CS4102 Algorithms

Fall 2018

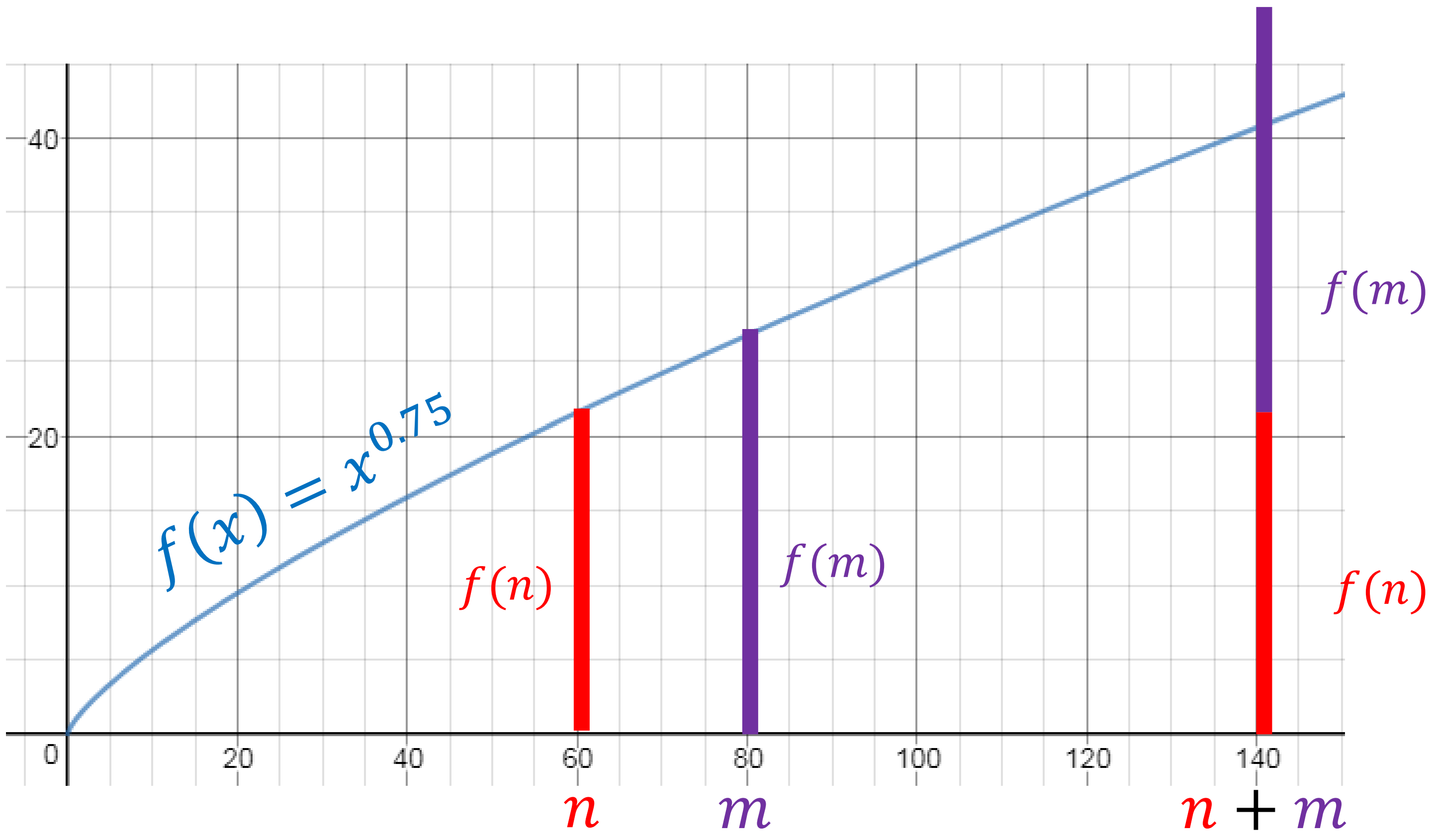
Warm up

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

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

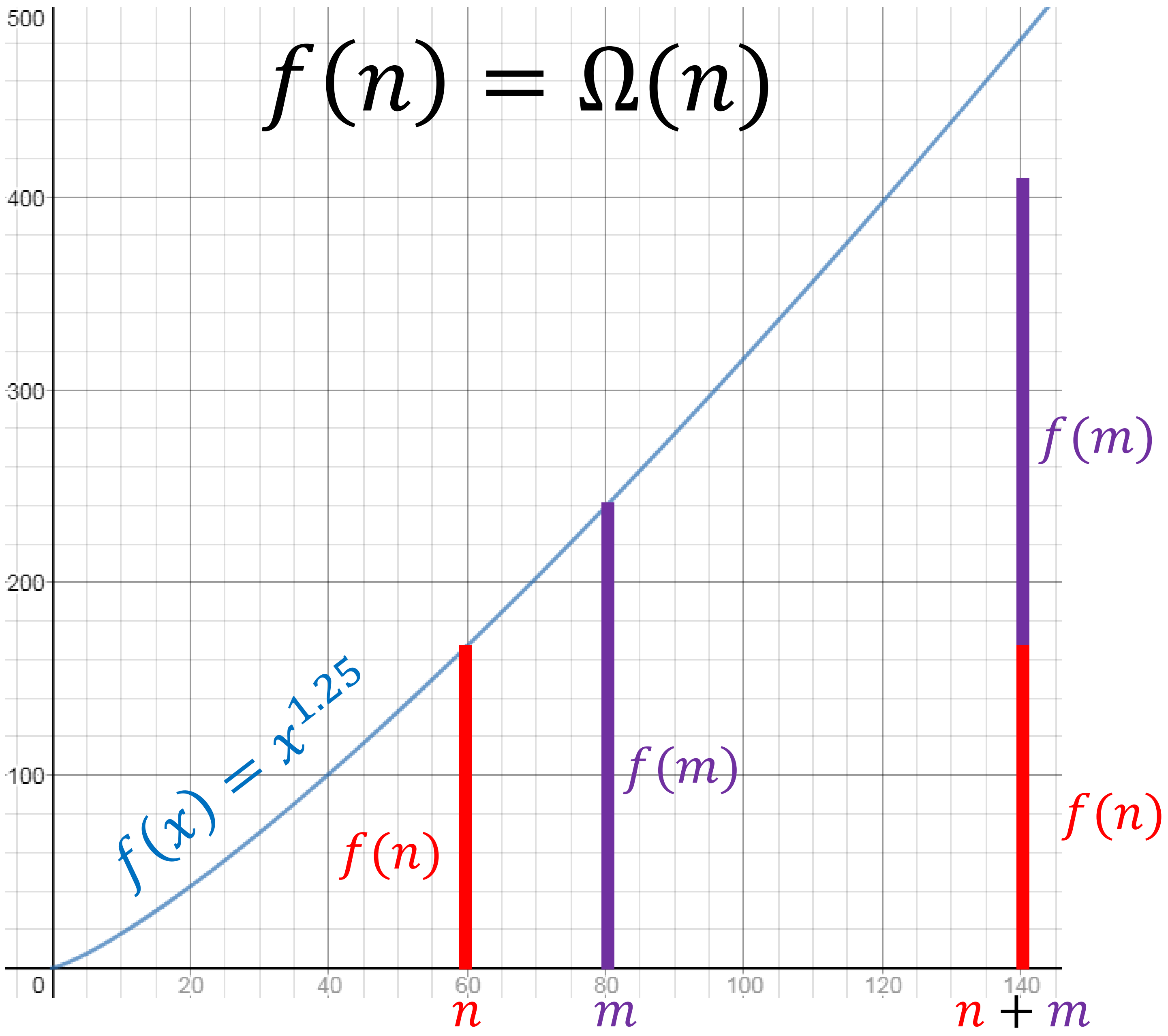
When $f(n) = \Omega(n)$

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



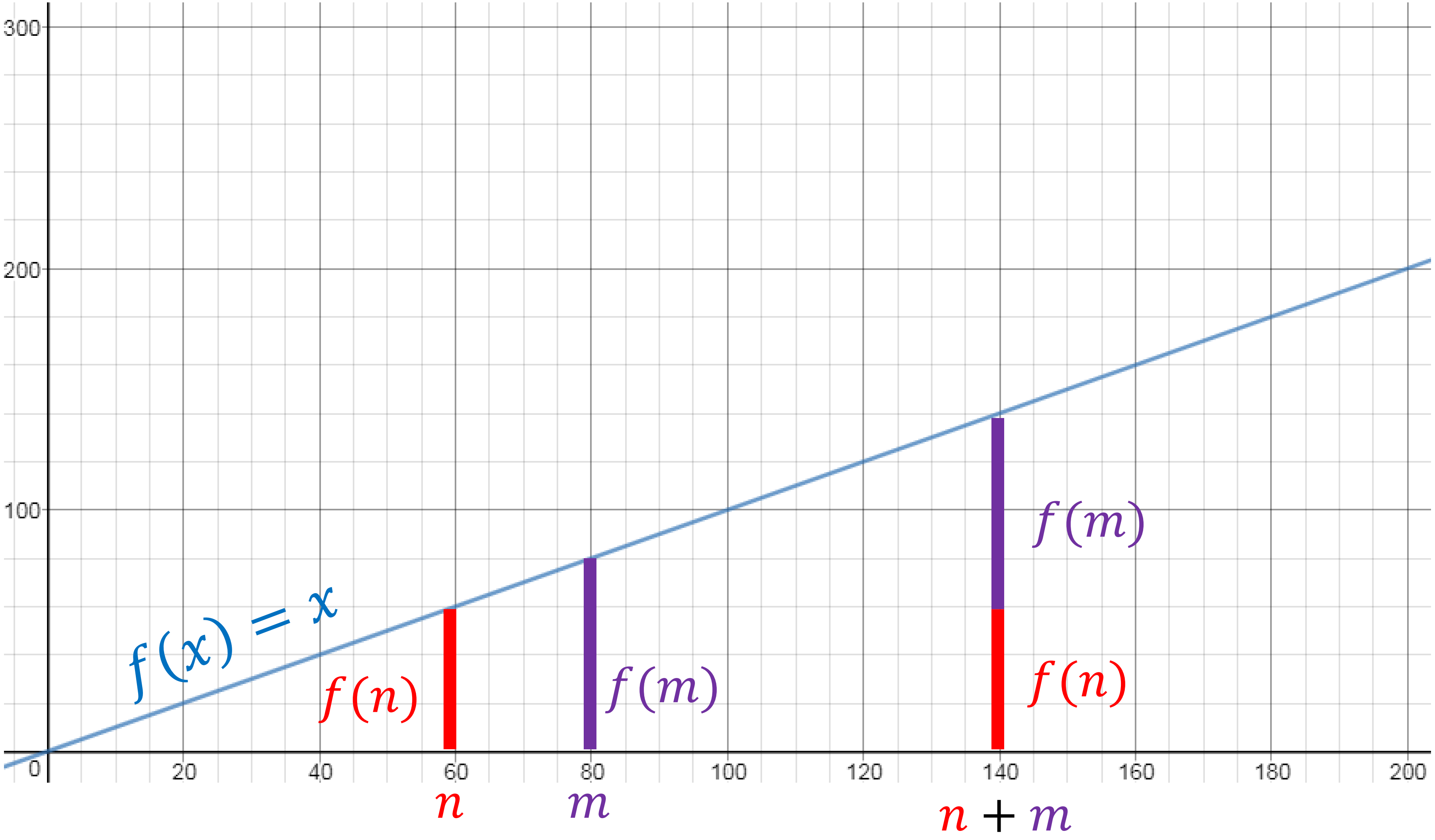
$$f(n+m) \leq f(n) + f(m)$$

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



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

$$f(n) = \Theta(n)$$



$$f(n+m) = f(n) + f(m)$$

Today’s Keywords

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

CLRS Readings

- Chapter 7

Homeworks

- Hw2 due 11pm Friday!
  - Programming (use Python or Java!)
  - Divide and conquer
  - Closest pair of points
- Hw3 released soon
  - Divide and conquer
  - Written (use LaTeX!)

More on HW2

- You must read from garden.txt file automatically (it's a fixed filename)
- That file has a list of pairs of **floats** (not ints)
- You must only output **one** floating point number (minimum distance)
- Uploaded files:
  - One python file, or
  - One or more java files (uploaded individually)
    - Don't use packages!
    - Don't use subdirectories!
  - DO NOT upload a zip file!
- Try it yourself:
  - Put the files you are going to upload in a directory (with a garden.txt file)
    - python closestpair_mst3k.py
    - javac *.java
java ClosestPair
  - Use the one for your language and you should get a result

Quicksort

- Idea: pick a pivot element, recursively sort two sublists around that element
- Divide: select an 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

