CSE 421 Winter 2025
Lecture 11: Quicksort and Medians

Nathan Brunelle

http://www.cs.uw.edu/421

$f(n+m)$ vs. $f(n) + f(m)$

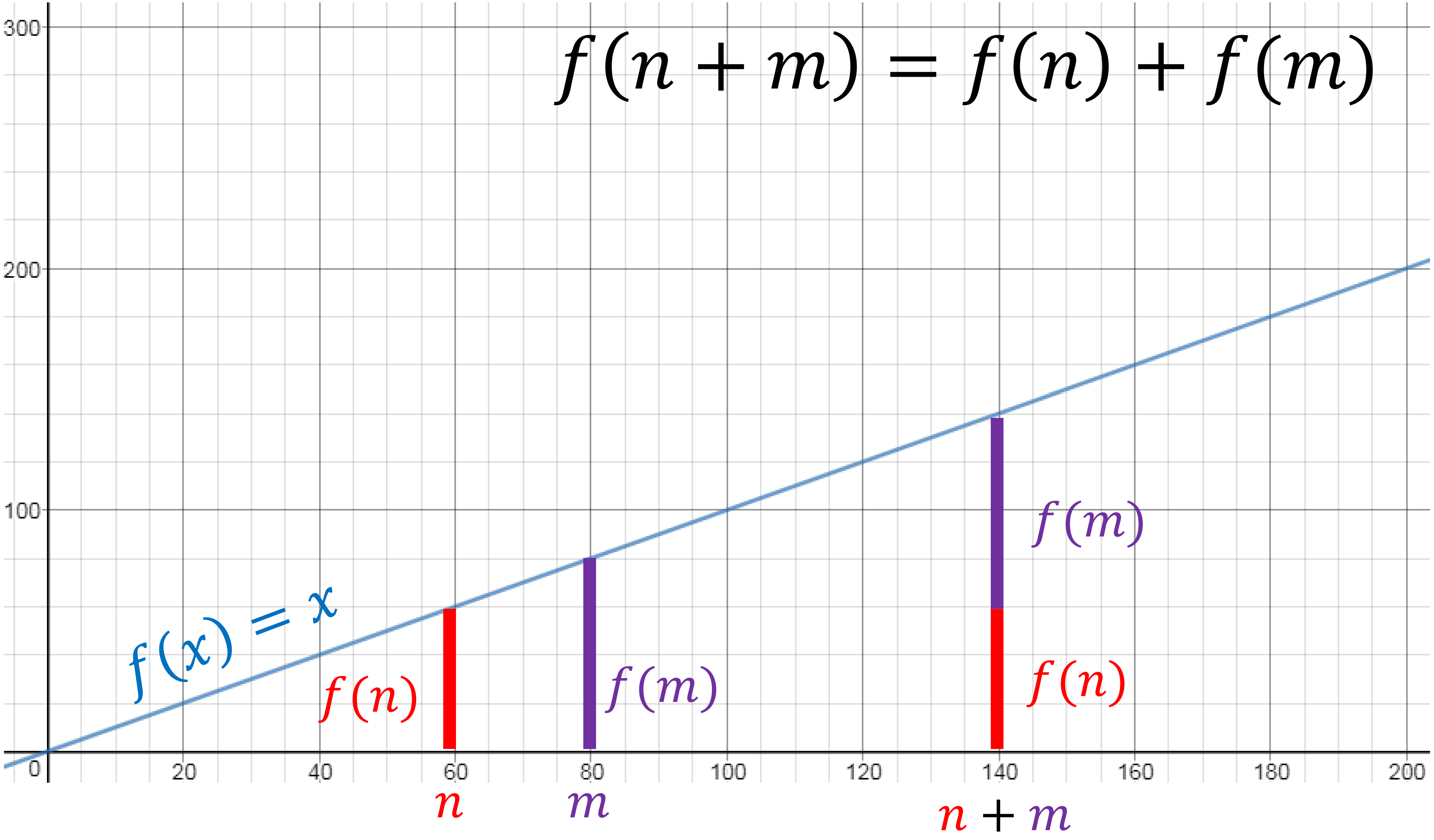
n^2

- When is each true?
  - $f(n+m) = f(n) + f(m)$ $3 \cdot n$
  - $f(n+m) < f(n) + f(m)$ $\sqrt{n}$
  - $f(n+m) > f(n) + f(m)$

