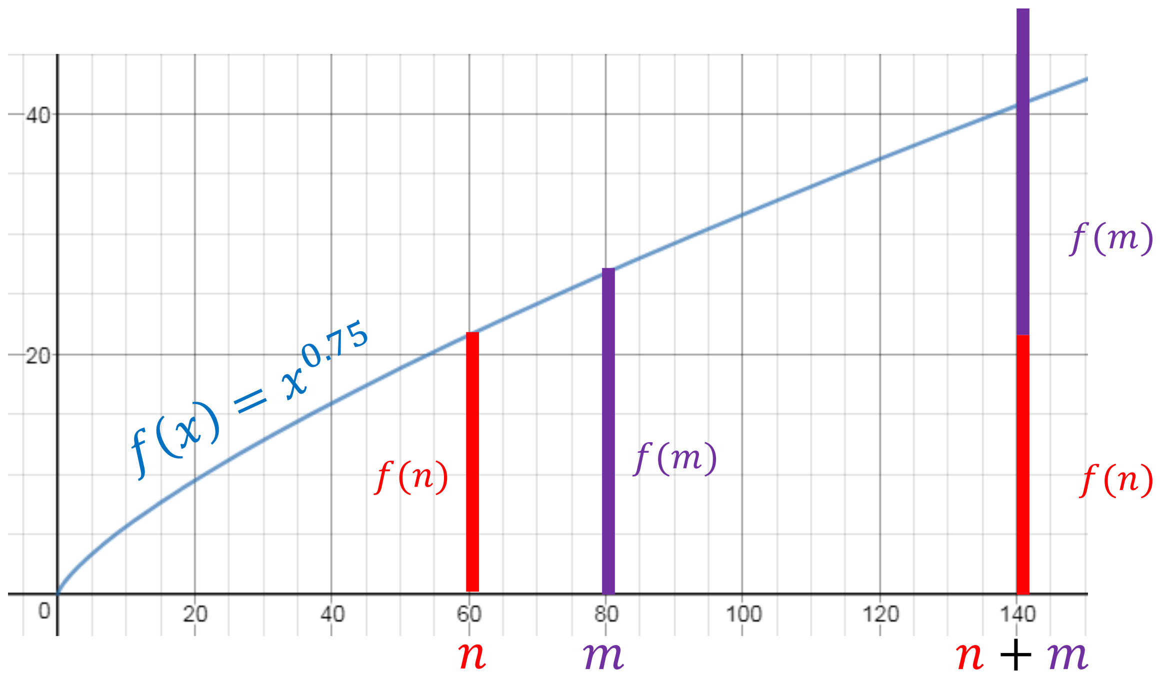
<u>Warm up</u>

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

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

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

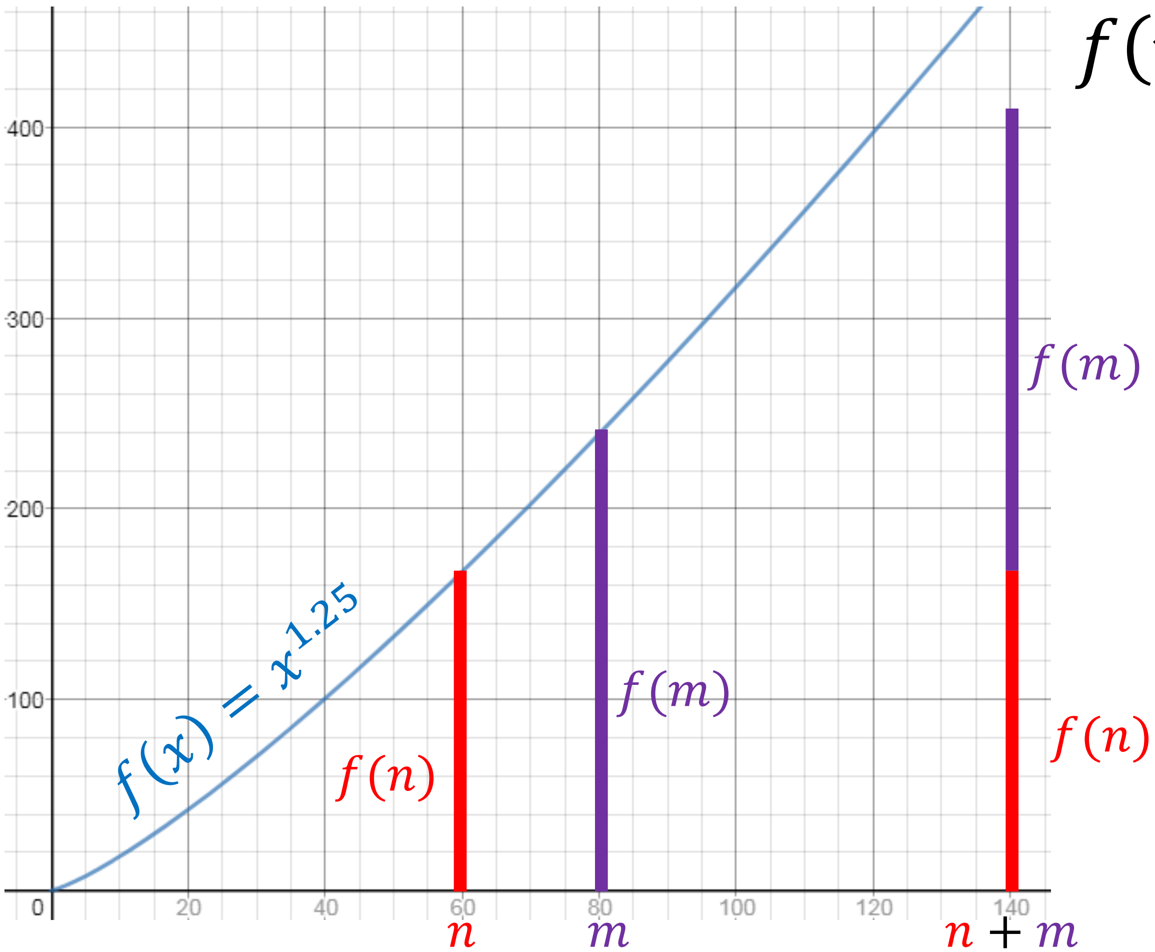
$f(n) \in O(n)$



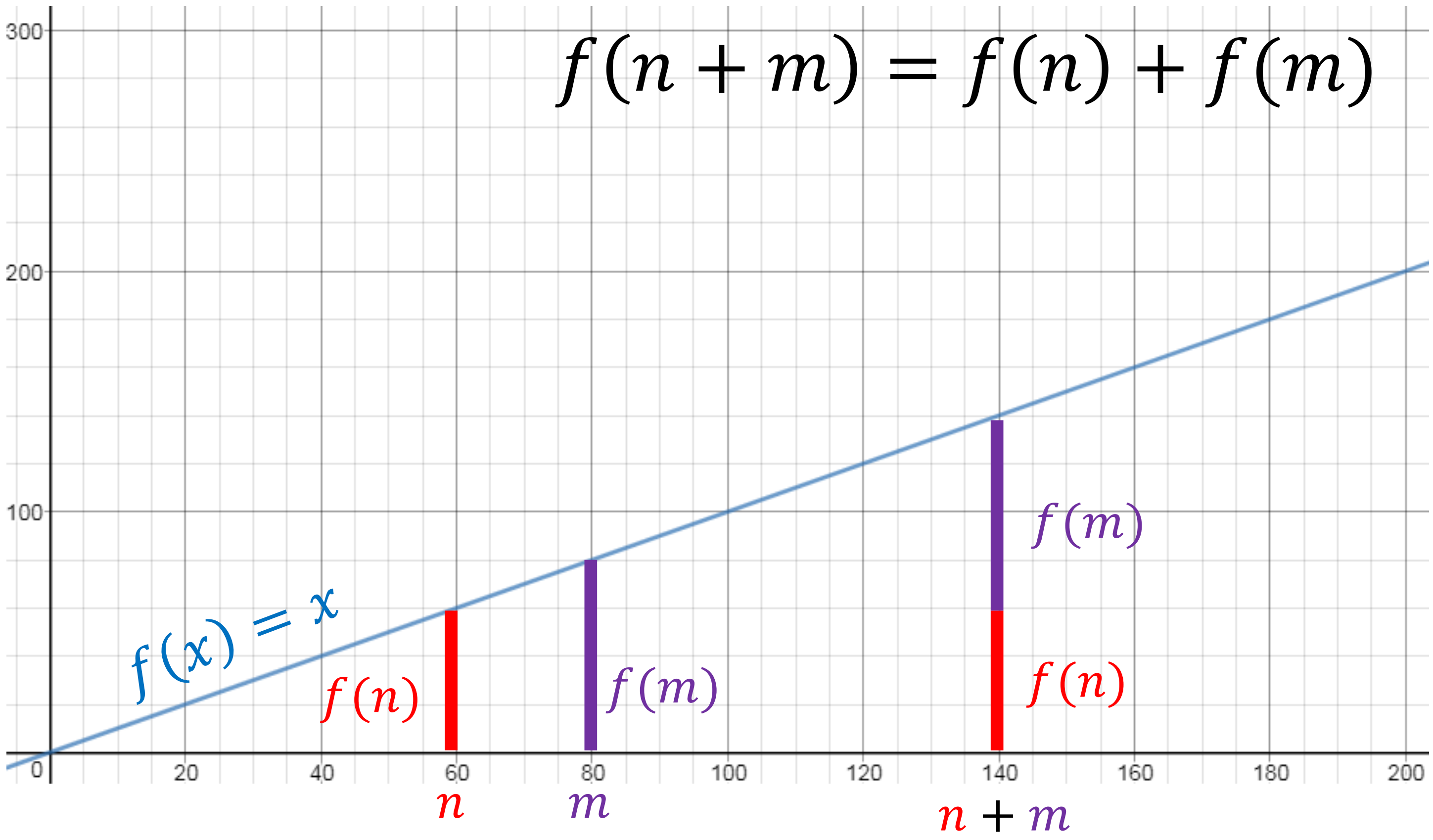
$$f(n+m) \leq f(n) + f(m)$$

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

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



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



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

Today's Keywords

- Divide and Conquer
- Strassen's Algorithm
- Sorting
- Quicksort

CLRS Readings

- Chapter 4
- Chapter 7

Homeworks

- HW3 due 11pm Thursday!
  - Programming (use Python or Java!)
  - Divide and conquer
  - Closest pair of points
  - Note: you will need to write a recursive function in:
    - closest_pair.py or
    - ClosestPair.java

Matrix Multiplication

$$
\begin{array}{cc} & n \\ n & \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \end{array} \times \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}