Quicksort 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

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

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

Quicksort 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

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

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

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

Quickselect

- Finds i^{th} order statistic
- 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

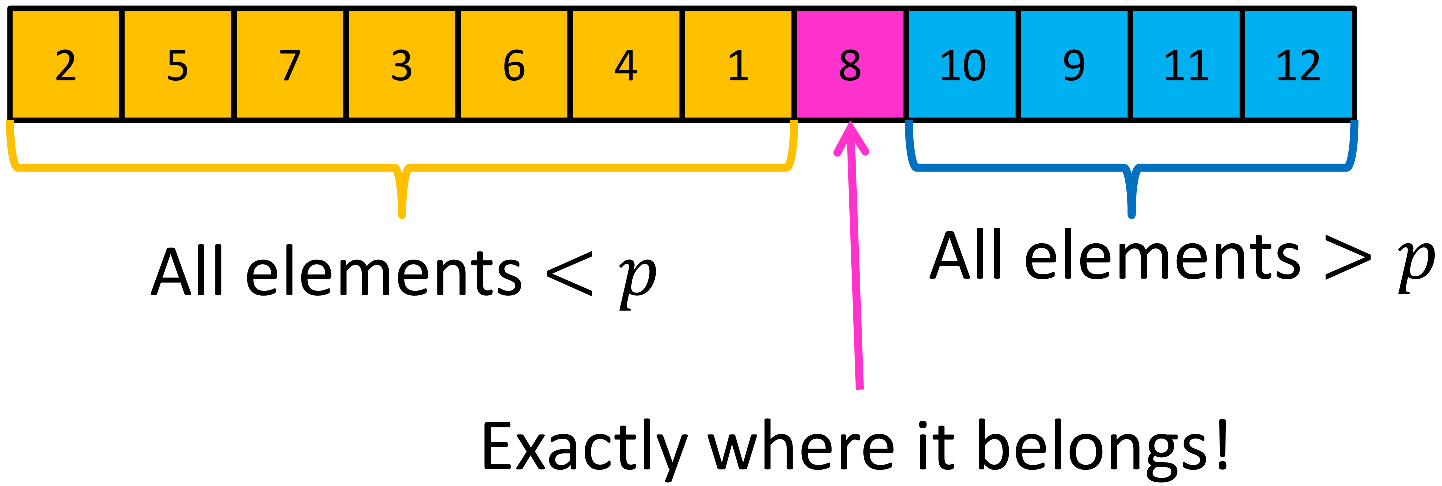
Start: unordered list

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

(add index of the pivot to i if recursing right)

