LECTURE 4: DIVIDE & CONQUER WITH BINARY SEARCH & MERGESORT

Harold Soh
harold@comp.nus.edu.sg

ADMINISTRATIVE ISSUES

Start 2019-08-25 16:00 UTC **Problem Set 1** **End** 2019-09-09 16:00 UTC

Time elapsed 15:19:47 **Time remaining** 344:40:13

PS1 was released on Monday. Due in 2 weeks.

Note:

- Sign up on nus.kattis.com
- Submit only Java code
- Don't plagiarize

ADMINISTRATIVE ISSUES

Feeling confused or have questions?

Discussion Groups (Labs) and Tutorials **<u>start this week</u>**.

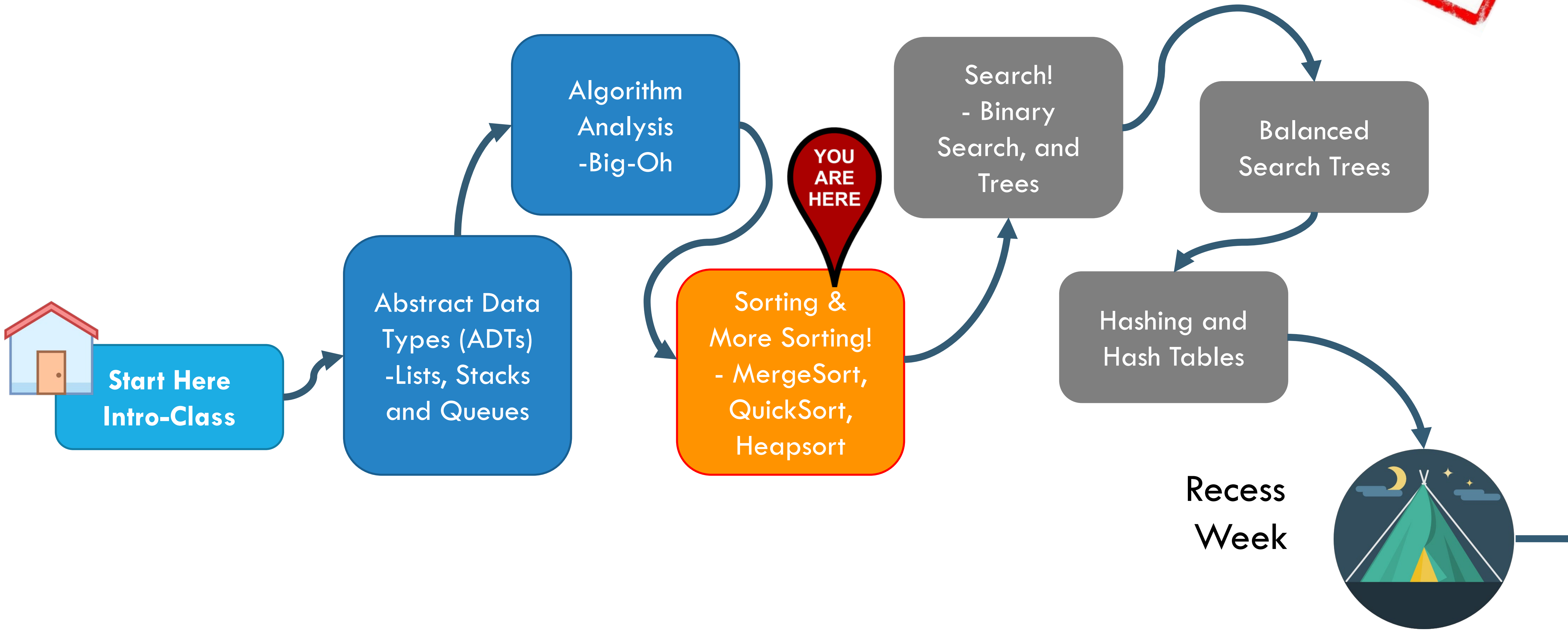
DG/Tutorials are being reorganized.

- Will announce on piazza

Group	Members
B01 (CS2040S)	1
B02 (CS2040S)	8
B03 (CS2040S)	23
B04 (CS2040S)	2
B05 (CS2040S)	10
B06 (CS2040S)	1
B07 (CS2040S)	14
B08 (CS2040S)	23
B09 (CS2040S)	4

PATH TO MASTERY / COURSE STRUCTURE

DRAFT



QUIZZES: 20%

Quiz 1 : Week 4 (during Lecture Slot)

Quiz 2 : Week 7 (during Lecture Slot)

Quiz 3 : Week 11 (during Lecture Slot)

All Dates To be Confirmed (TBC)

We will take the best 2/3 (10% each)

QUIZ 1

On **Sept 4th**

Please don't be late.

Covers everything up to **Quicksort** (tomorrow's lecture).

It's about **feedback.**

COURSE FEEDBACK

Please give your feedback! It's important!

https://forms.gle/9XYMGsWr2d98CzwQ7

Will post link on piazza

QUESTIONS BEFORE WE GET STARTED?

LEARNING OUTCOMES

By the end of this session, you should be able to:

- Use the **divide and conquer strategy** for problem solving
- Explain the **binary search algorithm** and prove its correctness
- Explain the **mergesort algorithm** as an example of the divide and conquer strategy.
- Analyze and describe the **performance of mergesort and binary search** using $O(g(n))$ notation.

PROBLEM: CUSTOMER LOYALTY REWARDS

LAST LECTURE

Get a list of customers ordered by their purchasing spend. Reward the k who spent the most.

PROBLEM: FIND CUSTOMERS SPENDING BETWEEN $a AND $b WHERE b > a

Data is in the customers array

Array sorted by customer spend

Homer proposes a simple algorithm:

```
start = findIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

Company Mascot

PROBLEM: FIND CUSTOMERS SPENDING BETWEEN $a AND $b WHERE b > a

Company Mascot

Data is in the customers array

Array sorted by customer spend

Homer proposes a simple algorithm:

```
start = findIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

Poll Everywhere

https://bit.ly/2LvG9bq

WHAT IF THE ARRAY WAS UNSORTED?

