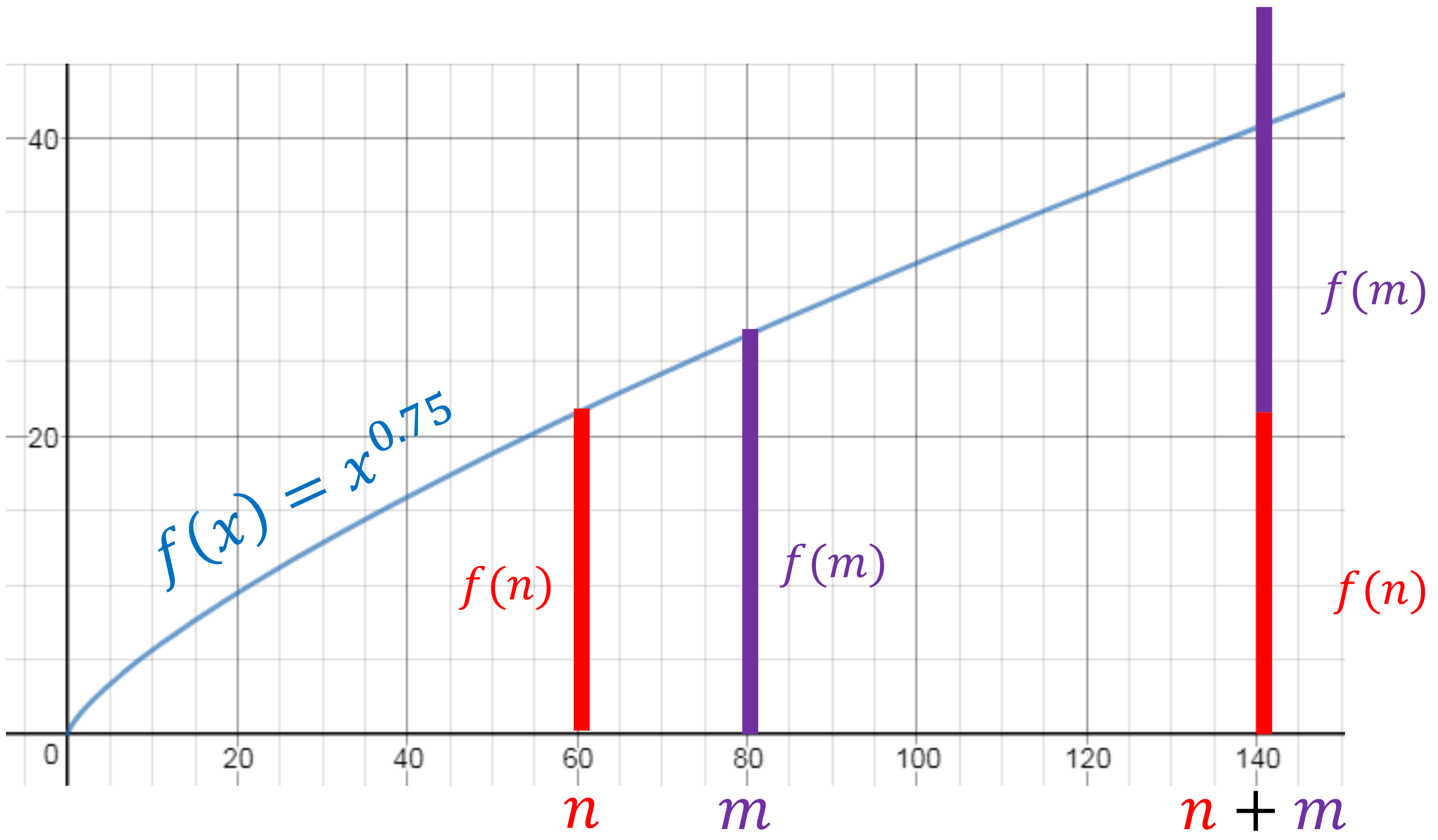
Reminder Warm-Up

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

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

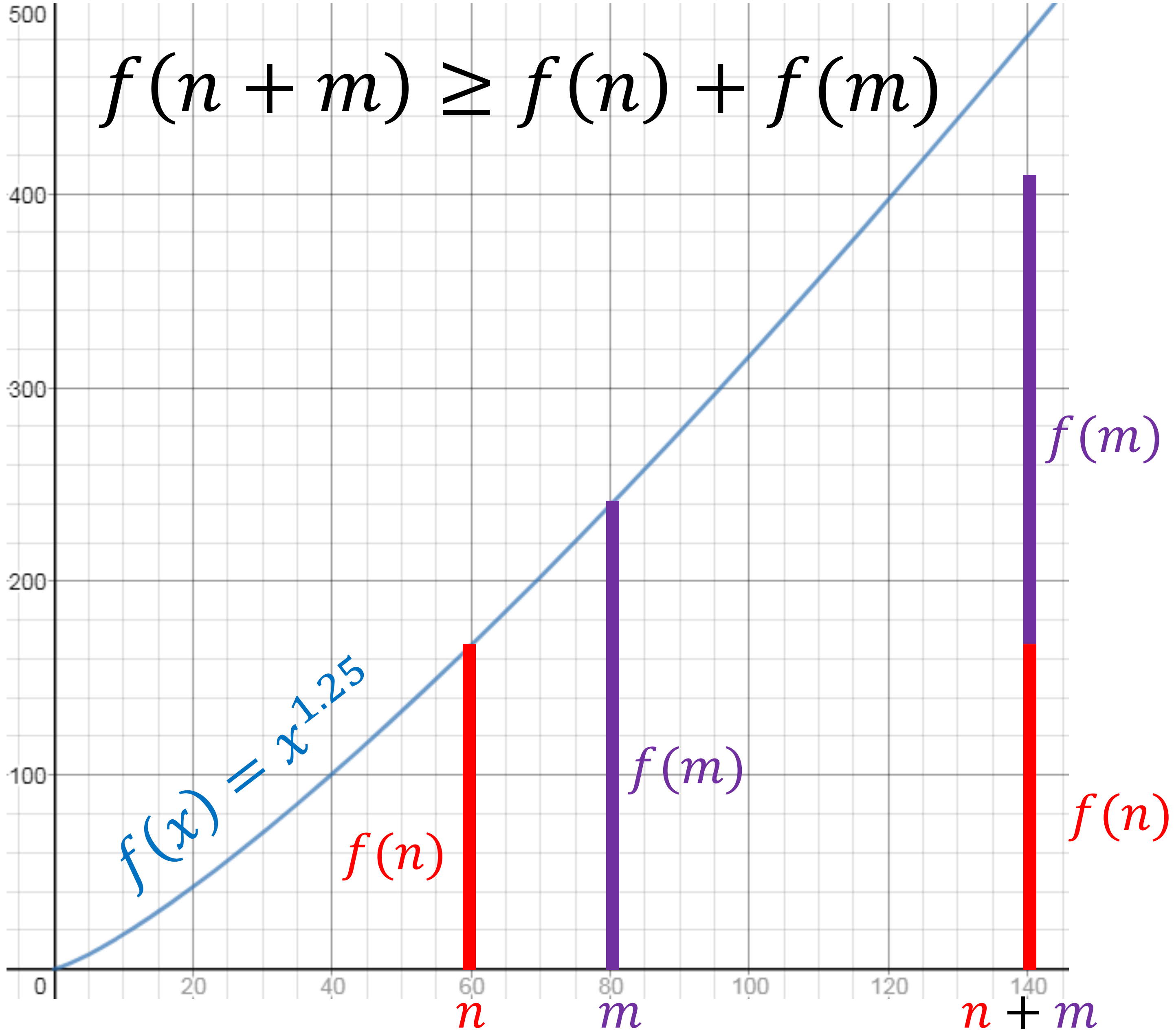
When $f(n) = \Omega(n)$

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



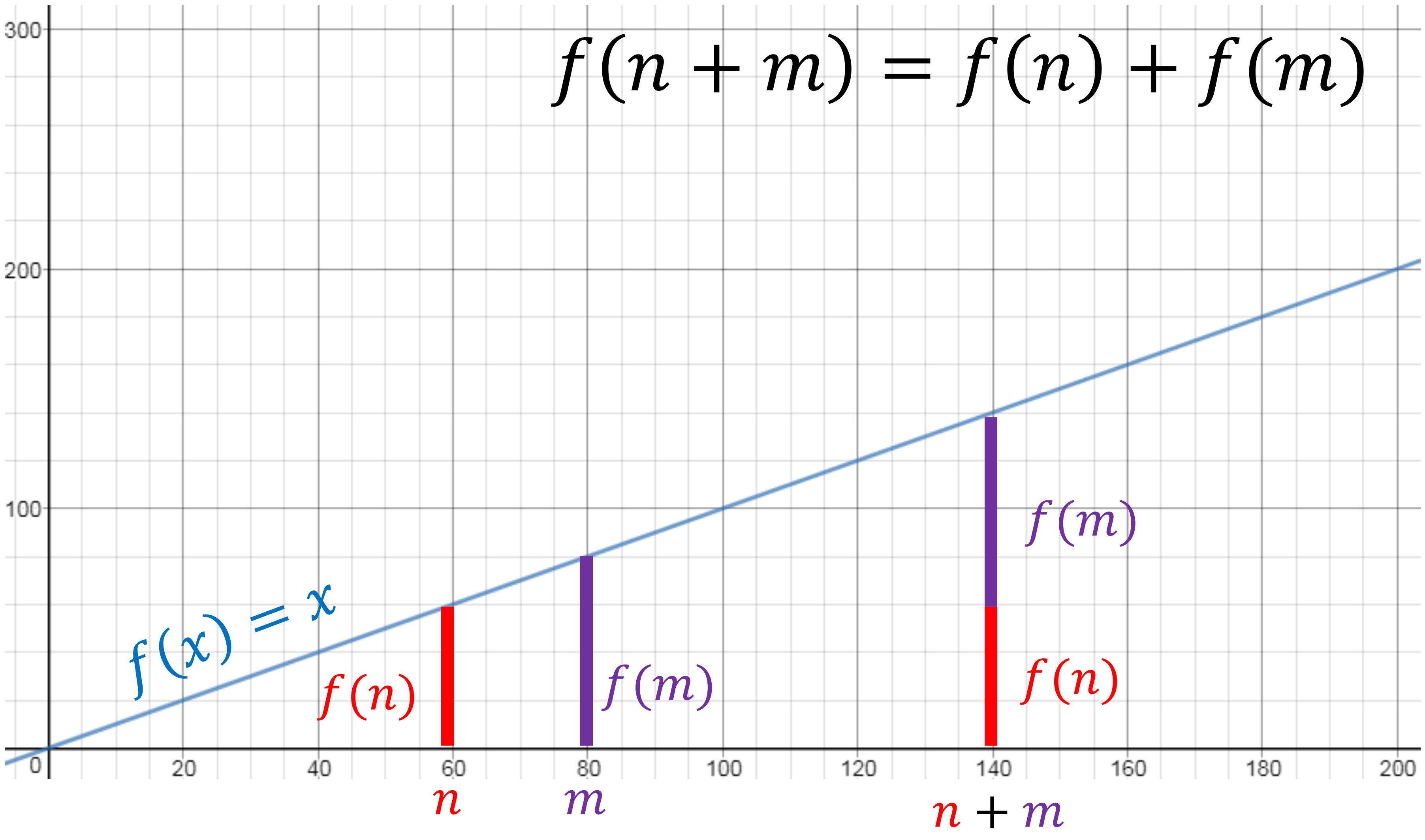
$$f(n+m) \le f(n) + f(m)$$

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



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

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



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

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a = 1$

$b = \frac{10}{9}$

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

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

If this was $T\left(\frac{9n}{10}\right)$ $= T\left(\frac{n}{10/9}\right)$, 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
- Strassen’s Algorithm
- Sorting
- Quicksort

CLRS Readings

- Chapter 7

Homeworks

- Hw2 due 11pm Tonight!
  - Programming (use Python or Java!)
  - Divide and conquer
  - Closest pair of points
- Hw3 coming Tonight!
  - Written (LaTeX)
  - Divide and Conquer

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	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

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

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

How to pick the pivot?

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
- 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