$$

$$
= \begin{bmatrix} 2+16+42 & 4+20+48 & 6+24+54 \\ . & . & . \\ . & . & . \end{bmatrix}
$$

$$
= \begin{bmatrix} 60 & 72 & 84 \\ 132 & 162 & 192 \\ 204 & 252 & 300 \end{bmatrix}
$$

Run time? $O(n^3)$ Lower Bound? $O(n^2)$

Matrix Multiplication D&C

Multiply $n \times n$ matrices (A and B)

Divide:

$$A = \left[\begin{array}{cc|cc} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \\ \hline a_9 & a_{10} & a_{11} & a_{12} \\ a_{13} & a_{14} & a_{15} & a_{16} \end{array}\right] \qquad B = \left[\begin{array}{cc|cc} b_1 & b_2 & b_3 & b_4 \\ b_5 & b_6 & b_7 & b_8 \\ \hline b_9 & b_{10} & b_{11} & b_{12} \\ b_{13} & b_{14} & b_{15} & b_{16} \end{array}\right]$$

Matrix Multiplication D&C

Multiply $n \times n$ matrices (A and B)

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \qquad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

Combine:

$$AB = \begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix}$$

Run time? $T(n) = 8T\left(\frac{n}{2}\right) + 4\left(\frac{n}{2}\right)^2$

Case 1!