How fast can you find the customer who spent $a in an **unsorted array**?

A. $O(n \log n)$
B. $O(n)$
C. $O(n^2)$
D. $O(1)$

Poll Everywhere

https://bit.ly/2LvG9bq

WHAT IF THE ARRAY WAS UNSORTED?

Linear search

How fast can you find the customer who spent $a in an **unsorted array**?

A. $O(n \log n)$
B. $\boldsymbol{O(n)}$
C. $O(n^2)$
D. $O(1)$

BUT OUR ARRAY IS **SORTED**

So we will exploit this fact!

Exploiting structure is a common approach in CS

5	3	2	1	4

vs

1	2	3	4	5

FINDING 11: THE IDEA

2	4	4	5	6	7	8	9	11	17	23	28

Question: if `A[i] < 11` then do we search to the left or right of i?

FINDING 11: THE IDEA

2	4	4	5	6	7	8	9	11	17	23	28
2	4	4	5	6	7	8	9	11	17	23	28

FINDING 11: THE IDEA

2	4	4	5	6	7	8	9	11	17	23	28

2	4	4	5	6	7	8	9	11	17	23	28

FINDING 11: THE IDEA

2	4	4	5	6	7	8	9	11	17	23	28

2	4	4	5	6	7	8	9	11	17	23	28

2	4	4	5	6	7	8	9	11	17	23	28

FINDING 11

2	4	4	5	6	7	8	9	11	17	23	28
2	4	4	5	6	7	8	9	11	17	23	28
2	4	4	5	6	7	8	9	11	17	23	28
2	4	4	5	6	7	8	9	11	17	23	28

PSEUDOCODE FOR BINARY SEARCH

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

2	5	7	11	13

IS THE CODE CORRECT?

Let's make sure it works.

How?

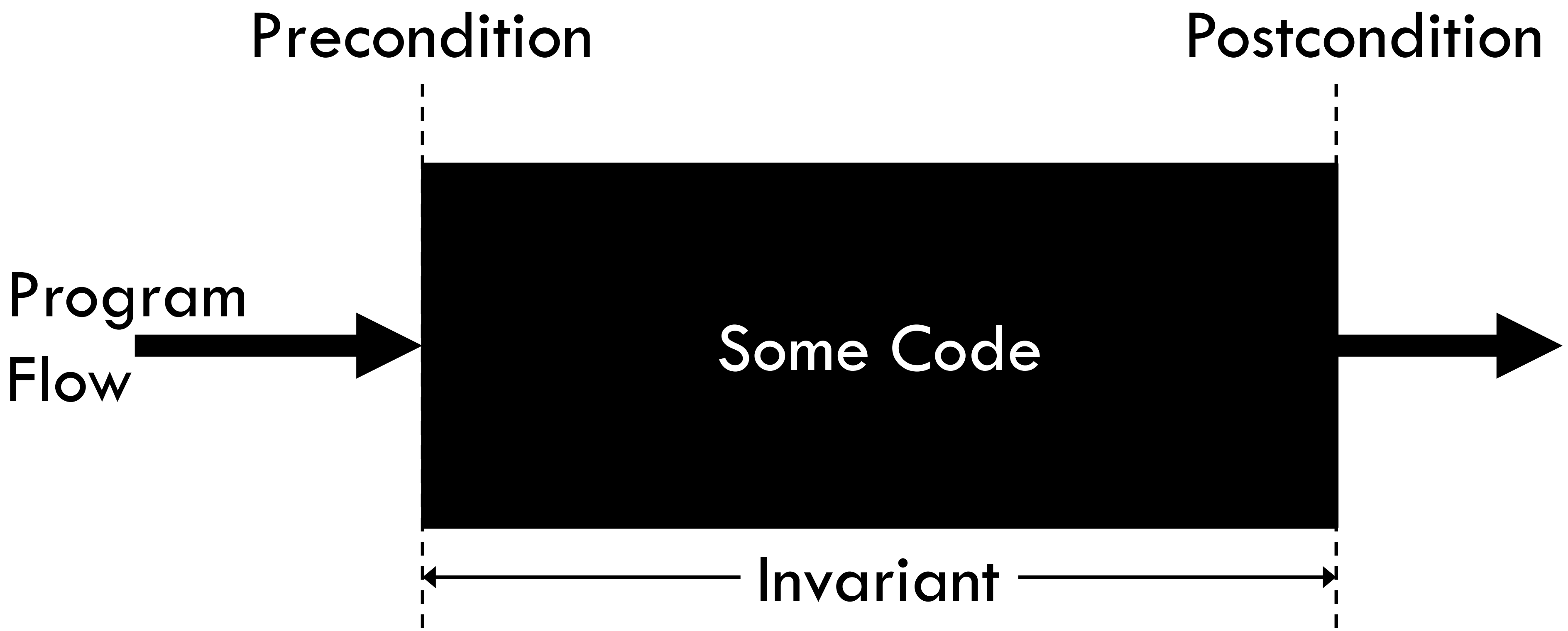
PRE- AND POST-CONDITIONS

Precondition: condition that is true *before* some set of operations (e.g., a loop or a function)

Postcondition: condition that is true *after* some set of operations

"Programming by Contract"

