MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT2004
Algorithms and Data Structures

Ian Wern Han Lim
lim.wern.han@monash.edu

Referencing materials by
Nathan Companez, Aamir Cheema, Arun Konagurthu and Lloyd Allison

GROUP OF EIGHT AUSTRALIA

Faculty of Information Technology, Monash University

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

This material has been reproduced and communicated to you by or on behalf of Monash University pursuant to Part VB of the Copyright Act 1968 (the Act). The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act. Do not remove this notice

Ready?

Agenda

- Quick-select

Agenda

- Quick-select
  - K-th order statistics

Agenda

- Quick-select
  - K-th order statistics
  - Using it to find the median

Agenda

- Quick-select
  - K-th order statistics
  - Using it to find the median
    - For Quick sort pivot?
    - Median of median?

Let us begin…

K-the Order Statistics
What is it?

K-the Order Statistics

What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array

K-the Order Statistics
What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array

- Example: 21,84,16,14,79,51,66,21,54,32

K-the Order Statistics

What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array

- Example: 21,84,16,14,79,51,66,21,54,32
- If k=1, we want the smallest item
  - 14

K-the Order Statistics
What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array

- Example: 21,84,16,14,79,51,66,21,54,32
- If k=1, we want the smallest item
  - 14
- If k=2, want the 2 smallest items
  - 16,14 (note: order is not important)

K-the Order Statistics

What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array

- Example: 21,84,16,14,79,51,66,21,54,32
- If k=1, we want the smallest item
  - 14
- If k=2, want the 2 smallest items
  - 16,14 (note: order is not important)
- If k=5, we want the 5 smallest items
  - 21,16,14,21,32

K-the Order Statistics

What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array

- Example: 21,84,16,14,79,51,66,21,54,32
- If k=1, we want the smallest item
  - 14
- If k=2, want the 2 smallest items
  - 16,14 (note: order is not important)
- If k=5, we want the 5 smallest items
  - 21,16,14,21,32
  - Isn't this partition?

K-the Order Statistics

What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array
  - First quartile (Q1); k=N/4
  - Median; k=N/2
  - Third quartile (Q3); k= 3N/4

K-the Order Statistics

What is it?

- Given an unsorted array
- Find the k-th smallest elements in the array
  - First quartile (Q1); k=N/4
  - Median; k=N/2
  - Third quartile (Q3) = k= 3N/4

- But how can we get it?

Questions?

K-the Order Statistics

Getting it

- Sort the list

K-the Order Statistics

Getting it

- Sort the list
- Then slice the list for the k-th we want

K-the Order Statistics

Getting it

- Sort the list
- Then slice the list for the k-th we want

K-the Order Statistics

Getting it

- Sort the list
- Then slice the list for the k-th we want
- Complexity is high!

K-the Order Statistics

Getting it

- Sort the list
  - Sorting gives us O(NM log N)
  - Where N is number of item in list
  - Where M is the comparison cost
- Then slice the list for the k-th we want
- Complexity is high!

K-the Order Statistics

Getting it

- Sort the list
  - Sorting gives us O(NM log N)
  - Where N is number of item in list
  - Where M is the comparison cost
  - We can't be sure counting sort work because item_max can be large!
- Then slice the list for the k-th we want
- Complexity is high!

K-the Order Statistics

Getting it

- Sort the list
  - Sorting gives us O(NM log N)
  - Where N is number of item in list
  - Where M is the comparison cost
  - We can’t be sure counting sort work because item_max can be large!
  - We can’t be sure radix sort work well as well…
- Then slice the list for the k-th we want
- Complexity is high!

K-the Order Statistics

Getting it

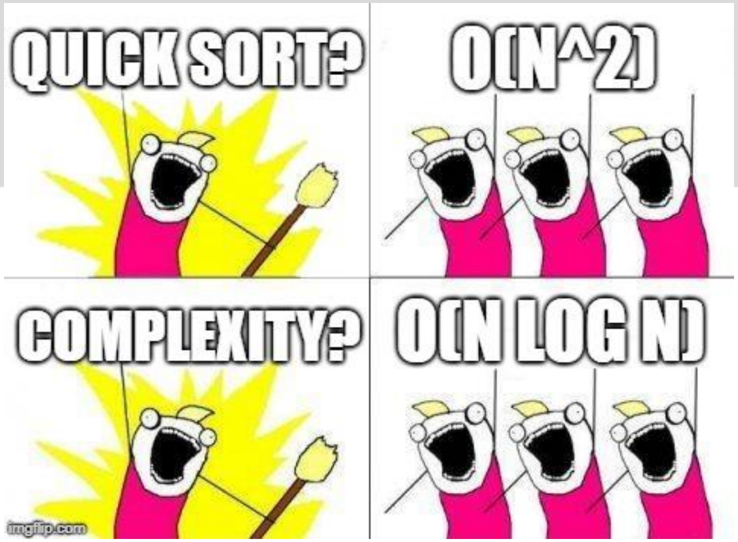
- Sort the list
  - Sorting gives us O(NM log N)
  - Where N is number of item in list
  - Where M is the comparison cost
  - We can’t be sure counting sort work because item_max can be large!
  - We can’t be sure radix sort work well as well…
- Then slice the list for the k-th we want
  - O(k) to slice
- Complexity is high!

K-the Order Statistics
Getting it

- Sort the list
  - Sorting gives us O(NM log N)
  - Where N is number of item in list
  - Where M is the comparison cost
  - We can’t be sure counting sort work because item_max can be large!
  - We can’t be sure radix sort work well as well…
- Then slice the list for the k-th we want
  - O(k) to slice
- Complexity is high!

K-the Order Statistics

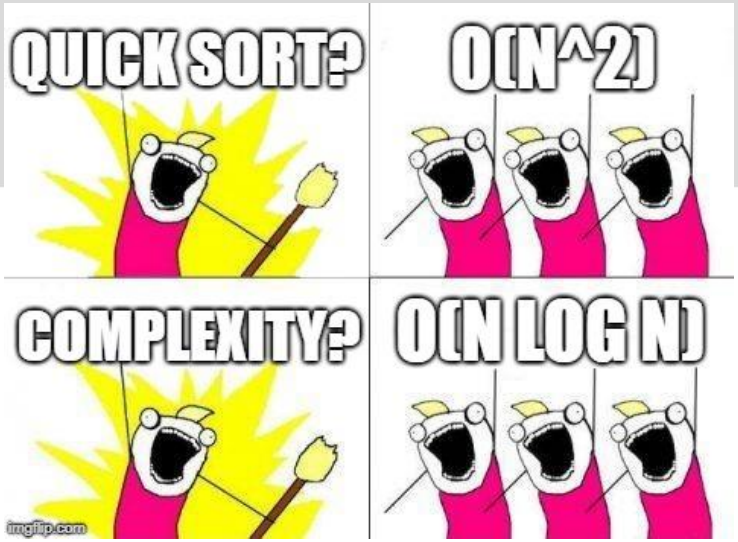
Getting it



- Sort the list
  - Sorting gives us O(NM log N)
  - Where N is number of item in list
  - Where M is the comparison cost
  - We can't be sure counting sort work because item_max can be large!
  - We can't be sure radix sort work well as well…
- Then slice the list for the k-th we want
  - O(k) to slice
- Complexity is high!

K-the Order Statistics

Getting it



- Sort the list
  - Sorting gives us O(NM log N)
  - Where N is number of item in list
  - Where M is the comparison cost
  - We can’t be sure counting sort work because item_max can be large!
  - We can’t be sure radix sort work well as well…
- Then slice the list for the k-th we want
  - O(k) to slice
- Complexity is high!

Questions?

Quick Select

What is it?

- In a nutshell, it is like quick sort

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want?

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want?
    - Index < what we want?

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want?
    - Index < what we want?
    - Index = what we want?

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

20	80	90	10	30	50	70	60

20	10	30	50	80	90	70	60

Pivot	X
In Sorted position	X
Others	X

In sorted position (at index 4, i.e., 4th smallest)

Quick Select

What is it?

MONASH University

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

20	80	90	10	30	50	70	60

20	10	30	50	80	90	70	60

k = 2

In sorted position (at index 4, i.e., 4th smallest)

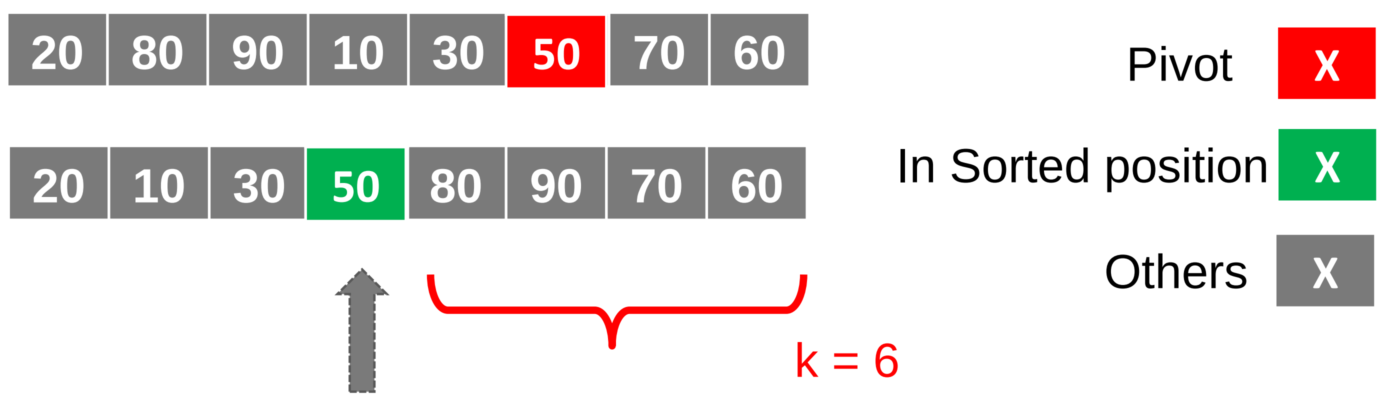
Pivot X

In Sorted position X

Others X

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!



In sorted position (at index 4, i.e., 4th smallest)

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

20	80	90	10	30	50	70	60

20	10	30	50	80	90	70	60

Pivot X

In Sorted position X

Others X

In sorted position (at index 4, i.e., 4th smallest), return everything on the left of the index…

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!
  - So it is really just like implementing a quick sort except we changed…

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

  - So it is really just like implementing a quick sort except we changed…
    - Go left or go right

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

  - So it is really just like implementing a quick sort except we changed…
    - Go left or go right… vs go both for quicksort

Quick Select
What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!
  - So it is really just like implementing a quick sort except we changed…
    - Go left or go right… vs go both for quicksort
  - Complexity?

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

  - So it is really just like implementing a quick sort except we changed…
    - Go left or go right… vs go both for quicksort
  - Complexity?
    - Best case O(N) cause we still need to partition once!

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!

  - So it is really just like implementing a quick sort except we changed…
    - Go left or go right… vs go both for quicksort
  - Complexity?
    - Best case O(N) cause we still need to partition once!
    - Worst case O(N^2) cause our pivot always fail!!!

Quick Select

What is it?

- In a nutshell, it is like quick sort
  - Select a pivot   repeat the process until it is found
  - Perform partition
  - What is the index of the pivot?
    - Index > what we want? Repeat for left
    - Index < what we want? Repeat for right
    - Index = what we want? Return, we found the item!
  - So it is really just like implementing a quick sort except we changed…
    - Go left or go right… vs go both for quicksort
  - Complexity?
    - Best case O(N) cause we still need to partition once!
    - Worst case O(N^2) cause our pivot always fail!!!
      the number need is at [0], but choosing [n-1] as pivot
- Allow us to find what we want without sorting!

