CS4102 Algorithms

Spring 2020

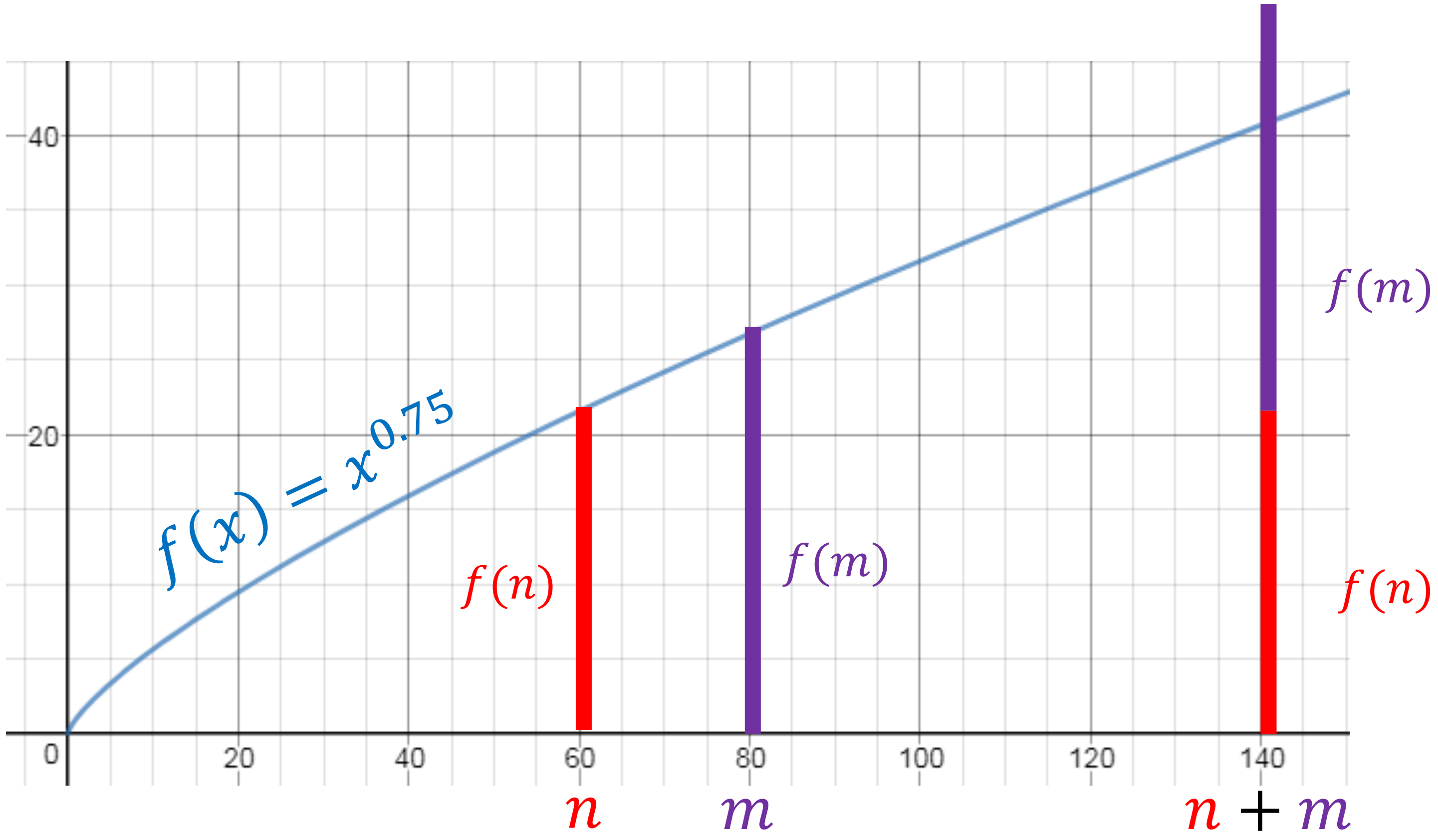
Reminder Warm-Up

Compare $f(n+m)$ with $f(n)+f(m)$

When $f(n) = O(n)$

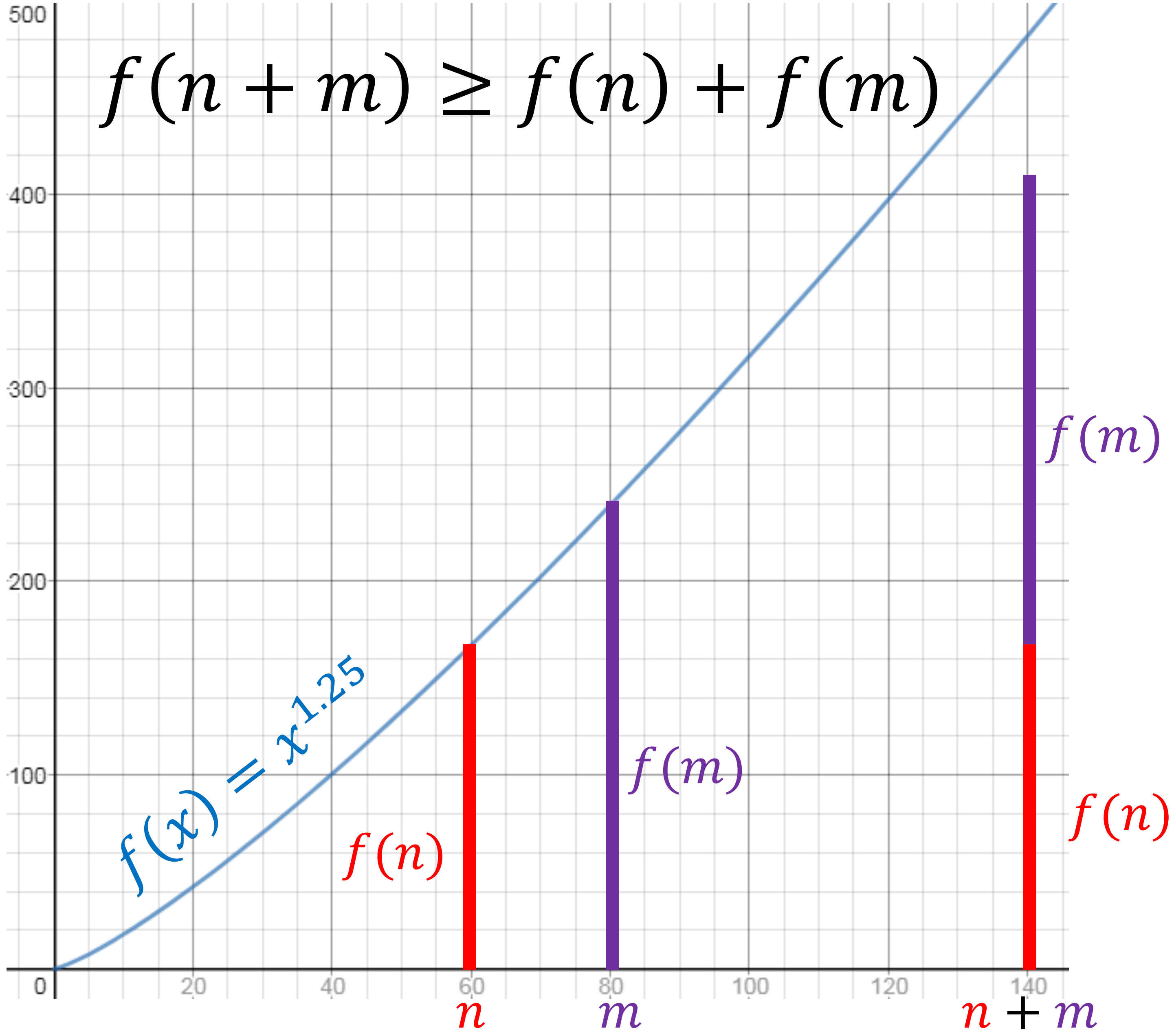
When $f(n) = \Omega(n)$

$f(n) \in O(n)$



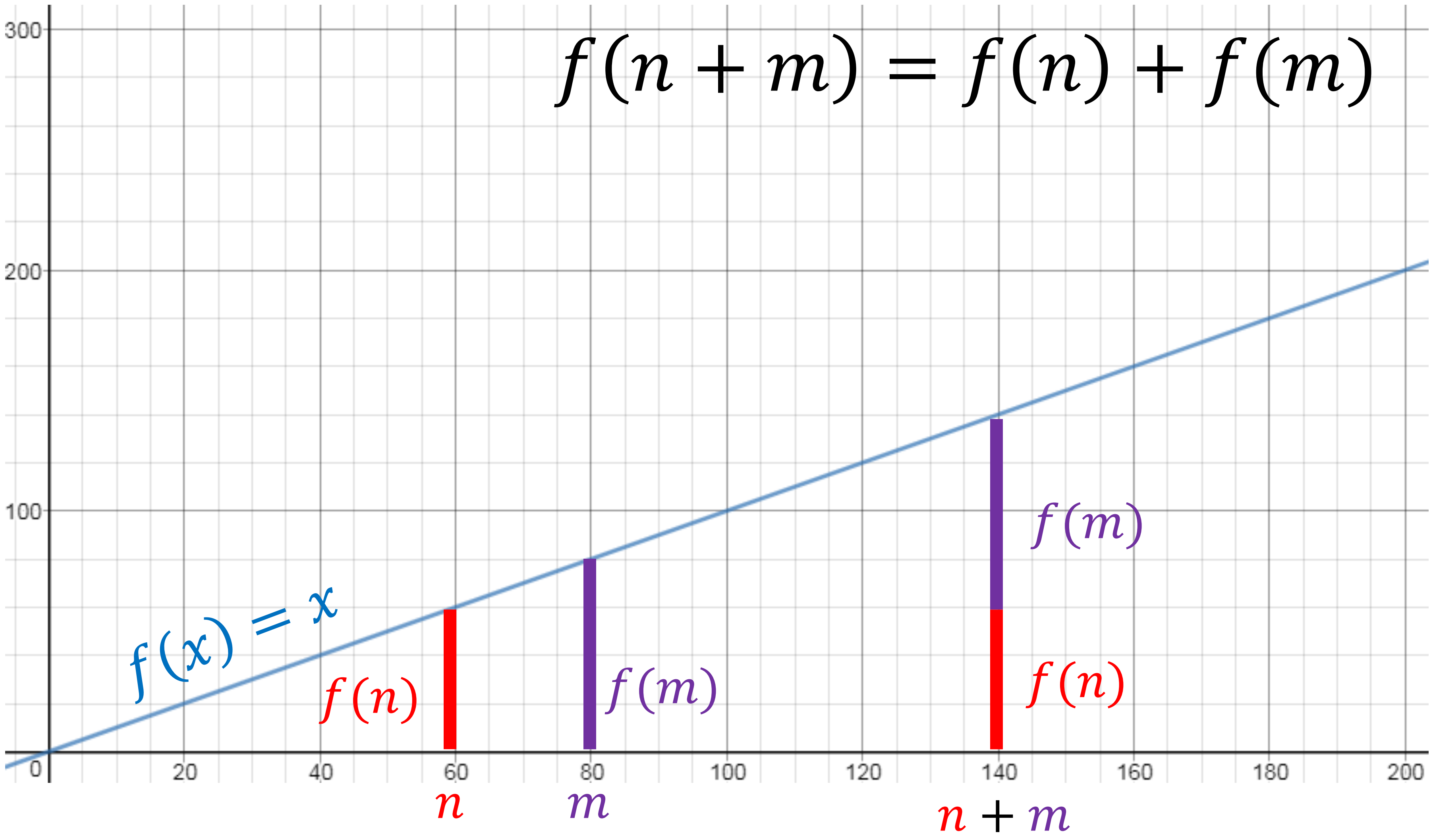
$$f(n+m) \leq f(n) + f(m)$$

$f(n) \in \Omega(n)$



$f(n+m) \geq f(n) + f(m)$
$f(x) = x^{1.25}$
$f(n)$
$f(m)$
$f(m)$
$f(n)$
n
m
$n+m$
0
20
40
60
80
100
120
140
100
200
300
400
500

$f(n) \in \Theta(n)$



$f(n+m) = f(n) + f(m)$
300
200
100
0
20
40
60
80
100
120
140
160
180
200
$f(x) = x$
$f(n)$
$f(m)$
$f(m)$
$f(n)$
n
m
$n + m$

Warm Up

Guess the solution to this recurrence:

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + c \cdot n$$

where $c \geq 1$ is a constant

Warm Up

$$T(n) = T(n/5) + T(7n/10) + c \cdot n$$

$$\frac{n}{5} + \frac{7n}{10} = \frac{9n}{10} < n$$

If this was $T\left(\frac{9n}{10}\right)$, then can use Master's Theorem to conclude $\Theta(n)$

