CS 4102: Algorithms

Lecture 7: Matrix Multiplication, Quicksort

David Wu

Fall 2019

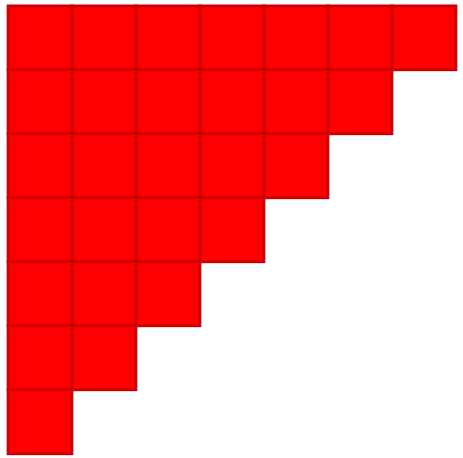
Warm Up

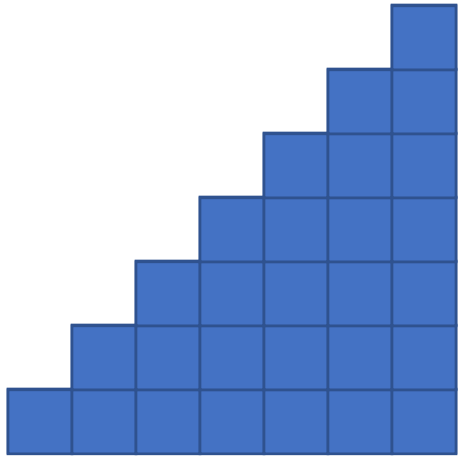
Simplify:

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

Warm Up

$$1 + 2 + 3 + \cdots + (n-1) + n = \frac{n(n+1)}{2}$$





n

$n + 1$

Today's Keywords

Matrix multiplication

Strassen's algorithm

Sorting

Quicksort

Quickselect (if there is time)

CLRS Readings: Chapter 4 and 7

Homework

HW2 due **Thursday, September 19, 11pm**

- Programming assignment (Python or Java)
- Divide and conquer (Closest pair of points)

Matrix Multiplication

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

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

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

Run time? $O(n^3)$

Lower Bound? $\Omega(n^2)$

Matrix Multiplication Divide and Conquer

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 Divide and Conquer

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? $T(n) = 8T\left(\frac{n}{2}\right) + 4\left(\frac{n}{2}\right)^2$ Cost of additions

Matrix Multiplication Divide and Conquer

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

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

$a = 8, b = 2, f(n) = n^2$

Case 1!

$n^{\log_b a} = n^{\log_2 8} = n^3$

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

Can we do better?

Matrix Multiplication D&C

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

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

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

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:

$$AB = \begin{bmatrix} Q_1 + Q_4 - Q_5 + Q_7 & Q_3 + Q_5 \\ Q_2 + Q_4 & Q_1 - Q_2 + Q_3 + Q_6 \end{bmatrix}$$

7 Multiplications

18 Additions

$$T(n) = 7T\left(\frac{n}{2}\right) + 18\frac{n^2}{4}$$

Strassen's Algorithm

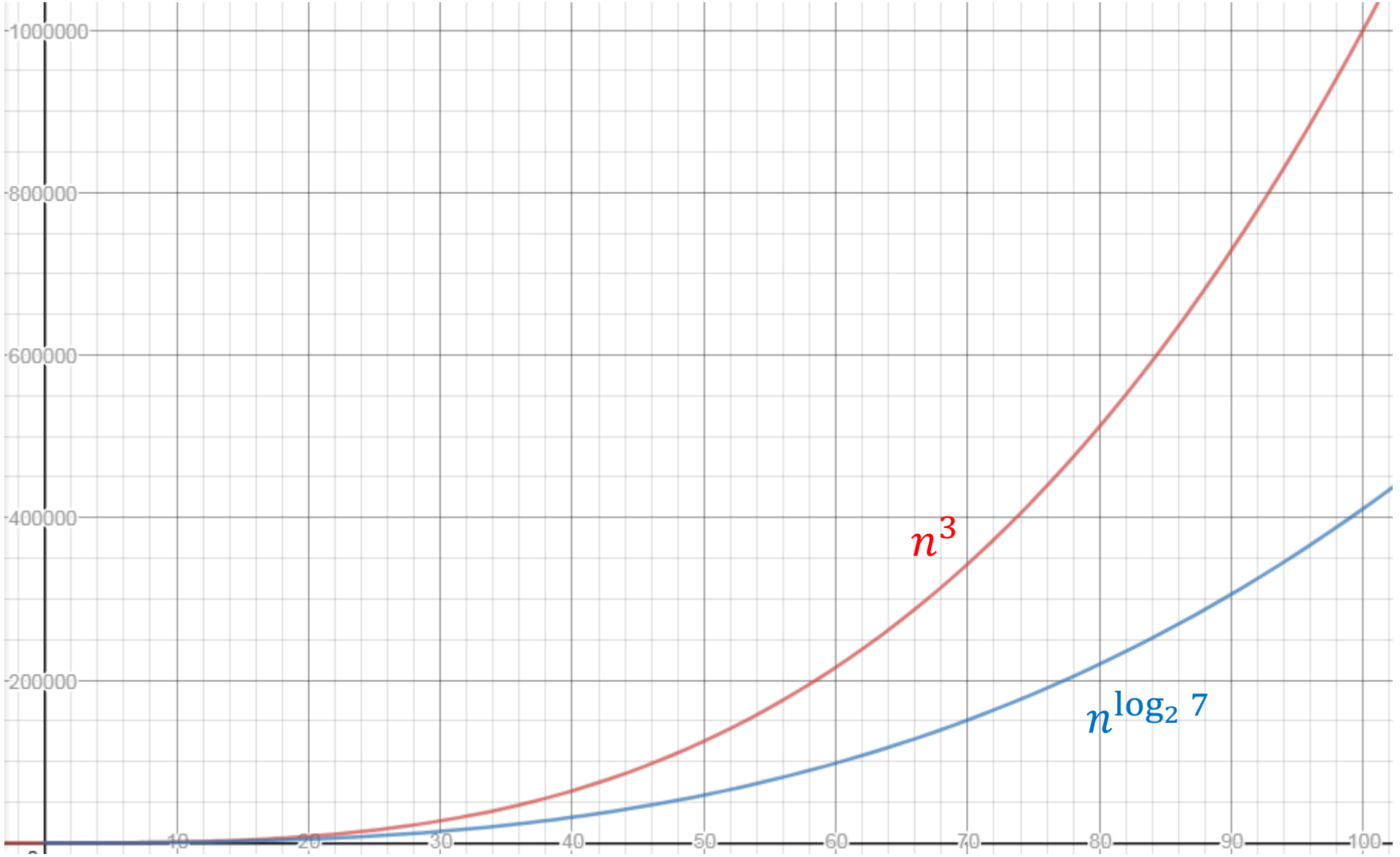
$$T(n) = 7T\left(\frac{n}{2}\right) + \frac{9}{2}n^2$$