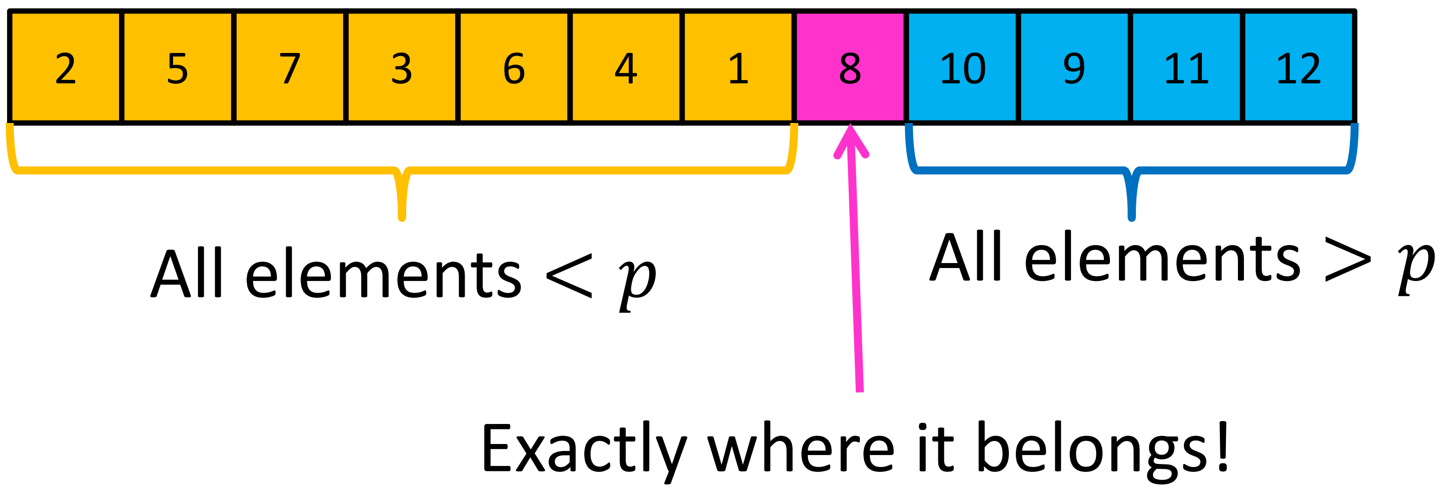
Start: unordered list

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)

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

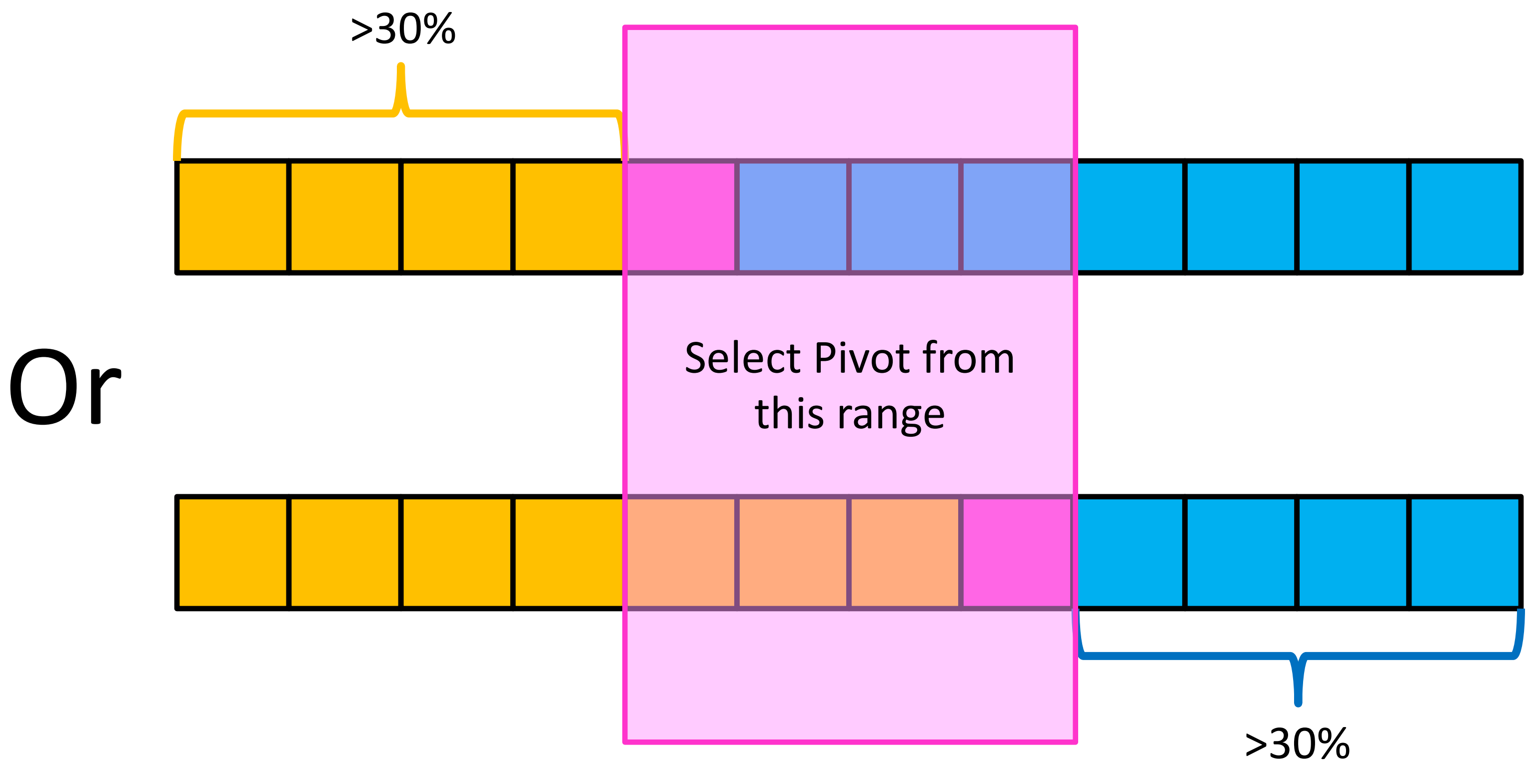
- What makes a good Pivot?
  - Roughly even split between left and right
  - Ideally: median