$(2+3)^2 = 25$

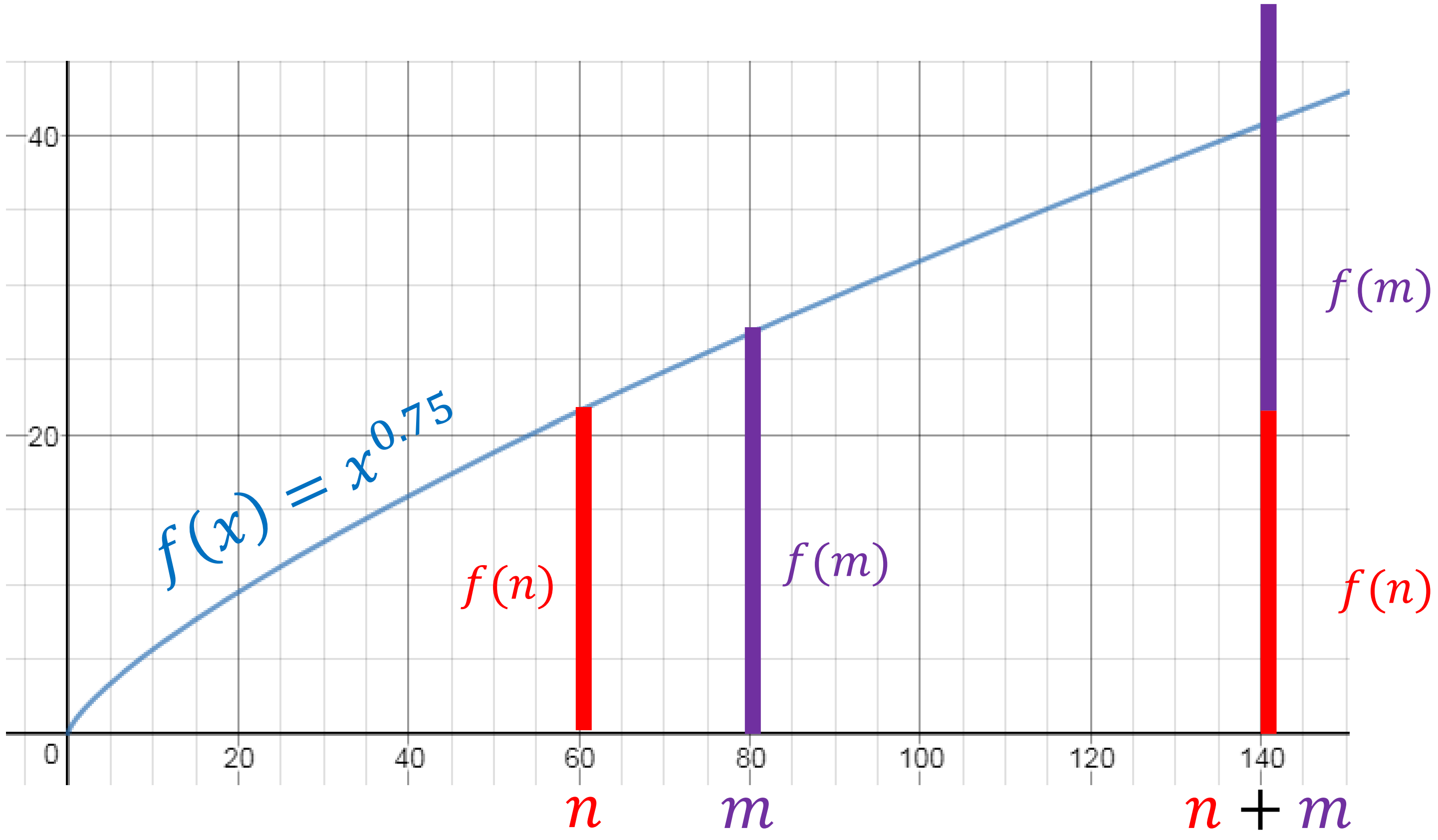
$2^2 + 3^2 = 13$

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



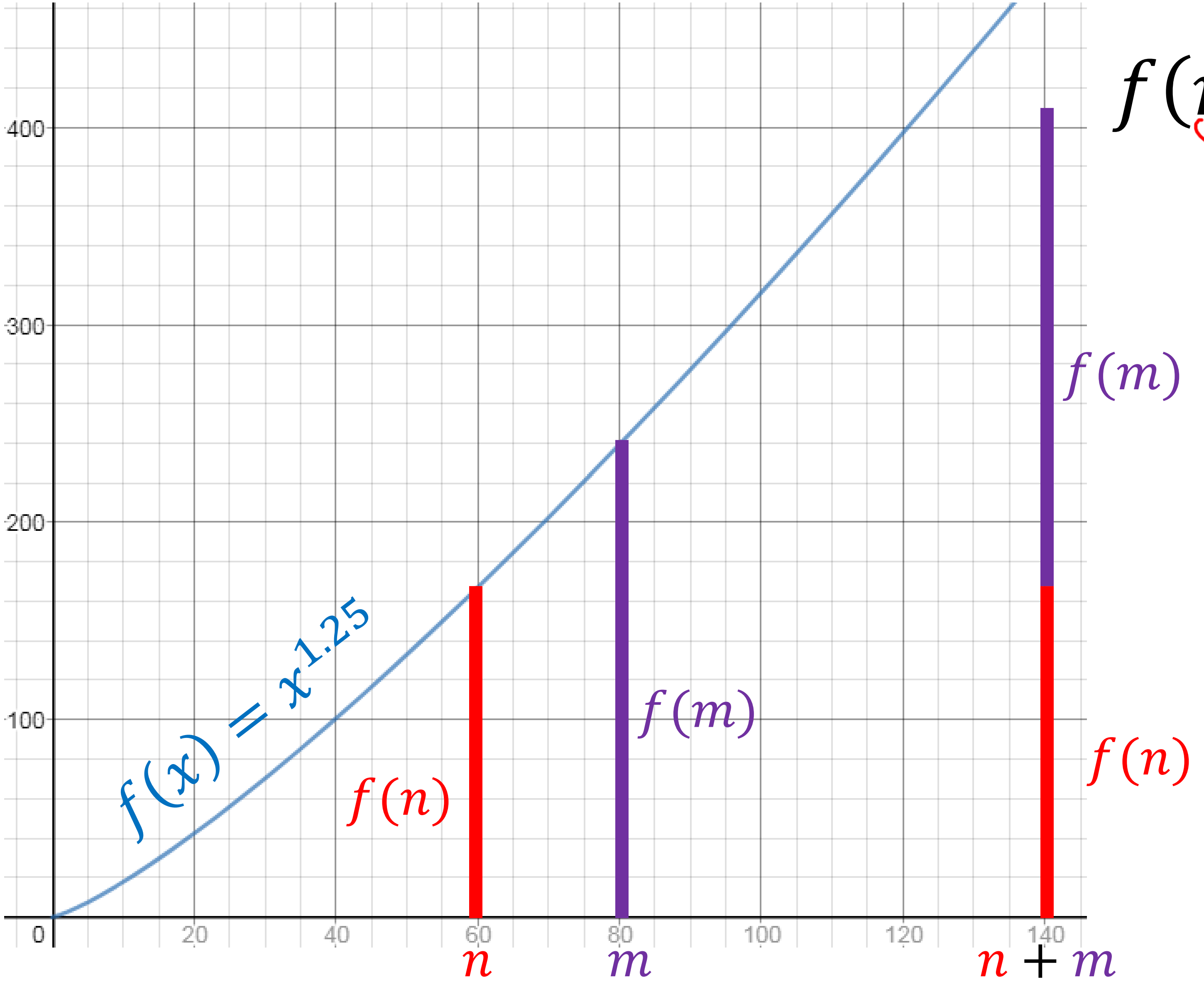
$f(n+m) = f(n) + f(m)$
$f(x) = x$
$f(n)$
$f(m)$
$f(m)$
$f(n)$
300
200
100
0
20
40
60
80
100
120
140
160
180
200
n
m
$n+m$

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



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

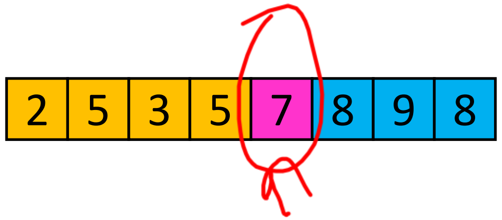
$$f(n) \in \Omega(n)$$



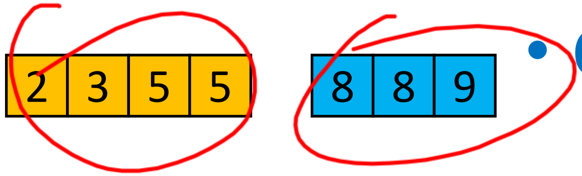
$$f(n+m) \geq f(n) + f(m)$$

Divide and Conquer (Quick Sort)

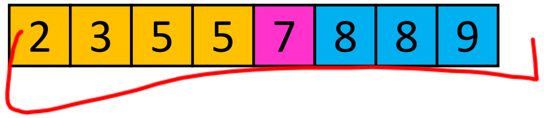
- **Base Case:**
  - If the list is of length 1 or 0, it’s already sorted, so just return
  - (Alternative: when length is ≤ 15, use insertion sort)



- **Divide:**
  - Select an element to use as a “pivot”
  - Partition: rearrange the list so that all elements less than the pivot are to the left of the pivot, all elements greater are to the right



- **Conquer:**
  - Sort the sublists to the left and right of the pivot recursively.



