Divide and Conquer

Overview

General structure,

Overview

General structure,

- Break problem into subproblems

Overview

General structure,

- Break problem into subproblems
- Solve subproblems

Overview

General structure,

- Break problem into subproblems
- Solve subproblems
- Merge partial solution

Overview

General structure,

- Break problem into subproblems
- Solve subproblems
- Merge partial solution
- DO NOT store solutions

Overview

General structure,

- Break problem into subproblems
- Solve subproblems
- Merge partial solution
- DO NOT store solutions

Most of the time recursive.

Modulo Exponentiation

Input: integer BASE, positive integer EXPO

Modulo Exponentiation

Input: integer BASE, positive integer EXPO

Output: $BASE^{EXPO}$

Modulo Exponentiation

Input: integer BASE, positive integer EXPO

Output: $\text{BASE}^{\text{EXPO}}$

The result can get large, so we typically do all this under some given modulo.

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $BASE^{EXPO}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $\text{BASE}^{\text{EXPO}}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

A trivial solution would,

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $BASE^{EXPO}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

A trivial solution would,

- Keep track of some answer starting at 1 ($BASE^{0}$)

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $\text{BASE}^{\text{EXPO}}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

A trivial solution would,

- Keep track of some answer starting at 1 (BASE^0)
- In a loop that runs EXPO times

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $BASE^{EXPO}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

A trivial solution would,

- Keep track of some answer starting at 1 ($BASE^0$)
- In a loop that runs EXPO times
  - Multiply the answer by BASE

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $\text{BASE}^{\text{EXPO}}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

A trivial solution would,

- Keep track of some answer starting at 1 (BASE^0)
- In a loop that runs EXPO times
  - Multiply the answer by BASE

$$\begin{aligned}\text{BASE}^{\text{EXPO}} &= (\text{BASE}*\text{BASE}*...*\text{BASE}) \\ &= (\text{BASE}*(\text{BASE}*(...)))\end{aligned}$$

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $BASE^{EXPO}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

A trivial solution would,

- Keep track of some answer starting at 1 ($BASE^0$)
- In a loop that runs EXPO times
  - Multiply the answer by BASE (remember to mod)

$$
\begin{aligned}
BASE^{EXPO} &= (BASE*BASE*...*BASE) \\
&= (BASE*(BASE*(...)))
\end{aligned}
$$

Modulo Exponentiation

Input: integer BASE, positive integer EXPO, positive integer MOD

Output: $\text{BASE}^{\text{EXPO}}$ mod MOD

The result can get large, so we typically do all this under some given modulo.

A trivial solution would,

- Keep track of some answer starting at 1 (BASE^0)
- In a loop that runs EXPO times
  - Multiply the answer by BASE (remember to mod)

Runtime?

Fast(er) Modulo Exponentiation

The runtime is a little slow for my tastes.

Fast(er) Modulo Exponentiation

The runtime is a little slow for my tastes.

Some math can help reduce the EXPO needed to find the answer.
(Fermat’s Little Theorem)

Fast(er) Modulo Exponentiation

The runtime is a little slow for my tastes.

Some math can help reduce the EXPO needed to find the answer.
(Fermat's Little Theorem)

Without intermediate modulo arithmetic there is another way to improve runtime.

Fast(er) Modulo Exponentiation

The runtime is a little slow for my tastes.

Some math can help reduce the EXPO needed to find the answer.
(Fermat's Little Theorem)

Without intermediate modulo arithmetic there is another way to improve runtime.

Consider the example 10^{26},

Fast(er) Modulo Exponentiation

The runtime is a little slow for my tastes.

Some math can help reduce the EXPO needed to find the answer.
(Fermat's Little Theorem)

Without intermediate modulo arithmetic there is another way to improve runtime.

Consider the example 10^{26},

10^{26} can be rewritten as $10^{2(13)}$

Fast(er) Modulo Exponentiation

The runtime is a little slow for my tastes.

Some math can help reduce the EXPO needed to find the answer.
    (Fermat's Little Theorem)

Without intermediate modulo arithmetic there is another way to improve runtime.

Consider the example 10^{26},

10^{26} can be rewritten as $10^{2(13)} = (10^{13})^2$

Fast(er) Modulo Exponentiation

The runtime is a little slow for my tastes.

Some math can help reduce the EXPO needed to find the answer.
(Fermat's Little Theorem)

Without intermediate modulo arithmetic there is another way to improve runtime.

Consider the example 10^{26},

10^{26} can be rewritten as $10^{2(13)} = (10^{13})^2$

Compute 10 to the 13 and then take 1 more operation to square the result.

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Don't compute 10^{13} the slow way.

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Don't compute 10^{13} the slow way. $10^{13} = 10^{12+1}$

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Don't compute 10^{13} the slow way. $10^{13} = 10^{12+1} = 10^{1}(10^{12})$

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Don't compute 10^{13} the slow way. $10^{13} = 10^{12+1} = 10^{1}(10^{12}) = 10^{1}(10^{2(6)})$

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Don't compute 10^{13} the slow way. $10^{13} = 10^{12+1} = 10^{1}(10^{12}) = 10^{1}(10^{2(6)}) = 10^{1}(10^{6})^{2}$

We go from 13 operations to 6 + 1 + 1.