Guess: $\Theta(n)$

Suffices to show $O(n)$ since non-recursive cost is already $\Omega(n)$

Warm Up

$$T(n) = T(n/5) + T(7n/10) + c \cdot n$$

Claim: $T(n) \leq 10cn$

Base Case: $T(0) = 0$

$T(1) = c \leq 10c$ which is true since $c \geq 1$

Strictly speaking, we can handle any $c > 0$, but assuming $c \geq 1$ to simplify the analysis here

Warm Up

$$T(n) = T(n/5) + T(7n/10) + c \cdot n$$

Inductive hypothesis: $\forall n \leq x_0 : T(n) \leq 10cn$

Inductive step:

$$T(x_0 + 1) = T\left(\frac{1}{5}(x_0 + 1)\right) + T\left(\frac{7}{10}(x_0 + 1)\right) + c(x_0 + 1)$$

$$\leq \left(\frac{1}{5} + \frac{7}{10}\right) 10c(x_0 + 1) + c(x_0 + 1)$$

$$= 9c(x_0 + 1) + c(x_0 + 1) = 10c(x_0 + 1)$$

Today's Keywords

- Divide and Conquer
- Sorting
- Quicksort
- Quickselect
- Median of Medians

CLRS Readings

- Chapter 7
- Chapter 9

Homeworks

- HW3 due 11pm tomorrow
  - Programming (use Python or Java!)
  - Divide and conquer
  - Closest pair of points
- HW4 coming soon
  - Written, using LaTeX

Quicksort

Idea: pick a pivot element, recursively sort two sublists around that element

- Divide: select pivot element p, Partition(p)
- Conquer: recursively sort left and right sublists
- Combine: Nothing!

Partition (Divide step)

Given: a list, a pivot p

Start: unordered list

8	5	7	3	12	10	1	2	4	9	6	11

Goal: All elements $< p$ on left, all $> p$ on right

5	7	3	1	2	4	6	8	12	10	9	11

Partition Summary

1. Put p at beginning of list
2. Put a pointer (Begin) just after p, and a pointer (End) at the end of the list
3. While Begin < End:
   1. If Begin value $<\ p$, move Begin right
   2. Else swap Begin value with End value, move End Left
4. If pointers meet at element $< p$: Swap p with pointer position
5. Else If pointers meet at element $> p$: Swap p with value to the left

Run time? $O(n)$

Conquer

2	5	7	3	6	4	1	8	10	9	11	12

All elements $< p$

All elements $> p$

Exactly where it belongs!

Recursively sort Left and Right sublists

Quicksort Run Time (Best)

If the pivot is always the median:

2	5	1	3	6	4	7	8	10	9	11	12

2	1	3	5	6	4	7	8	9	10	11	12

Then we divide in half each time

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

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

Quicksort Run Time (Worst)

If the pivot is always at the extreme:

1	2	2	5	6	4	7	8	10	9	11	12

1	2	3	5	6	4	7	8	10	9	11	12

Then we shorten by 1 each time

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

$$T(n) = O(n^2)$$

How to pick the pivot?

CLRS, Chapter 9

Good Pivot

- What makes a good Pivot?
  - Roughly even split between left and right
  - Ideally: median
- Can we find median in linear time?
  - Yes!
  - Quickselect

Quickselect

- Finds i^{th} order statistic
  - i^{th} smallest element in the list
  - 1^{st} order statistic: minimum
  - n^{th} order statistic: maximum
  - $\frac{n}{2}^{\text{th}}$ order statistic: median
- CLRS, Section 9.1
  - **Selection problem**: Given a list of distinct numbers and value *i*, find value *x* in list that is larger than exactly *i-1* list elements

Quickselect

Idea: pick a pivot element, partition, then recurse on sublist containing index i

- Divide: select an element p, Partition(p)
- Conquer: if $i = $ index of p, done!
  - if $i <$ index of p recurse left. Else recurse right
- Combine: Nothing!

Partition (Divide step)