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

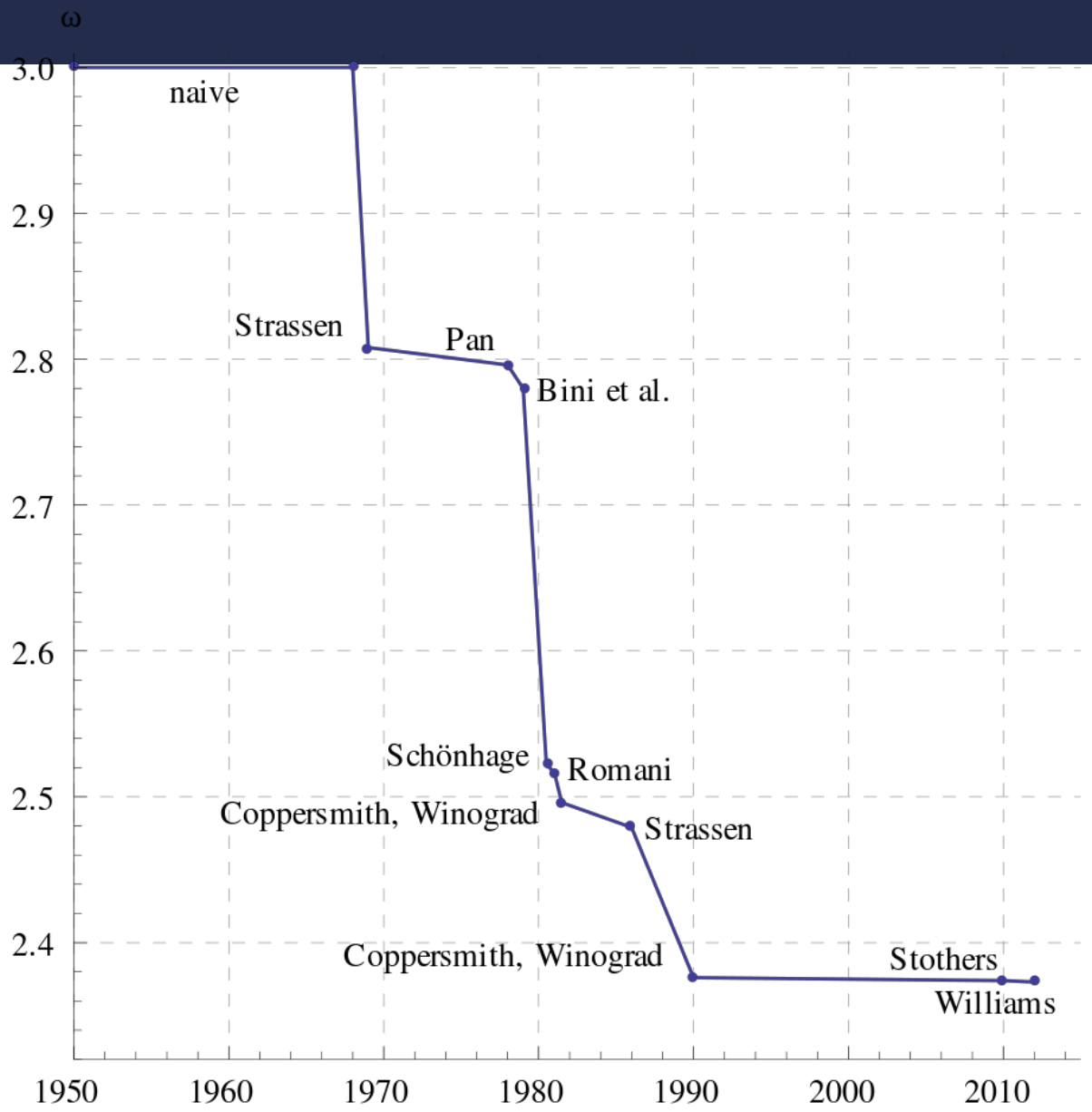
Case 1!

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

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


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 still unknown

Best lower bound: $\Omega(n^2)$

Divide and Conquer Algorithms (Thus Far)

Mergesort

Naïve Multiplication

Karatsuba Multiplication

Closest Pair of Points

Strassen's Algorithm

What they have in common:

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

Combine: More complex ($\Omega(n)$)

Quicksort

Like Mergesort:

- Divide and conquer algorithm
- $O(n \log n)$ run time (on expectation)

Unlike Mergesort:

- **Divide** step is the hard part
- Typically faster than Mergesort (often is the basis of sorting algorithms in standard library implementations)

Quicksort

General idea: choose a pivot element, recursively sort two sublists around that element

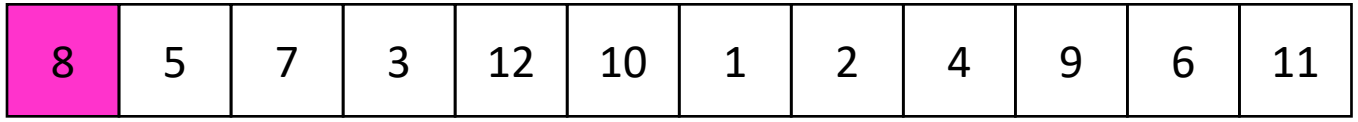
Divide: select pivot element p, Partition(p)

Conquer: recursively sort left and right sublists

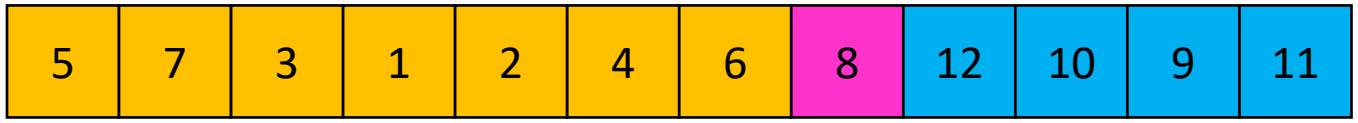
Combine: nothing!

Partition Procedure (Divide Step)