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Don't compute 10^{13} the slow way. $10^{13} = 10^{12+1} = 10^1(10^{12}) = 10^1(10^{2(6)}) = 10^1(10^6)^2$

We go from 13 operations to 6 + 1 + 1.

$10^6 = (10^3)^2$

Fast(er) Modulo Exponentiation (cont.)

Continuing the example 10^{26},

Compute 10 to the 13 and then take 1 more operation to square the result.

Don't compute 10^{13} the slow way. $10^{13} = 10^{12+1} = 10^{1}(10^{12}) = 10^{1}(10^{2(6)}) = 10^{1}(10^{6})^{2}$

We go from 13 operations to 6 + 1 + 1.

$10^{6} = (10^{3})^{2}$

We go from 6 operations to 3 + 1.

Fast Mod Expo Algorithm

```
Function expo(base, expo, mod)
    If (expo is trivial [e.g. < 3])
        return using linear expo
    End If
    If (expo is odd)
        make it even by multing expo(base, expo - 1, mod) by base
    Else
        Do the square root trick expo(base, expo / 2, mod) * itself
        // DO NOT CALL THE FUNCTION TWICE!!!
    End If
    Return the result
End Function
```

Fast Mod Expo Analysis

What is the runtime?

Meet in the Middle

How does one find the shortest path distance between two nodes in a unit graph?

Meet in the Middle

How does one find the shortest path distance between two nodes in a unit graph?

Runtime?

Meet in the Middle

How does one find the shortest path distance between two nodes in a unit graph?

Runtime?

Suppose the graph is really big, and branches fast.

Meet in the Middle

How does one find the shortest path distance between two nodes in a unit graph?

Runtime?

Suppose the graph is really big, and branches fast. [Friend network / Chess]

Meet in the Middle

How does one find the shortest path distance between two nodes in a unit graph?

Runtime?

Suppose the graph is really big, and branches fast. [Friend network / Chess]

The visited nodes is roughly $(\text{Branch Factor})^{\text{distance}}$

Meet in the Middle

How does one find the shortest path distance between two nodes in a unit graph?

Runtime?

Suppose the graph is really big, and branches fast. [Friend network / Chess]

The visited nodes is roughly $(\text{Branch Factor})^{\text{distance}}$

We can improve the runtime by searching from both ends.

Meet in the Middle

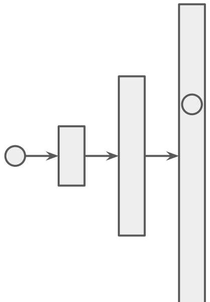
How does one find the shortest path distance between two nodes in a unit graph?

Runtime?

Suppose the graph is really big, and branches fast. [Friend network / Chess]

The visited nodes is roughly $(\text{Branch Factor})^{\text{distance}}$

We can improve the runtime by searching from both ends.



Meet in the Middle

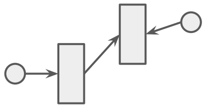
How does one find the shortest path distance between two nodes in a unit graph?

Runtime?

Suppose the graph is really big, and branches fast. [Friend network / Chess]

The visited nodes is roughly (Branch Factor)$^{\text{distance}}$

We can improve the runtime by searching from both ends.



Meet in the Middle

How does one find the shortest path distance between two nodes in a unit graph?

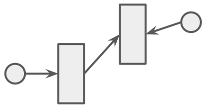
Runtime?

Suppose the graph is really big, and branches fast. [Friend network / Chess]

The visited nodes is roughly (Branch Factor)$^{\text{distance}}$

We can improve the runtime by searching from both ends.

Effectively cutting the distance in half.



Integer Multiplication

Suppose we want to multiply integers x and y together.

Integer Multiplication

Suppose we want to multiply integers x and y together.

Let x be expressed as digits $X_n...X_1X_0$.

Let y be expressed as digits $Y_n...Y_1Y_0$.

Integer Multiplication

Suppose we want to multiply integers x and y together.

Let x be expressed as digits $X_n...X_1X_0$.

Let y be expressed as digits $Y_n...Y_1Y_0$.

Let z be their product,

Integer Multiplication

Suppose we want to multiply integers x and y together.

Let x be expressed as digits $X_n...X_1X_0$.

Let y be expressed as digits $Y_n...Y_1Y_0$.

Let z be their product,

$Z_0=X_0Y_0$

Integer Multiplication

Suppose we want to multiply integers x and y together.

Let x be expressed as digits $X_n...X_1X_0$.

Let y be expressed as digits $Y_n...Y_1Y_0$.

Let z be their product,

$Z_0=X_0Y_0$

$Z_1=X_1Y_0 + X_0Y_1$

Integer Multiplication

Suppose we want to multiply integers x and y together.

Let x be expressed as digits $X_n...X_1X_0$.

Let y be expressed as digits $Y_n...Y_1Y_0$.