- **Combine:**
  - Nothing!

Partition

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

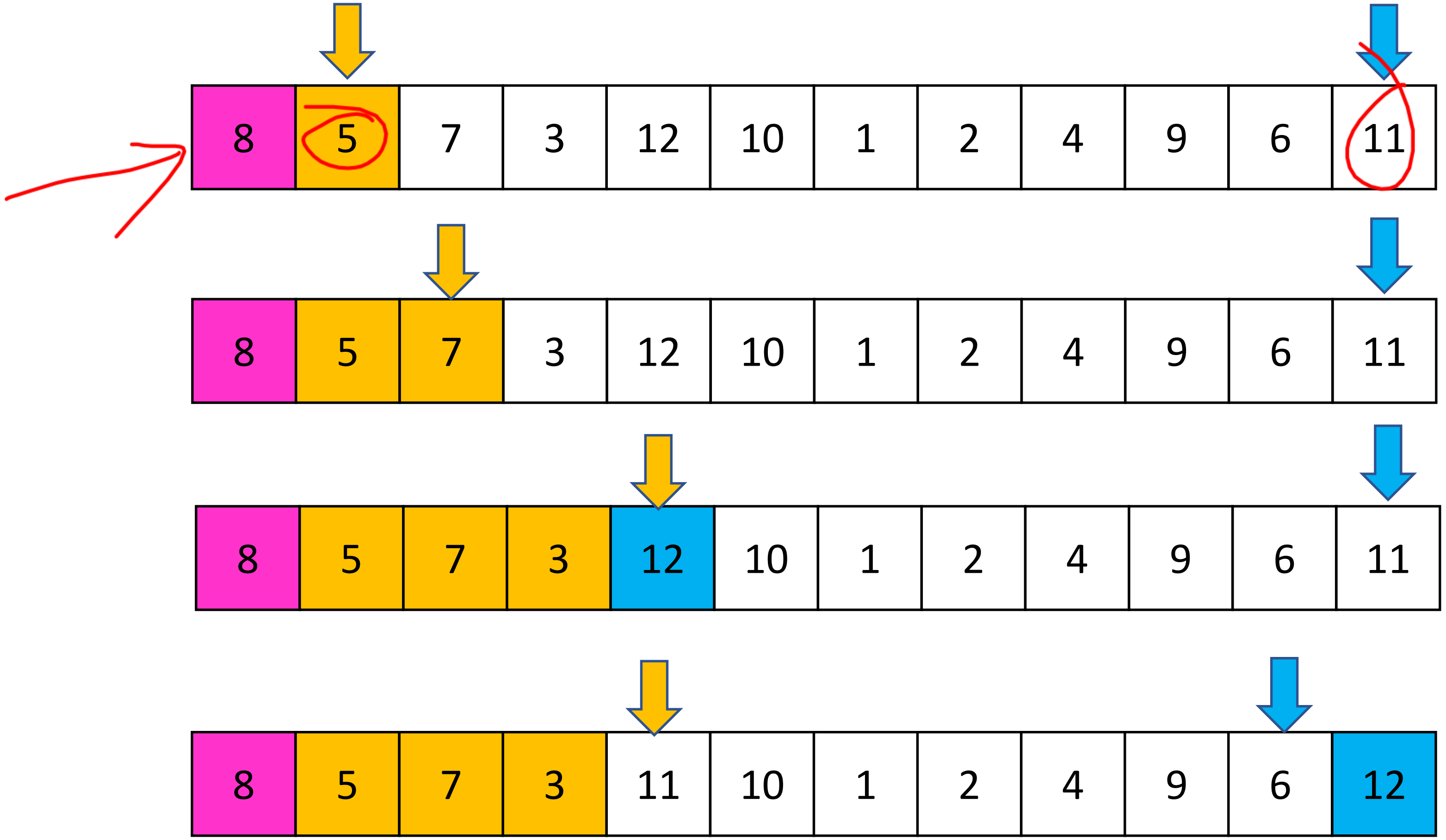
Run time? $O(n)$

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	11	10	1	2	4	9	6	12

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

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

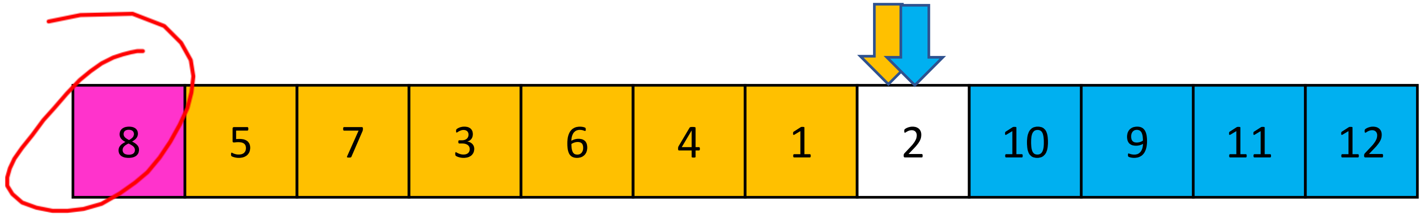
8	5	7	3	6	10	1	2	4	9	11	12

Partition, Procedure

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

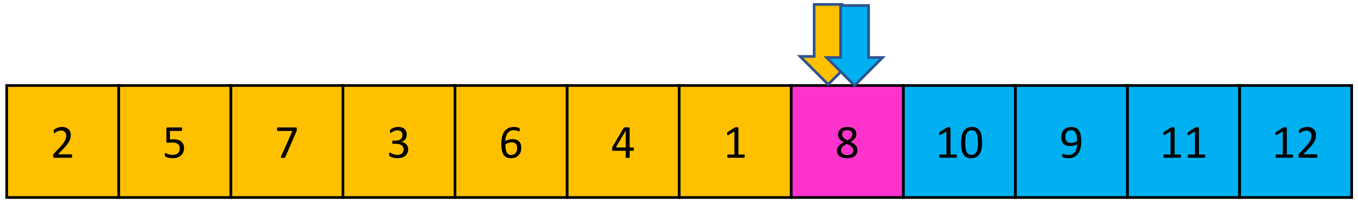
Else swap Begin value with End value, move End Left

Done when Begin = End



Case 1: meet at element $< p$

Swap p with pointer position (2 in this case)



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 2: meet at element $> p$

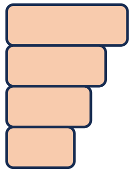
Swap p with value to the left (2 in this case)

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

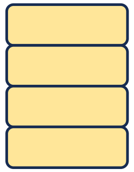
Quick Sort Running Time

Master Theorem: Suppose that $T(n) = a \cdot T(n/b) + O(n^k)$ for $n > b$.

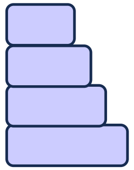
- If $a < b^k$ then $T(n)$ is $O(n^k)$
  - Cost is dominated by work at top level of recursion



- If $a = b^k$ then $T(n)$ is $O(n^k \log n)$
  - Total cost is the same for all $\log_b n$ levels of recursion



- If $a > b^k$ then $T(n)$ is $O(n^{\log_b a})$
  - Note that $\log_b a > k$ in this case
  - Cost is dominated by total work at lowest level of recursion