Questions?

Quick Sort
With quick select

- So we will now
  - Use quick select to find the median as a pivot
  - Use quick sort to sort

- So we will now
  – Use quick select to find the median as a pivot
    - Note: Since quick-select do perform the partition as well, we can avoid doing partition in the quick sort phase itself!
  – Use quick sort to sort

Quick Sort

With quick select

- So we will now
  - Use quick select to find the median as a pivot
    - Worst case complexity of O(N^2) when pivot != k till the last final iteration…
    - Note: Since quick-select do perform the partition as well, we can avoid doing partition in the quick sort phase itself!
  - Use quick sort to sort

Quick Sort

With quick select

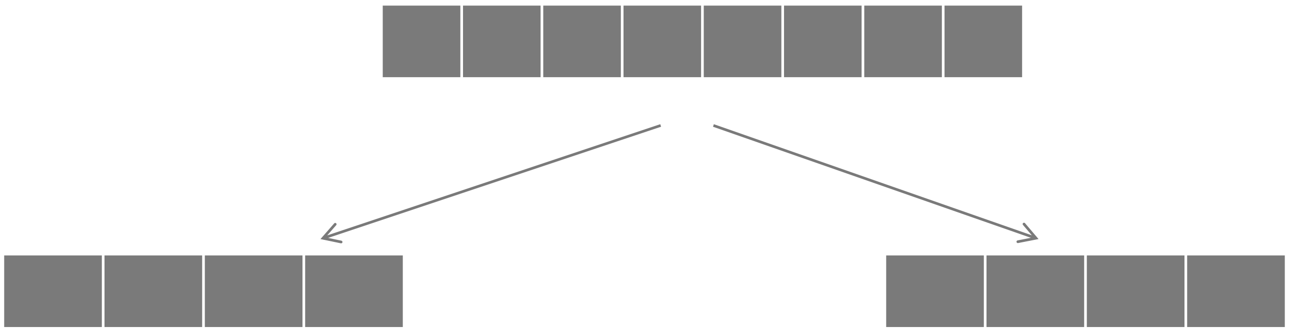
Quick Sort

With quick select

N^2

Quick Sort
With quick select

N^2



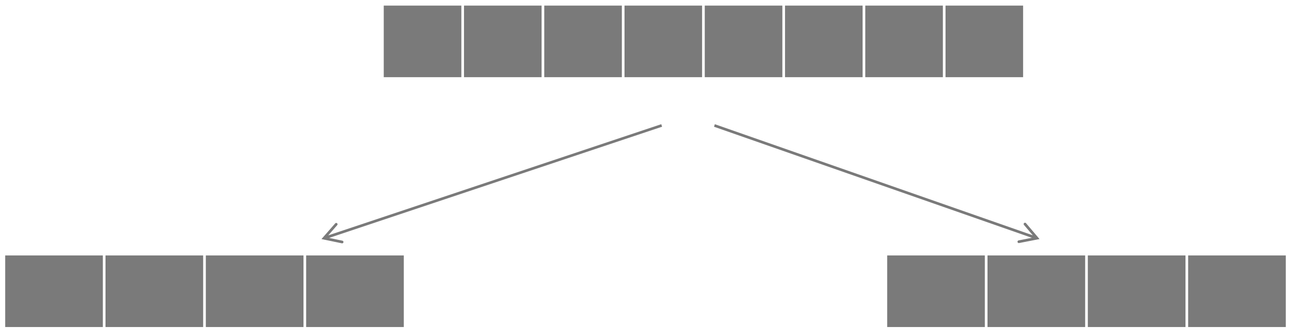
N/2 size

N/2 size

Quick Sort

With quick select

N^2



N/2 size

N^2/4 complexity

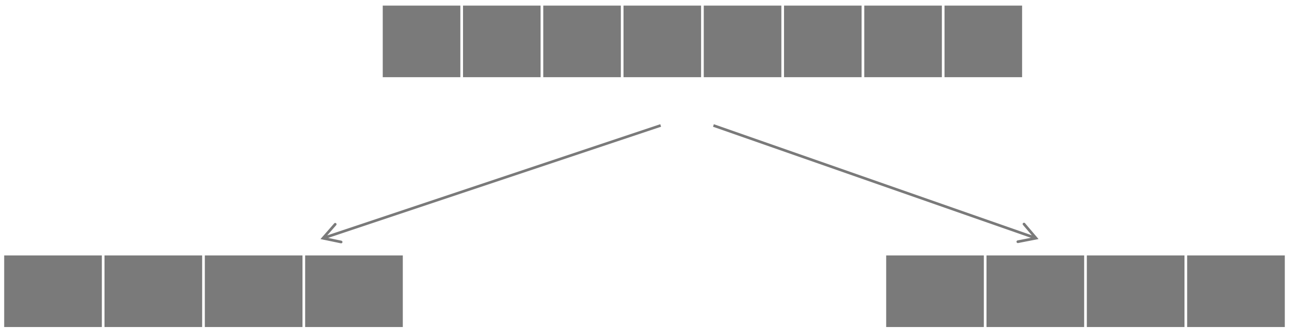
N/2 size

N^2/4 complexity

Quick Sort

With quick select

N^2



N/2 size

N^2/4 complexity

+

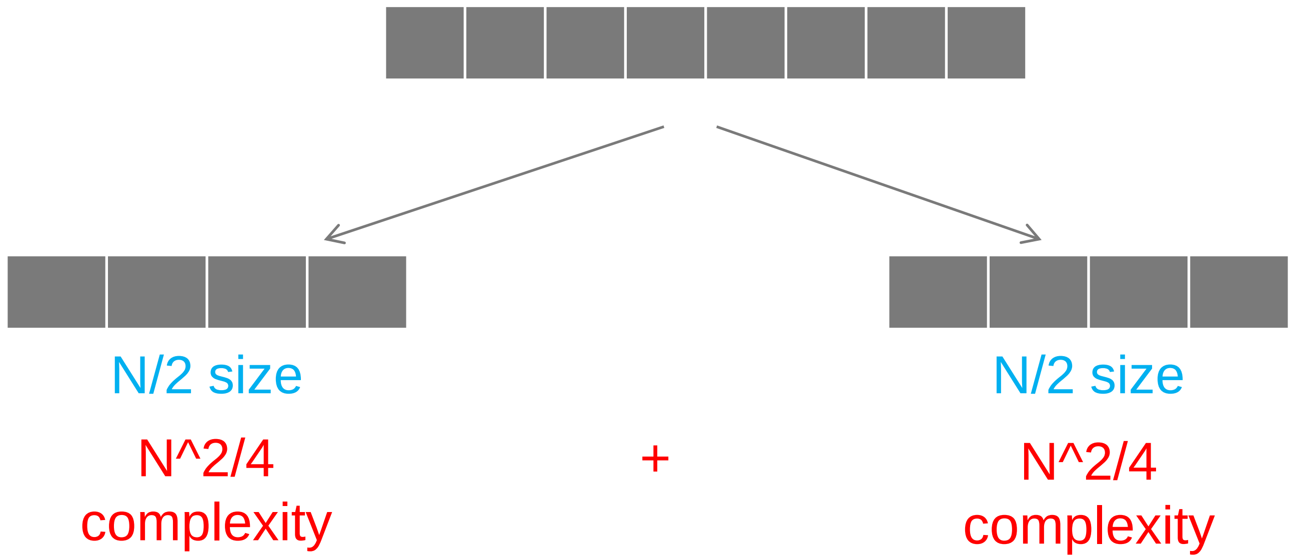
N/2 size

N^2/4 complexity

Quick Sort
With quick select

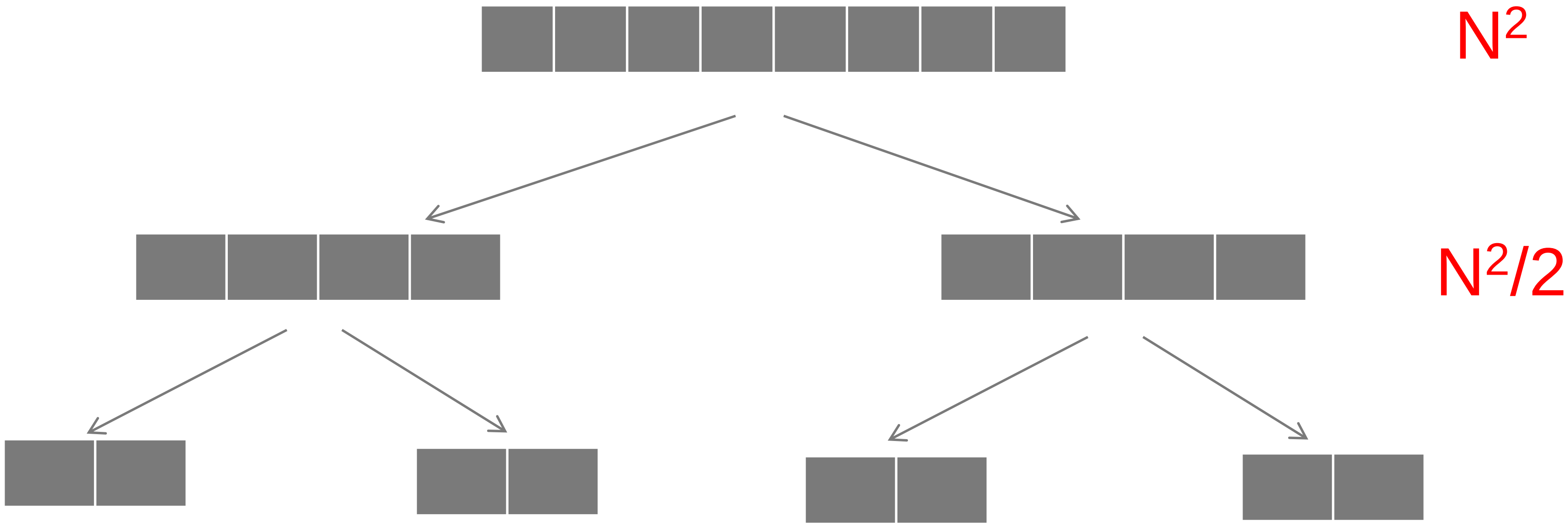
N^2