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%

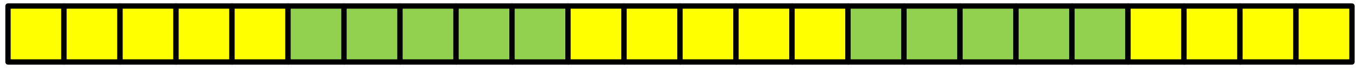


Median of Medians

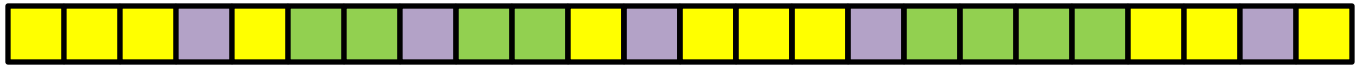
- Fast way to select a “good” pivot
- Guarantees pivot is greater than 30% of elements and less than 30% of the elements
- Idea: break list into chunks, find the median of each chunk, use the median of those medians

Median of Medians

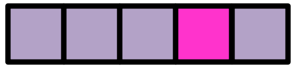
1. Break list into chunks of size 5



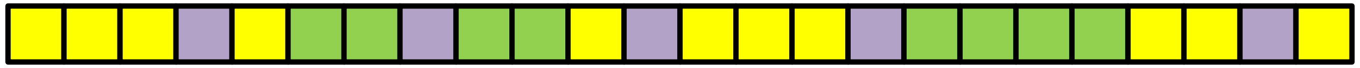
2. Find the median of each chunk



3. Return median of medians (using Quickselect)

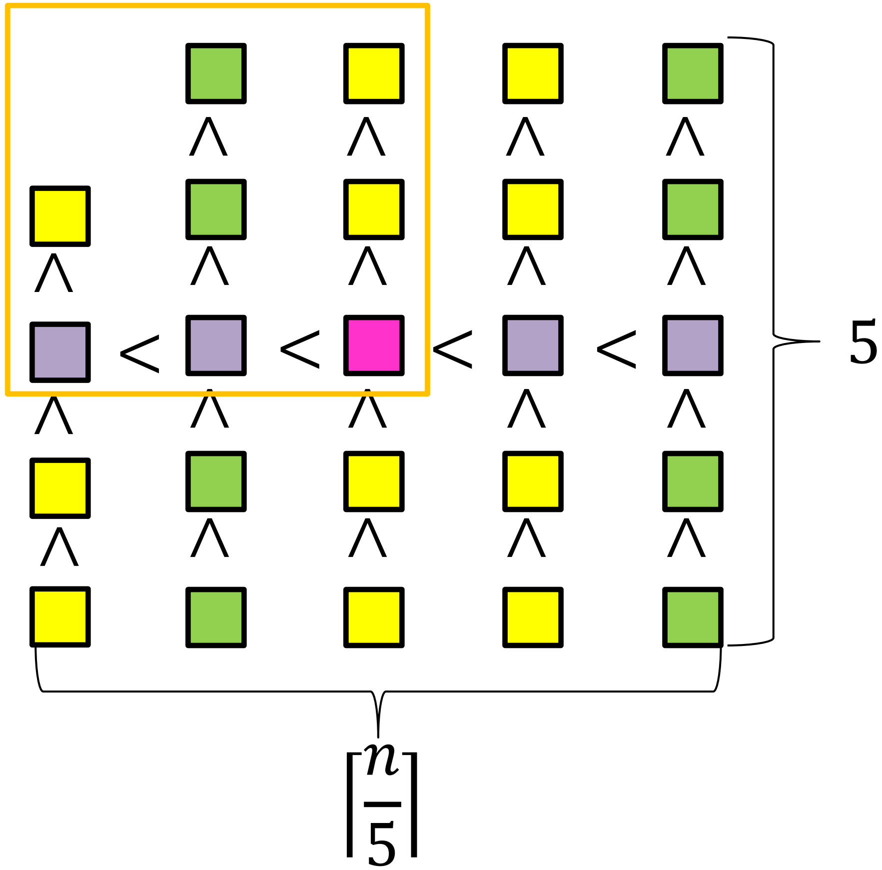


Why is this good?



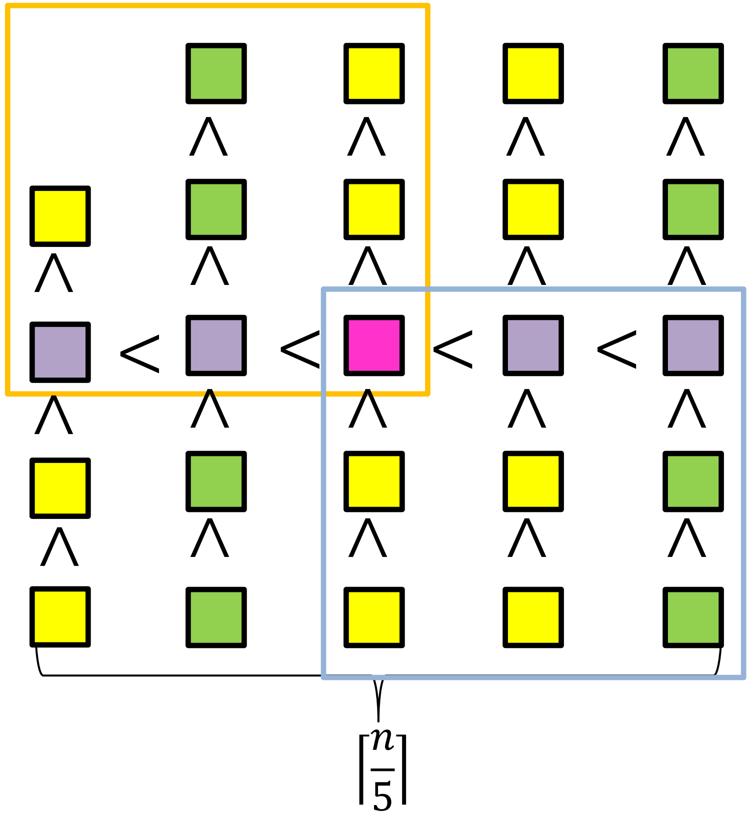
Each chunk sorted, chunks ordered by their medians

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

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

Similarly:

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

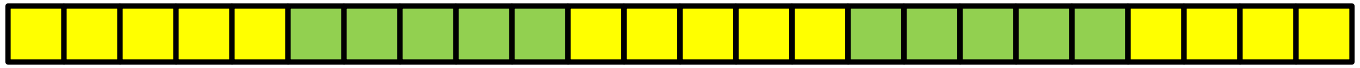
Quickselect

- 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!

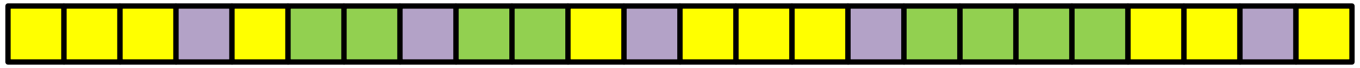
$$S(n) \leq S\left(\frac{7}{10}n\right) + M(n) + \Theta(n)$$

Median of Medians, Run Time

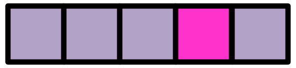
1. Break list into chunks of 5 $\Theta(n)$



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

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

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

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

... 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?