$T(n) = \Theta(n^3)$

Matrix Multiplication D&C

Multiply $n \times n$ matrices (A and B)

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \qquad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

$$AB = \boxed{\begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix}}$$

Idea: Use a Karatsuba-like technique on this

Strassen's Algorithm

Multiply $n \times n$ matrices (A and B)

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \qquad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

Calculate:

$$Q_1 = (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2})$$
$$Q_2 = (A_{2,1} + A_{2,2})B_{1,1}$$
$$Q_3 = A_{1,1}(B_{1,2} - B_{2,2})$$
$$Q_4 = A_{2,2}(B_{2,1} - B_{1,1})$$
$$Q_5 = (A_{1,1} + A_{1,2})B_{2,2}$$
$$Q_6 = (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2})$$
$$Q_7 = (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2})$$

Find AB:

$$\begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix}$$
$$=$$
$$\begin{bmatrix} Q_1 + Q_4 - Q_5 + Q_7 & Q_3 + Q_5 \\ Q_2 + Q_4 & Q_1 - Q_2 + Q_3 + Q_6 \end{bmatrix}$$

Number Mults.: 7 Number Adds.: 18

$$T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2$$

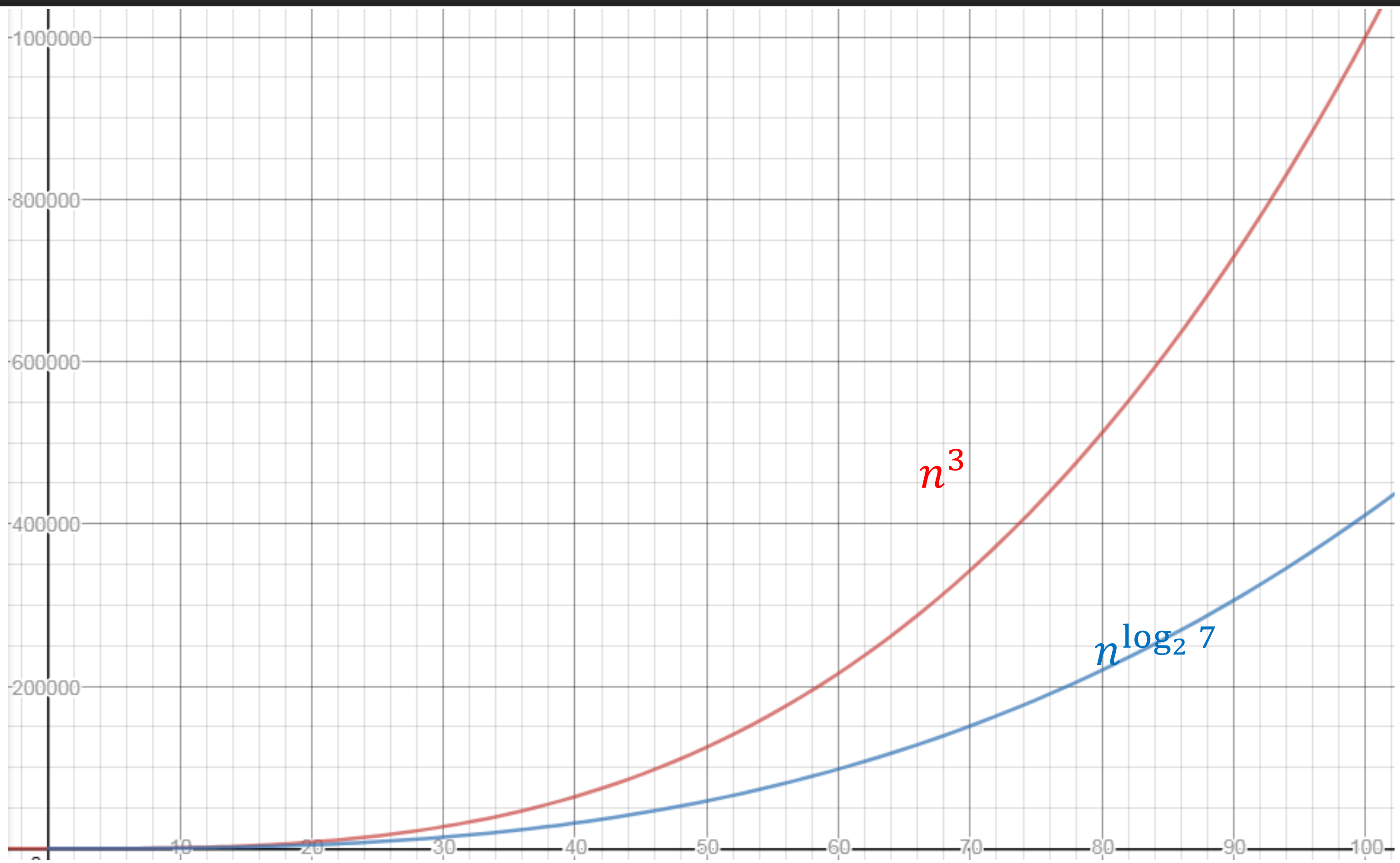
Strassen's Algorithm