$N^2/2$



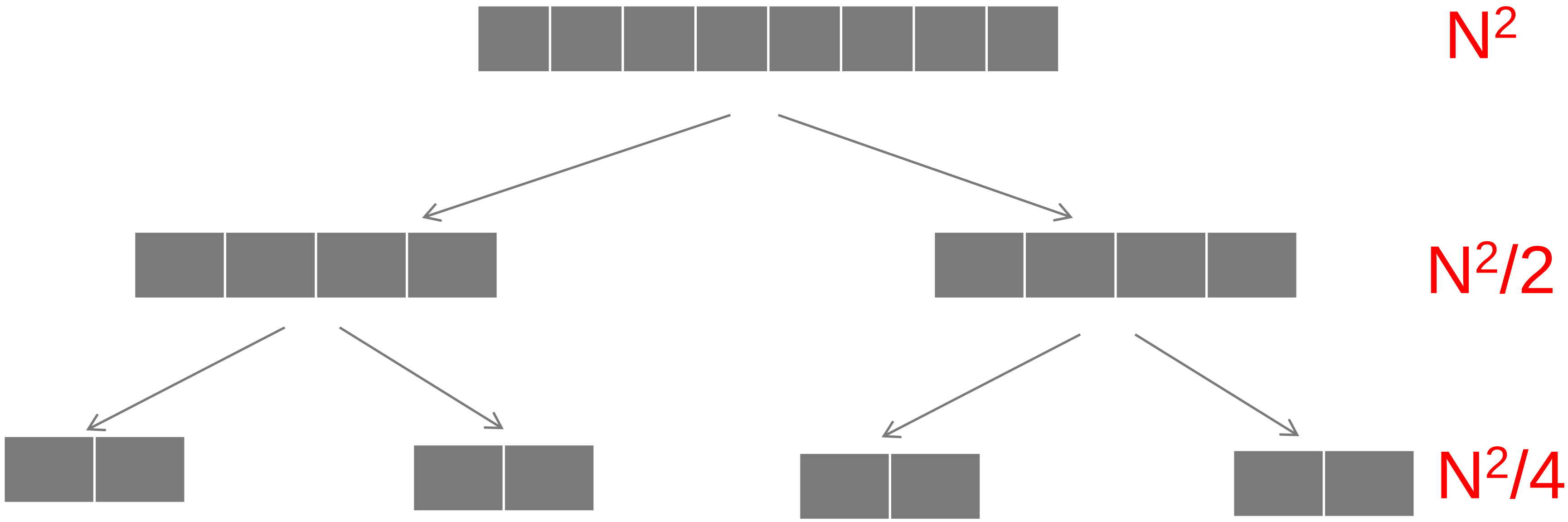
Quick Sort

With quick select



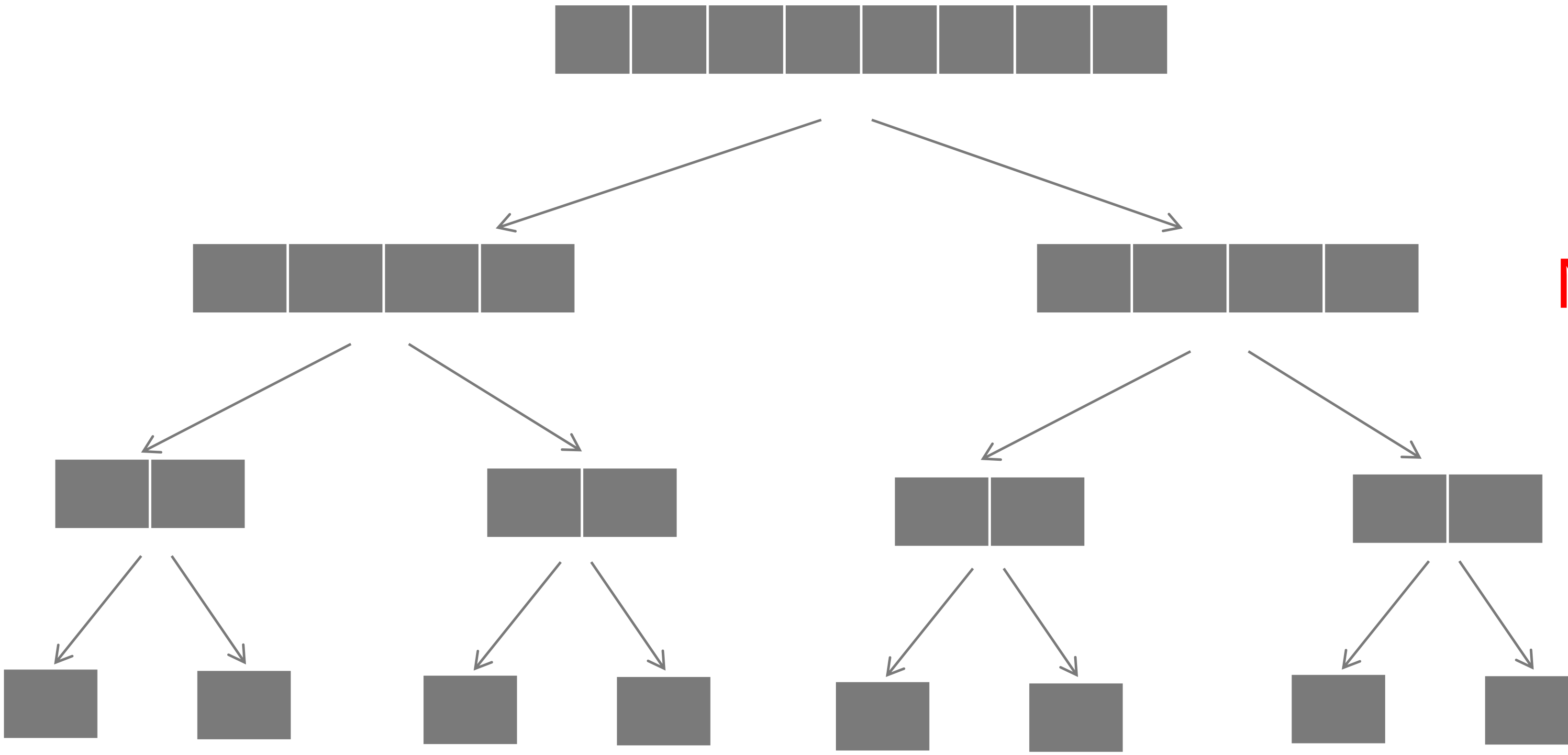
Quick Sort

With quick select



Quick Sort

With quick select

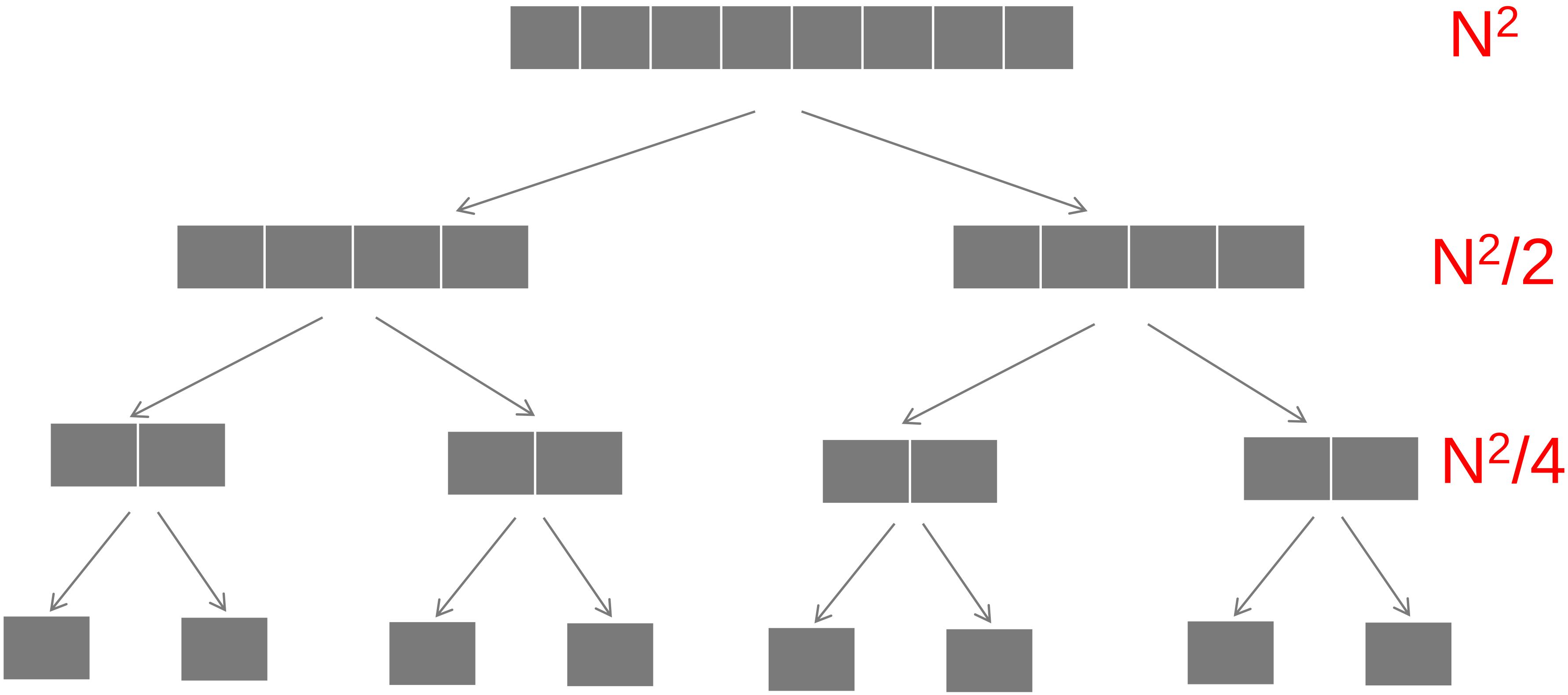


N^2

$N^2/2$

$N^2/4$

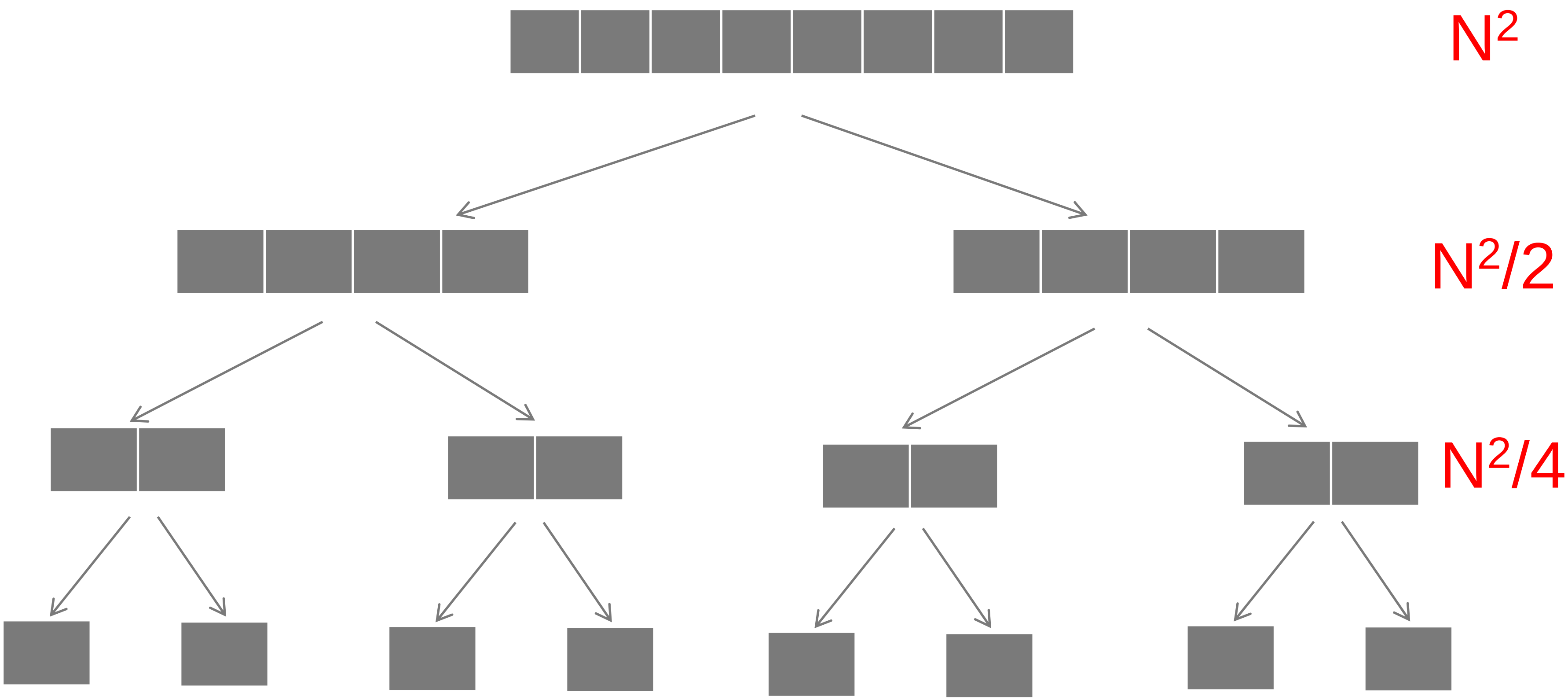
Quick Sort
With quick select



Worst-case cost at level k: $N^2/2^k$

Quick Sort

With quick select

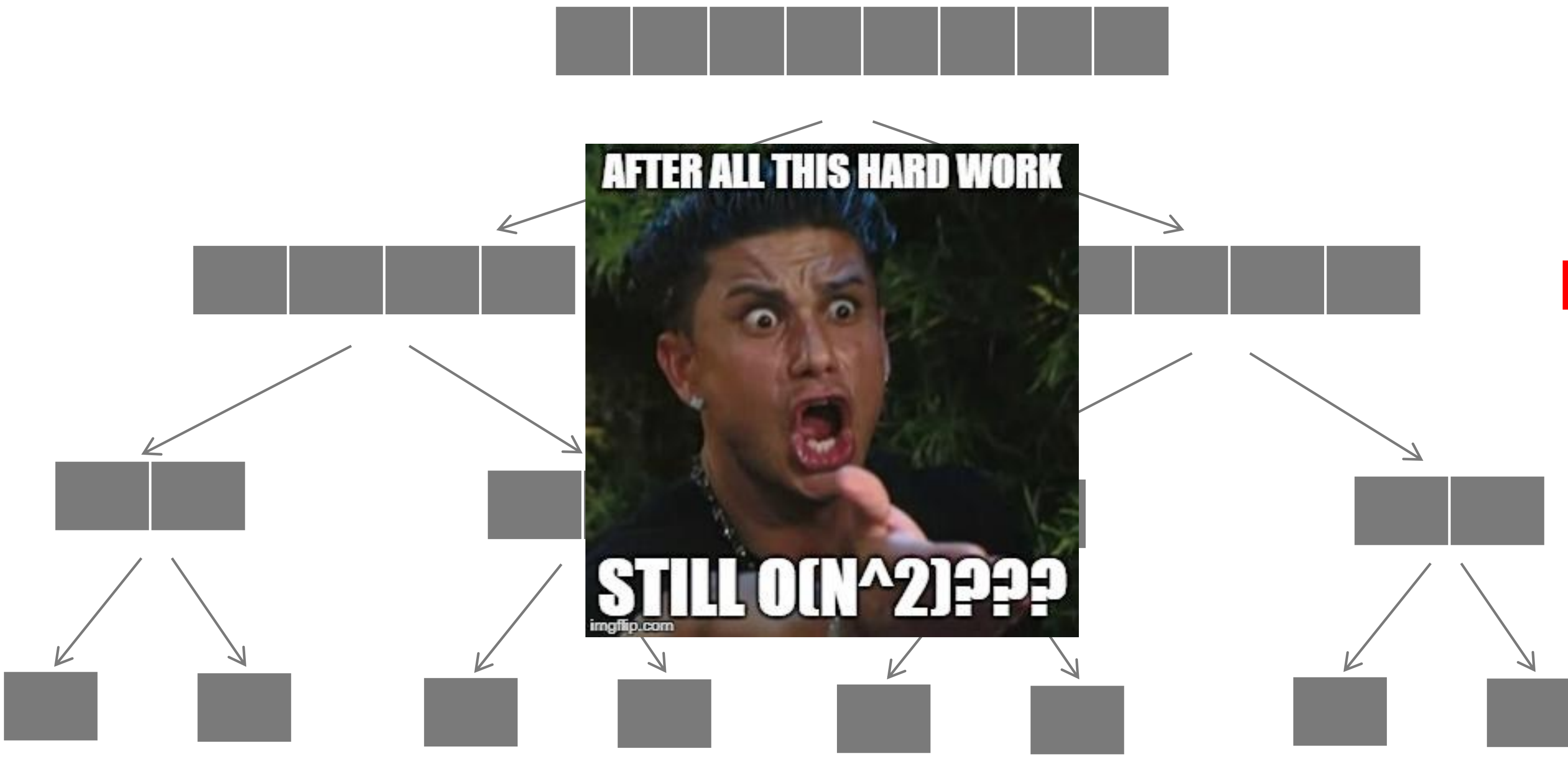


Worst-case cost at level k: $N^2/2^k$

Total cost: $N^2 + N^2/2 + N^2/4 + \ldots + 1 = N^2(1 + ½ + ¼ + \ldots) = O(N^2)$

Quick Sort

With quick select



N^2

$N^2/2$

$N^2/4$

Worst-case cost at level k: $N^2/2^k$

Total cost: $N^2 + N^2/2 + N^2/4 + \ldots + 1 = N^2(1 + \frac{1}{2} + \frac{1}{4} + \ldots) = O(N^2)$

Quick Sort

With quick select

- So what is the complexity?
  - Worst case…
  - O(N^2)

Questions?

Quick Sort
With quick select

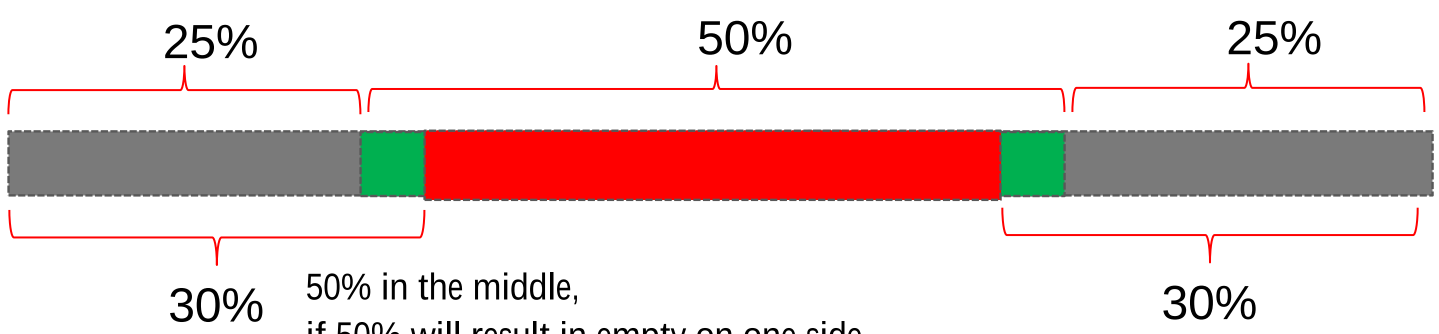
- So what is the complexity?
  - Worst case…
  - O(N^2)
  - … so is it useless?

- So what is the complexity?
  - Worst case…
  - O(N^2)
  - … so is it useless?

  - Why not we adjust it now, to not find the median but find just “good enough” known as the median of median…

- So what is the complexity?
  - Worst case…