Input: an unordered list, a pivot p

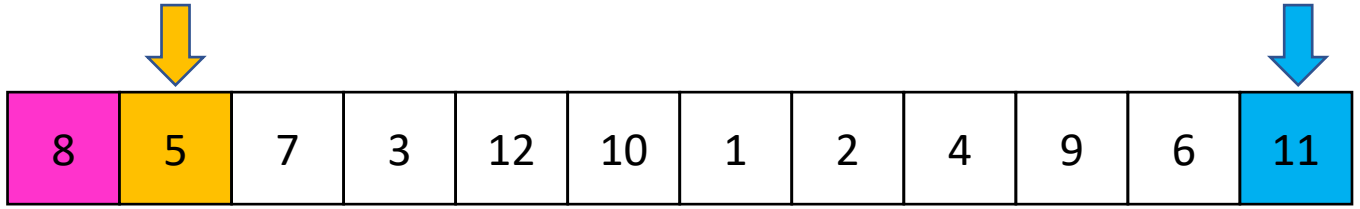


Goal: All elements $< p$ on left, all $\geq p$ on right



Partition Procedure

Initialize two pointers Begin and End

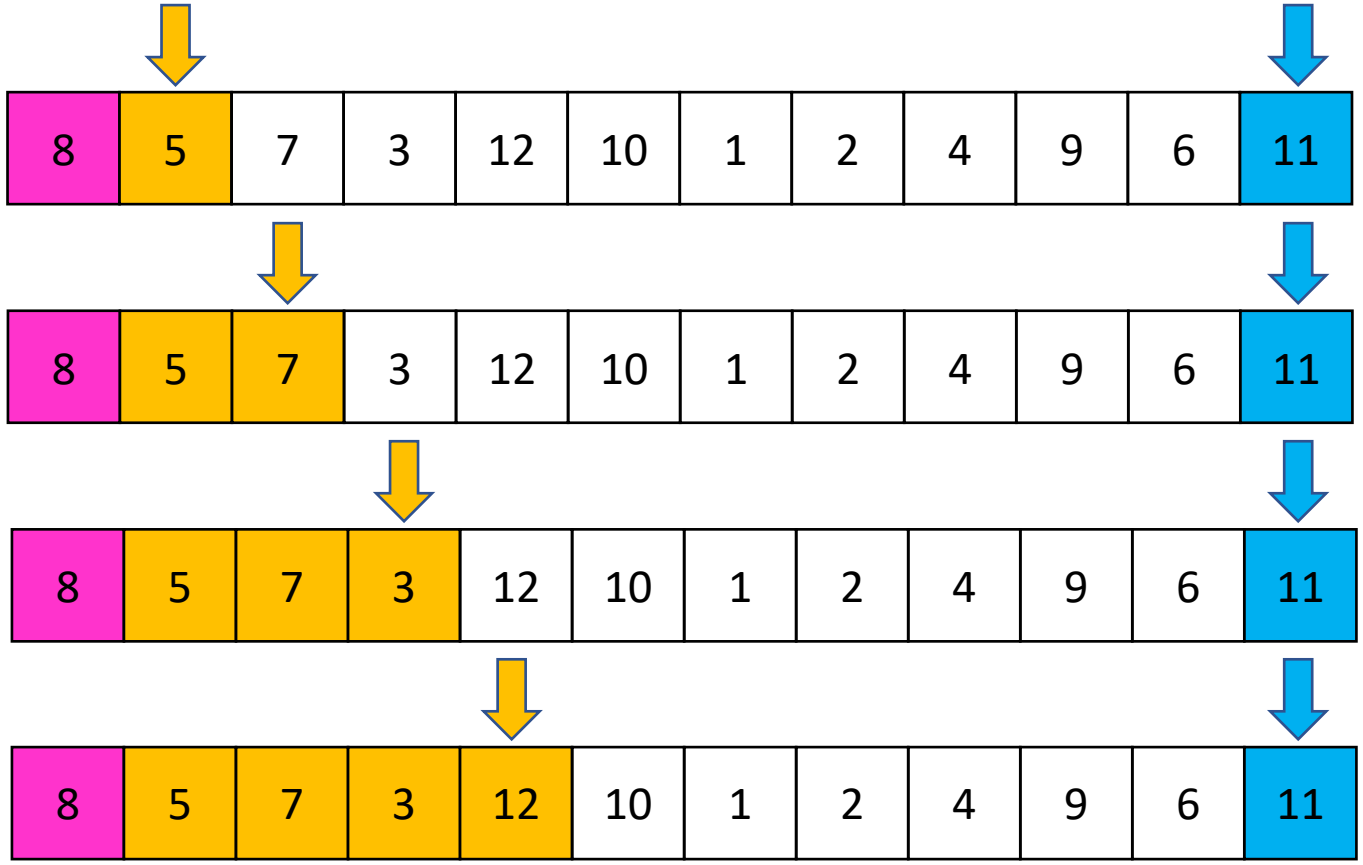


Partition Procedure

If Begin value $<$ p, move Begin right

Else swap Begin value with End value, move End Left

Stop when Begin = End



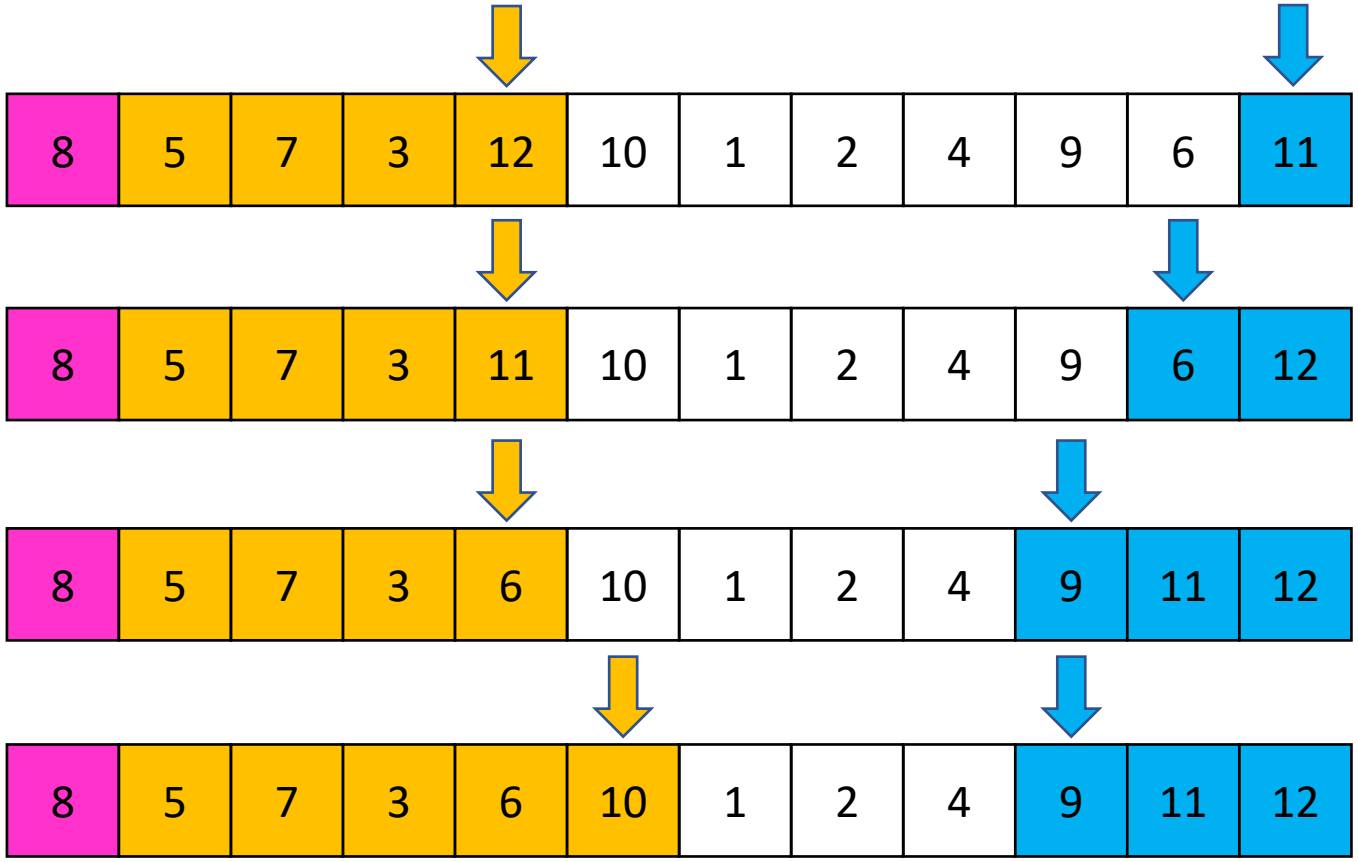
Swap!