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

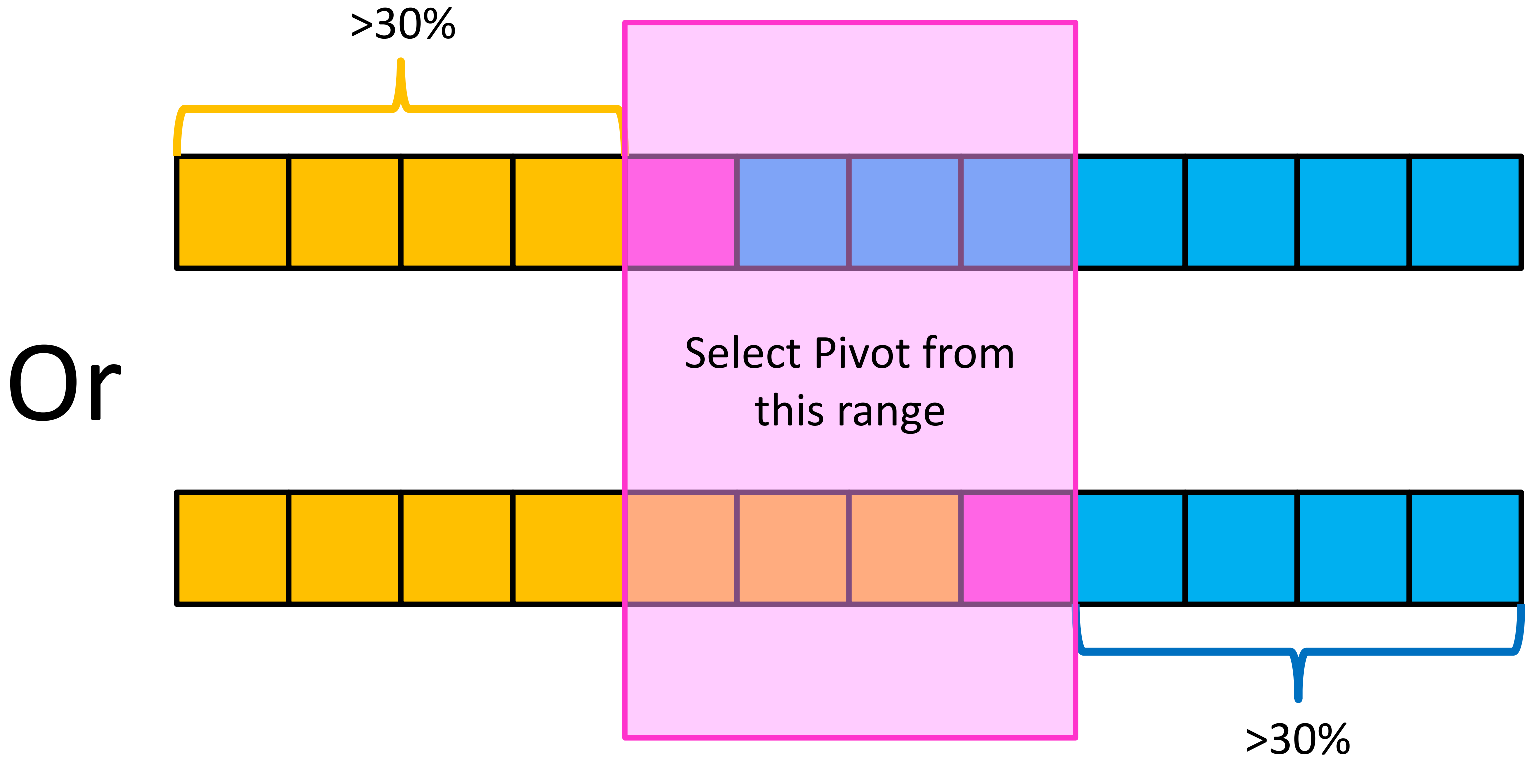
- What makes a good Pivot?
  - Roughly even split between left and right
  - Ideally: median

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

Déjà vu?

Good Pivot

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



Median of Medians

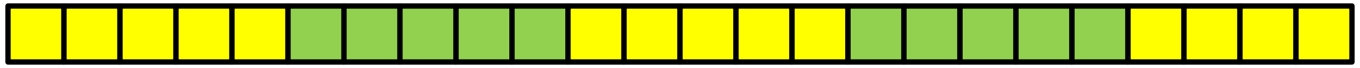
- Fast way to select a “good” pivot

- Guarantees pivot is greater than 30% of elements and less than 30% of the elements

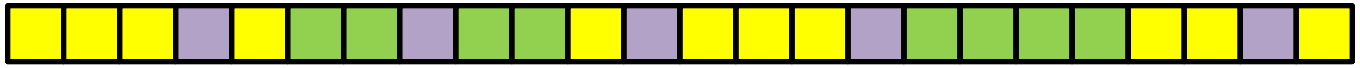
- Idea: break list into chunks, find the median of each chunk, use the median of those medians