Let z be their product,

$Z_0=X_0Y_0$

$Z_1=X_1Y_0 + X_0Y_1$

$Z_2=X_2Y_0 + X_1Y_1 + X_0Y_2$

Integer Multiplication

Suppose we want to multiply integers x and y together.

Let x be expressed as digits $X_n \ldots X_1 X_0$.

Let y be expressed as digits $Y_n \ldots Y_1 Y_0$.

Let z be their product,

$Z_0 = X_0 Y_0$

$Z_1 = X_1 Y_0 + X_0 Y_1$

$Z_2 = X_2 Y_0 + X_1 Y_1 + X_0 Y_2$

We will compute each digit of Z.
What would be the runtime?

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

We will let a be the most significant n/2 digits of x and
b be the least significant n/2 digits of x

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

We will let a be the most significant n/2 digits of x and
b be the least significant n/2 digits of x

$x = a*10^{n/2}+b$

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

We will let a be the most significant n/2 digits of x and
b be the least significant n/2 digits of x

$x = a*10^{n/2}+b$

We will let c be the most significant n/2 digits of y and
d be the least significant n/2 digits of y

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

We will let a be the most significant n/2 digits of x and
b be the least significant n/2 digits of x

$x = a*10^{n/2}+b$

We will let c be the most significant n/2 digits of y and
d be the least significant n/2 digits of y

$y = c*10^{n/2}+d$

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

We will let a be the most significant n/2 digits of x and
b be the least significant n/2 digits of x

$x = a*10^{n/2}+b$

We will let c be the most significant n/2 digits of y and
d be the least significant n/2 digits of y

$y = c*10^{n/2}+d$

We need to compute ac, ad, bc, and bd.

$xy = ac(10^n)+(ad+bc)(10^{n/2})+bd$

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

We will let a be the most significant n/2 digits of x and
b be the least significant n/2 digits of x

$x = a*10^{n/2}+b$

We will let c be the most significant n/2 digits of y and
d be the least significant n/2 digits of y

$y = c*10^{n/2}+d$

$xy = ac(10^{n})+(ad+bc)(10^{n/2})+bd$

We need to compute ac, ad, bc, and bd.
This would have the recurrence relation of
$T(N) = 4T(N/2)+O(N)$

Karatsuba

We will try to improve the runtime by breaking the problem into 4 subproblems.

We will let a be the most significant n/2 digits of x and
b be the least significant n/2 digits of x

$x = a*10^{n/2}+b$

We will let c be the most significant n/2 digits of y and
d be the least significant n/2 digits of y

$y = c*10^{n/2}+d$

$xy = ac(10^n)+(ad+bc)(10^{n/2})+bd$

We need to compute ac, ad, bc, and bd.
This would have the recurrence relation of
$T(N) = 4T(N/2)+O(N)$
...
$T(N) = N^2$

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Instead compute ac, bd, and (a+b)(c+d).

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Instead compute ac, bd, and (a+b)(c+d).

Note that a+b and c+d has around N digits.

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Instead compute ac, bd, and (a+b)(c+d).

Note that a+b and c+d has around N digits.

$(a+b)(c+d) = ac+ad+bc+bd$

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Instead compute ac, bd, and (a+b)(c+d).

Note that a+b and c+d has around N digits.

$(a+b)(c+d) = ac+ad+bc+bd$

$(a+b)(c+d) - ac - bd = ac+ad+bc+bd -ac - bd$

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Instead compute ac, bd, and (a+b)(c+d).

Note that a+b and c+d has around N digits.

$(a+b)(c+d) = ac+ad+bc+bd$

$(a+b)(c+d) - ac - bd = ac+ad+bc+bd -ac - bd$

$(a+b)(c+d) - ac - bd = ad+bc$

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Instead compute ac, bd, and (a+b)(c+d).

Note that a+b and c+d has around N digits.

$(a+b)(c+d) = ac+ad+bc+bd$

$(a+b)(c+d) - ac - bd = ac+ad+bc+bd - ac - bd$

$(a+b)(c+d) - ac - bd = ad+bc$
(this is the term needed for the middle part of the product)

Karatsuba (cont.)

We need to compute ac, ad, bc, and bd.

Instead compute ac, bd, and (a+b)(c+d).

Note that a+b and c+d has around N digits.

$(a+b)(c+d) = ac+ad+bc+bd$

$(a+b)(c+d) - ac - bd = ac+ad+bc+bd - ac - bd$

$(a+b)(c+d) - ac - bd = ad+bc$
(this is the term needed for the middle part of the product)

$xy = ac(10^n)+(ad+bc)(10^{n/2})+bd$

Karatsuba Algorithm Analysis

Runtime?

Karatsuba Algorithm Analysis

Runtime?

$T(N) = 3T(N/2) + O(N)$

Median of an Array

Input: An unsorted array of N integers.

Median of an Array

Input: An unsorted array of N integers.

Output: The value of the median.

Median of an Array

Input: An unsorted array of N integers.

Output: The value of the median.

Straightforward solution,

Median of an Array

Input: An unsorted array of N integers.

Output: The value of the median.

Straightforward solution,

- Sort array

Median of an Array

Input: An unsorted array of N integers.

Output: The value of the median.