Partition Procedure

If Begin value $<$ p, move Begin right

Else swap Begin value with End value, move End Left

Stop when Begin = End

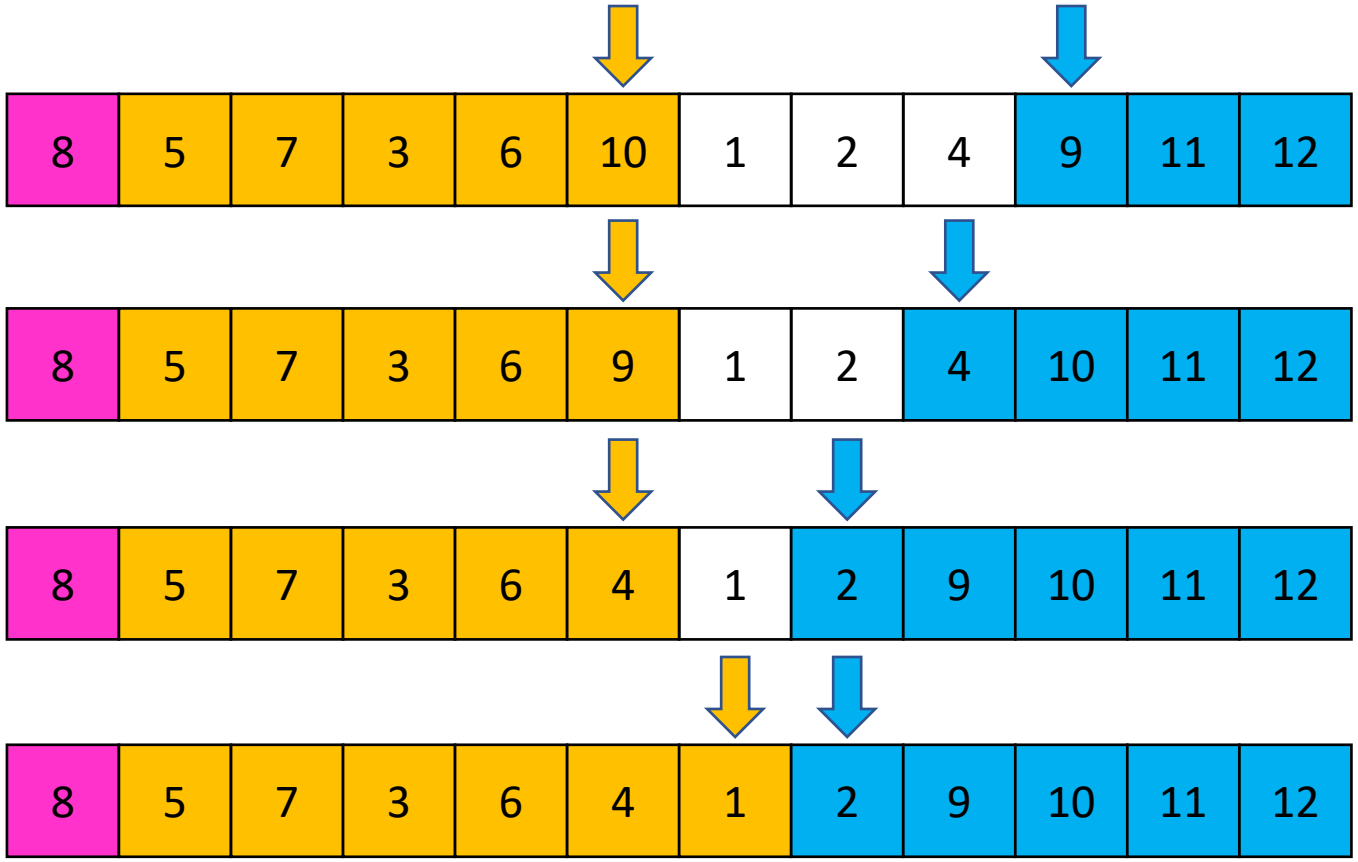


Partition Procedure

If Begin value $<$ p, move Begin right

Else swap Begin value with End value, move End Left

Stop when Begin = End

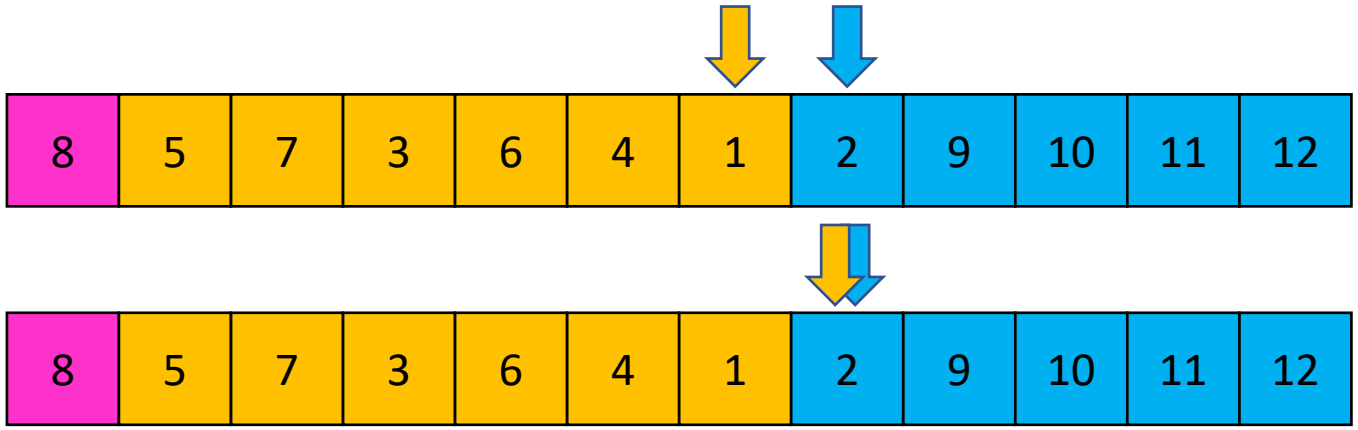


Partition Procedure

If Begin value $<$ p, move Begin right

Else swap Begin value with End value, move End Left

Stop when Begin = End



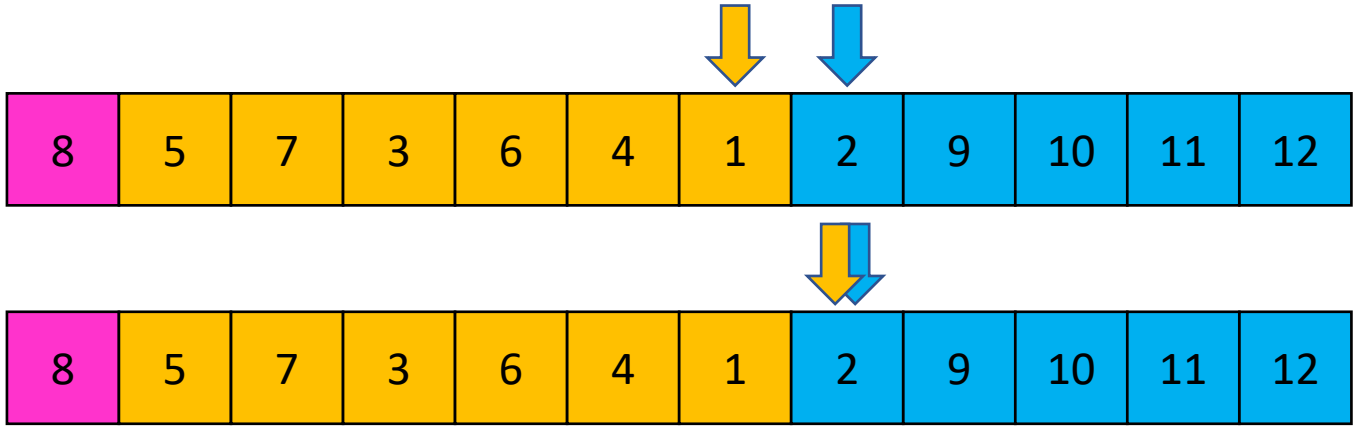
Remaining item: where do we place the pivot?

Partition Procedure

If Begin value $<$ p, move Begin right

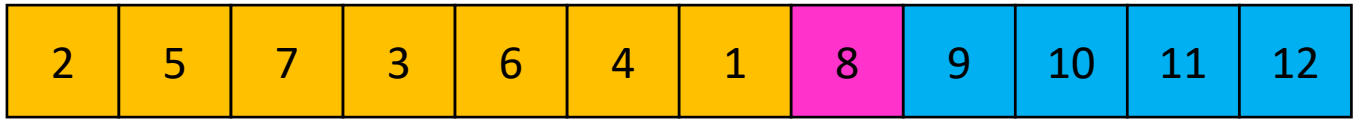
Else swap Begin value with End value, move End Left

Stop when Begin = End



Case 1: meet at element $< p$

Swap p with pointer position

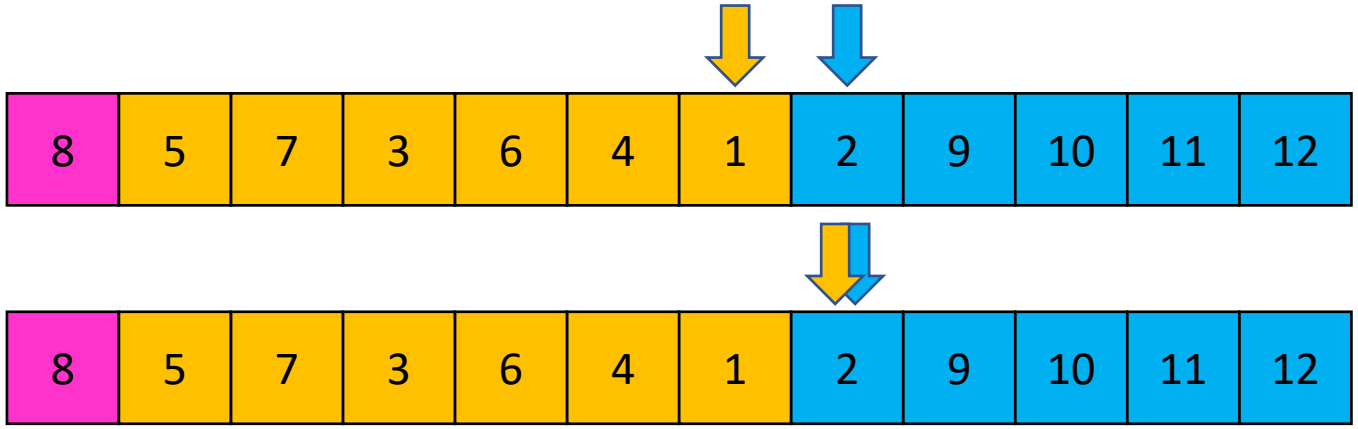


Partition Procedure