  - O(N^2)
  - … so is it useless?

  - Why not we adjust it now, to not find the median but find just “good enough” known as the median of median…



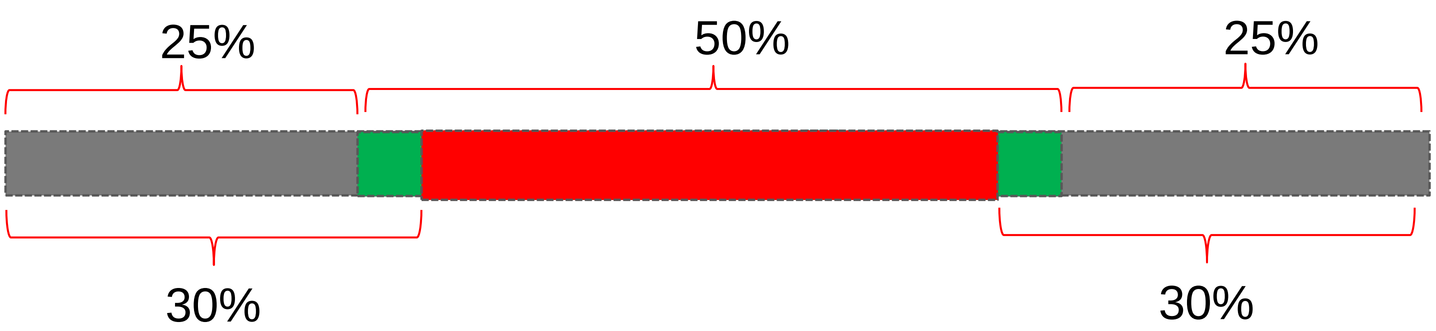
50% in the middle,
if 50% will result in empty on one side
and n elements on another side

Quick Sort

With quick select

- So what is the complexity?
  - Why not we adjust it now, to not find the median but find just "good enough" known as the median of median…
    - Find pivot within the green area using quick select
    - Do normal quick sort

prevent worst case for quick sort



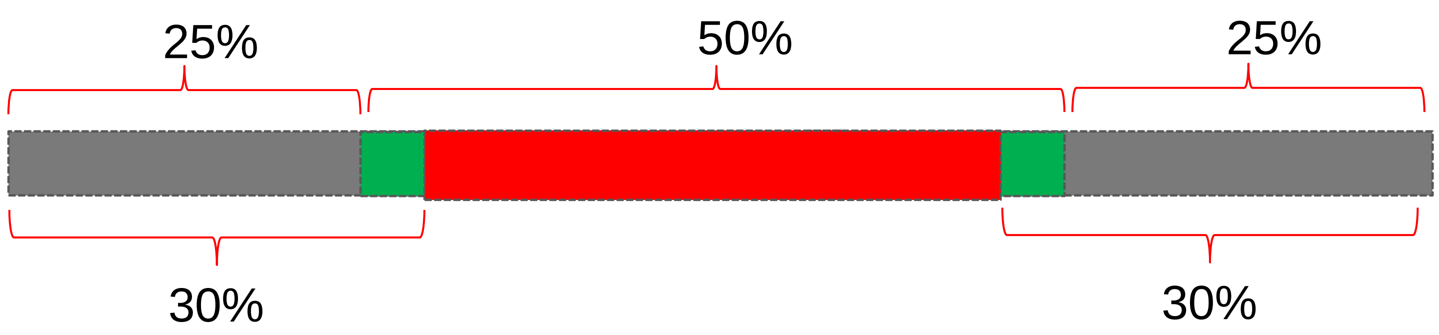
Quick Sort

With quick select

- So what is the complexity?
  - Why not we adjust it now, to not find the median but find just "good enough" known as the median of median…
    - Find pivot within the green area using quick select
    - Do normal quick sort

loop through every elements to find the suitable one

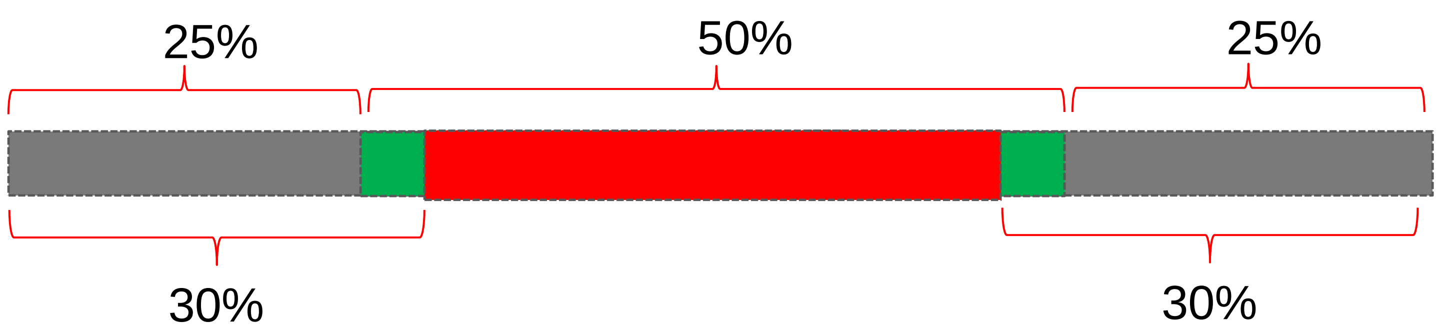
O(N)



Quick Sort
With quick select

- So what is the complexity?
  - Why not we adjust it now, to not find the median but find just “good enough” known as the median of median…
    - Find pivot within the green area using quick select   O(N)
    - Do normal quick sort   O(1) why?