Straightforward solution,

- Sort array
- Return the middle value

Median of an Array

Input: An unsorted array of N integers.

Output: The value of the median.

Straightforward solution,

- Sort array
- Return the middle value

Runtime?

Median of an Array (D and C)

~~Suppose we knew of 100 values that were greater than the median and we knew of 100 values that are smaller than the median.~~

Median of an Array (D and C)

~~Suppose we knew of 100 values that were greater than the median and we knew of 100 values that are smaller than the median.~~

Not important!

Median of an Array (D and C)

~~Suppose we knew of 100 values that were greater than the median and we knew of 100 values that are smaller than the median.~~

Not important!

Instead we will find the k-th smallest value of the array.

Median of an Array (D and C)

We will start by breaking the array into groups of 5.

Median of an Array (D and C)

We will start by breaking the array into groups of 5.

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, \ldots$

Median of an Array (D and C)

We will start by breaking the array into groups of 5.

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, \ldots$

Median of an Array (D and C)

We will start by breaking the array into groups of 5.

We then sort those groups of 5

$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, \ldots$

Median of an Array (D and C)

We will start by breaking the array into groups of 5.

We then sort those groups of 5, 5*n [25 *(n/5)] operations

$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, \ldots$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)

$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, \ldots$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)

We take the median of each group and emplace them into a new array.

$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, \ldots$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)

We take the median of each group and emplace them into a new array.

$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, \ldots$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)

We take the median of each group and emplace them into a new array.

$(a'_3, a'_8, a'_{13}, ...)$

$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, ...$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)

We take the median of each group and emplace them into a new array.

$(a'_3, a'_8, a'_{13}, ...)$

Find the median of this new array using our “method”, m’

$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, ...$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)+F(N/5)

We take the median of each group and emplace them into a new array.

$(a'_3, a'_8, a'_{13}, \ldots)$

Find the median of this new array using our “method”, m’

(Takes F(N/5) operations)

$a'_1, a'_2, \boxed{a'_3}, a'_4, a'_5, a'_6, a'_7, \boxed{a'_8}, a'_9, a'_{10}, a'_{11}, a'_{12}, \boxed{a'_{13}}, a'_{14}, a'_{15}, \ldots$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)+F(N/5)

We take the median of each group and emplace them into a new array.

  (a'_3, a'_8, a'_{13}, ...)

Find the median of this new array using our “method”, m’

  (Takes F(N/5) operations)

We examine the groups that have a middle value that is less than m’

$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, \ldots$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)+F(N/5)

We take the median of each group and emplace them into a new array.

$(a'_3, a'_8, a'_{13}, ...)$

Find the median of this new array using our “method”, m’

(Takes F(N/5) operations)

We examine the groups that have a middle value that is less than m’

$(b_1, b_2, b_3, b_4, b_5)$

$a'_1, a'_2, \boxed{a'_3}, a'_4, a'_5, a'_6, a'_7, \boxed{a'_8}, a'_9, a'_{10}, a'_{11}, a'_{12}, \boxed{a'_{13}}, a'_{14}, a'_{15}, ...$

Median of an Array (D and C) (cont.)

Total operations so far (5*n operations)+F(N/5)

We take the median of each group and emplace them into a new array.

$(a'_3, a'_8, a'_{13}, ...)$

Find the median of this new array using our “method”, m’

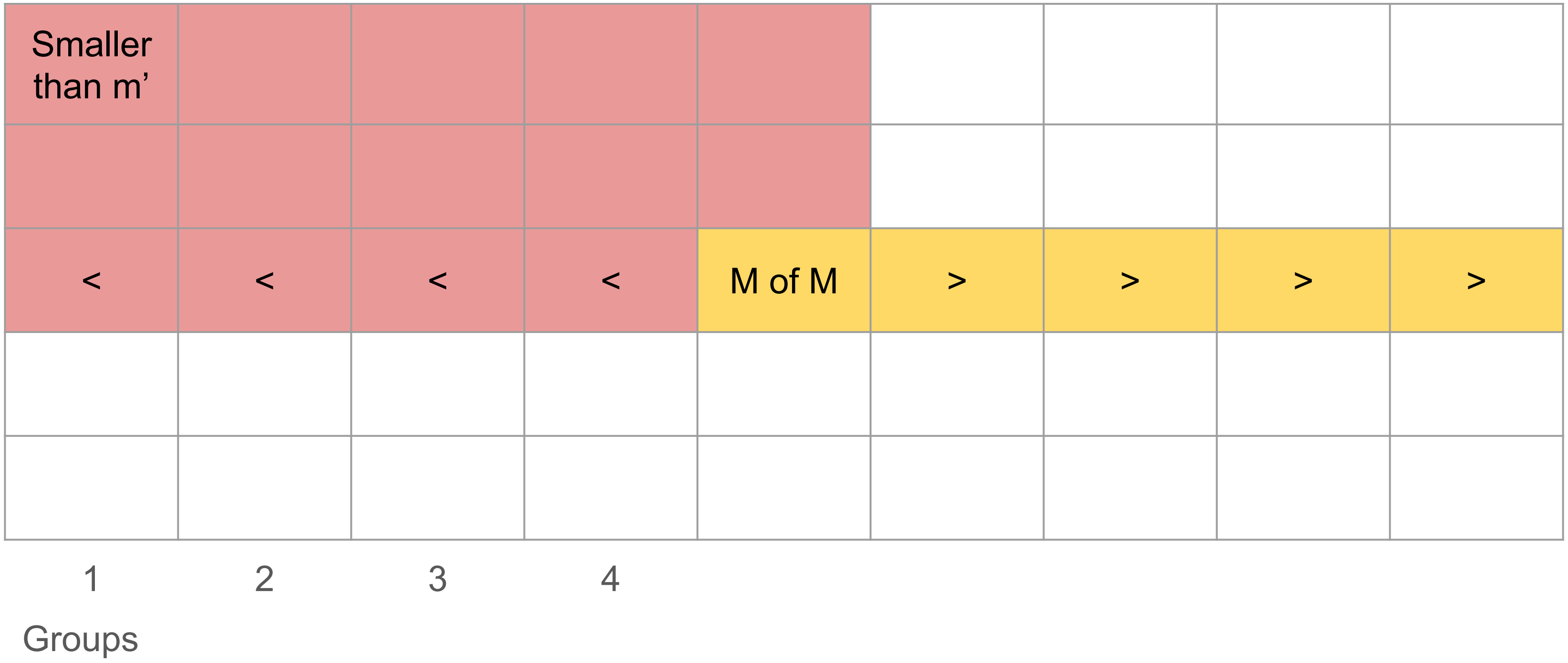
(Takes F(N/5) operations)