Median of Medians

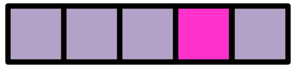
1. Break list into chunks of size 5



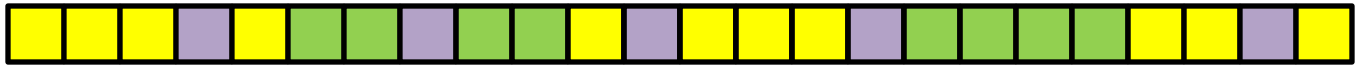
2. Find the median of each chunk



3. Return median of medians (using Quickselect)

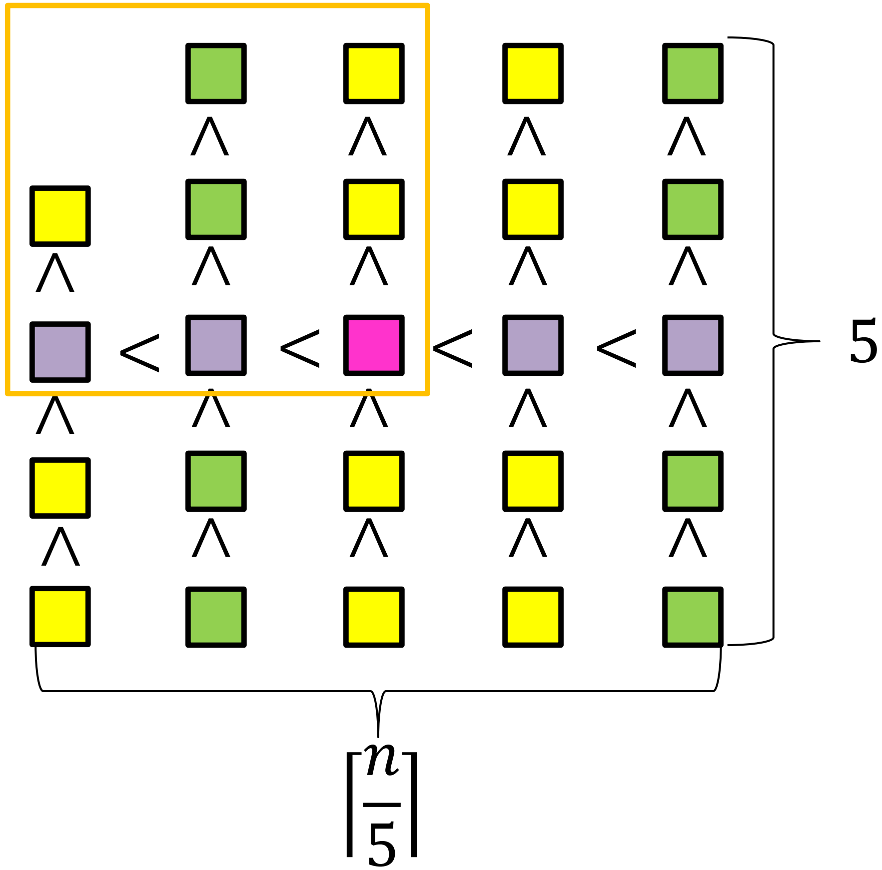


Why is this good?



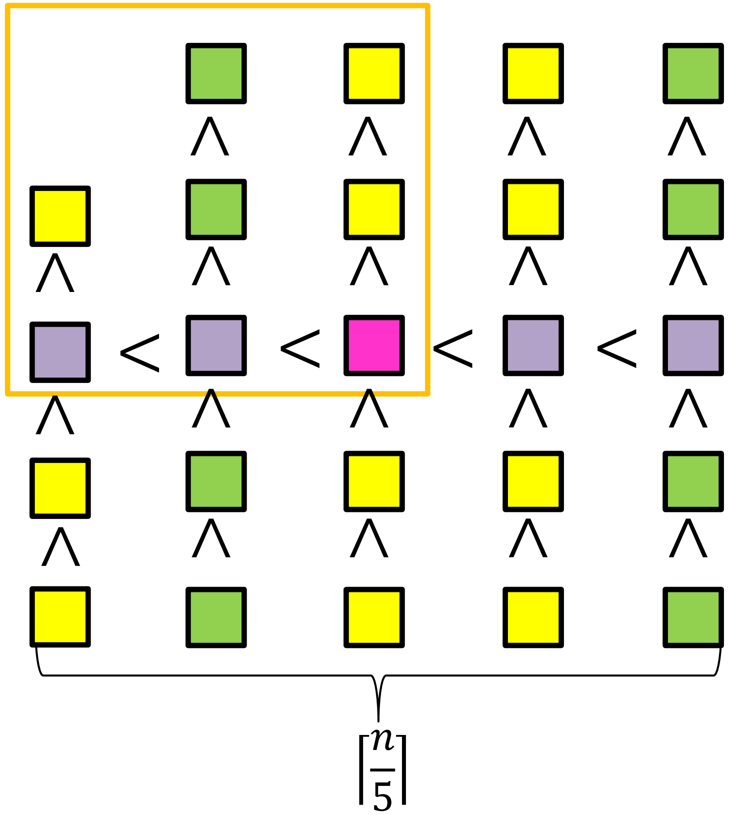
Each chunk sorted, chunks ordered by their medians

MedianofMedians
is Greater than all
of these



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

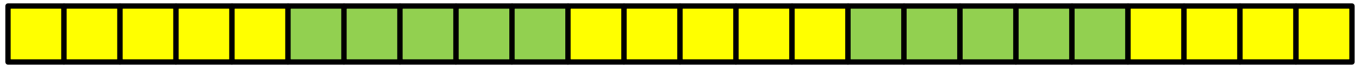
  $\leq S\left(\frac{7}{10}n\right)$

- Combine: Nothing!