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

$a = 2, b = 2, k = 1$ so $a = b^k$: Solution: $O(n\log n)$

Worst Case: $T(n) = T(n-1) + n$

The master theorem does not apply, but the solution is $O(n^2)$

Expected: $O(n\log n)$ … Our first task today is to show this!

Expected Runtime for QuickSort: "Global analysis"

Runtime is the # of comparisons

Recurrence & Master Theorem kind of analysis won't work ...

Instead, use a "global" analysis:

- Number elements $\boldsymbol{a_1, a_2, \ldots, a_n}$ based on **final** sorted order
- Let $\boldsymbol{p_{i,j}}$ = Probability that QuickSort compares $\boldsymbol{a_i}$ and $\boldsymbol{a_j}$

Expected number of comparisons:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{i,j}$$

Observation – We only compare to pivot

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 $\boxed{< p,}$ move Begin right
    2. Else swap Begin value with End value, move End Left
4. If pointers meet at element $\boxed{< p}$: Swap p with pointer position
5. Else If pointers meet at element $\boxed{> p}$: Swap p with value to the left

Conclusion: we only compare a_i with a_j when both of these are true:

- a_i and a_j are in the same subproblem (no previous pivot fell between them)
- One of a_i and a_j is the pivot

Finding $p_{i,j}$ - Adjacent Values

We always compare consecutive elements

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

$$p_{i,i+1} = 1$$

Why?