We examine the groups that have a middle value that is less than m’

$(b_1, b_2, b_3, b_4, b_5)$ b_1 and b_2 will not be the median.

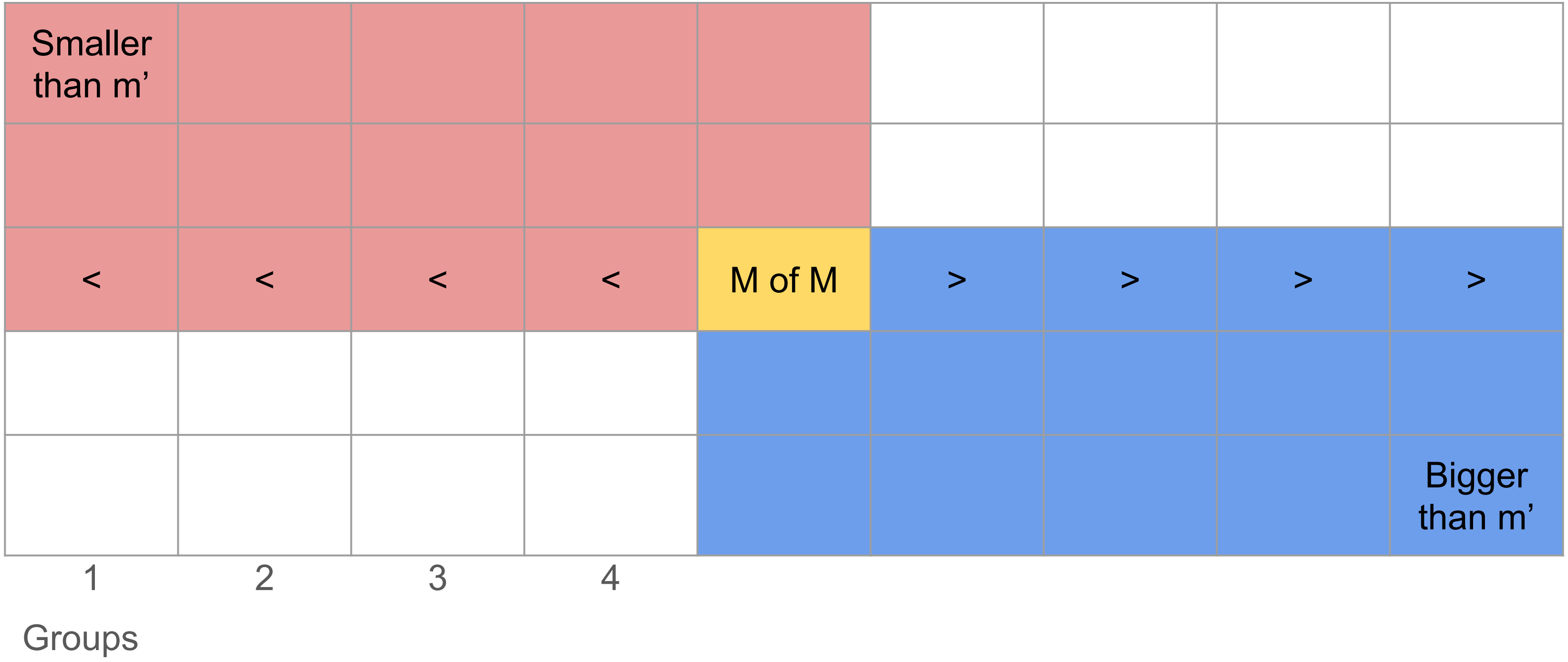
$a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7, a'_8, a'_9, a'_{10}, a'_{11}, a'_{12}, a'_{13}, a'_{14}, a'_{15}, ...$

Median of an Array (D and C) (cont.) (cont.)