Given: a list, a pivot value p

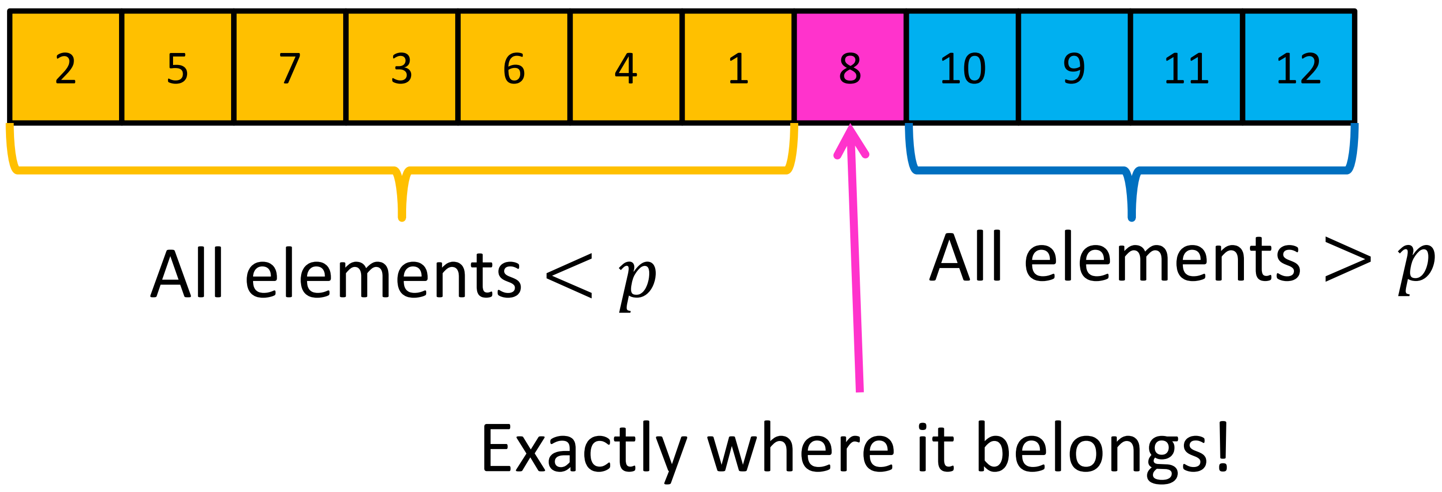
Start: unordered list

8	5	7	3	12	10	1	2	4	9	6	11

Goal: All elements $< p$ on left, all $> p$ on right

5	7	3	1	2	4	6	8	12	10	9	11

Conquer



Recurse on sublist that contains index i

(adjust i accordingly if recursing right)