CONDITIONS AND INVARIANTS



INVARIANTS

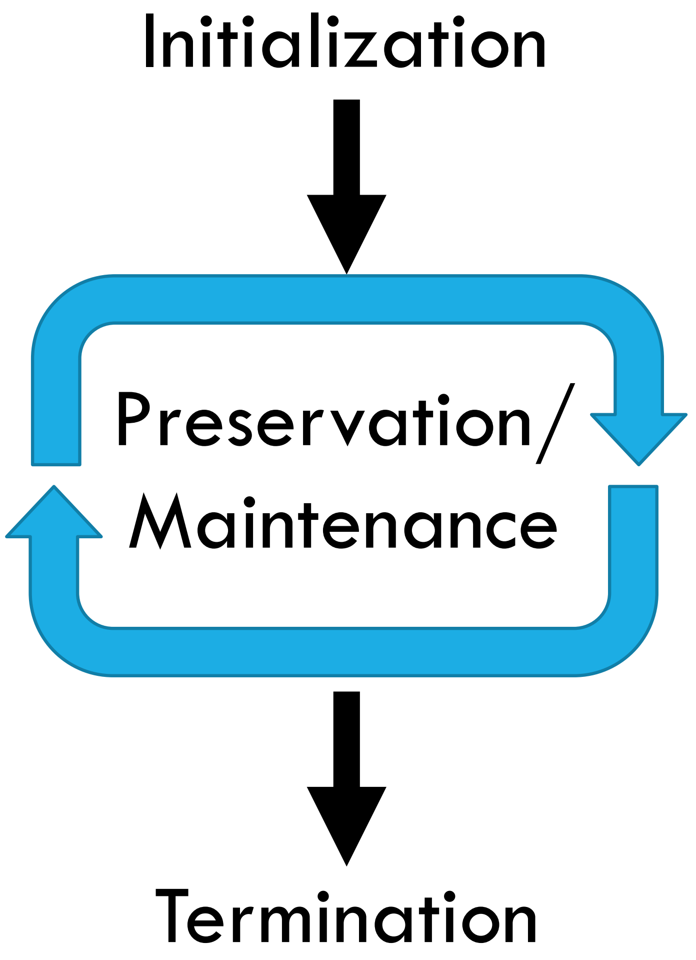
Invariant: condition that is true during some execution of a program or a set of operations

Loop Invariant: a condition that is true before (and after) each iteration of a loop

SO, IS BINARY SEARCH CORRECT?

Strategy: Establish 3 properties for the main loop

- **Initialization:** we've set up our invariant
- **Preservation:** the invariant is true at *every* iteration
- **Termination:** when the loop terminates, the desired result is true



[Programming Pearls, Bentley]

BINARY SEARCH CORRECTNESS

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

Preconditions:

-
-

Postcondition:

-
-

BINARY SEARCH CORRECTNESS

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

Preconditions:

- A is of size n
- A is sorted

Postcondition:

- A[mid] = key
- or key not found

"Programming by Contract"

BINARY SEARCH CORRECTNESS

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

Loop Invariant:
A[low] ≤ key ≤ A[high]

(if key is in A)

Strategy: Work through the loop and consider each case.

BINARY SEARCH CORRECTNESS

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

Loop Termination:
A[mid] = key
or high < low (not_found)

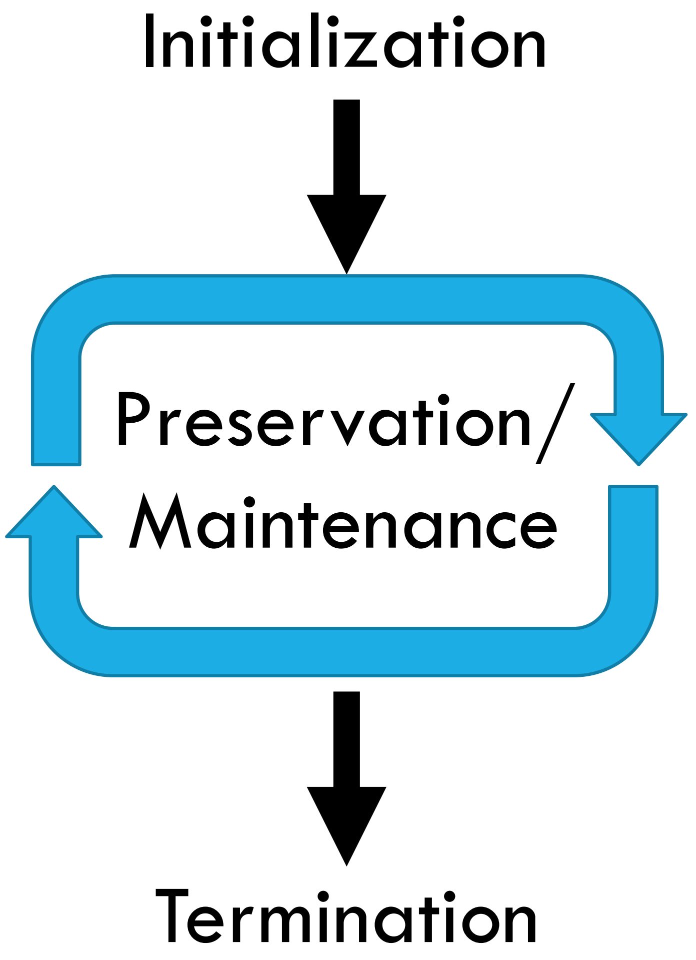
Check: Is the desired condition true?

Ensure that our postcondition holds.

SO, IS BINARY SEARCH CORRECT?

Strategy: Establish 3 properties for the main loop

- ✓**Initialization:** we've set up our invariant
- ✓**Preservation:** the invariant is true at every iteration
- ✓**Termination:** when the loop terminates, the desired result is true



[Programming Pearls, Bentley]

SO, IS BINARY SEARCH CORRECT?

Yes!

(well, almost, we'll cover that later)

What is the worst-case time complexity for binary search?

A. $O(\log n)$
B. $O(n)$
C. $O(n^2)$
D. $O(1)$

Poll Everywhere

https://bit.ly/2LvG9bq

HOW FAST IS BINARY SEARCH?