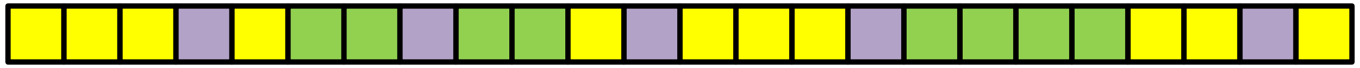
$$S(n) \leq S\left(\frac{7}{10}n\right) + M(n) + \Theta(n)$$

Median of Medians, Run Time

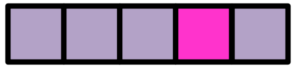
1. Break list into chunks of 5 $\Theta(n)$



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

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

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

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

$$= S\left(\frac{7n}{10}\right) + S\left(\frac{2n}{10}\right) + \Theta(n)$$

$$\le S\left(\frac{9n}{10}\right) + \Theta(n)$$ Because $S(n) = \Omega(n)$

Master theorem Case 3!

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

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?

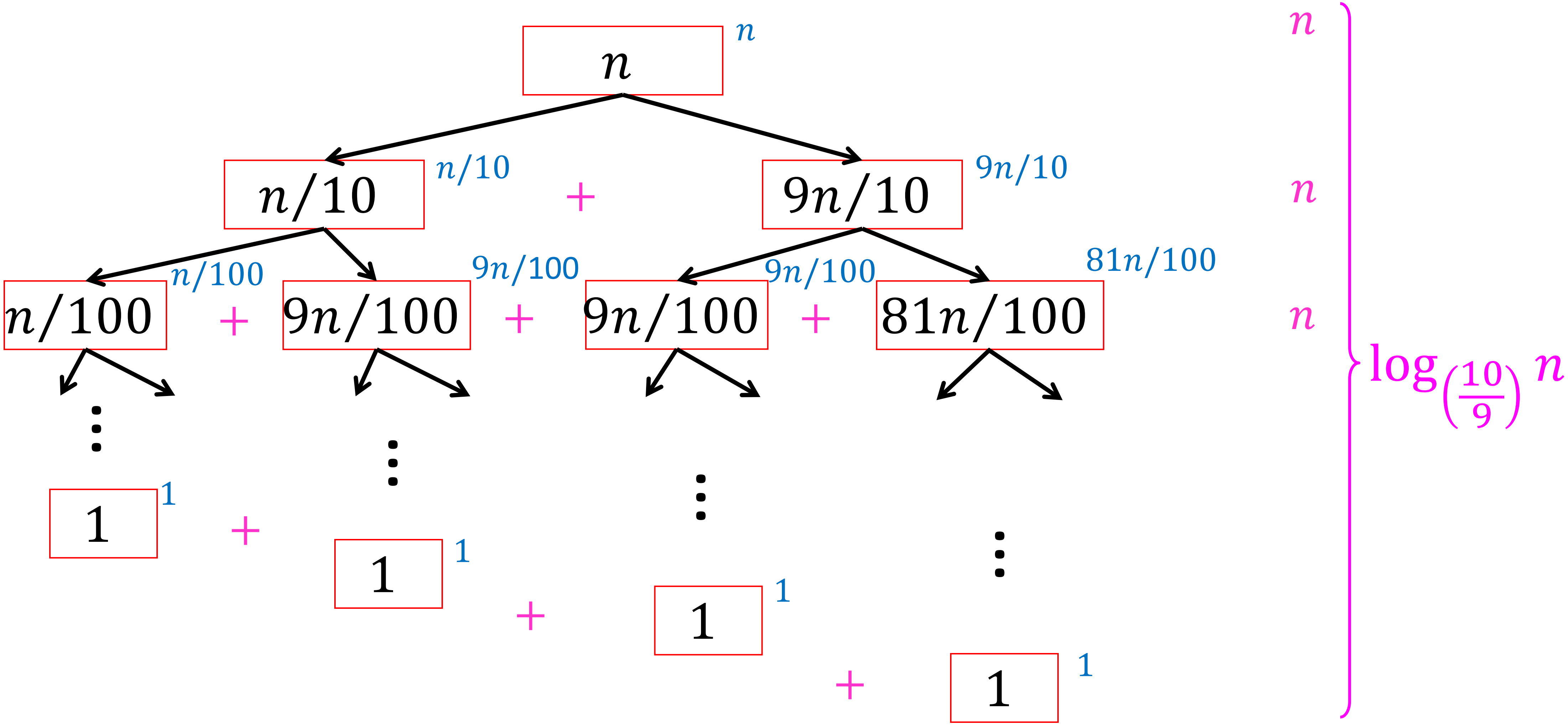
- Using Quickselect to pick median guarantees $\Theta(n \log n)$ run time
- Approach has very large constants
  - If you really want $\Theta(n \log n)$, better off using MergeSort
- Better approach: Random pivot
  - Very small constant (very fast algorithm)
  - Expected to run in $\Theta(n \log n)$ time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