- Intuitively, we MUST compare adjacent items to guarantee we get them in the right order
- Precisely, a_i and a_{i+1} can't be on opposite sides of any third value, so they'll only end up in separate subproblems when one was the pivot

Finding $p_{i,j}$ - Extreme Values

We rarely compare the min and the max

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

$$p_{1,n} = \frac{2}{n}$$

Why?

– The only way we will compare the smallest and largest items is if one or the other was the very first pivot chosen ($\frac{1}{n}$ chance of each)

Finding $p_{i,j}$ - In General

For any pair of elements, the probability we compare them in proportional to their distance

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

$$p_{i,j} = \frac{2}{j-i+1} \quad \text{if } i < j$$

Why?

- a_i and a_j will only be compared if one of them was the very first pivot chosen from among the range $a_i, a_{i+1}, \dots, a_j$
- There are $j - i + 1$ items in this range, two of which result in this comparison

Expected Runtime for QuickSort: "Global analysis"

For $i < j$ we have $p_{i,j} = \frac{2}{j-i+1}$.

Expected number of comparisons:

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

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

for $k = j - i$

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

$$< 2\,n\,H_n$$

$$= 2\,n \ln n + O(n) \leq 1.387\, n \log_2 n$$

Harmonic series sum:

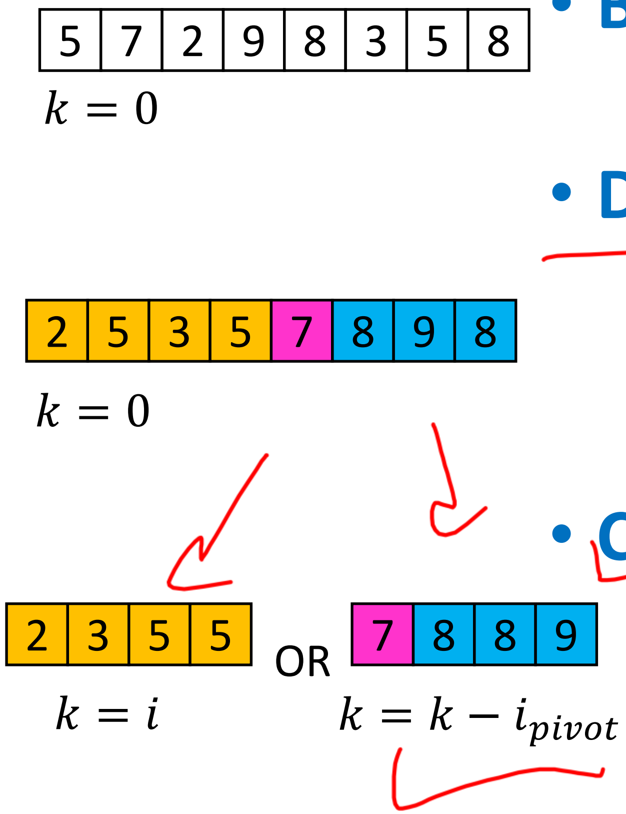
$$H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$$

Fact: $H_n = \ln n + O(1)$

QuickSelect

- Finds k^{th} order statistic
  - k^{th} smallest element in the list
  - 1^{st} order statistic: minimum
  - n^{th} order statistic: maximum
  - $\left\lfloor \frac{n}{2} \right\rfloor^{\text{th}}$ order statistic: median (for odd n)

QuickSelect(S, k)

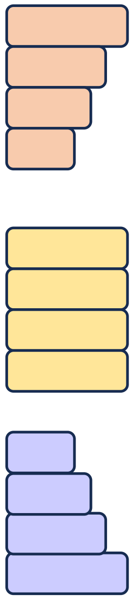


- **Base Case:**
  - If $k = 0$ then return $S[0]$
- **Divide:**
  - Select an element to use as a "pivot"
  - Partition: rearrange S so that all elements less than the pivot are to the left of the pivot, all elements greater are to the right
- **Conquer:**
  - Let i_{pivot} be the index of the pivot after partitioning
  - If $k < i_{pivot}$ then call QuickSelect(S_{left}, k)
  - Otherwise call Quickselect$(S_{right}, k - i_{pivot})$
- **Combine:**
  - Nothing!

QuickSelect Running Time

Master Theorem: Suppose that $T(n) = a \cdot T(n/b) + O(n^k)$ for $n > b$.