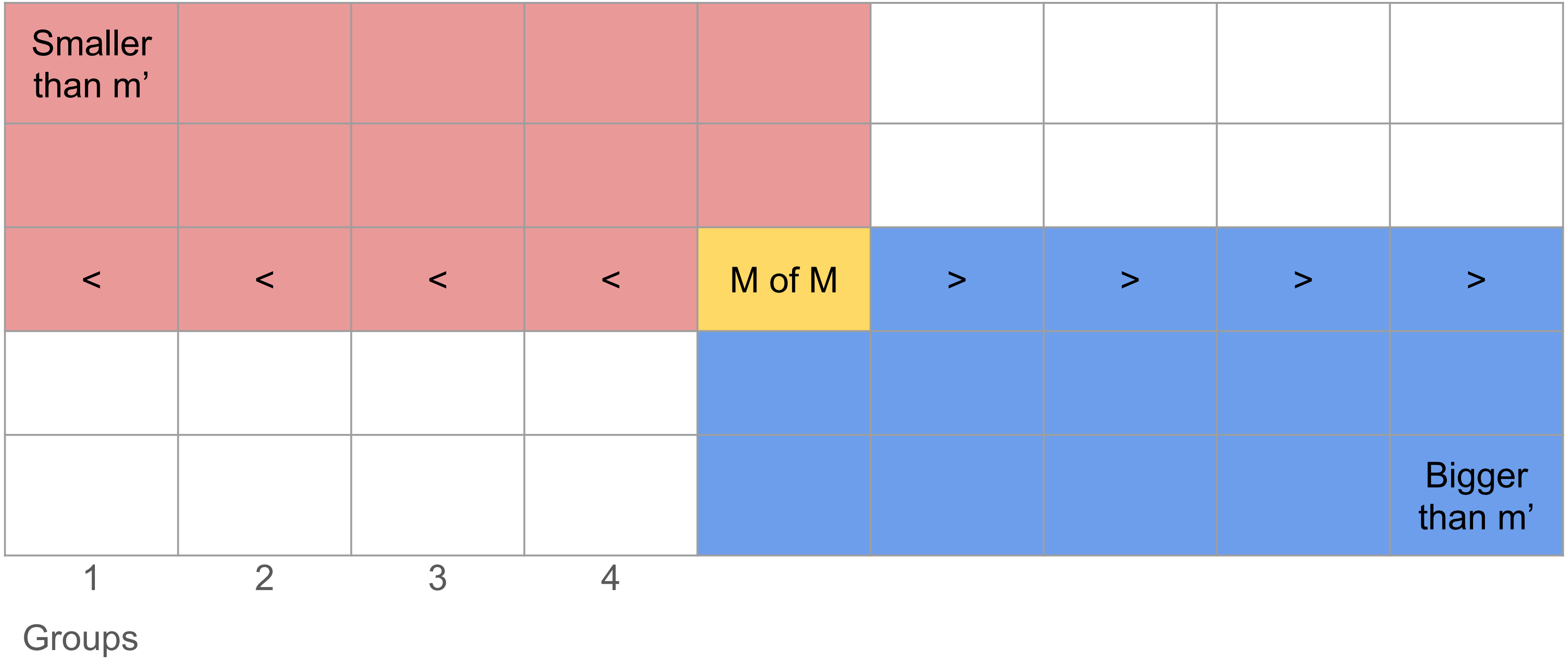
Median of an Array (D and C) (cont.) (cont.)

Similarly.