Quicksort Run Time

- If the pivot is always $\frac{n}{10}$th order statistic:

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

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



Quicksort Run Time

- If the pivot is always $\frac{n}{10}$th order statistic:

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

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

Quicksort Run Time

- If the pivot is always d^{th} order statistic:

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 d each time

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

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

What's the probability of this occurring?

Probability of n^2 run time

We must consistently select pivot from within the first d terms

Probability first pivot is among d smallest: $\frac{d}{n}$

Probability second pivot is among d smallest: $\frac{d}{n-d}$

Probability all pivots are among d smallest:

$$\frac{d}{n} \cdot \frac{d}{n-d} \cdot \frac{d}{n-2d} \cdot \ldots \cdot \frac{d}{2d} \cdot 1 = \frac{1}{\left(\frac{n}{d}\right)!}$$

Formal Argument for $n \log n$ Average

- Remember, run time counts comparisons!
- Quicksort only compares against a pivot
  - Element i only compared to element j if one of them was the pivot

Formal Argument for $n \log n$ Average

- What is the probability of comparing two given elements?

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

- (Probability of comparing 3 and 4) = 1
  - Why? Otherwise I wouldn’t know which came first
  - ANY sorting algorithm must compare adjacent elements

Formal Argument for $n \log n$ Average

- What is the probability of comparing two given elements?

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

- (Probability of comparing 1 and 12) = $\frac{2}{12}$
  - Why?
    - I only compare 1 with 12 if either was chosen as the first pivot
    - Otherwise they would be divided into opposite sublists

Formal Argument for $n \log n$ Average

- Probability of comparing i with j ($j > i$):
  - dependent on the number of elements between i and j
  - $\frac{1}{j-i+1}$
- Expected number of comparisons:
  - $\sum_{i<j} \frac{1}{j-i+1}$

Expected number of Comparisons

Consider when $i = 1$

$$\sum_{i<j} \frac{1}{j-i+1}$$

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

Compared if 1 or 2 are chosen as pivot
(these will always be compared)

Sum so far: $\frac{2}{2}$

Expected number of Comparisons

$$\sum_{i<j} \frac{1}{j - i + 1}$$

Consider when $i = 1$

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

Compared if 1 or 3 are chosen as pivot
(but never if 2 is ever chosen)

Sum so far: $\frac{2}{2} + \frac{2}{3}$

Expected number of Comparisons

Consider when $i = 1$

$$\sum_{i<j} \frac{1}{j - i + 1}$$

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

Compared if 1 or 4 are chosen as pivot
(but never if 2 or 3 are chosen)

Sum so far: $\frac{2}{2} + \frac{2}{3} + \frac{2}{4}$

Expected number of Comparisons

$$\sum_{i<j} \frac{1}{j-i+1}$$

Consider when $i = 1$

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

Compared if 1 or 12 are chosen as pivot
(but never if 2 -> 11 are chosen)

Overall sum: $\frac{2}{2} + \frac{2}{3} + \frac{2}{4} + \frac{2}{5} + \cdots + \frac{2}{n}$

Expected number of Comparisons

$$\sum_{i<j} \frac{1}{j-i+1}$$

When $i = 1$: $2\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right)$

n terms overall

$$\sum_{i<j} \frac{1}{j-i+1} \leq 2n \boxed{\left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)} \quad \Theta(\log n)$$

Quicksort overall: expected $\Theta(n \log n)$