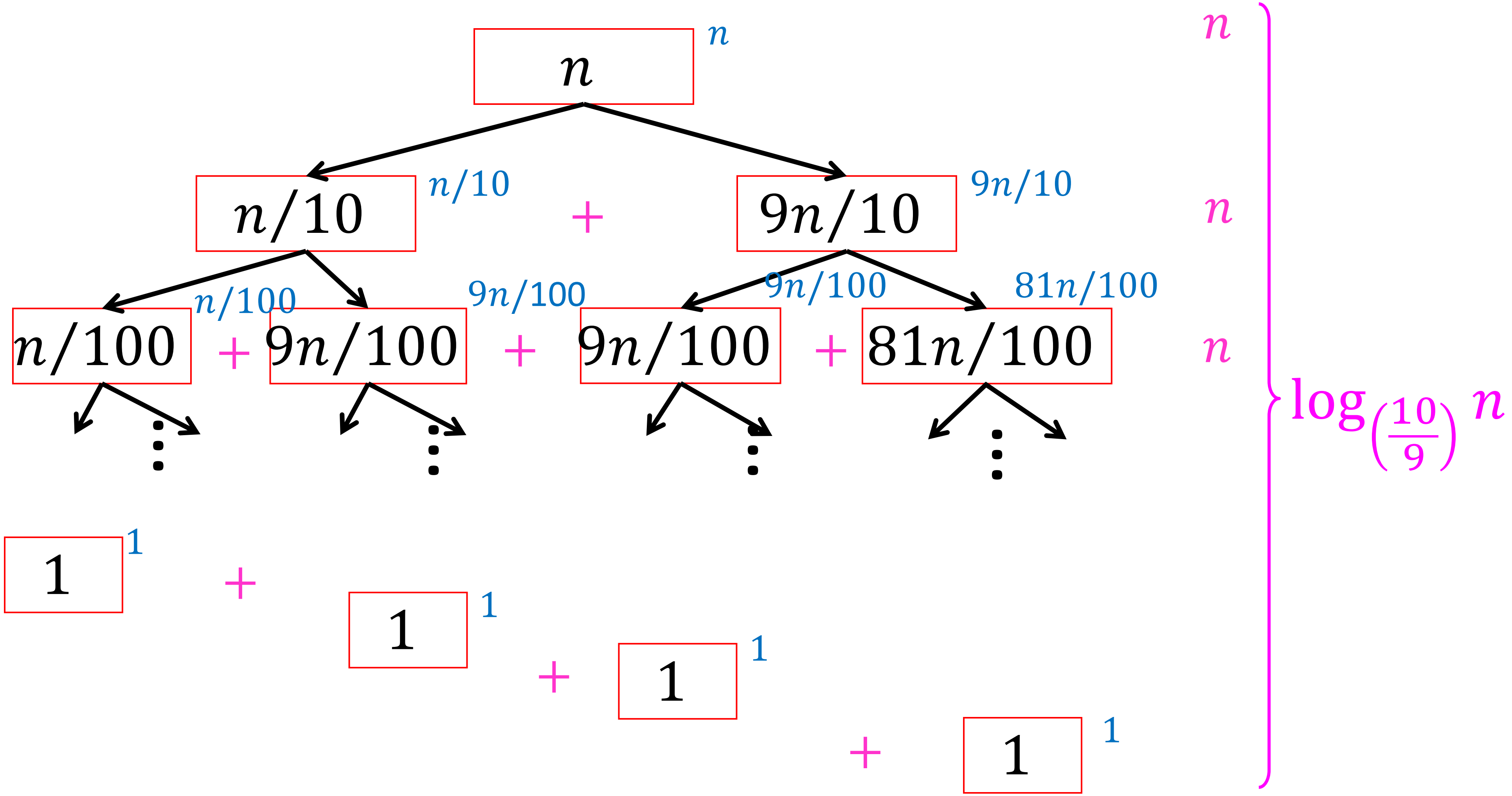
- Using Quickselect to pick median guarantees $\Theta(n \log n)$ run time
- Approach has very large constants
  - If you really want $\Theta(n \log n)$, better off using MergeSort
- Better approach: Random pivot
  - Very small constant (very fast algorithm)
  - Expected to run in $\Theta(n \log n)$ time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

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

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

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

Consider the sorted version of the list

Observation: Adjacent elements must be compared

- **Why?** Otherwise I would not know which came first
- **Every** sorting algorithm **must** compare adjacent elements

In quicksort: adjacent elements always end up in same sublist, unless one is 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

Consider the sorted version of the list

$$\Pr[\text{we compare } 1 \text{ and } 12] = \frac{2}{12}$$

Assuming pivot is chosen uniformly at random

Only compared if 1 or 12 was chosen as the first pivot since otherwise they are in different sublists

Formal Argument for $\boldsymbol{n \log n}$ Average

What is the probability of comparing two given elements?