- If $a < b^k$ then $T(n)$ is $O(n^k)$
  - Cost is dominated by work at top level of recursion
- If $a = b^k$ then $T(n)$ is $O(n^k \log n)$
  - Total cost is the same for all $\log_b n$ levels of recursion
- If $a > b^k$ then $T(n)$ is $O(n^{\log_b a})$
  - Note that $\log_b a > k$ in this case
  - Cost is dominated by total work at lowest level of recursion



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

$a = 1, b = 2, k = 1$ so $a < b^k$: Solution: $O(n)$

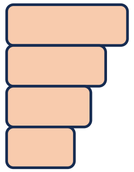
Worst Case: $T(n) = T(n-1) + n$

The master theorem does not apply, but the solution is $O(n^2)$

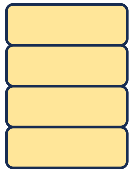
We don’t need the “ideal” for $O(n)$!

Master Theorem: Suppose that $T(n) = a \cdot T(n/b) + O(n^k)$ for $n > b$.

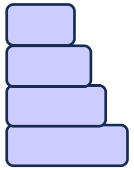
- If $a < b^k$ then $T(n)$ is $O(n^k)$
  - Cost is dominated by work at top level of recursion



- If $a = b^k$ then $T(n)$ is $O(n^k \log n)$
  - Total cost is the same for all $\log_b n$ levels of recursion



- If $a > b^k$ then $T(n)$ is $O(n^{\log_b a})$
  - Note that $\log_b a > k$ in this case
  - Cost is dominated by total work at lowest level of recursion



$$T(n) = T\left(\frac{3n}{4}\right) + n$$

$a = 1$, $b = 4/3$, $k = 1$ so $a < b^k$: Solution: $O(n)$

QuickSelect: Random Choice of Pivot

Consider a call to QuickSelect. We will say the pivot is "good enough" if it is among the middle half of the value

Elements of $\boldsymbol{S}$ listed in sorted order

Say $\boldsymbol{p}$ is "good enough" iff it is in the middle half

With probability $\geq \mathbf{1/2}$ pivot $\boldsymbol{p}$ is good enough

- For any good enough pivot the recursive call has subproblem size $\leq \mathbf{3}\boldsymbol{n}/\mathbf{4}$
- After 2 calls, QuickSelect has expected problem size $\leq \mathbf{3}\boldsymbol{n}/\mathbf{4}$

So $\boldsymbol{T(n) \leq 2T'(n)}$ where

$$\boldsymbol{T'(n) = T'\left(\frac{3n}{4}\right) + n} \text{ for } \boldsymbol{b = 4/3 > 1}$$

$\Rightarrow$ Expected $\boldsymbol{O(n)}$ time

Doing Quickselect in $O(n)$ Worst Case

- We can make adapt Quickselect by running in $O(n)$ worst case by applying some tricky extra recursion!
- Median-of-Medians:
  1. Break S into chunks of size 5, sort them
  2. Sort each chunk by its median value (i.e. value at index 2)
  3. Use Quickselect to find the median of these medians, use that as the pivot

M-o-M, Step 1: Construct sets of size 5; Step 2: sort each set

Input: 13, 15, 32, 14, 95, 5, 16, 45, 86, 65, 62, 41, 81, 52, 32, 32, 12, 73, 25, 81, 47, 8, 69, 9, 7, 81, 18, 25, 42, 91, 64, 98, 96, 91, 6, 51, 21, 12, 36, 11, 11, 9, 5, 17, 77

Group:

13	5	62	32	47	81	64	51	11
15	16	41	12	8	18	98	21	9
32	45	81	73	69	25	96	12	5
14	86	52	25	9	42	91	36	17
95	65	32	81	7	91	6	11	77

Sort each group:

95	86	81	81	69	91	98	51	77
32	65	62	73	47	81	96	36	17
15	45	52	32	9	42	91	21	11
14	16	41	25	8	25	64	12	9
13	5	32	12	7	18	6	11	5

$O(\boldsymbol{n})$

M-o-M, Step 3: Find median of column medians

Column medians:

95	86	81	81	69	91	98	51	77
32	65	62	73	47	81	96	36	17
15	45	52	32	9	42	91	21	11
14	16	41	25	8	25	64	12	9
13	5	32	12	7	18	6	11	5

$T(n/5)$

Choose pivot to be that median of medians

95	86	81	81	69	91	98	51	77
32	65	62	73	47	81	96	36	17
15	45	52	32	9	42	91	21	11
14	16	41	25	8	25	64	12	9
13	5	32	12	7	18	6	11	5

This Pivot is "Good Enough"!