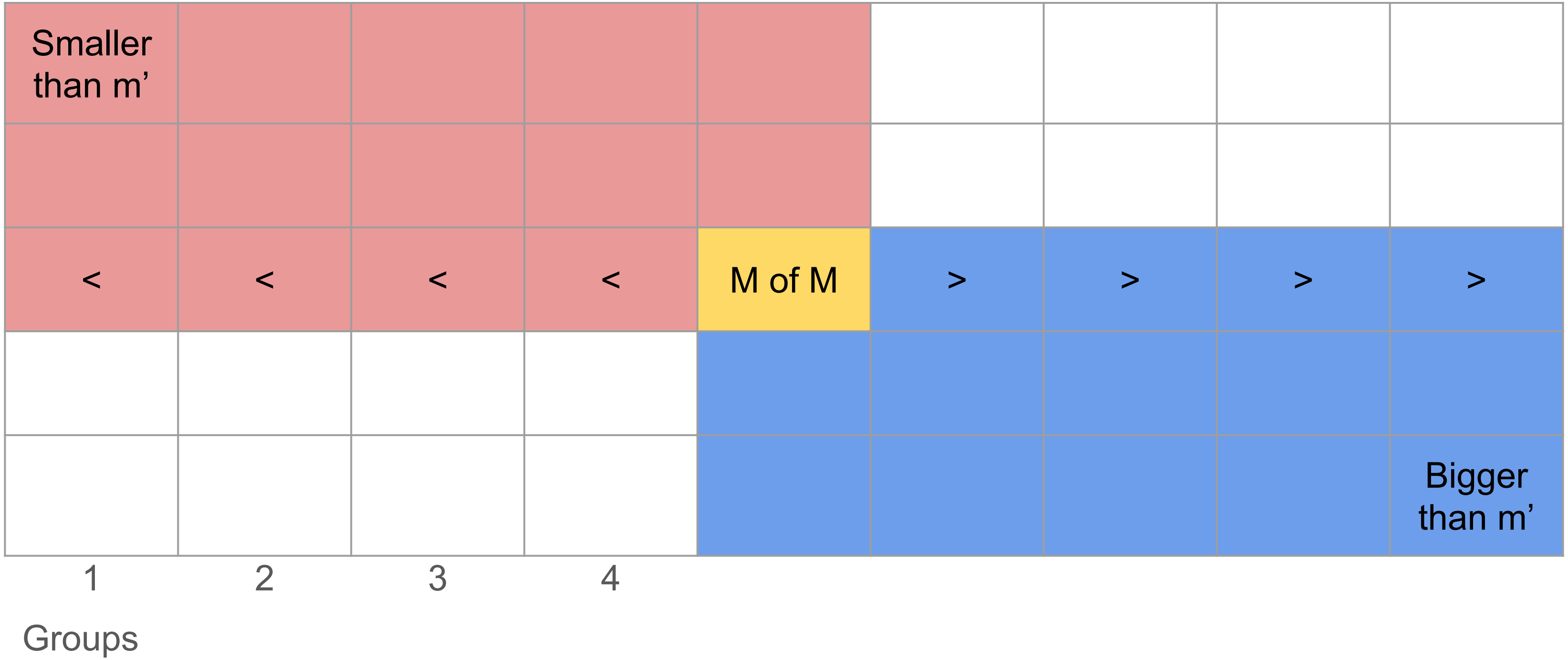
Median of an Array (D and C) (cont.) (cont.)

We use m’ as a pivot.



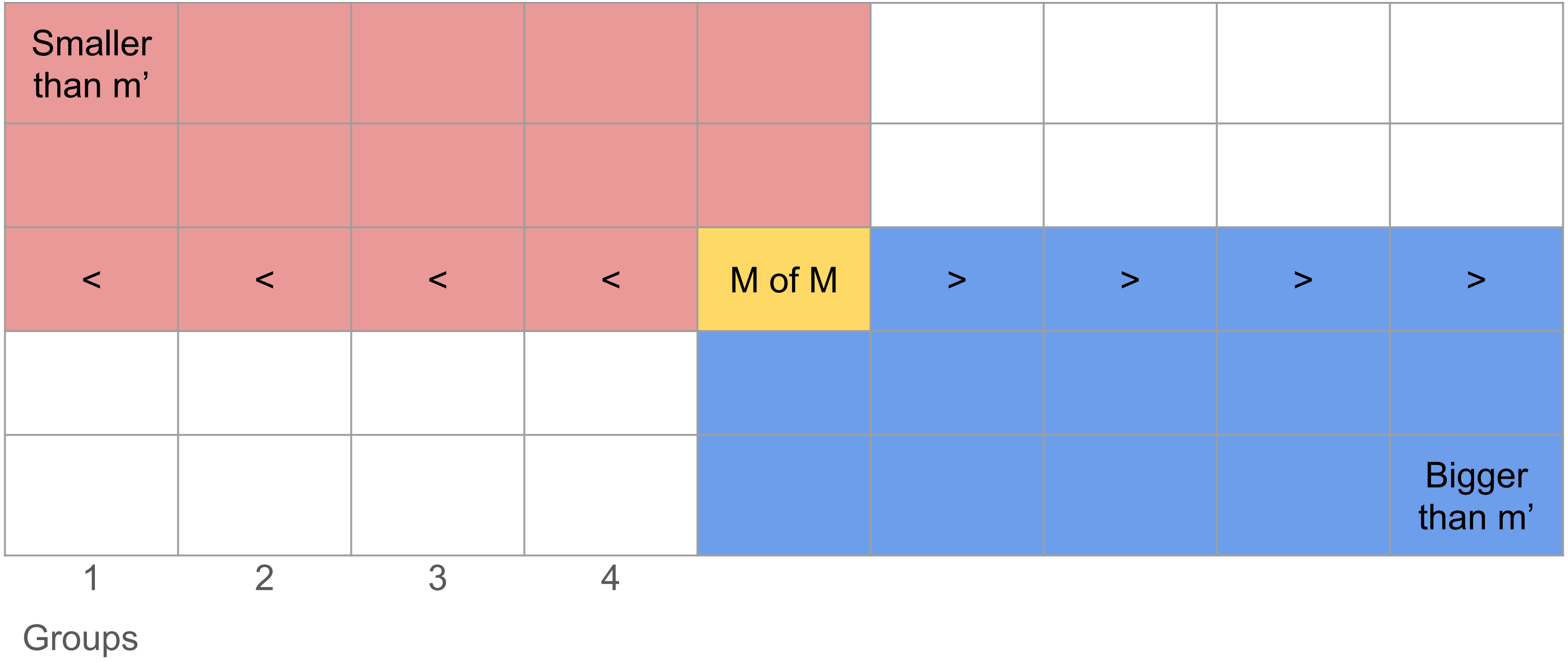
Median of an Array (D and C) (cont.) (cont.)