$$T(n) = 7T\left(\frac{n}{2}\right) + \frac{9}{2}n^2$$

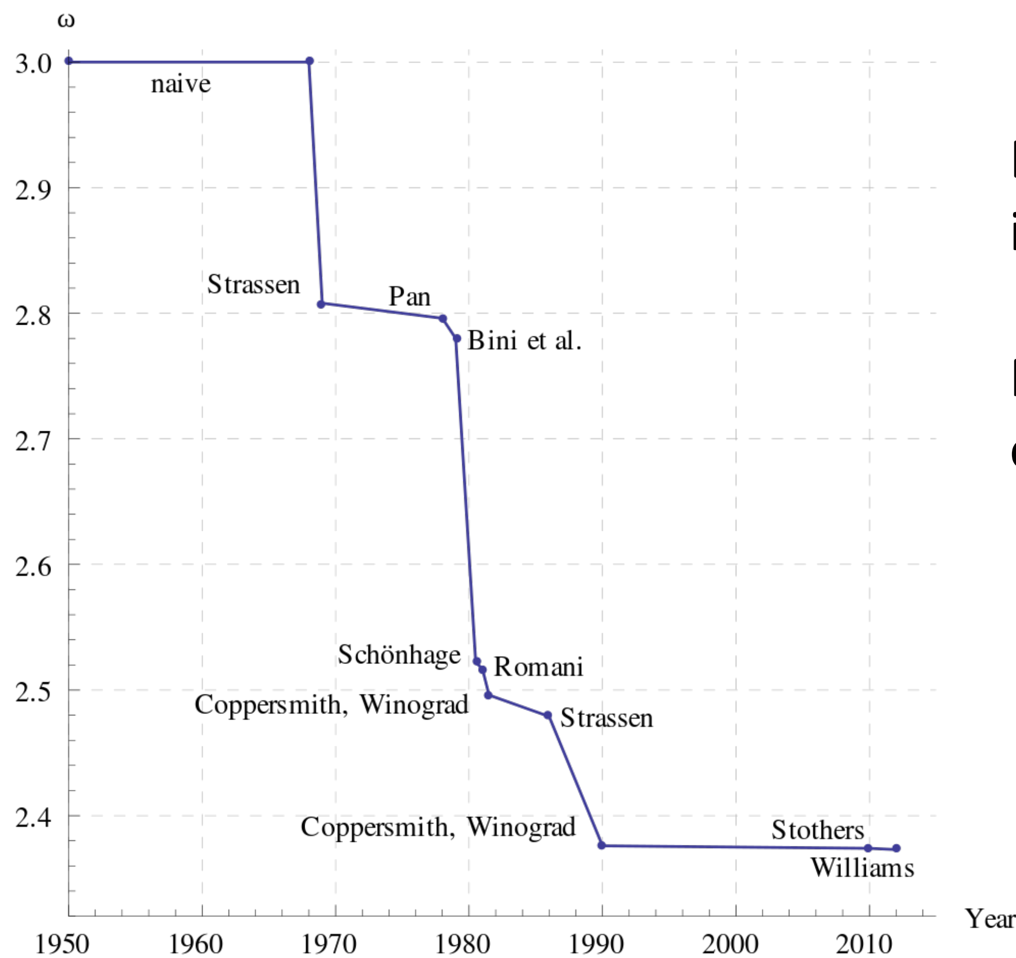
$$a = 7, b = 2, f(n) = \frac{9}{2}n^2$$

$$n^{\log_b a} = n^{\log_2 7} \approx n^{2.807}$$ Case 1!

$$T(n) = \Theta\left(n^{\log_2 7}\right) \approx \Theta(n^{2.807})$$


1000000
800000
600000
400000
200000
10
20
30
40
50
60
70
80
90
100
n^3
$n^{\log_2 7}$

Is this the fastest?



Best possible is unknown

May not even exist!

Divide and Conquer, so far

What do they have in common?

Divide: Very easy (i.e. $O(1)$)

Combine: Hard work ($\Omega(n)$)

- Mergesort
- Naïve Multiplication
- Karatsuba
- Closest Pair of Points
- Naïve Matrix-Matrix Multiplication
- Strassen's

Quicksort

- Like Mergesort:
  - Divide and conquer
  - $O(n \log n)$ run time (kind of…)
- Unlike Mergesort:
  - Divide step is the hard part
  - *Typically* faster than Mergesort

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

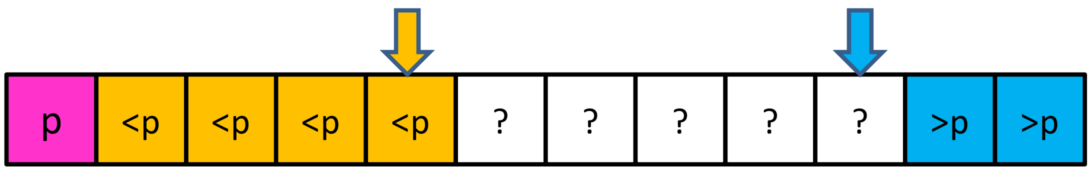
Goal: All elements $< p$ on left, all $> p$ on right

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

Partition, Invariant

Invariant:

- Begin is index of last item known to be < pivot p
- End is index of last item that hasn't be compared to p



Strategy:

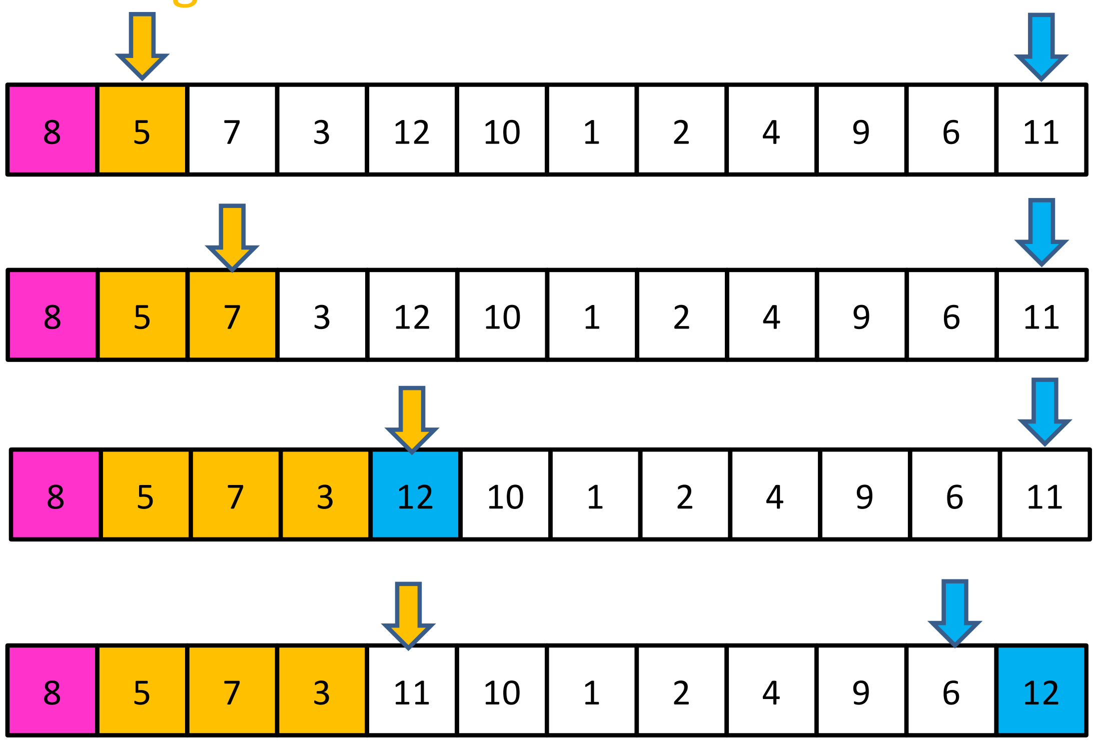
- Increment Begin, restore invariant
- If Begin value $< p$, no swaps. (Begin just moves right.)
- Else: Swap Begin value with End value, move End Left
- Done when Begin = End

Partition, Procedure

If Begin value $<\ p$, move Begin right

Else swap Begin value with End value, move End Left

Done when Begin = End

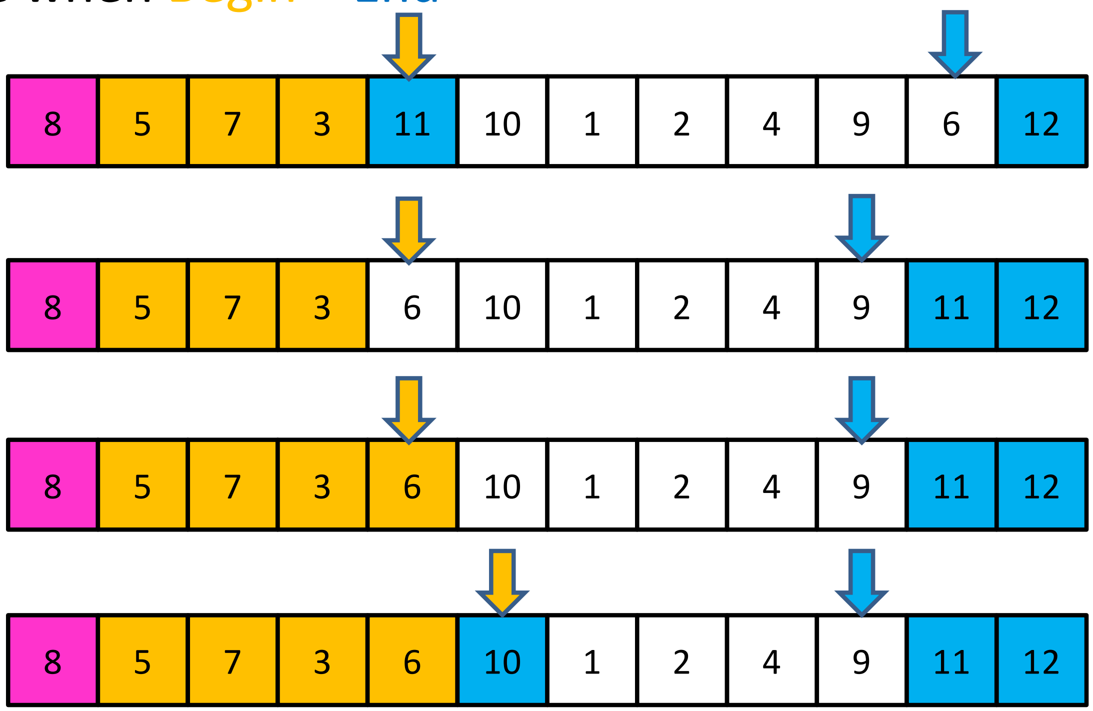


Partition, Procedure

If Begin value $<\ p$, move Begin right

Else swap Begin value with End value, move End Left

Done when Begin = End



Partition, Procedure

If Begin value $<\ p$, move Begin right

Else swap Begin value with End value, move End Left

Done when Begin = End

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

Case 1: meet at element $< p$

Swap p with pointer position (2 in this case)

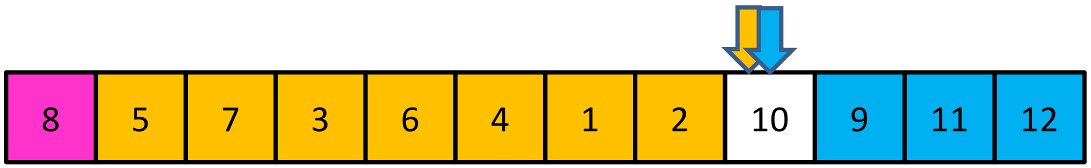
2	5	7	3	6	4	1	8	10	9	11	12

Partition, Procedure

If Begin value $<$ p, move Begin right

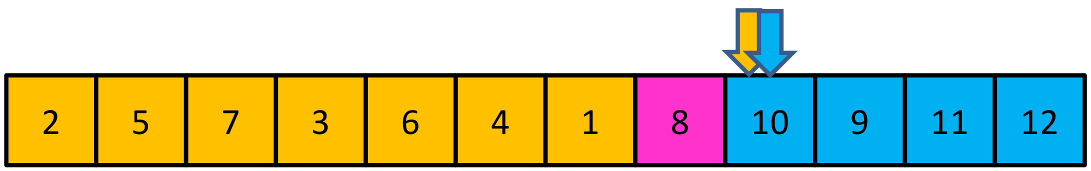
Else swap Begin value with End value, move End Left