CLRS Pseudocode

```
RANDOMIZED-SELECT(A, p, r, i)
if p == r
    return A[p]
q = RANDOMIZED-PARTITION(A, p, r)   // q is the position of pivot
k = q − p + 1        // number of points on the left of the pivot
if i == k            // the pivot value is the answer
    return A[q]
elseif i < k
    return RANDOMIZED-SELECT(A, p, q − 1, i)
else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

adjust *i*

Quickselect Run Time

If the pivot is always the median:

2	5	1	3	6	4	7	8	10	9	11	12

2	1	3	5	6	4	7	8	9	10	11	12

Then we divide in half each time

$$S(n) = S\left(\frac{n}{2}\right) + n$$

$$S(n) = O(n)$$

Quickselect Run Time

If the partition is always unbalanced:

1	5	2	3	6	4	7	8	10	9	11	12

1	2	3	5	6	4	7	8	10	9	11	12

Then we shorten by 1 each time

$$S(n) = S(n-1) + n$$

$$S(n) = O(n^2)$$

Good Pivot

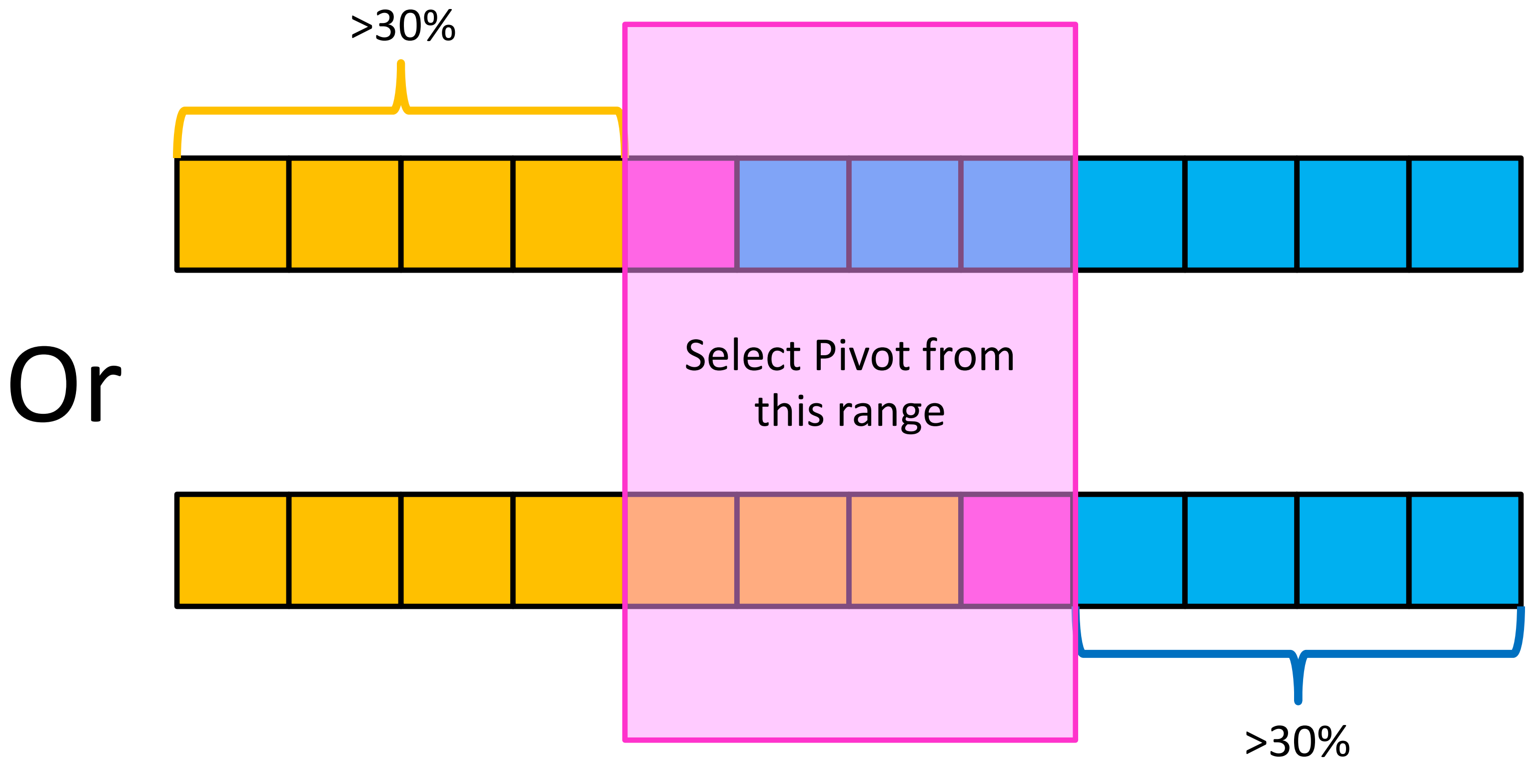
- What makes a good Pivot?
  - Roughly even split between left and right
  - Ideally: median

Déjà vu?

- Here’s what’s next:
  - An algorithm for finding a “rough” split (Median of Medians)
  - This algorithm uses Quickselect as a subroutine

Good Pivot

- What makes a good Pivot?
  - Both sides of Pivot >30%

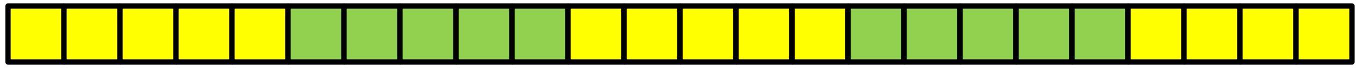


Median of Medians

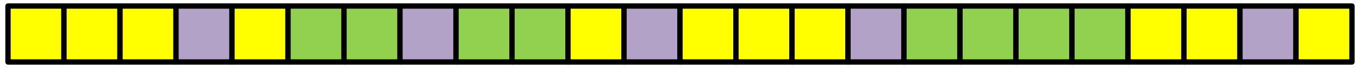
- Fast way to select a "good" pivot
