6	7

$k = 8$

5
5
6
7
7

6
6
6
7

Sorting

EXAMPLE

5	7	7	5	6	6	6	6	7

$k = 8$

5
5
6
7
7

6
6
6
7

Computing medians

EXAMPLE

5	7	7	5	6	6	6	6	7

$k = 8$

5	
5	6
6	6
7	6
7	7

6	6

What is the median of median?

EXAMPLE

5	7	7	5	6	6	6	6	7

$k = 8$

6

median of medians?

Left: 6

5	5	6	6	6	6

≤ 6

Right: 3

7	7	7

> 6

$K = 8 \geq 6$ ➔ we must search on the right side.

The new index we are looking for is: 8-6 ➔ 2th smallest element in the right array

EXAMPLE

7	7	7

$k = 2$

The input is too small now, → Sort

The 25th smallest element in the original list is: 7

EXAMPLE

If we were to sort the original array!!

4	8	2	3	1	5	4	7	4	7	9	8	1	3	3	2	4	1	4	10	5	1	4	3	6	6	6	9	10	4	10	6	7	9	7

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	1	1	2	2	3	3	3	3	4	4	4	4	4	4	4	5	5	6	6	6	6	7	7	7	7	8	8	9	9	9	10	10	10

MEDIAN-OF-MEDIAN ALGORITHMS | ANALYSIS

2	18	110	0	3
7	41	110	3	4
99	110	111	115	116
106	112	112	120	129
200	113	113	190	6000

99 is median ➔ 99 > 7, 2

110 is median ➔ 110 > 41, 18

111 is median ➔ 111 is > 110, 110

111 is median of median ➔ 111 > 110, 99

➔111 > 110, 110, 110, 41, 18, 99, 7, 2 ≥ 9/25

MEDIAN-OF-MEDIAN ALGORITHMS | ANALYSIS

2	18	110	0	3
7	41	110	3	4
99	110	111	115	116
106	112	112	120	129
200	113	113	190	6000

- Top left: Every item in this quadrant is strictly less than the median
- Bottom left: These items may be bigger (or smaller!) than the median
- Top right: These items may be bigger (or smaller!) than the median
- Bottom right: Every item in this quadrant is strictly greater than the median

MEDIAN-OF-MEDIAN ALGORITHMS | ANALYSIS

2	18	110	0	3
7	41	110	3	4
99	110	111	115	116
106	112	112	120	129
200	113	113	190	6000

- Every item in the top left is strictly less than our pivot.
- How many items are there as a function of n?
- Each column has 5 items, of which we'll take 3;
- We're taking half of the columns, thus: $((n/5)/2) \times 3 = (3/10)\, n$ elements

Therefore, at minimum, at each step, we will remove 30% of the elements.

➔ In the worst case, we will always look at $(7/10)$ elements which is $< (9/10)$

➔ Linear time for the Rand-Select algorithm

MEDIAN-OF-MEDIAN ALGORITHMS | ANALYSIS

```
RAND-SELECT2 (A, p, q, i ) // i -th smallest of A[ p . . q]
    if p = q then return A [ p ]
    x ← GOOD_PIVOT (A, p, q )
    r ← PARTITION_WITH_PIVOT (A, p, q, x )
    k ← r – p + 1 //k = rank(A [ r])
    if i = k then     //ith smallest element is found
        return A [ r ]
    if i < k then   // search on the left array
        return RAND-SELECT ( A, p, r – 1, i )
    else                    // search on the right array
        return RAND-SELECT ( A, r + 1, q, i – k )
```

```
Good-Pivot(A, p, r)
B ← array of length ⌈(r − p + 1)/5⌉
for j ← p to r by 5
    B[1 + (j − p)/5] ← median of A[j.. min(j + 4, r)]
return RAND-SELECT2(B, 1, B.length, ⌈B.length/2⌉)
```

MEDIAN-OF-MEDIAN ALGORITHMS | ANALYSIS

•$\theta(n)$ work to partition the elements

•Solve 1 subproblem $(1/5)$ the size of the original to compute the median of medians
•Solve 1 subproblem $(7/10)$ the size of the original as the recursive step

•This yields the following equation for the total runtime:

$$T(n) = \theta(n) + T(\frac{1}{5}n) + T(\frac{7}{10}n)$$

•It's not straightforward to prove why this is $O(n)$

•The master theorem can not be used to show that this recurrence equals $O(n)$

•Proof by substitution/induction?

MEDIAN-OF-MEDIAN ALGORITHMS | ANALYSIS

$$T(n) \; = \theta(n) + \; T(\tfrac{1}{5}n) \; + \; T(\tfrac{7}{10}n)$$

- It's not straightforward to prove why this is $O(n)$
- To prove $\mathrm{T}(n) = \; O(n)$, we must prove that there exists constants $c \; > \; 0, n_0 \; > \; 0,$
- such that $0 \; \leq T(n) \leq \; c.n \; \forall \; n \; \geq n_0$

$$0 \; \leq \theta(n) + \; T(\tfrac{1}{5}n) \; + \; T(\tfrac{7}{10}n) \leq \; c.n \; \forall \; n \; \geq n_0, \; c \; > \; 0, n_0 \; > \; 0 \; ?$$

- By Induction: Suppose it is true for any $k < n$ ➔ $T(k) \; \leq \; c.k$, then prove it is true for n

- $T(k) \leq c.k$ ➔ $T(\frac{1}{5}n) \leq \mathrm{c}\frac{1}{5}n \; (k = \frac{1}{5}n)$
- $T(k) \leq c.k$ ➔ $T(\frac{7}{10}n) \leq \mathrm{c}\frac{7}{10}n \; (k = \frac{7}{10}n)$
- $T(n) \leq \theta(n) + \mathrm{c}\frac{1}{5}n + \mathrm{c}\frac{7}{10}n$

$$\leq \theta(n) + \frac{9c}{10}n$$

$$\leq \theta(n) + \frac{10c}{10}n - \frac{c}{10}n$$

$$\leq cn - (\frac{c}{10}n \; - \; \theta(n))$$

$\leq cn$ if ($(\frac{c}{10}n \; - \; \theta(n)) \geq 0$), true for some constant $\mathrm{c} > 0$

QUESTIONS/ANSWERS