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

Case 1: Pivot less than i

Then sublist $[i, i+1, \dots, j]$ will be in right sublist and will be processed in future recursive invocation of Quicksort

$$\Pr[\text{we compare } i \text{ and } j] = \Pr[\text{we compare } i \text{ and } j \text{ in Quicksort}([p+1, \dots, n])]$$

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
				i				j			

Case 1: Pivot less than i

Then sublist $[i, i+1, \dots, j]$ will be processed in future recursive invo

$[p+1, \dots, n]$ denotes the right sublist (in some order) that we are recursively sorting

$$\Pr[\text{we compare } i \text{ and } j] = \Pr[\text{we compare } i \text{ and } j \text{ in Quicksort}([p+1, \dots, n])]$$

Formal Argument for $\boldsymbol{n \log n}$ Average

What is the probability of comparing two given elements?

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

Case 2: Pivot greater than j

Then sublist $[i, i+1, \dots, j]$ will be in left sublist and will be processed in future recursive invocation of Quicksort

$$\Pr[\text{we compare } i \text{ and } j] = \Pr[\text{we compare } i \text{ and } j \text{ in Quicksort}([1, \dots, p])]$$

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
				i				j			

Case 3.1: Pivot contained in $[i+1, \dots, j-1]$

Then i and j are in different sublists and will <u>never</u> be compared

$$\Pr[\text{we compare } i \text{ and } j] = 0$$

Formal Argument for $\boldsymbol{n \log n}$ Average

What is the probability of comparing two given elements?

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

Case 3.2: Pivot is either i or j

Then we will always compare i and j

$$\Pr[\text{we compare } i \text{ and } j] = 1$$

Formal Argument for $\boldsymbol{n \log n}$ Average

What is the probability of comparing two given elements?

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

Case 1: Pivot less than i

$$\Pr[\text{we compare } i \text{ and } j] = \Pr[\text{we compare } i \text{ and } j \text{ in Quicksort}([p+1, \dots, n])]$$

Case 2: Pivot greater than j

$$\Pr[\text{we compare } i \text{ and } j] = \Pr[\text{we compare } i \text{ and } j \text{ in Quicksort}([1, \dots, p])]$$

Case 3: Pivot in $[i, i+1, \dots, j]$

$$\Pr[\text{we compare } i \text{ and } j] = \Pr[i \text{ or } j \text{ is selected as pivot}] = \frac{2}{j - i + 1}$$

Formal Argument for $n \log n$ Average

Probability of comparing i with j ($j > i$):

- dependent on the number of elements between (and including) i and j

$$\frac{2}{j - i + 1}$$

Expected number of comparisons for Quicksort:

$$\sum_{i<j} \frac{2}{j - i + 1}$$

$$\boxed{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1}}$$

Expected number of Comparisons

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Consider when $i = 1$

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

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

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

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Consider when $i = 1$

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

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

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

$$\sum_{i<j} \frac{2}{j-i+1}$$

When $i = 1$:

$$2\left(\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}\right) < 2\boxed{\sum_{x=1}^{n}\frac{1}{x}} \qquad \Theta(\log n)$$

Formal Argument for $\boldsymbol{n \log n}$ Average

- Probability of comparing element i with element j:
- $\Pr[\text{we compare } i \text{ and } j] = \frac{2}{j-i+1}$

- <u>Expected</u> number of comparisons:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}$$

Substitution:
$k = j - i$

$$\frac{1}{k+1} < \frac{1}{k}$$

Formal Argument for $\boldsymbol{n \log n}$ Average

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}$$

Substitution:
$k = j - i$

$$\frac{1}{k+1} < \frac{1}{k}$$

Useful fact: $\displaystyle\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$

Intuition (not proof!):

$$\sum_{i=1}^{n} \frac{1}{i} \approx \int_{1}^{n} \frac{1}{x} dx = \ln n$$

Formal Argument for $\boldsymbol{n \log n}$ Average

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}$$

$$= 2 \sum_{i=1}^{n-1} \Theta(\log n) = \Theta(n \log n)$$

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

Sorting, so far

- Sorting algorithms we have discussed:
  - Mergesort $O(n \log n)$
  - Quicksort $O(n \log n)$
- Other sorting algorithms (will discuss):
  - Bubblesort $O(n^2)$
  - Insertionsort $O(n^2)$
  - Heapsort $O(n \log n)$

Can we do better than $O(n \log n)$?