- Guarantees pivot is greater than 30% of elements and less than 30% of the elements
- Idea:
  - break list into chunks,
  - find the median of each chunk,
  - use the median of those medians (with Quickselect)

Median of Medians

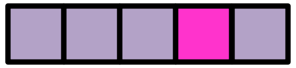
1. Break list into chunks of size 5



2. Find the median of each chunk

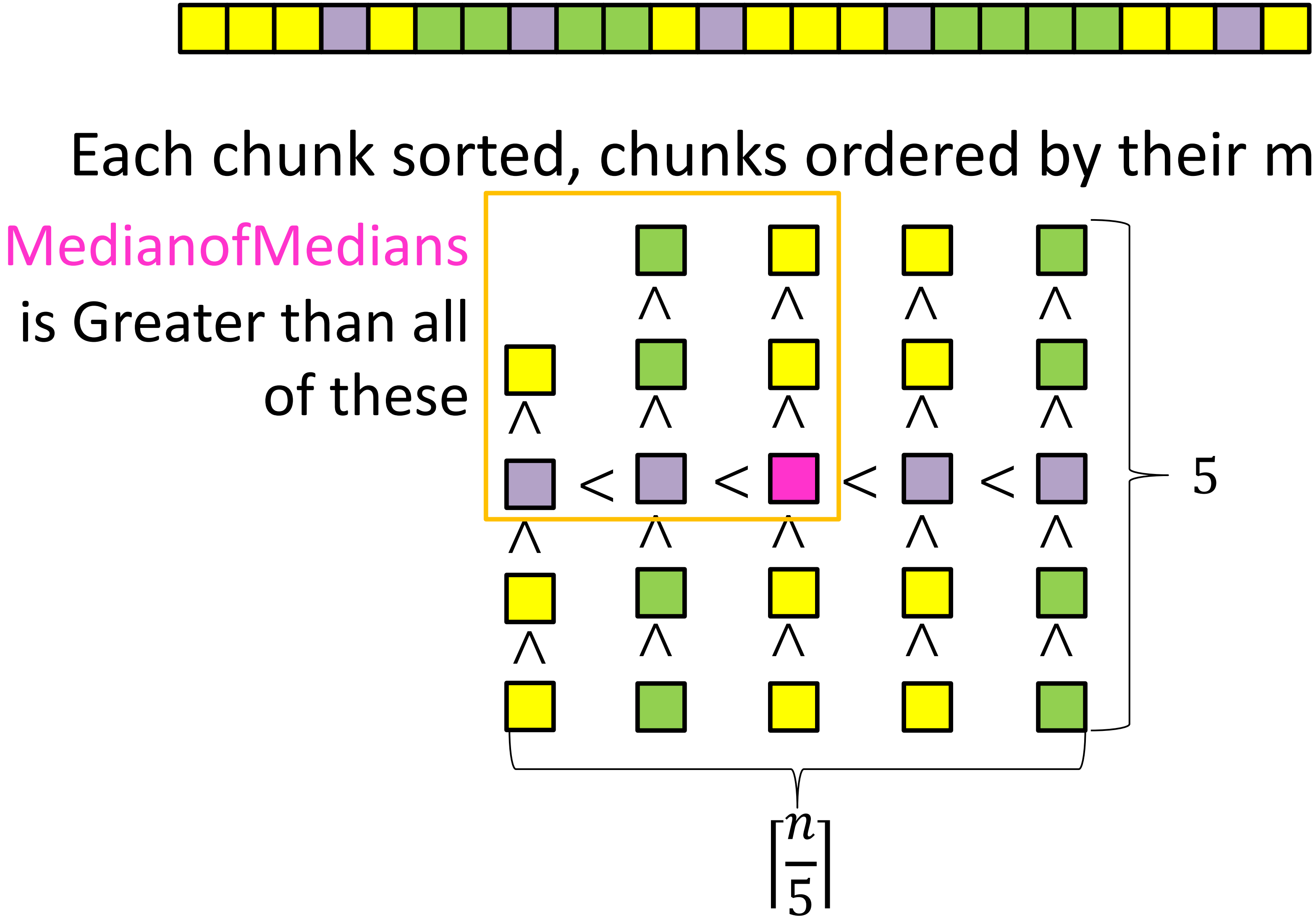


3. Return median of medians (using Quickselect)



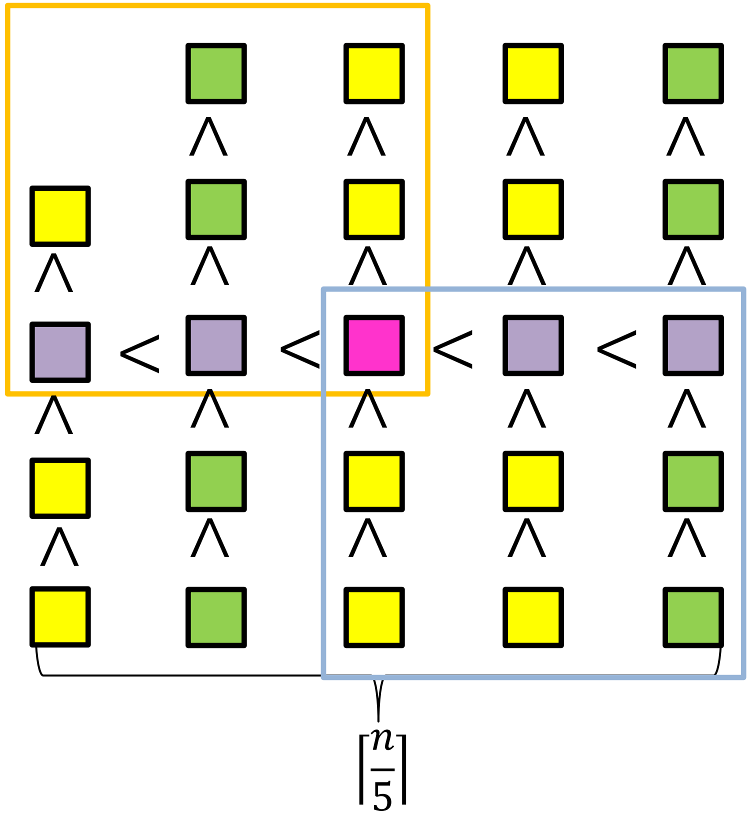
Why is this good?

Each chunk sorted, chunks ordered by their medians



Why is this good?

MedianofMedians is larger than all of these



Larger than 3 things in each (but one) list to the left

$$3\left(\frac{1}{2}\cdot\left\lceil\frac{n}{5}\right\rceil - 2\right) \approx \frac{3n}{10} - 6 \text{ elements } < \blacksquare$$

Similarly:

$$3\left(\frac{1}{2}\cdot\left\lceil\frac{n}{5}\right\rceil - 2\right) \approx \frac{3n}{10} - 6 \text{ elements } > \blacksquare$$

Quickselect

- Divide: select an element p using Median of Medians, Partition(p)

$$M(n) + \Theta(n)$$

- Conquer: if $i =$ index of p, done, if $i <$ index of p recurse left. Else recurse right

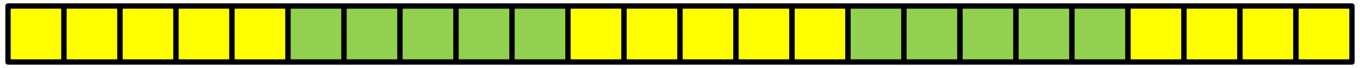
$$\le S\left(\frac{7}{10}n\right)$$

- Combine: Nothing!

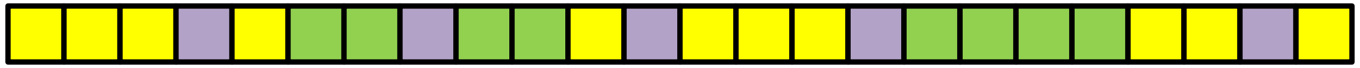