If Begin value $<$ p, move Begin right

Else swap Begin value with End value, move End Left

Stop when Begin = End



Case 2: meet at element $>$ p

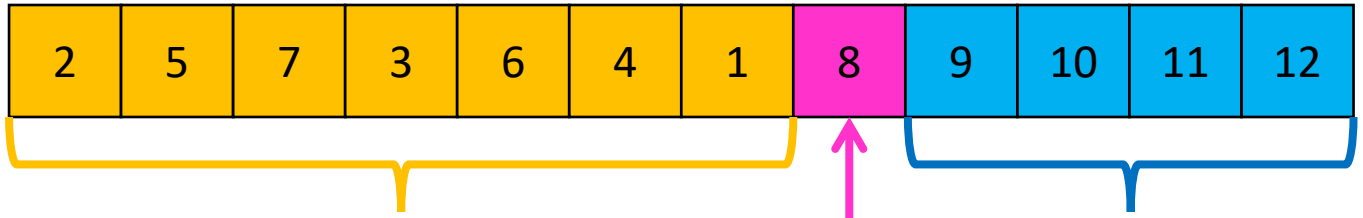
Swap p with value to the left

Partition Procedure Summary

1. Choose the pivot p to be the first element of the list
2. Initialize two pointers Begin (just after p), and End (at end of list)
3. While Begin < End:
   - If value of Begin $< p$, advance Begin to the right
   - Otherwise, swap value of Begin value with value of End value, and advance End to the left
4. If pointers meet at element $< p$: swap p with pointer position
5. Otherwise, if pointers meet at element $> p$: swap p with value to the left

Run time? $\Theta(n)$

Conquer Step



All elements $< p$

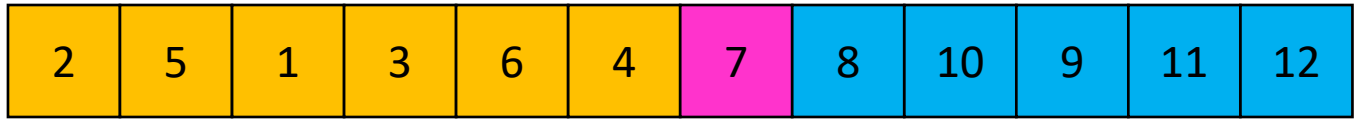
All elements $> p$

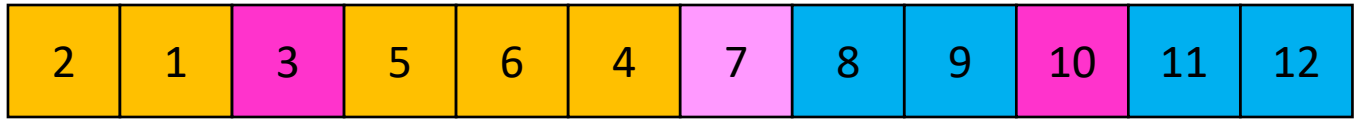
Exactly where it belongs!