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

What is the worst-case time complexity for binary search?

A. $O(\log n)$
B. $O(n)$
C. $O(n^2)$
D. $O(1)$

Poll Everywhere
https://bit.ly/2LvG9bq

HOW FAST IS BINARY SEARCH?

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

What is the worst-case time complexity for binary search?

A. $\boldsymbol{O(\log n)}$
B. $O(n)$
C. $O(n^2)$
D. $O(1)$

TIME COMPLEXITY OF BINARY SEARCH

n	2	4	4	5	6	7	8	9	11	17	23	28

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

At each iteration, how much of the array are we "shaving off"?

Poll Everywhere

https://bit.ly/2LvG9bq

n

TIME COMPLEXITY OF BINARY SEARCH

n											
2	4	4	5	6	7	8	9	11	17	23	28

Iteration 1: n

Iteration 2: $n/2$

Iteration 3: $n/4$

...

Iteration k: $n/2^k = 1$

If $n/2^k = 1$, what is k in terms of n ?

A. $k = 2^n$

B. $k = 2/n$

C. $k = \log(n)$

D. Am I in CS or Math?

Poll Everywhere

https://bit.ly/2LvG9bq

n

TIME COMPLEXITY OF BINARY SEARCH

n	2	4	4	5	6	7	8	9	11	17	23	28

Iteration 1: n

Iteration 2: $n/2$

Iteration 3: $n/4$

...

Iteration k: $n/2^k = 1$

Each iteration takes c time

so $T(n) = c\log n = O(\log n)$

If $n/2^k = 1$, what is k in terms of n ?

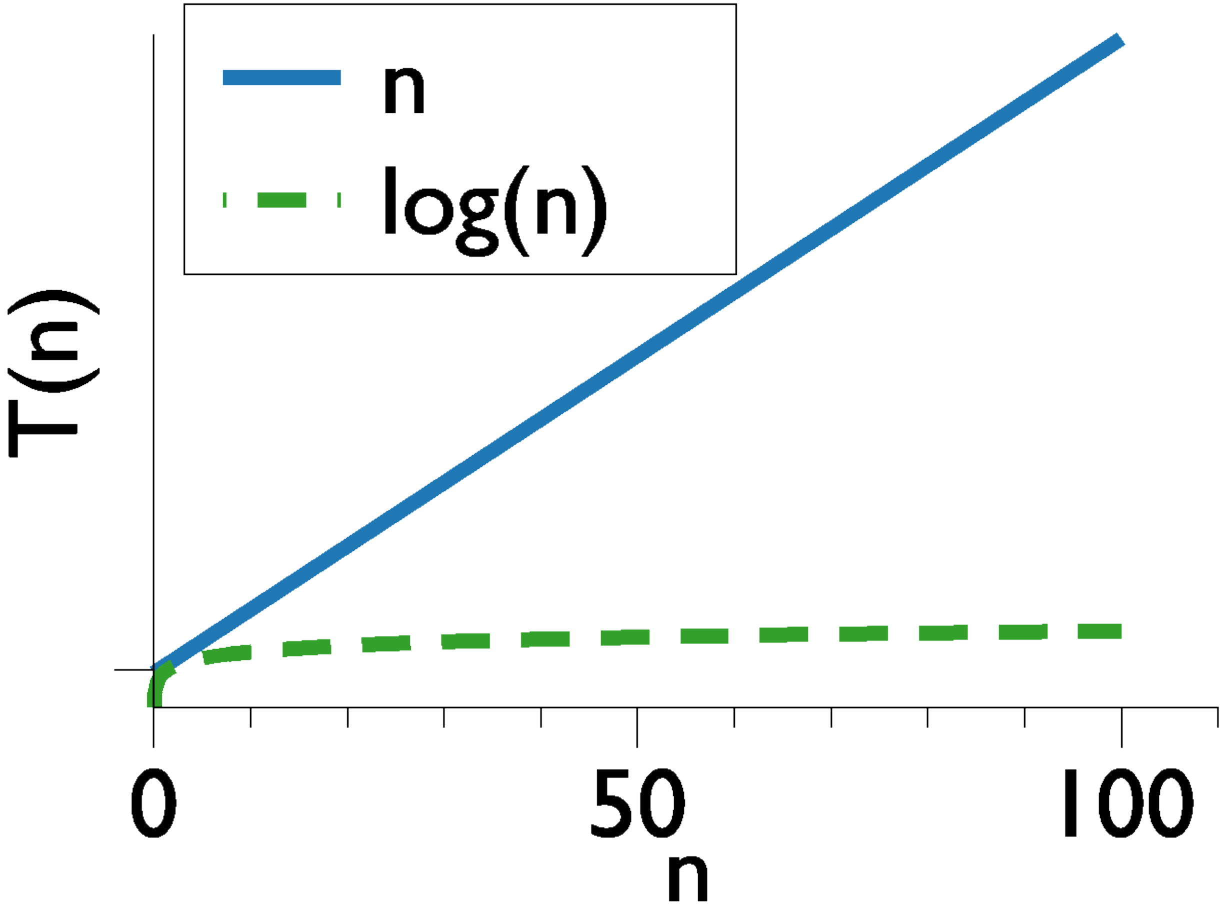
A. $k = 2^n$

B. $k = 2/n$

C. $k = \log(n)$

D. Am I in CS or Math?

WHAT IS THE DIFFERENCE?