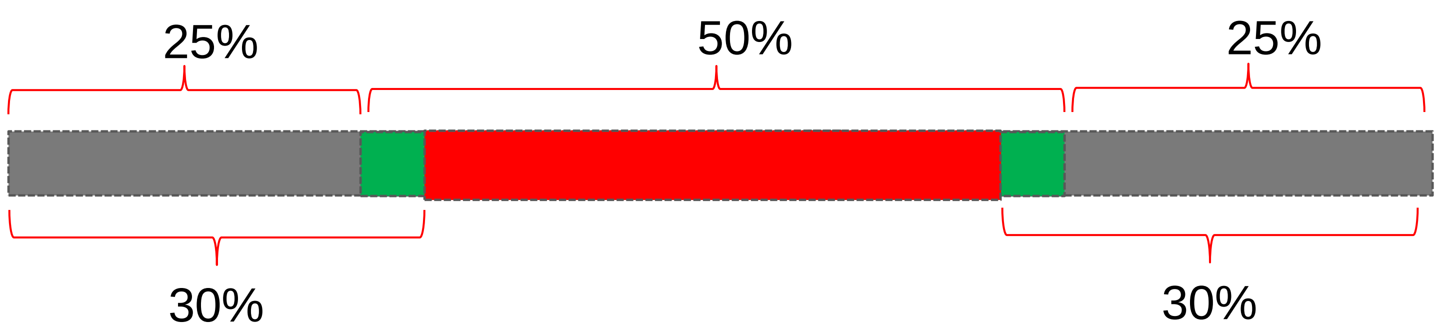
not repeating partition
in quicksort



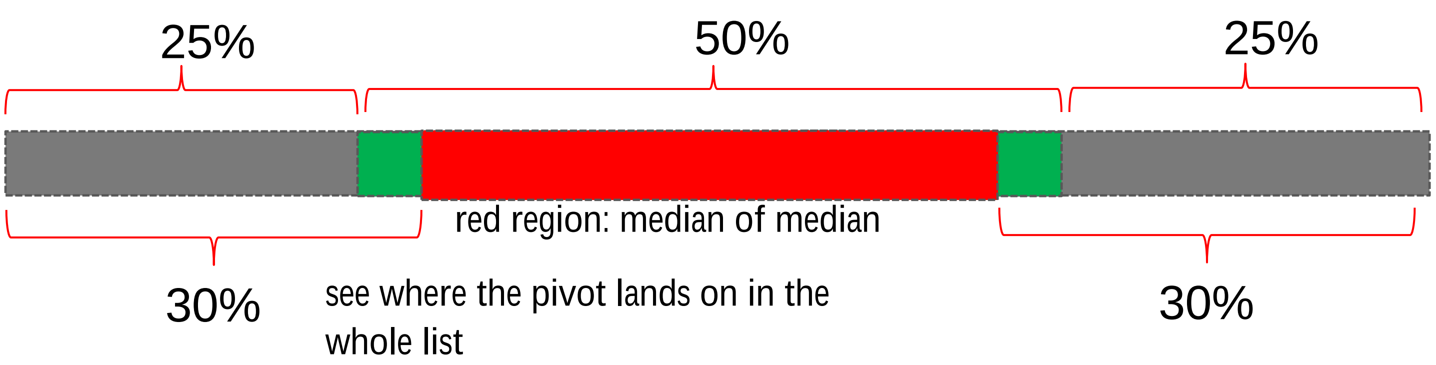
Quick Sort

With quick select

- So what is the complexity?
  - Why not we adjust it now, to not find the median but find just "good enough" known as the median of median…
    - Find pivot within the green area using quick select O(N)
    - Do normal quick sort O(1) why?
    - Maximum height is roughly log n



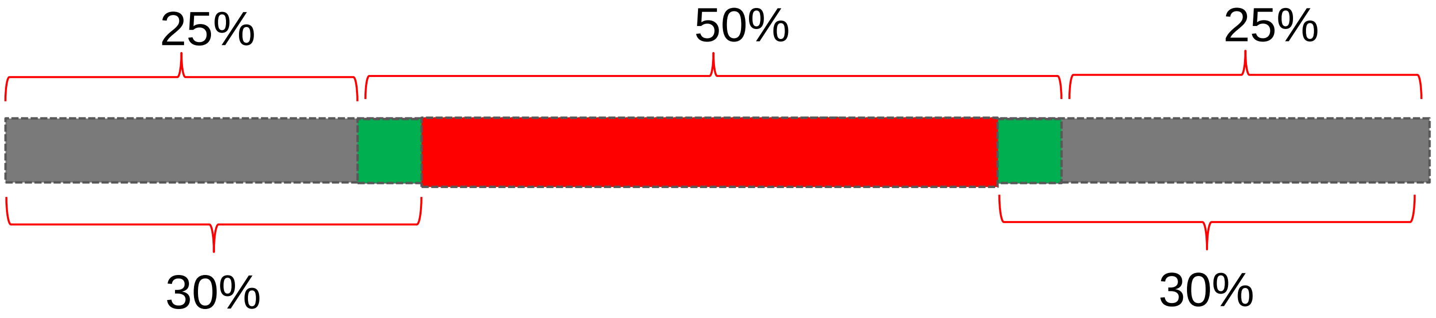
- So what is the complexity?
  - Why not we adjust it now, to not find the median but find just "good enough" known as the median of median…
    - Find pivot within the green area using quick select O(N)
    - Do normal quick sort O(1) why?
    - Maximum height is roughly log n
    - Final complexity? O(N log N) still lol



Quick Sort

With quick select

- So what is the complexity?
  - Why not we adjust it now, to not find the median but find just "good enough" known as the median of median…
    - Find pivot within the green area using quick select O(N)
    - Do normal quick sort O(1) why?
    - Maximum height is roughly log n
    - Final complexity? O(N log N) still lol



Quick Sort
With quick select

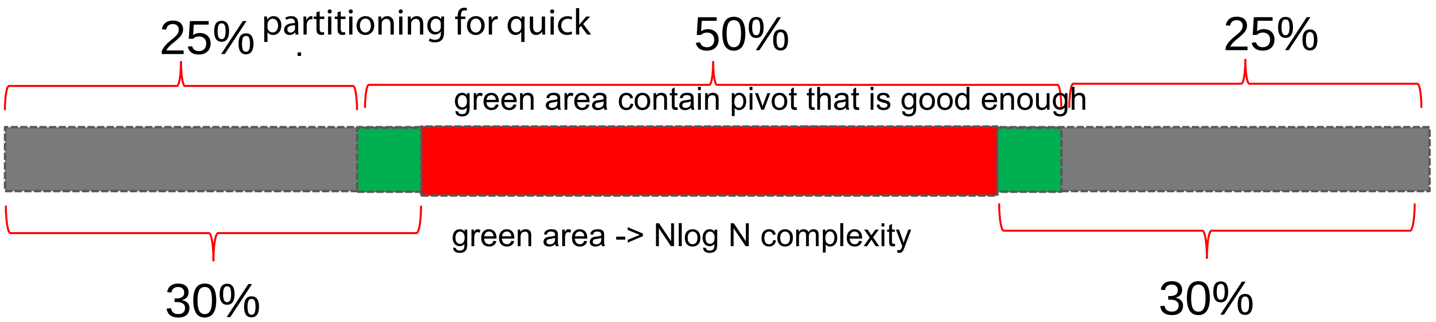
- So what is the complexity?
  - Why not we adjust it now, to not find the median but find just "good enough" known as the median of median…

2h/2 = 1/2 * log2 (N^2)^1/2 = log2 N
since only halve of the original height or recursion to get suitable pivot

    - Find pivot within the green area using quick select O(N)
    - Do normal quick sort O(1) why?
    - Maximum height is roughly log n
    - Final complexity? O(N log N) still lol

prevent worst case for quick sort

partitioning for quick



- In reality, random pivot works well due to probability…

MONASH University

Questions?

Quick Sort

With quick select median of medians

- Now a way to make quick select better is via median-of-medians

Quick Sort

With quick select median of medians

- Now a way to make quick select better is via median-of-medians
  - This is not examinable
  - I will be using Nathan's slides

Quick Sort
With quick select median of medians

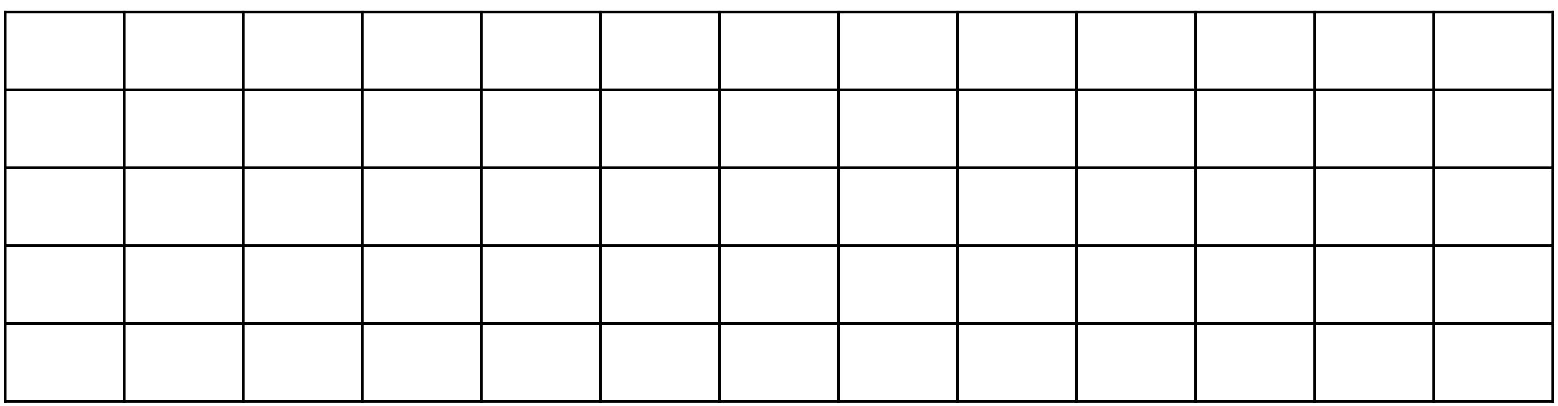
- Now a way to make quick select better is via median-of-medians
  - This is not examinable
  - I will be using Nathan's slides