Recursively sort Left and Right sublists

Quicksort Run Time (Optimistic)

If the pivot is the median:



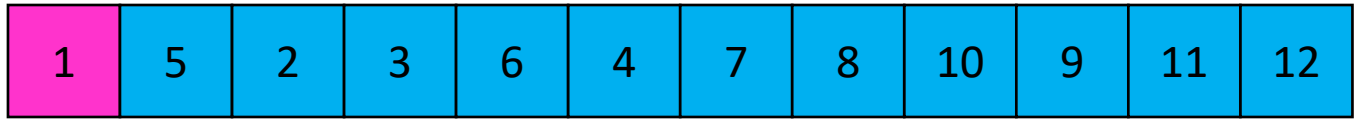


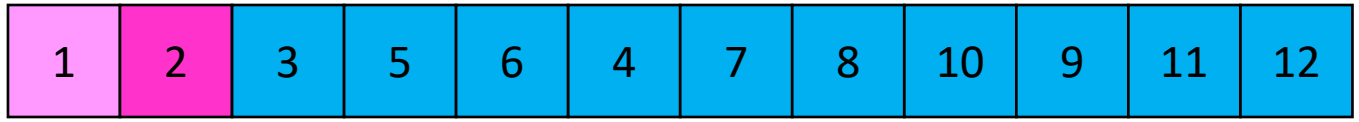
Then we divide in half each time

$$T(n) = 2T(n/2) + n = \Theta(n \log n)$$

Quicksort Run Time (Worst-Case)

If the pivot is the extreme (min/max):



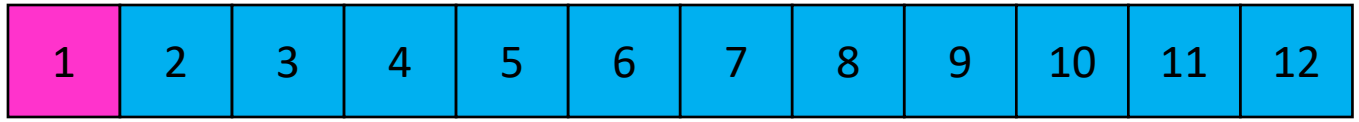


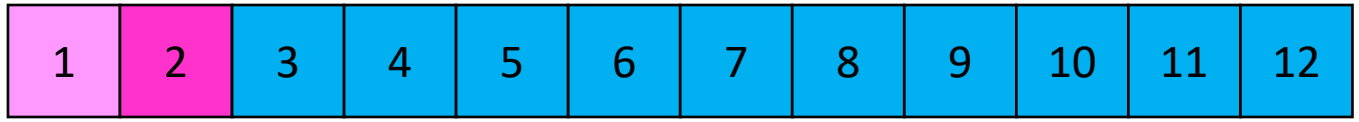
Then we shorten by 1 each time

$$T(n) = T(n-1) + n$$
$$= n + (n-1) + \cdots + 2 + 1$$
$$= \frac{n(n+1)}{2} = \Theta(n^2)$$

Quicksort on a Nearly Sorted List

First element always yields unbalanced pivot





Then we shorten by 1 each time

$$T(n) = \Theta(n^2)$$

How to Choose the Pivot?

Good choice: $\Theta(n \log n)$

Bad choice: $\Theta(n^2)$

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 algorithm

Quickselect Algorithm

Algorithm to compute the i^{th} order statistic

- i^{th} smallest element in the list
- 1^{st} order statistic: minimum
- n^{th} order statistic: maximum
- $(n/2)^{\text{th}}$ order statistic: median

Quickselect

Finds i^{th} order statistic

General idea: choose a pivot element, partition around the pivot, and recurse on sublist containing index i

Divide: select pivot element p, Partition(p)

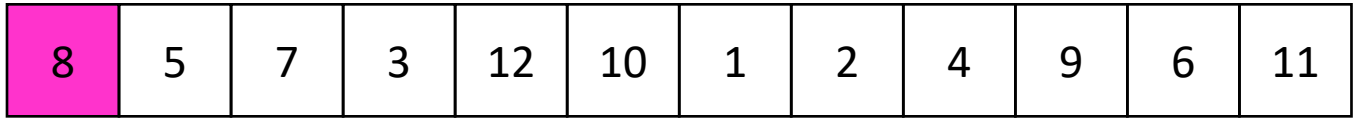
Conquer:

- if $i =$ index of p, then we are done and return p
- if $i <$ index of p recurse left. Otherwise, recurse right

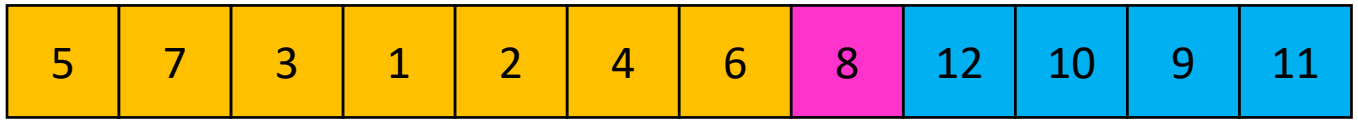
Combine: Nothing!

Partition Procedure (Divide Step)