Column medians:

95	86	81	81	69	91	98	51	77
32	65	62	73	47	81	96	36	17
15	45	52	32	9	42	91	21	11
14	16	41	25	8	25	64	12	9
13	5	32	12	7	18	6	11	5

$T(n/5)$

Imagining rearranging columns by columns' medians

95	51	77	69	81	91	98	86	81
32	36	17	47	73	81	96	65	62
15	21	11	9	32	42	91	45	52
14	12	9	8	25	25	64	16	41
13	11	5	7	12	18	6	5	32

This Pivot is “Good Enough”!

Column medians:

95	86	81	81	69	91	98	51	77
32	65	62	73	47	81	96	36	17
15	45	52	32	9	42	91	21	11
14	16	41	25	8	25	64	12	9
13	5	32	12	7	18	6	11	5

$T(n/5)$

Choose pivot to be that median of medians

All $\leq$ pivot

Not in S_{right}

Size $\geq n/4$

95	51	77	69	81	91	98	86	81
32	36	17	47	73	81	96	65	62
15	21	11	9	32	42	91	45	52
14	12	9	8	25	25	64	16	41
13	11	5	7	12	18	6	5	32

Size of S_{right} is $\leq \frac{3n}{4}$

This Pivot is “Good Enough”!

Column medians:

95	86	81	81	69	91	98	51	77
32	65	62	73	47	81	96	36	17
15	45	52	32	9	42	91	21	11
14	16	41	25	8	25	64	12	9
13	5	32	12	7	18	6	11	5

$\boldsymbol{T(n/5)}$

Choose pivot to be that median of medians

95	51	77	69	81	91	98	86	81
32	36	17	47	73	81	96	65	62
15	21	11	9	32	42	91	45	52
14	12	9	8	25	25	64	16	41
13	11	5	7	12	18	6	5	32

All $\geq$ pivot

Not in $\boldsymbol{S_{left}}$

Size $\geq \boldsymbol{n/4}$

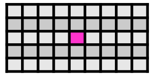
Size of $\boldsymbol{S_{left}}$ is $\leq \frac{3n}{4}$

QuickSelect With Median-of-Medians

5	7	2	9	8	3	5	8

$i = 0$

- **Base Case**:
  - If $i = 0$ then return $S[0]$
- **Divide**:
  - Use median-of-medians to select an element to use as a "pivot"
    - Break S into $\frac{n}{5}$ chunks of size 5 each
    - Sort each chunk
    - Make a new list M containing all chunks' medians
    - Use QuickSelect($M, \frac{n}{10}$) as the pivot
  - Partition



$T\left(\frac{n}{5}\right) + n$

n

2	5	3	5	7	8	9	8

$i = 0$

- **Conquer**:
  - Let i_{pivot} be the index of the pivot after partitioning
  - If $i < i_{pivot}$ then call QuickSelect(S_{left}, i)
  - Otherwise call Quickselect($S_{right}, i - i_{pivot}$)
- **Combine**:
  - Nothing!

2	3	5	5

$i = i$

OR

7	8	8	9

$i = i - i_{pivot}$

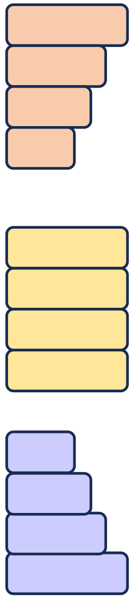
$T\left(\frac{3n}{4}\right)$

$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + n$

Solving the QS with MoM Recurrence

Master Theorem: Suppose that $\boldsymbol{T(n) = a \cdot T(n/b) + O(n^k)}$ for $\boldsymbol{n > b}$.

- If $\boldsymbol{a < b^k}$ then $\boldsymbol{T(n)}$ is $\boldsymbol{O(n^k)}$
  - Cost is dominated by work at top level of recursion
- If $\boldsymbol{a = b^k}$ then $\boldsymbol{T(n)}$ is $\boldsymbol{O(n^k \log n)}$
  - Total cost is the same for all $\boldsymbol{\log_b n}$ levels of recursion
- If $\boldsymbol{a > b^k}$ then $\boldsymbol{T(n)}$ is $\boldsymbol{O(n^{\log_b a})}$
  - Note that $\boldsymbol{\log_b a > k}$ in this case
  - Cost is dominated by total work at lowest level of recursion



$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + n$$

$$T(n) \leq T\left(\frac{19n}{20}\right) + n$$

$\boldsymbol{a = 1}$, $\boldsymbol{b = 20/19}$, $\boldsymbol{k = 1}$ so $\boldsymbol{a < b^k}$: Solution: $\boldsymbol{O(n)}$