- And this ensure the worst case or quick-sort is O(N log N)

Median of medians (not examinable)

- Sort groups of size five

Bigger



Smaller

Median of medians (not examinable)

Sort groups of size five

Find the medians

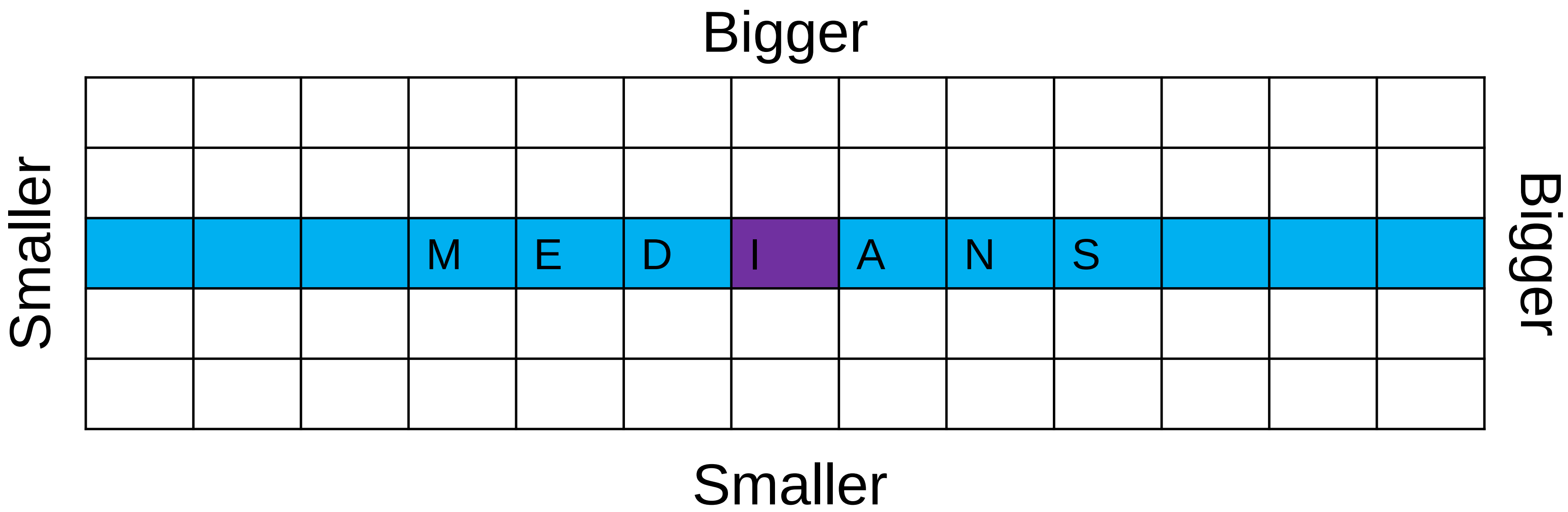
Bigger

			M	E	D	I	A	N	S			

Smaller

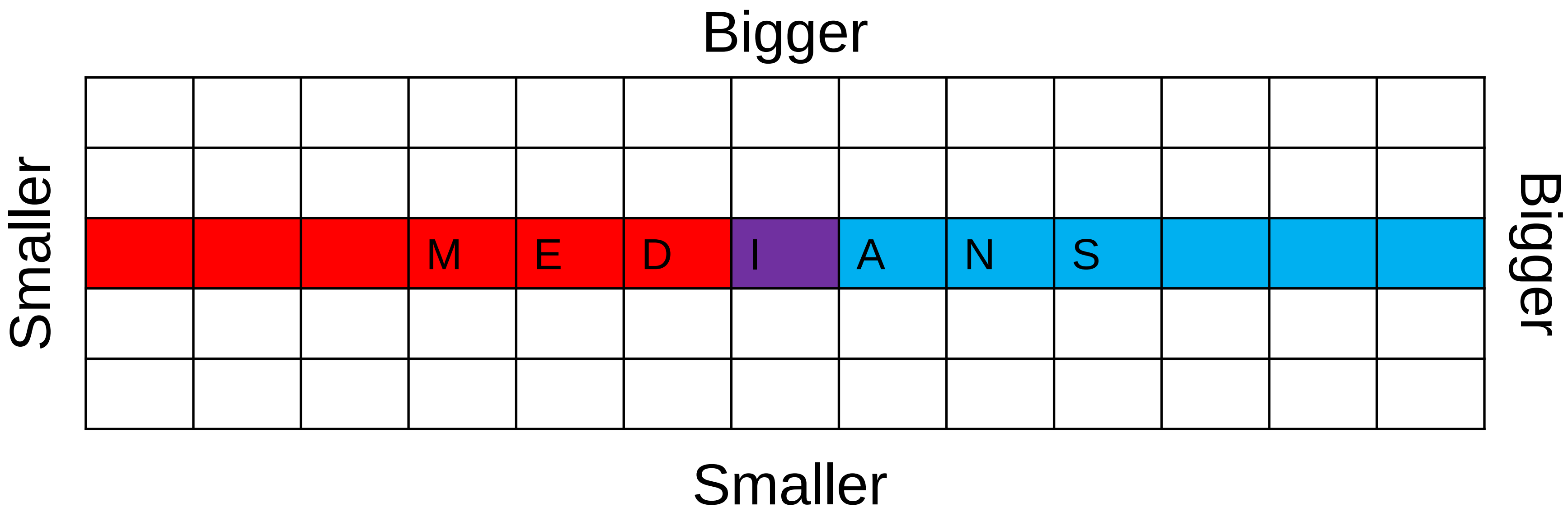
Median of medians (not examinable)

- Sort groups of size five
- Find the medians
- Find the median of those!
- (Note that the groups of 5 are not actually sorted, just shown here in sorted order for clarity)



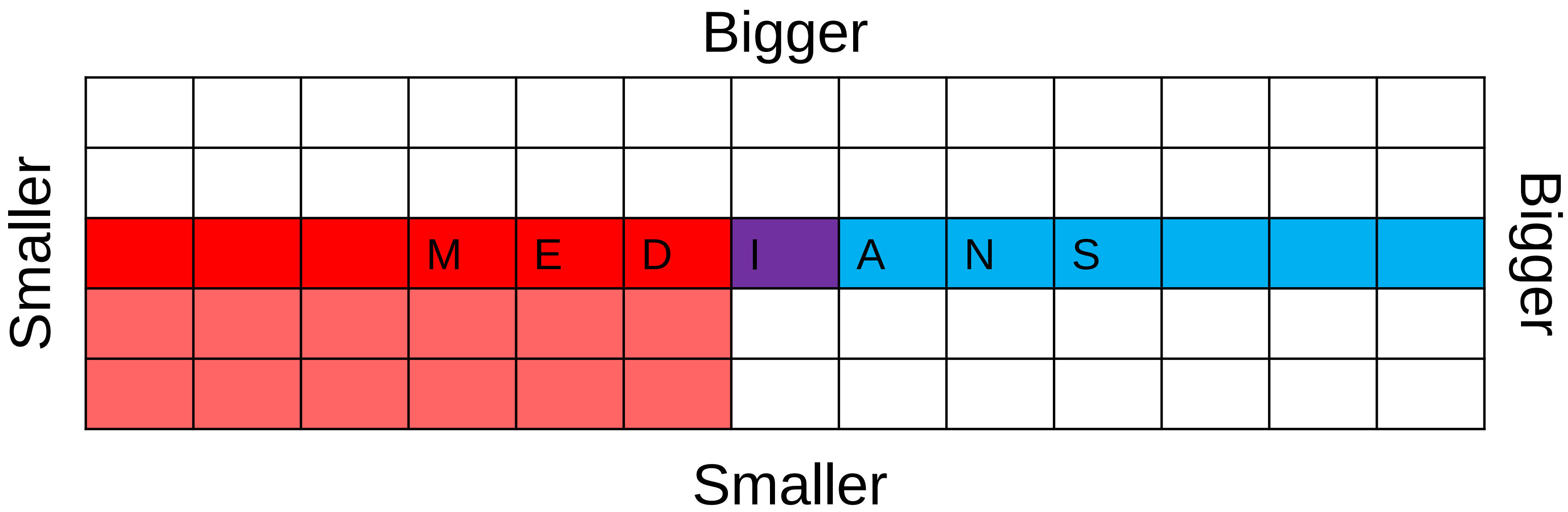
Median of medians (not examinable)

- Median of medians is bigger than half the medians



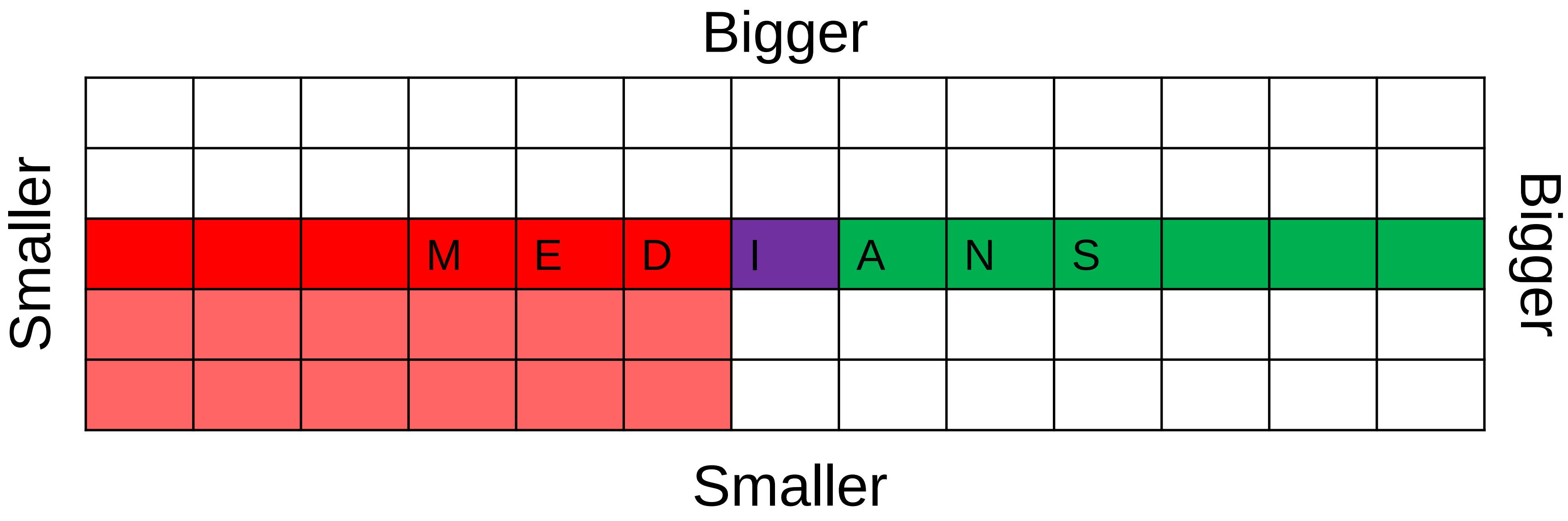
Median of medians (not examinable)

- Median of medians is bigger than half the medians
- So it is bigger than all the red values as well