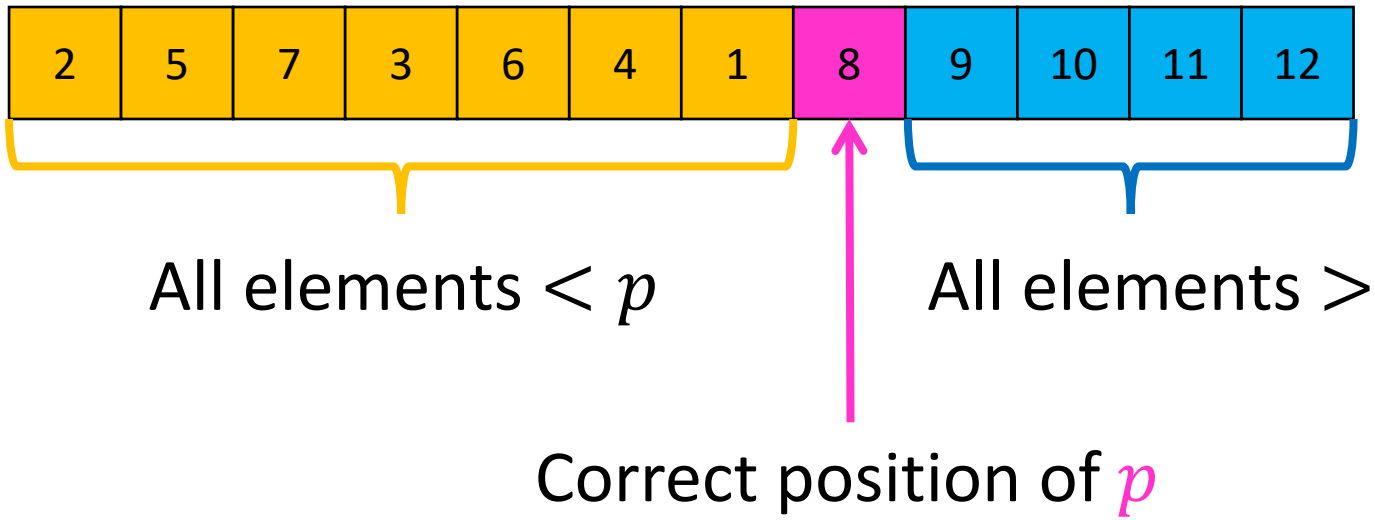
Input: an unordered list, a pivot p



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



Conquer Step

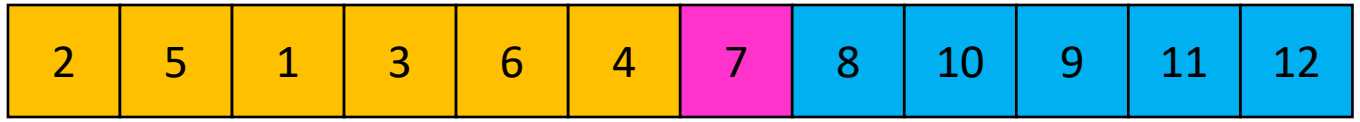


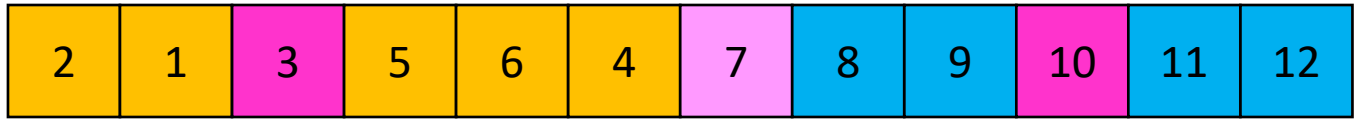
Recurse on sublist that contains index i

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

Quickselect Run Time (Optimistic)

If the pivot is the median:



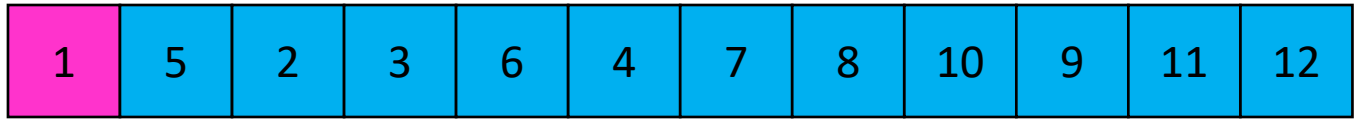


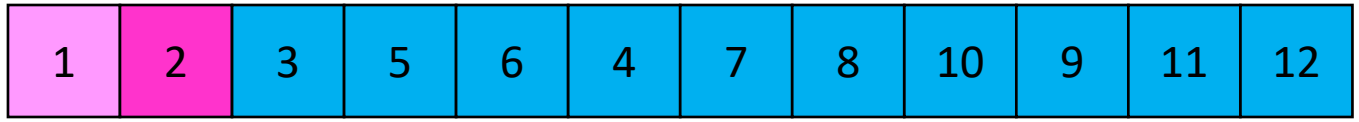
Then we divide in half each time

$$T(n) = T(n/2) + n = \Theta(n)$$

Quickselect Run Time (Worst-Case)

If the pivot is the extreme (min/max):





Then we shorten by 1 each time

$$T(n) = T(n-1) + n = \Theta(n^2)$$

How to Choose the Pivot?

Good choice: $\Theta(n)$

Bad choice: $\Theta(n^2)$

Good Pivot

What makes a good pivot?

- Roughly even split between left and right
- Ideally: median

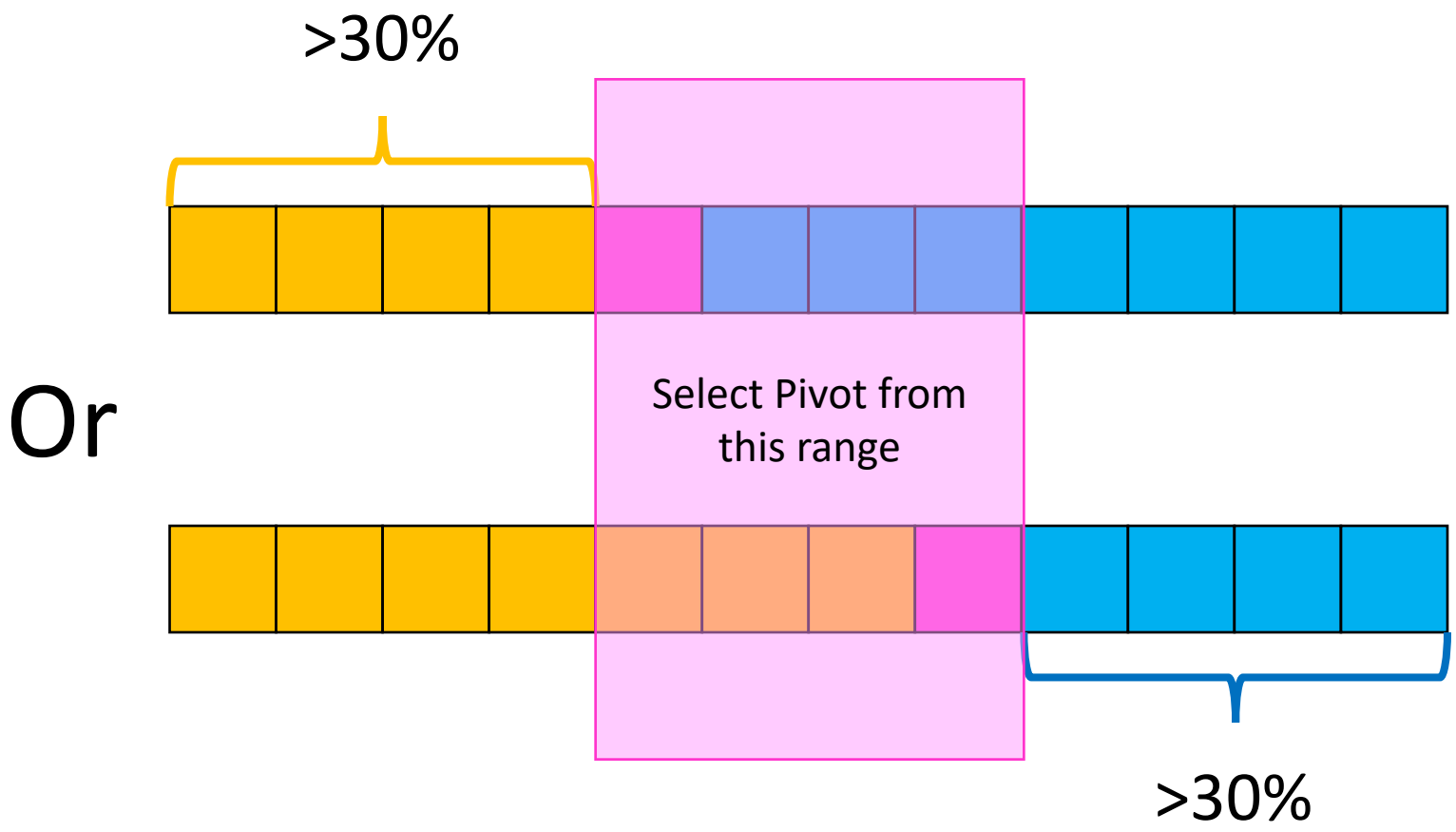
But this is the problem that Quickselect is supposed to solve!

Déjà vu?