Poll Everywhere

https://bit.ly/2LvG9bq

SPACE COMPLEXITY OF BINARY SEARCH

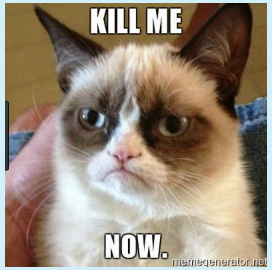
```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

What is the worst-case space complexity for binary search?

A. $O(n \log n)$

B. $O(n)$

C. $O(1)$

D. 

Poll Everywhere

https://bit.ly/2LvG9bq

SPACE COMPLEXITY OF BINARY SEARCH

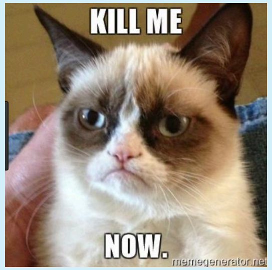
```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

What is the worst-case space complexity for binary search?

A. $O(n \log n)$

B. $O(n)$

C. $\boldsymbol{O(1)}$

D. 

DIVIDE & CONQUER STRATEGY: THE 3 STEPS

1. (*Deviously*) Split your problem into subproblems.
2. (*Gleefully*) Solve each subproblem (recursively).
3. (*Cleverly*) Combine the solutions for each subproblem to produce a solution to the original problem.

is the basis for many efficient algorithms

BACK TO OUR INITIAL PROBLEM

find customers spending between \$a and \$b where b $\geq$ a

Array sorted by customer spend

```
start = findIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
    println(customers[i])
```

What is the worst-case time complexity for algorithm on the left if you use binary search for findIndex?

A. $O(n \log n)$
B. $O(n)$
C. $O(1)$
D. 

Poll Everywhere
https://bit.ly/2LvG9bq

BACK TO OUR INITIAL PROBLEM

find customers spending between $a and $b where b ≥ a

Array sorted by customer spend

```
start = findIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

What is the worst-case time complexity for algorithm on the left if you use binary search for findIndex?

A. $O(n \log n)$
B. $\boldsymbol{O(n)}$
C. $O(1)$
D.

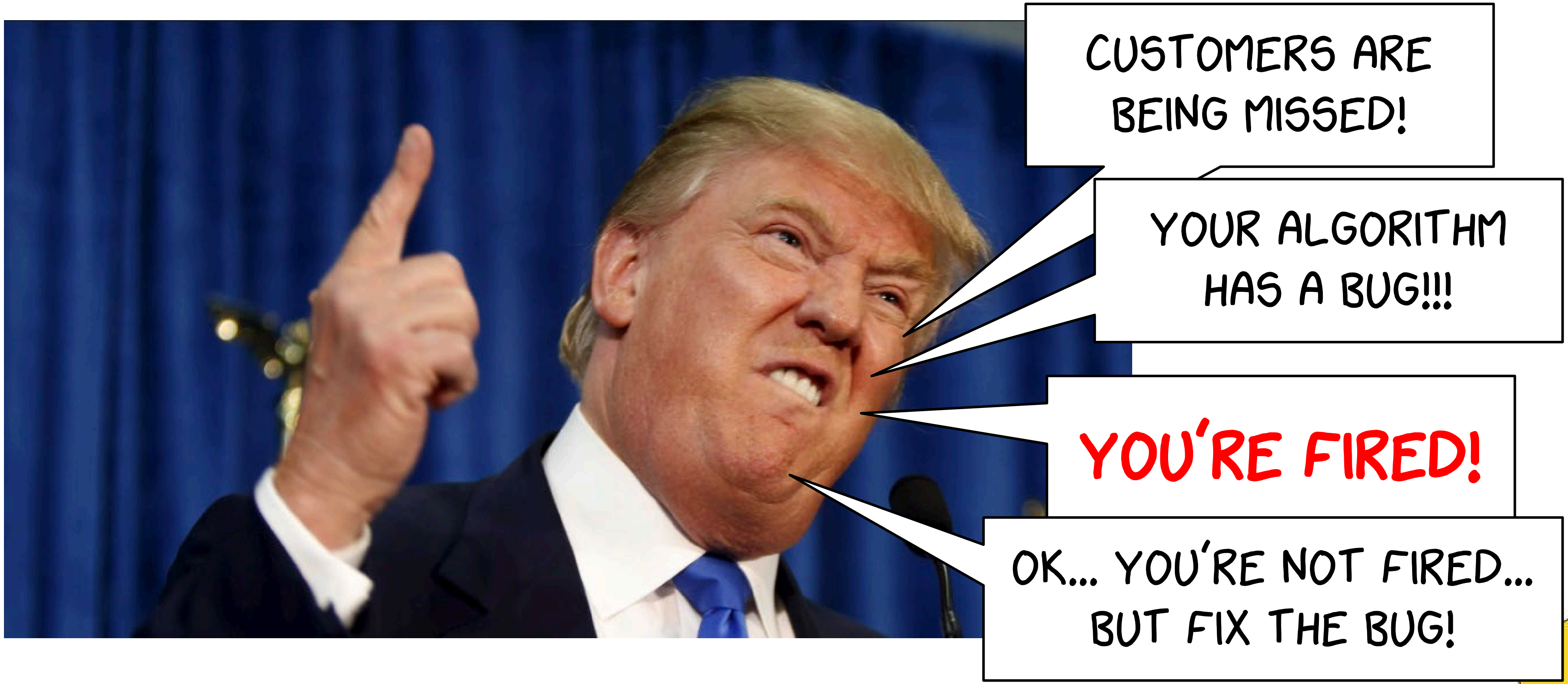
What is the worst-case time complexity for algorithm on the left if you use binary search for findIndex?

A. $O(n \log n)$
B. $\boldsymbol{O(n)}$
C. $O(1)$
D.

PROBLEM SOLVED! ☺

BUT…

BOSS CALLS YOU INTO HIS OFFICE ☹



Poll Everywhere
https://bit.ly/2LvG9bq

WHAT'S WRONG WITH THE ALGORITHM?

Data is in the `customers` array

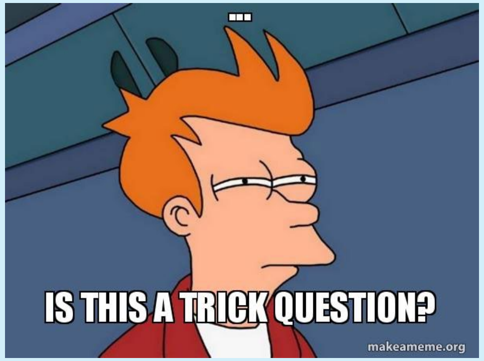
Array sorted by customer spend