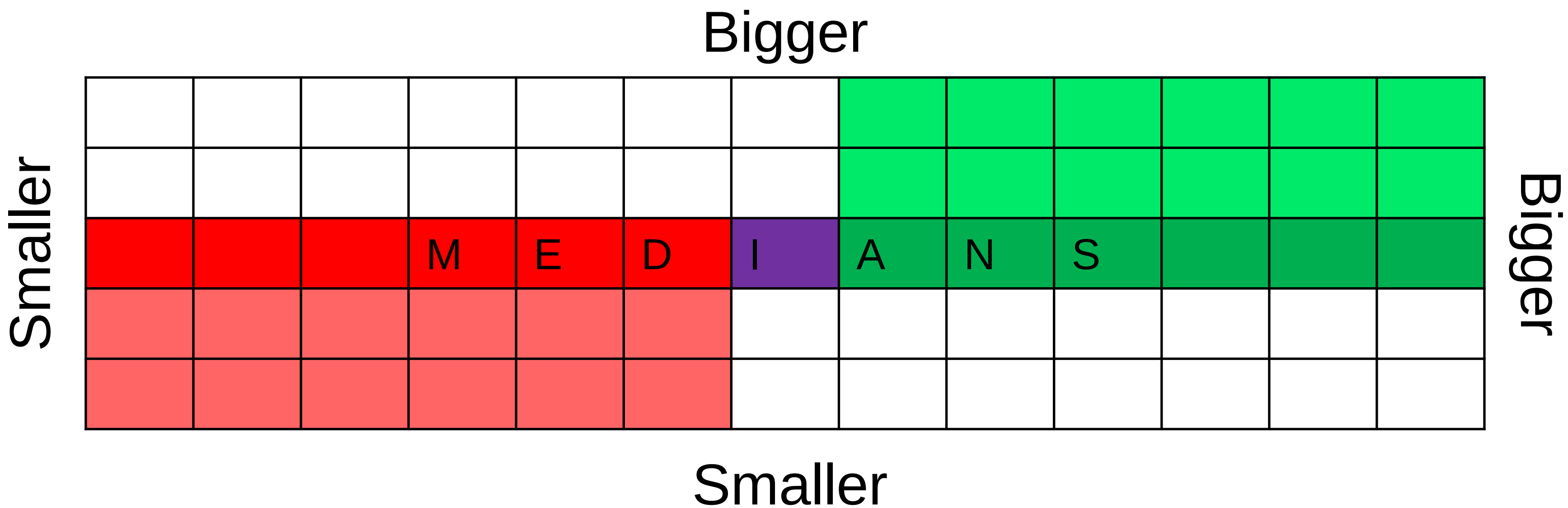
Median of medians (not examinable)

- Median of medians is smaller than half the medians



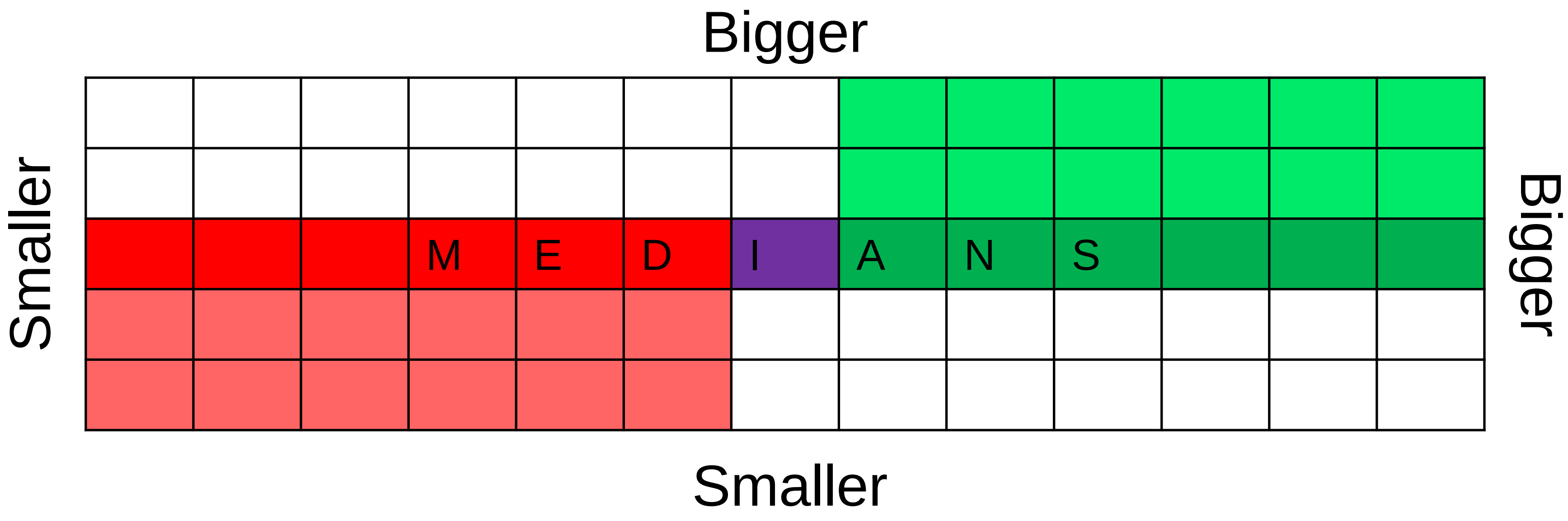
Median of medians (not examinable)

- Median of medians is smaller than half the medians
- So it is smaller than the green values as well



Median of medians (not examinable)

- Median of medians is greater than 30% and also less than 30%, so its in the middle 40%
- The worst split we can get using the MoM is 70:30!
- However, we did need to find the exact median of n/5 items… how?



Quicksort with O(N log N) in worst-case

Median_of_medians(list[1..n])

divide into sublists of size 5

medians = [median of each sublist]

use quickselect to find the median of **medians**

get one median from each sublist = medians

Quicksort with O(N log N) in worst-case

Median_of_medians(list[1..n])

if n <= 5

O(1) due to just 5 elements in sublist and insertion sort just run once

use insertion sort to find the median, and return it

all together run insertion sort for n/5 times

divide into sublists of size 5

O(n)

medians = [median of each sublist]

use quickselect to find the median of **medians**

Log(N)

Quicksort with O(N log N) in worst-case

Median_of_medians(list[1..n])

if n <= 5

use insertion sort to find the median, and return it

divide into sublists of size 5

medians = [median of each sublist]

return *quickselect*(medians, (len(medians)+1)/2)

Quicksort with O(N log N) in worst-case

This call uses quickselect!
But with a weaker pivot

Quickselect(list, lo, hi, k)
    if lo > hi
        return array[k]
    pivot = **median_of_medians**(list, lo, hi, k)
    mid = *partition*(array, lo, hi, pivot)
    if mid > k
        return *quickselect*(array, lo, mid-1, k)
    elif k > mid
        return *quickselect*(array, mid+1, hi, k)
    else
        return array[k]

Quick Sort
with O(N log N) in worst-case

- Wait what?
  - Median of median calls quick select
  - Quick select calls median of median

Quick Sort
with O(N log N) in worst-case

- Wait what?
  - Median of median calls quick select
  - Quick select calls median of median

  - This is called co-recursion…

Quicksort with O(N log N) in worst-case

This call uses quickselect!
But with a weaker pivot

```
Quickselect(list, lo, hi, k)
    if lo > hi
        return array[k]
    pivot = median_of_medians(list, lo, hi, k) (
    mid = partition(array, lo, hi, pivot) (70:30 pivot in worst)
    if mid > k
        return quickselect(array, lo, mid-1, k) (n/7 in worst)
    elif k > mid
         return quickselect(array, mid+1, hi, k) (n/7 in worst)
    else
        return array[k]
```

Quicksort with O(N log N) in worst-case

Quickselect time complexity recurrence

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + an$$

- $T\left(\frac{n}{5}\right)$ for recursing on the list of the medians of groups of 5 (inside the call to median of medians)
- $T\left(\frac{7n}{10}\right)$ for the main recursive call, which is guaranteed to have split the list at least 30:70 (because the pivot was selected by MoM)
- an for the linear time partition algorithm + time to find medians of groups of five

Solving this give linear time!

So armed with a linear time quickselect, we can now quicksort in NlogN worst case…

Questions?

- Note: NOT EXAMINABLE for math approach

Quick Select
and average case complexity

- What algorithm is quick select?

Quick Select
and average case complexity

- What algorithm is quick select?
  - Partial sorting with partition
  - Divide and conquer

Quick Select
and average case complexity

- What algorithm is quick select?
  - Partial sorting with partition
  - Divide and conquer
  - Recursion

Quick Select
and average case complexity

- What algorithm is quick select?
  - Partial sorting with partition
  - Divide and conquer
  - Recursion

$$T_N = N + 1 + \begin{cases} T(N-i) & \text{if } i < k, \\ 1 & \text{if } i = k, \\ T(i-1) & \text{if } i > k \end{cases} .$$

Quick Select
and average case complexity

- What algorithm is quick select?
  - Partial sorting with partition
  - Divide and conquer
  - Recursion

$$T_N = N + 1 + \begin{cases} T(N-i) & \text{if } i < k, \text{ got to right N - i} \\ 1 & \text{if } i = k, . \\ T(i-1) & \text{if } i > k \text{ 0 to i - 1} \end{cases}$$

  - N is the size of list
  - i is the rank of the pivot
  - k is the k-th element we are looking for

Quick Select
and average case complexity

- What algorithm is quick select?
  - Partial sorting with partition
  - Divide and conquer
  - Recursion

$$T_N = N + 1 + \begin{cases} T(N-i) & \text{if } i < k, \\ 1 & \text{if } i = k, \\ T(i-1) & \text{if } i > k \end{cases}.$$

  - N is the size of list
  - i is the rank of the pivot
  - k is the k-th element we are looking for
  - With this, we can now figure out the average complexity

Quick Select
and average case complexity

- What algorithm is quick select?
  – With this, we can now figure out the average complexity
  – This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

  – What does this means?