Done when Begin = End



Case 2: meet at element $> p$

Swap p with value to the left (2 in this case)

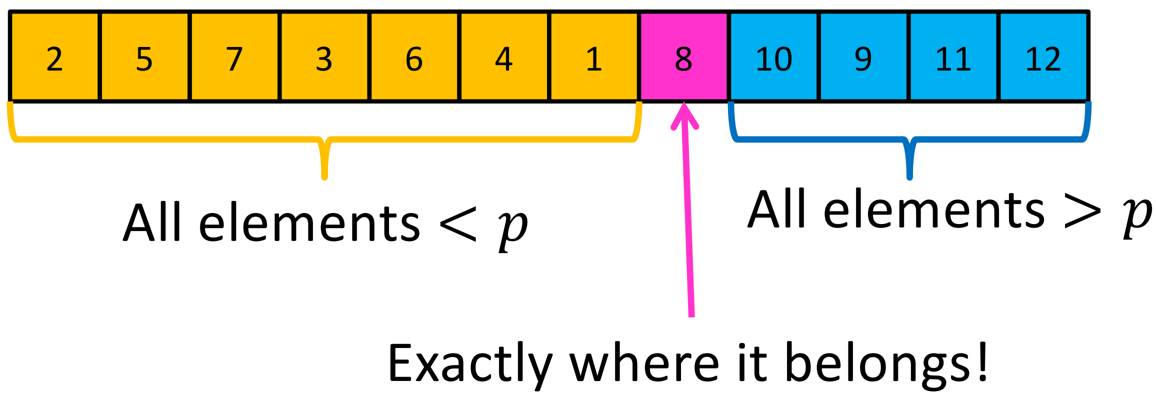


Partition Summary

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



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

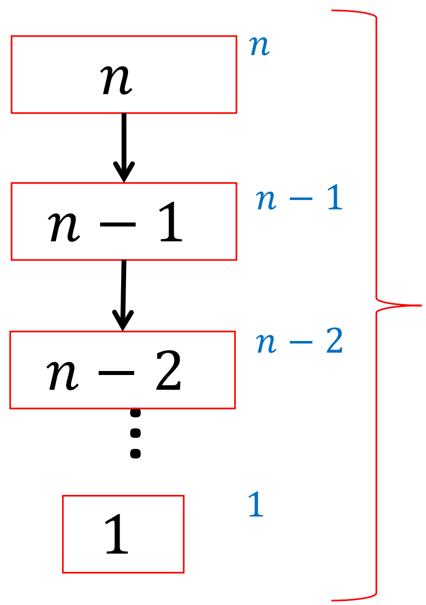
Then we shorten by 1 each time

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

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

Quicksort Run Time (Worst)

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



$$T(n) = 1 + 2 + 3 + \cdots + n$$

$$T(n) = \frac{n(n+1)}{2}$$

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

Quicksort on a (nearly) Sorted List

First element always yields unbalanced pivot

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

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

So 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
- Estimate the median etc?
  - Median-of-three: pick 3 items, choose their median
  - Choose a random-element as the pivot
- Can we find median in linear time?
  - Yes!
  - Quickselect

Quickselect

- CLRS, Section 9.1
  - **Selection problem**: Give list of distinct numbers and value *i*, find value *x* in list that is larger than exactly *i-1* list elements
- 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

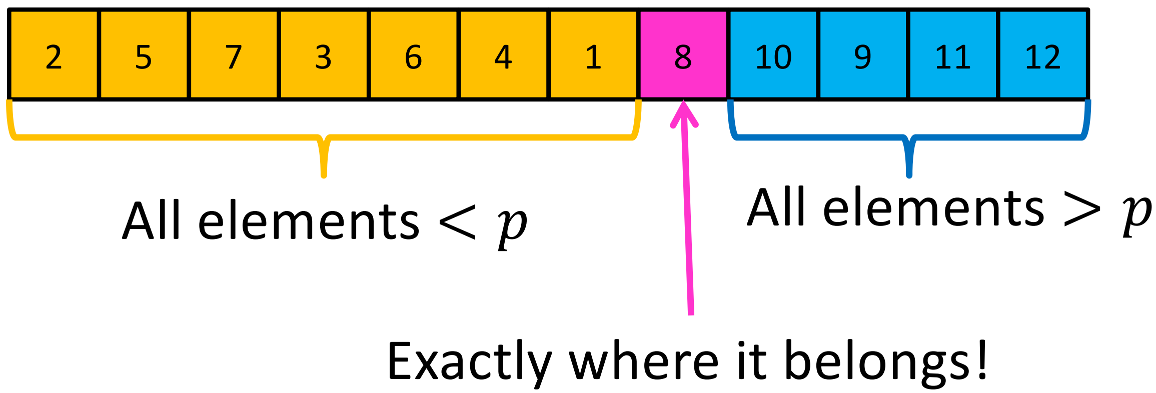
Start: unordered list

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

(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)
k = q − p + 1
if i == k                  // the pivot value is the answer
    return A[q]  // number of elements on left-side of partition
elseif i < k
    return RANDOMIZED-SELECT(A, p, q − 1, i)
else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
                                        // note adjustment to next call's 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