```
start = findIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
    println(customers[i])
```

Is something wrong with the algorithm?

A. Yes!

B. Nooooo! Boss is dumb

C. 

WHAT'S WRONG WITH THE ALGORITHM?

Poll Everywhere

https://bit.ly/2LvG9bq

Data is in the `customers` array

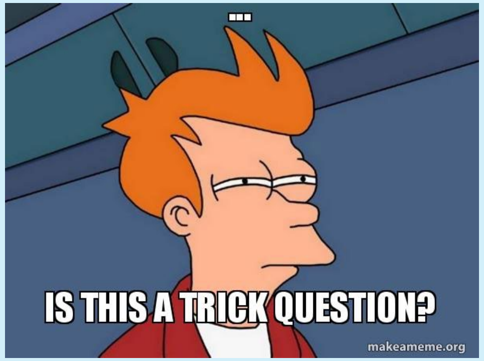
Array sorted by customer spend

```
start = findIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
    println(customers[i])
```

Is something wrong with the algorithm?

A. **Yes!**

B. Noooooo! Boss is dumb

C. 

BINARY SEARCH: FINDING 11

2	4	4	11	11	11	11	11	11	17	23	28

```
function binarySearch (A, key, n)
      low = 0
      high = n - 1
      while low <= high
            mid = (low + high )/2
            if key < A[mid] then
                  high = mid - 1
            else if key > A[mid]
                  low = mid + 1
            else
                  return mid
      return not_found
```

Question: what happens now?

Uh oh! A Bug!

HOW TO FIX THIS?

2	4	4	11	11	11	11	11	11	17	23	28

```
start = findIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

HOW TO FIX THIS?

2	4	4	11	11	11	11	11	11	17	23	28

```
start = findFirstIndex(customers, $a)
end = findIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

HOW TO FIX THIS?

2	4	4	11	11	11	11	11	11	17	23	28

```
start = findFirstIndex(customers, $a)
end = findLastIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

HOW TO FIX THIS?

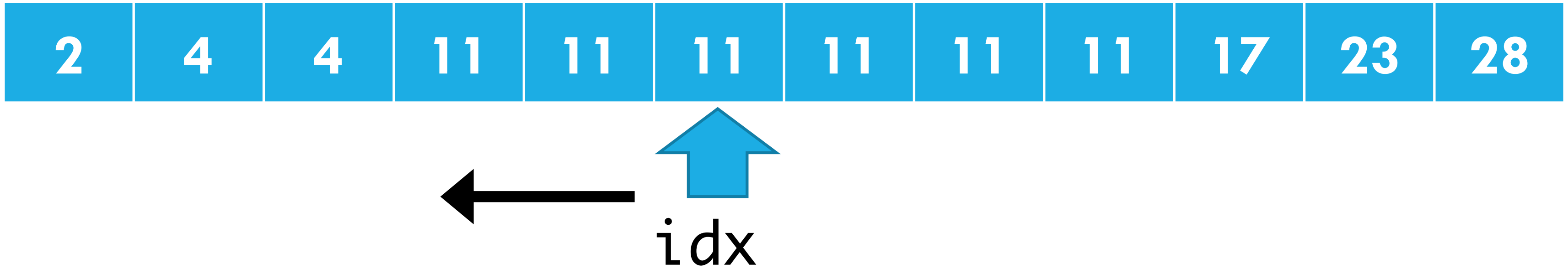
2	4	4	11	11	11	11	11	11	17	23	28

```
start = findFirstIndex(customers, $a)
end = findLastIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

HOW CAN WE MODIFY BINARY SEARCH TO FIND THE FIRST INDEX?

HOW CAN WE MODIFY BINARY SEARCH TO FIND THE FIRST INDEX?

```
idx = binarySearch(customers, $a)
start = searchLeft(customers, idx, $a)
```



What is the time complexity of this algorithm?

$$O(n)$$

HOW CAN WE MODIFY BINARY SEARCH TO FIND THE FIRST INDEX?

2	4	4	11	11	11	11	11	11	17	23	28

`mid = 5`

We know that idx **may** be the first occurence of 11, but anything to the right will not be.

- Still can get rid of ½ the array!
- Store a candidate `result = 5`

HOW CAN WE MODIFY BINARY SEARCH TO FIND THE FIRST INDEX?

2	4	4	11	11	11	11	11	11	17	23	28

2	4	4	11	11	11	11	11	11	17	23	28

`mid = 2`

HOW CAN WE MODIFY BINARY SEARCH TO FIND THE FIRST INDEX?

2	4	4	11	11	11	11	11	11	17	23	28

2	4	4	11	11	11	11	11	11	17	23	28

2	4	4	11	11	11	11	11	11	17	23	28

`mid = 3`

- Update candidate `result = 3`

HOW CAN WE MODIFY BINARY SEARCH TO FIND THE FIRST INDEX?

2	4	4	11	11	11	11	11	11	17	23	28

2	4	4	11	11	11	11	11	11	17	23	28

2	4	4	11	11	11	11	11	11	17	23	28

2	4	4	11	11	11	11	11	11	17	23	28

- Return result = 3

Poll Everywhere

https://bit.ly/2LvG9bq

BINARY SEARCH FIRST