Quick Select
and average case complexity

- What algorithm is quick select?
  – With this, we can now figure out the average complexity
  – This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

  – What does this means?
    - Every pivot case
    - Sum it all up
    - Get the average

Quick Select
and average case complexity

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = \boxed{N} + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

Quick Select
and average case complexity

- What algorithm is quick select?
  – With this, we can now figure out the average complexity
  – This is the expected run time on average

$$T_N = N \boxed{+1} + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

Quick Select

and average case complexity

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = N + 1 + \boxed{\frac{1}{N}} \left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1 \right)$$

Quick Select
and average case complexity

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\boxed{\sum_{i=1}^{k-1} T(N-i)} + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

Quick Select
and average case complexity

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \boxed{\sum_{i=k+1}^{N} T(i-1)} + 1\right)$$

Quick Select
and average case complexity

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

Quick Select
and average case complexity

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

  - Now we try to solve this

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

  - Now we try to solve this
    - Multiply both side with N

$$NT_N = N^2 + N + \sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + \frac{1}{N}$$

Quick Select
and average case complexity

MONASH University

- What algorithm is quick select?
  - With this, we can now figure out the average complexity
  - This is the expected run time on average

$$T_N = N + 1 + \frac{1}{N}\left(\sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + 1\right)$$

  - Now we try to solve this
    - Multiply both side with N

$$NT_N = N^2 + N + \sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + \boxed{\frac{1}{N}}$$

Quick Select

and average case complexity

- So now we have

$$NT_N = N^2 + N + \sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + \frac{1}{N}$$

- How do we solve it?
  - Not easy

Quick Select
and average case complexity

- So now we have

$$NT_N = N^2 + N + \sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + \frac{1}{N}$$

- How do we solve it?
  - Not easy
  - We create another equation, instead of N it is N-1

$$(N-1)T_{N-1} = (N-1)^2 + (N-1) + \sum_{i=2}^{k} T(N-i) + \sum_{i=k+1}^{N+1} T(i-1) + \frac{1}{N-1}$$

Quick Select
and average case complexity

- So now we have

$$NT_N = N^2 + N + \sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + \frac{1}{N}$$

- How do we solve it?
  - Not easy
  - We create another equation, instead of N it is N-1. Why?

$$(N-1)T_{N-1} = (N-1)^2 + (N-1) + \sum_{i=2}^{k} T(N-i) + \sum_{i=k+1}^{N+1} T(i-1) + \frac{1}{N-1}$$

Quick Select

and average case complexity

- So now we have

$$NT_N = N^2 + N + \sum_{i=1}^{k-1} T(N-i) + \sum_{i=k+1}^{N} T(i-1) + \frac{1}{N}$$

- How do we solve it?
  - Not easy
  - We create another equation, instead of N it is N-1. Why? N – (N-1) = 1!!!

$$(N-1)T_{N-1} = (N-1)^2 + (N-1) + \sum_{i=2}^{k} T(N-i) + \sum_{i=k+1}^{N+1} T(i-1) + \frac{1}{N-1}$$

Quick Select
and average case complexity

- Doing the N – (N-1), we have

$$NT_N - (N-1)T_{N-1} = N^2 + N - (N-1)^2 - (N-1) - T(N-1) + T(N-1) + \frac{1}{N} - \frac{1}{N-1}$$

- Doing the N – (N-1), we have

$$NT_N - (N-1)T_{N-1} = N^2 + N - (N-1)^2 - (N-1) - T(N-1) + T(N-1) + \frac{1}{N} - \frac{1}{N-1}$$

- We then clean it

$$NT_N = (N-1)T_{N-1} + 2N + \frac{1}{N(N-1)}$$

Quick Select
and average case complexity

- Doing the N – (N-1), we have

$$NT_N - (N-1)T_{N-1} = N^2 + N - (N-1)^2 - (N-1) - T(N-1) + T(N-1) + \frac{1}{N} - \frac{1}{N-1}$$

- We then clean it

$$NT_N = (N-1)T_{N-1} + 2N + \frac{1}{N(N-1)}$$

- Then have the left side of T_N

$$T_N = \frac{N-1}{N}T_{N-1} + 2 + \frac{1}{N^2(N-1)}$$

- Finally our average case complexity is from the equation:

$$T_N = \frac{N-1}{N} T_{N-1} + 2 + \frac{1}{N^2(N-1)}$$

- Finally our average case complexity is from the equation:

$$T_N = \frac{N-1}{N} T_{N-1} + 2 + \frac{1}{N^2(N-1)}$$

- What is the complexity?

MONASH University

- Finally our average case complexity is from the equation:

$$T_N = \frac{N-1}{N} T_{N-1} + 2 + \frac{1}{N^2(N-1)}$$

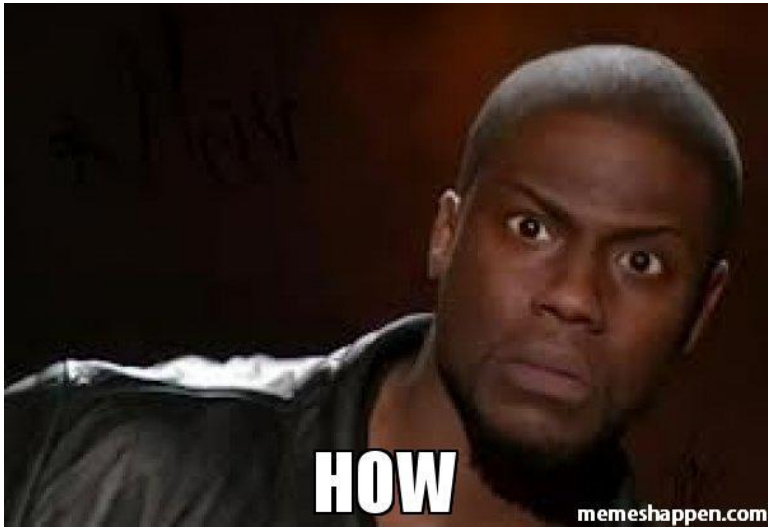
- What is the complexity? O(N)

Quick Select
and average case complexity

- Finally our average case complexity is from the equation:

$$T_N = \frac{N-1}{N} T_{N-1} + 2 + \frac{1}{N^2(N-1)}$$

- What is the complexity? O(N)



Quick Select
and average case complexity

- Finally our average case complexity is from the equation:

$$T_N = \boxed{\frac{N-1}{N}} T_{N-1} + 2 + \boxed{\frac{1}{N^2(N-1)}}$$

- What is the complexity? O(N)
  - Both red boxes are < 1

Quick Select
and average case complexity

- Finally our average case complexity is from the equation:

$$T_N = \boxed{\frac{N-1}{N}} T_{N-1} + 2 + \boxed{\frac{1}{N^2(N-1)}}$$

- What is the complexity? O(N)
  - Both red boxes are < 1

$$T_N < T_{N-1} + 3$$

  - And for complexity, we are only concerns with bounds!

Quick Select
and average case complexity

- Finally our average case complexity is from the equation:

$$T_N = \boxed{\frac{N-1}{N}} T_{N-1} + 2 + \boxed{\frac{1}{N^2(N-1)}}$$

<1 <1

- What is the complexity? O(N)
  - Both red boxes are < 1

make right part bigger to have < rather <=

$$T_N < T_{N-1} + 3$$

  - And for complexity, we are only concerns with bounds!

$$T_N < 3N = O(N)$$

Questions?

- What are online algorithms?

MONASH University

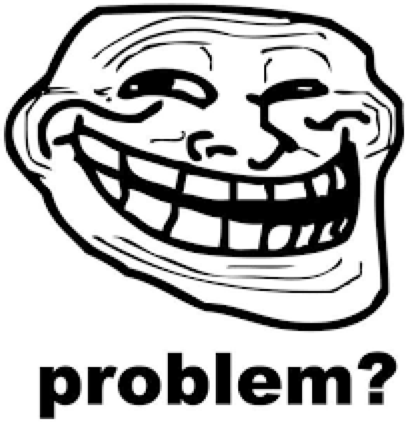
- What are online algorithms?
  – Let say I give you a list of number, ask you to sort it…

- What are online algorithms?
  - Let say I give you a list of number, ask you to sort it…
  - Then I say I left out a number, and give it to you…

- What are online algorithms?
  - Let say I give you a list of number, ask you to sort it…
  - Then I say I left out a number, and give it to you…
  - Then you re-sort it…

- What are online algorithms?
  - Let say I give you a list of number, ask you to sort it…
  - Then I say I left out a number, and give it to you…
  - Then you re-sort it…
  - Then oops, I forgot another number lol

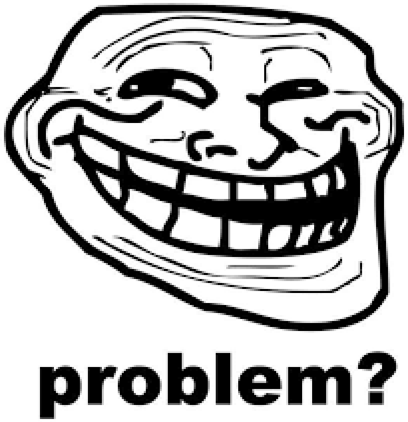
- What are online algorithms?
  – Let say I give you a list of number, ask you to sort it…
  – Then I say I left out a number, and give it to you…
  – Then you re-sort it…
  – Then oops, I forgot another number lol



Bonus
Online algorithms

- What are online algorithms?
  - Let say I give you a list of number, ask you to sort it…
  - Then I say I left out a number, and give it to you…
  - Then you re-sort it…
  - Then oops, I forgot another number lol

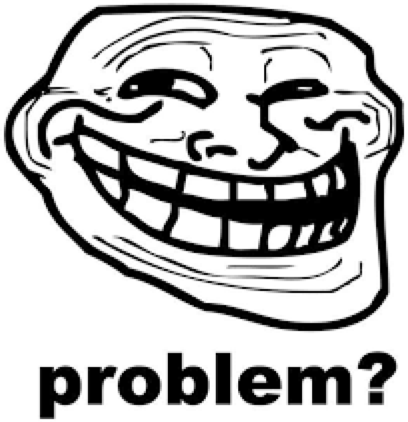
  - Algorithms that can process new information without re-processing the old one



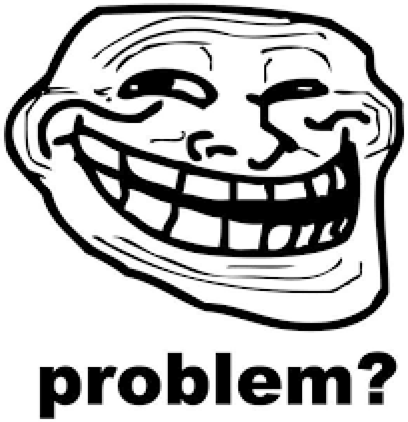
- What are online algorithms?
  - Let say I give you a list of number, ask you to sort it…
  - Then I say I left out a number, and give it to you…
  - Then you re-sort it…
  - Then oops, I forgot another number lol

  - Algorithms that can process new information without re-processing the old one
    - Insertion sort

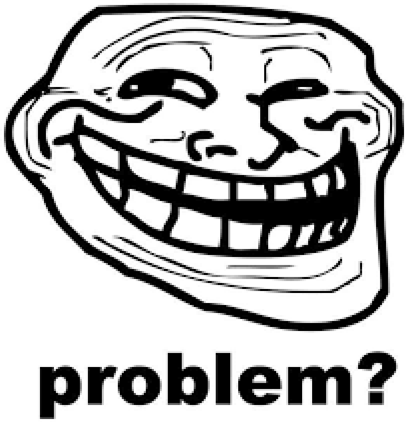
the left part is sorted already
so can do the insertion sort for the new number individually



- What are online algorithms?
  - Let say I give you a list of number, ask you to sort it…
  - Then I say I left out a number, and give it to you…
  - Then you re-sort it…
  - Then oops, I forgot another number lol
  - Algorithms that can process new information without re-processing the old one
    - Insertion sort
    - What about k-th order statistic?



- What are online algorithms?
  - Let say I give you a list of number, ask you to sort it…
  - Then I say I left out a number, and give it to you…
  - Then you re-sort it…
  - Then oops, I forgot another number lol
  - Algorithms that can process new information without re-processing the old one
    - Insertion sort
    - What about k-th order statistic?
      - Does quick select still work?



Bonus
Online algorithms

- From your earlier studio
  - Note: Question number changes between semester

Problem 8. Devise an efficient online algorithm[1] that finds the smallest k elements of a sequence of integers. Write psuedocode for your algorithm. [Hint: Use a data structure that you have learned about in a previous unit]

- From your Tutorial 03 Question 08
  - Using quick select?
  - Using a new approach?

Problem 8. Devise an efficient online algorithm[1] that finds the smallest k elements of a sequence of integers. Write psuedocode for your algorithm. [Hint: Use a data structure that you have learned about in a previous unit]

Questions?

Thank You