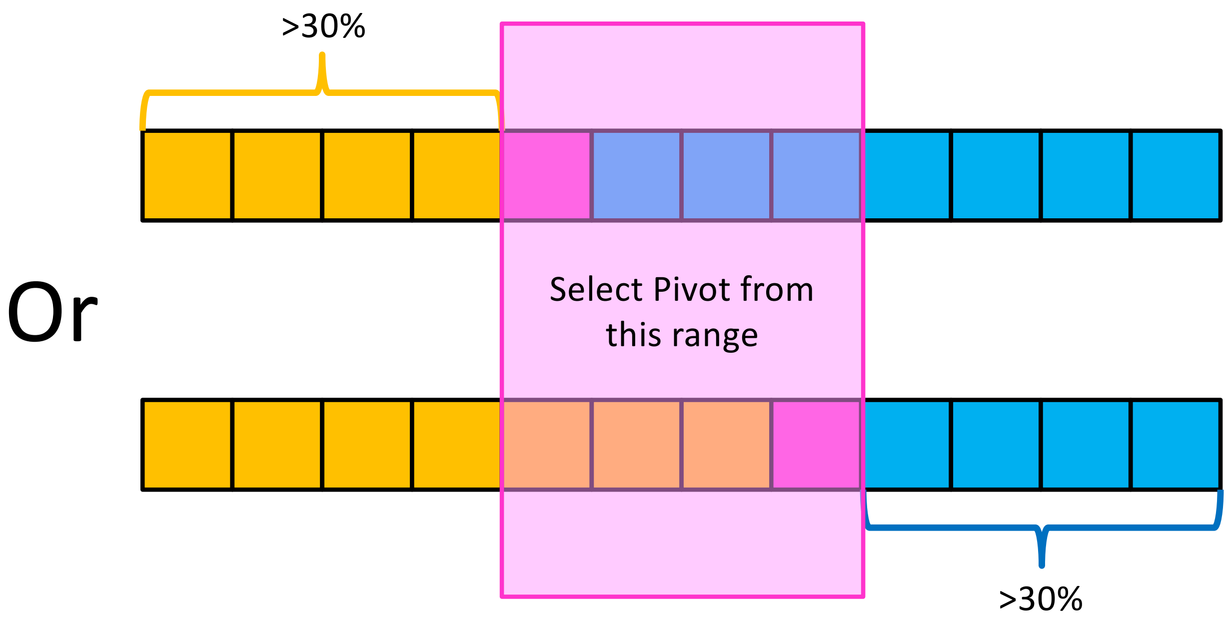
- What makes a good Pivot?
  - Roughly even split between left and right
  - Ideally: median

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%

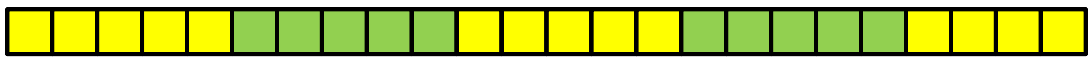


Median of Medians

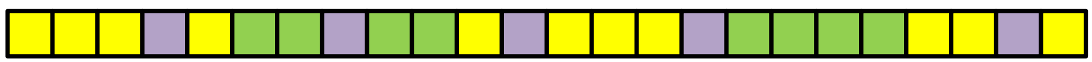
- Fast way to select a “good” pivot
- Guarantees pivot is greater than 30% of elements and less than 30% of the elements
- Idea: break list into chunks, find the median of each chunk, use the median of those medians

Median of Medians

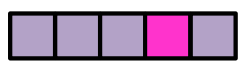
1. Break list into chunks of size 5



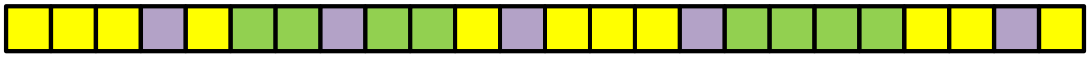
2. Find the median of each chunk



3. Return median of medians (using Quickselect)

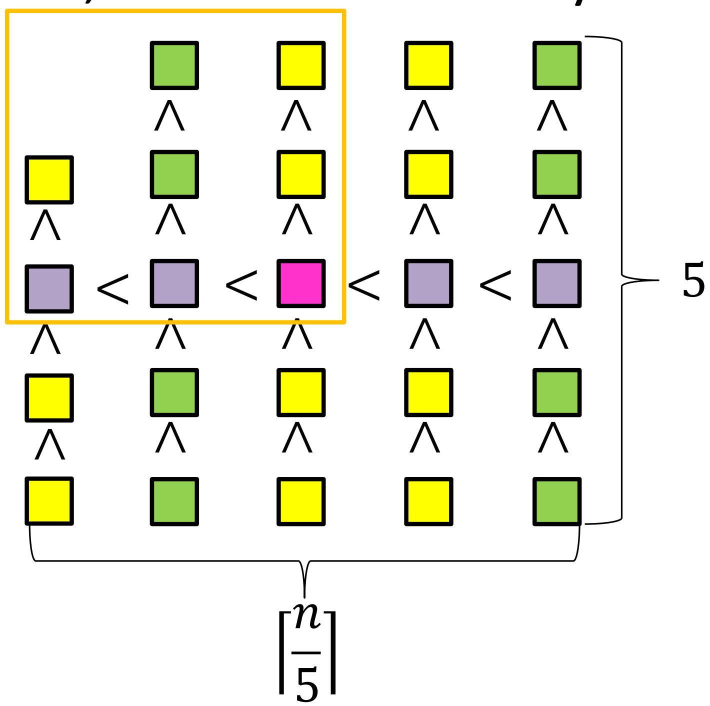


Why is this good?