What's next: an algorithm for choosing a "decent" pivot (median of medians)

Good Pivot

Decent 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

Main idea: break list into blocks, find the median of each blocks, use the median of those medians

Median of Medians

1. Break list into blocks of size 5

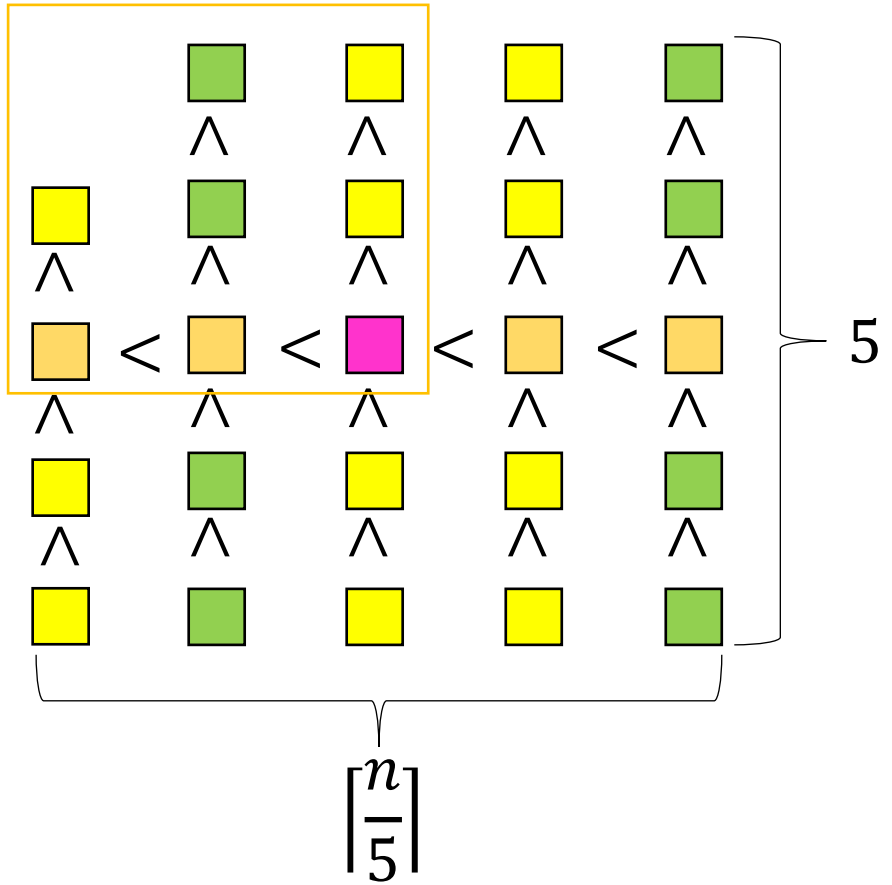
2. Find the median of each chunk

3. Return median of medians (using Quickselect)

Median of Medians

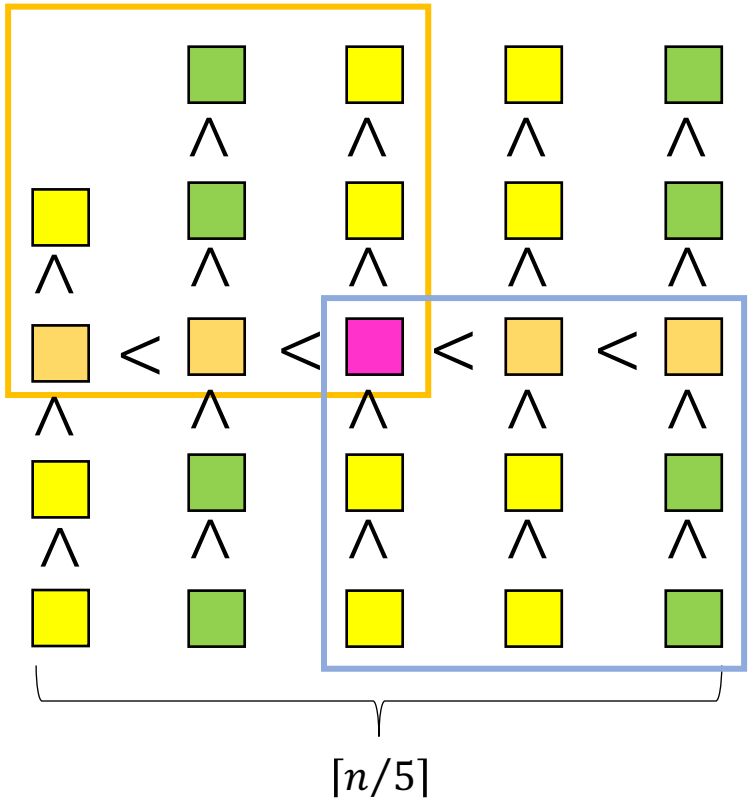
Each chunk sorted, chunks ordered by their medians

MedianofMedians is larger than all of these



Median of Medians

MedianofMedians is larger than all of these



Elements smaller than MedianofMedians:

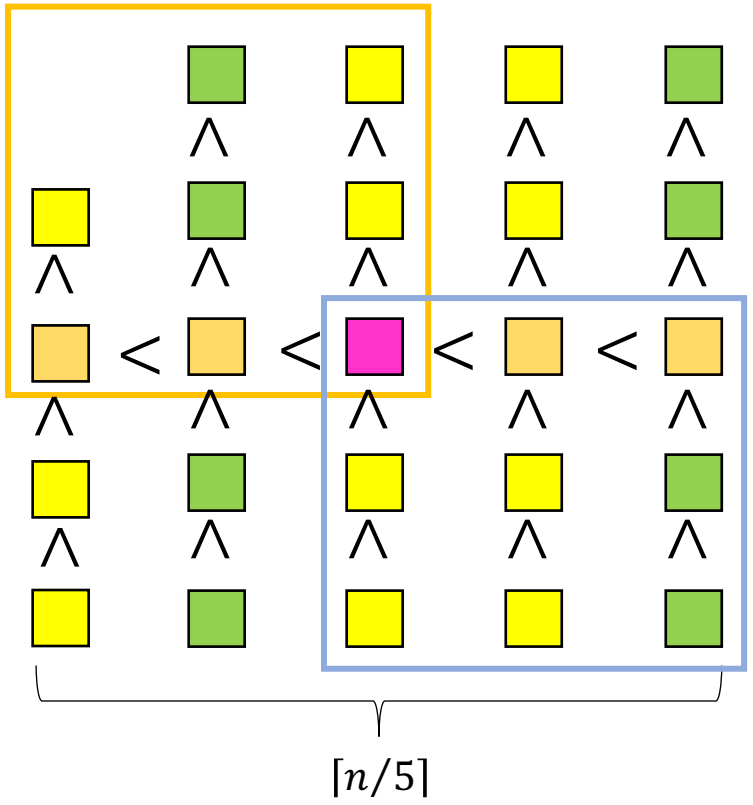
$$3\left(\left\lceil \frac{1}{2} \cdot \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6 \text{ elements}$$

Number of lists to the "left"

Exclude list on the endpoint, and "middle" list

Median of Medians

MedianofMedians is larger than all of these



Elements smaller than MedianofMedians:

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

Elements greater than MedianofMedians:

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

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

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

Combine: Nothing!

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

Median of Medians

1. Break list into blocks of size 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

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

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

Combine: Nothing!

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

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

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

Combine: Nothing!

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