```
function binarySearchFirst (A, key, n)
       low = 0
       high = n – 1
       result = not_found
       while low <= high
              mid = (low + high)/2
              if key < A[mid] then
                     high = mid - 1
              else if key == A[mid] then
                     result = mid
                     high = mid - 1
              else
                     low = mid + 1
       return result
```

What is the worst case computational complexity of binarySearchFirst()?

A. $O(\log n)$
B. $O(n)$
C. $O(n^2)$
D. $O(1)$

BINARY SEARCH FIRST

Poll Everywhere
https://bit.ly/2LvG9bq

```
function binarySearchFirst (A, key, n)
        low = 0
        high = n - 1
        result = not_found
        while low <= high
                mid = (low + high)/2
                if key < A[mid] then
                        high = mid - 1
                else if key == A[mid] then
                        result = mid
                        high = mid - 1
                else
                        low = mid + 1
        return result
```

What is the worst case computational complexity of binarySearchFirst()?

A. $\boldsymbol{O(logn)}$
B. $O(n)$
C. $O(n^2)$
D. $O(1)$

BINARY SEARCH FIRST CORRECTNESS

```
function binarySearchFirst (A, key, n)
        low = 0
        high = n – 1
        result = not_found
        while low <= high
                mid = (low + high)/2
                if key < A[mid] then
                        high = mid - 1
                else if key == A[mid] then
                        result = mid
                        high = mid - 1
                else
                        low = mid + 1
        return result
```

Preconditions:

- A is of size n
- A is sorted

Postcondition:

- A[mid] = key
- A[mid] is the first occurrence of key

YOU CAN FIND THE LAST INDEX IN A SIMILAR WAY

2	4	4	11	11	11	11	11	11	17	23	28

```
start = findFirstIndex(customers, $a)
end = findLastIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

ADDENDUM

The code has another bug! Can you spot it?

- What if start or end is not_found? Can we modify the pre and postconditions?
- Or can we modify the code below to do proper checking?
- Or can we modify binary search to return the next largest/smallest item?

```
start = findFirstIndex(customers, $a)
end = findLastIndex(customers, $b)
for (int i=start; i<=end; i++)
     println(customers[i])
```

WRITING CORRECT PROGRAMS

Programming Pearls by Jon Bentley (published in 1986).

Binary search used in Sun JDK.

But there's a bug.

BUG WITH BINARY SEARCH

```
function binarySearchFirst (A, key, n)
        low = 0
        high = n - 1
        result = not_found
        while low <= high
            mid = (low + high)/2
            if key < A[mid] then
                high = mid - 1
            else if key == A[mid] then
                result = mid
                high = mid - 1
            else
                low = mid + 1
        return result
```

This can overflow!

BUG WITH BINARY SEARCH

```
function binarySearchFirst (A, key, n)
        low = 0
        high = n - 1
        result = not_found
        while low <= high
                mid = low + ((high-low)/2)
                if key < A[mid] then
                        high = mid - 1
                else if key == A[mid] then
                        result = mid
                        high = mid - 1
                else
                        low = mid + 1
        return result
```

PROBLEM SOLVED! ☺

INTERIM SUMMARY

Divide and Conquer:

- Split into sub-problems
- Solve sub-problems
- Combine sub-solutions

Program Correctness

- Be careful with your
  Pre & Post Conditions
- Invariances are useful!

QUESTIONS?

PROBLEM: NARUTO'S SORT IS TOO SLOW!

We now have more than a million customers!

Naruto's Insertion sort is now too slow.

We need a better method!

Can we apply divide and conquer?

Yes!

DIVIDE & CONQUER STRATEGY: THE 3 STEPS

1. (*Deviously*) **Split** your problem into subproblems.
2. (*Gleefully*) **Solve** each subproblem (recursively).
3. (*Cleverly*) **Combine** the solutions for each subproblem to produce a solution to the original problem.

MERGESORT: THE ALGORITHM

7	3	9	5	7	1	6	2

MERGESORT: THE ALGORITHM

Split

7	3	9	5	7	1	6	2

MERGESORT: THE ALGORITHM

7	3	9	5	7	1	6	2

Split

Solve (Sort)

3	5	7	9

Solve (Sort)

1	2	6	7

MERGESORT: THE ALGORITHM

Split

7	3	9	5	7	1	6	2

Solve (Sort)

3	5	7	9

Solve (Sort)

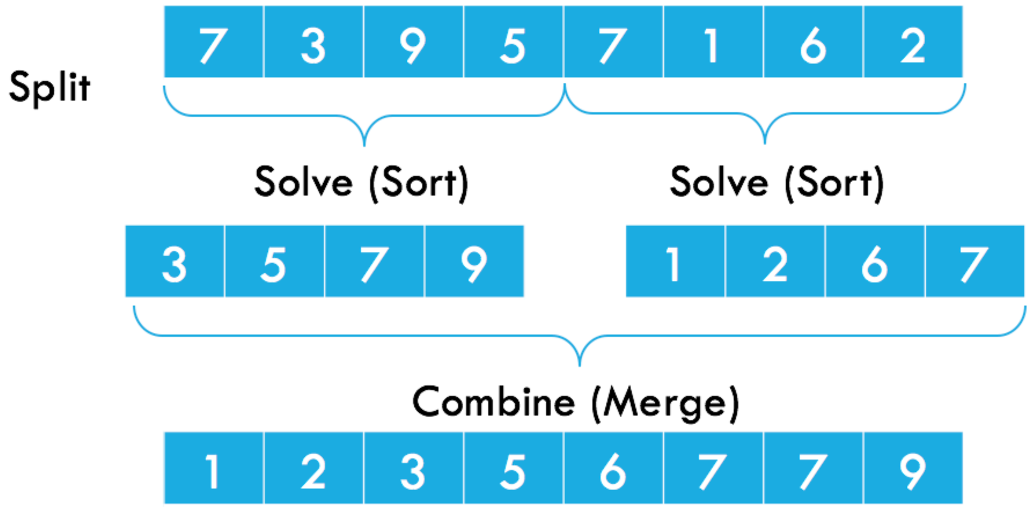
1	2	6	7

Combine (Merge)