We use m’ as a pivot. Find values greater than m’; find values smaller than m’.



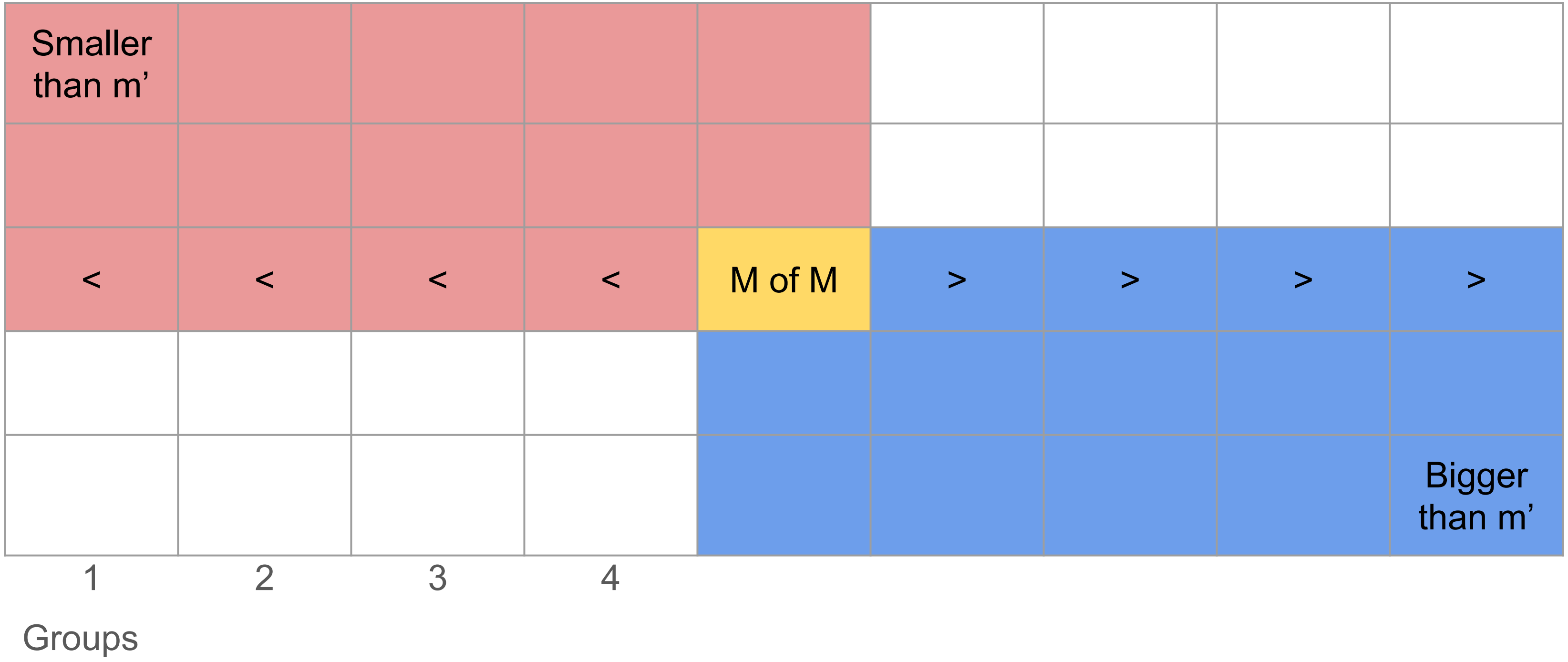
Median of an Array (D and C) (cont.) (cont.)

We recursive search in the necessary array and adjust the k as necessary.



Median of an Array (D and C) (cont.) (cont.)

The worst case array size is 7/10 * n



Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10)$

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10)$

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \le 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Upper bound!

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Upper bound!

$F(N) = F(9/10N) + 5N$

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Upper bound!

$F(N) = F(9/10N) + 5N$ = some math

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Upper bound!

$F(N) = F(9/10N) + 5N =$ some math

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Upper bound!

$F(N) = F(9/10N) + 5N =$ some math $= 10*(5N)$

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Upper bound!

$F(N) = F(9/10N) + 5N =$ some math $= 10*(5N) = 50N$

Analysis Median of an Array

Total operations so far (5*N operations)+F(N/5)+F(7N/10)

If F(N) is larger than linear (i.e. $\omega(N)$), then $F(aN) + F(bN) < F((a+b)N)$.

Thus $F(N) = 5N+F(N/5)+F(7N/10) \leq 5N + F(N/5+7N/10) = 5N + F(9/10N)$

Upper bound!

$F(N) = F(9/10N) + 5N =$ some math $= 10*(5N) = 50N \in O(N)$