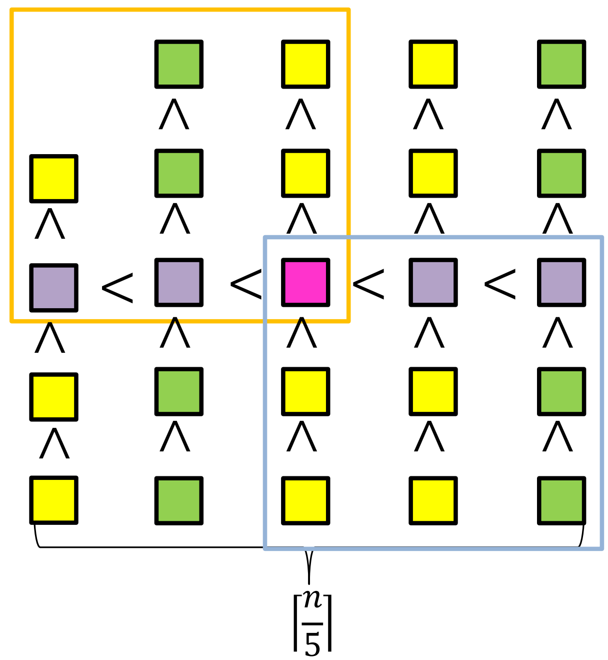
Each chunk sorted, chunks ordered by their medians

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

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

Similarly:

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

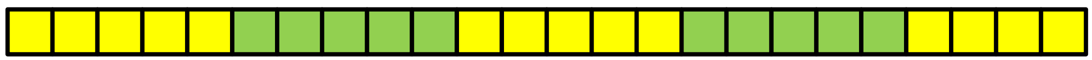
$$\leq S\left(\frac{7}{10}n\right)$$

- Combine: Nothing!

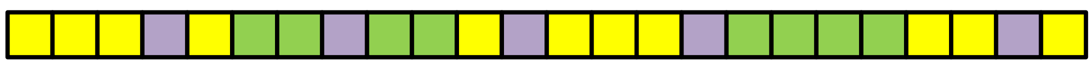
$$S(n) \leq S\left(\frac{7}{10}n\right) + M(n) + \Theta(n)$$

Median of Medians, Run Time

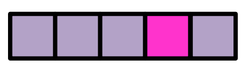
1. Break list into chunks of 5 $\Theta(n)$



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

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

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

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

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

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

Master theorem Case 3!

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

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

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?

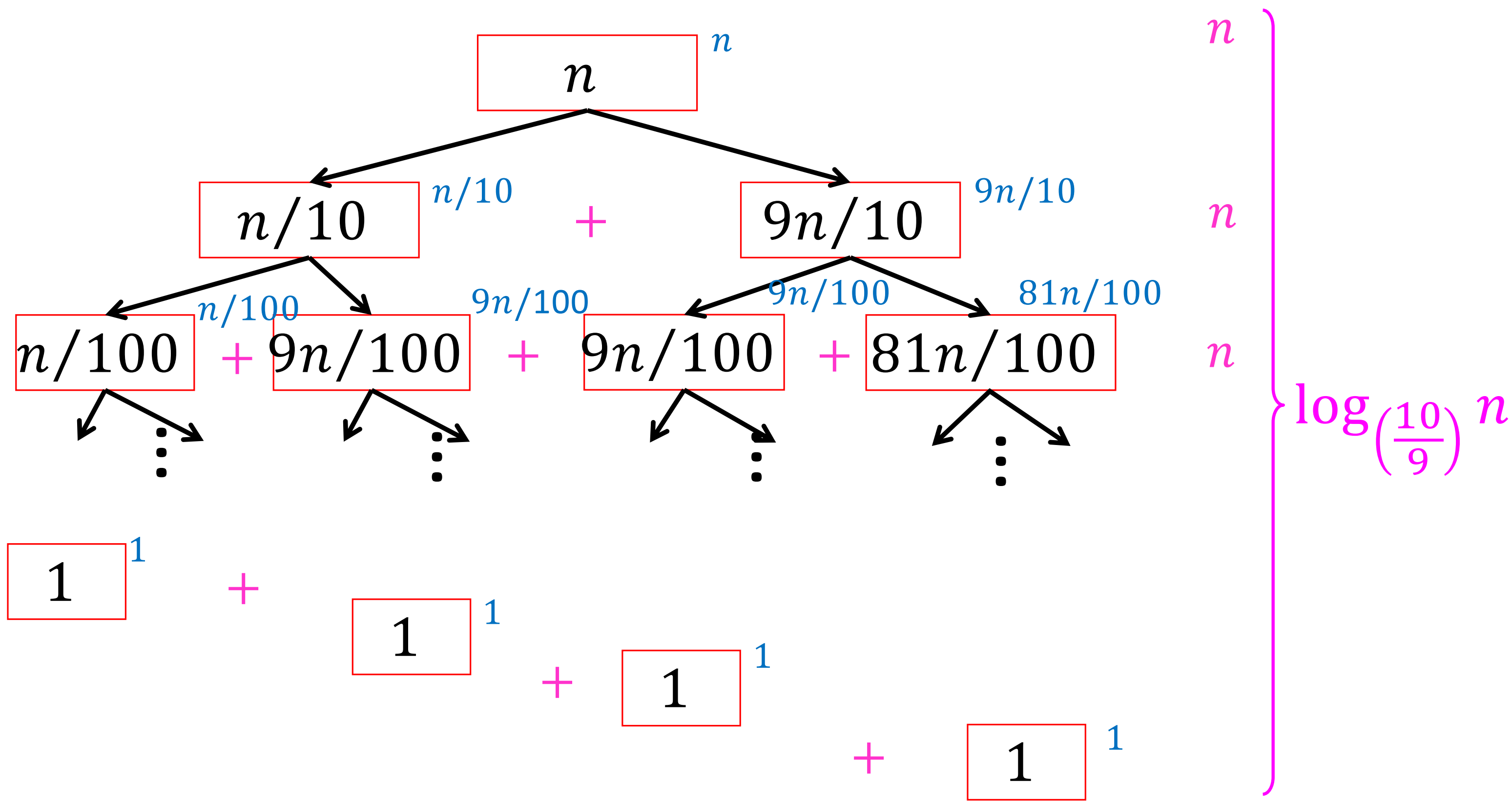
- Using Quickselect to pick median guarantees $\Theta(n \log n)$ run time
- Approach has very large constants
  - If you really want $\Theta(n \log n)$, better off using MergeSort
- Better approach: Random pivot
  - Very small constant (very fast algorithm)
  - Expected to run in $\Theta(n \log n)$ time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

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

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 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

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 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 $\Theta(n \log n)$