$$S(n) \le S\left(\frac{7}{10}n\right) + M(n) + \Theta(n)$$

Median of Medians, Run Time

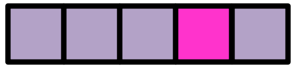
1. Break list into chunks of 5 $\Theta(n)$



2. Find the median of each chunk $\Theta(n)$



3. Return median of medians (using Quickselect)



$S\left(\frac{n}{5}\right)$

$$M(n) = S\left(\frac{n}{5}\right) + \Theta(n)$$

Quickselect

$$S(n) \leq S\left(\frac{7n}{10}\right) + M(n) + \Theta(n)$$

$$M(n) = S\left(\frac{n}{5}\right) + \Theta(n)$$

$$= S\left(\frac{7n}{10}\right) + S\left(\frac{n}{5}\right) + \Theta(n)$$

... Guess and Check ...

Warm Up!

$$S(n) = O(n)$$

$$S(n) = \Omega(n)$$

Linear work done at top level

$$S(n) = \Theta(n)$$

Phew! Back to Quicksort

Using Quickselect, with a median-of-medians partition:

2	5	1	3	6	4	7	8	10	9	11	12

2	1	3	5	6	4	7	8	9	10	11	12

Then we divide in half each time

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

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

Is it worth it?

- Using Quickselect to pick median guarantees $\Theta(n \log n)$ run time
- Approach has very large constants
  - If you really want $\Theta(n \log n)$, better off using MergeSort
- Better approach: Random pivot
  - Very small constant (very fast algorithm)
  - Expected to run in $\Theta(n \log n)$ time
    - Why? Unbalanced partitions are very unlikely