1	2	3	5	6	7	7	9

MERGESORT: THE ALGORITHM

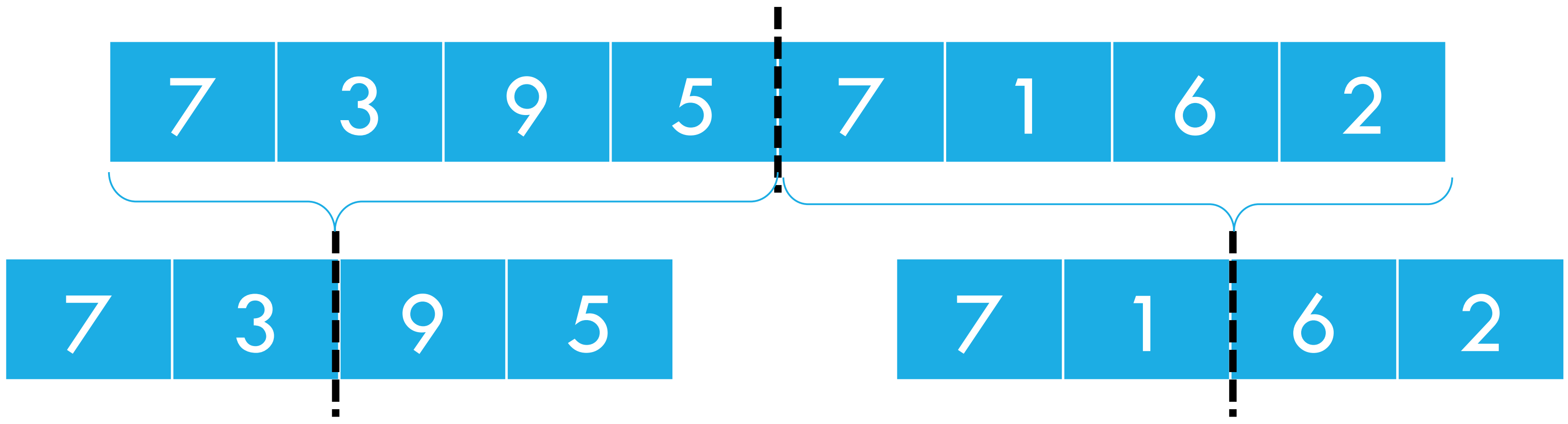
```
function mergeSort(A, low, high)
    if low < high
        mid = (high + low)/2
        mergeSort(A, low, mid)
        mergeSort(A, mid+1, high)
        merge(A, low, mid, high)
```



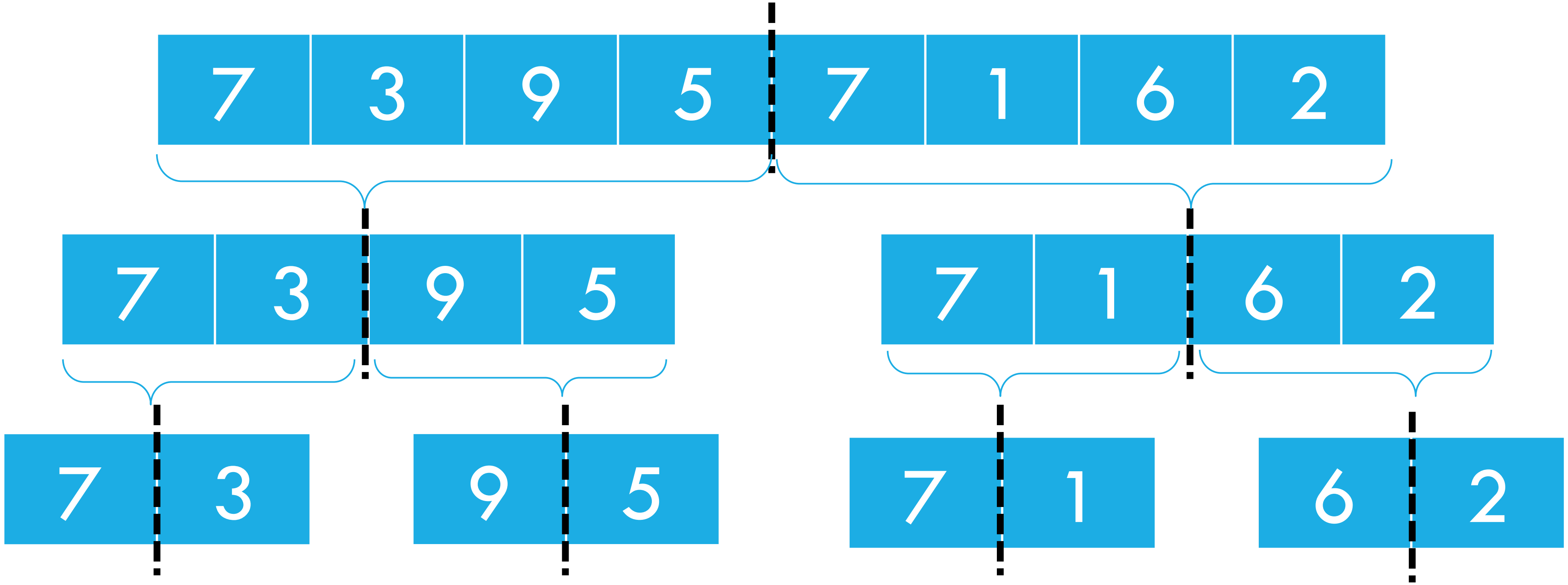
MERGESORT: RECURSE “DOWNWARDS”

7	3	9	5	7	1	6	2

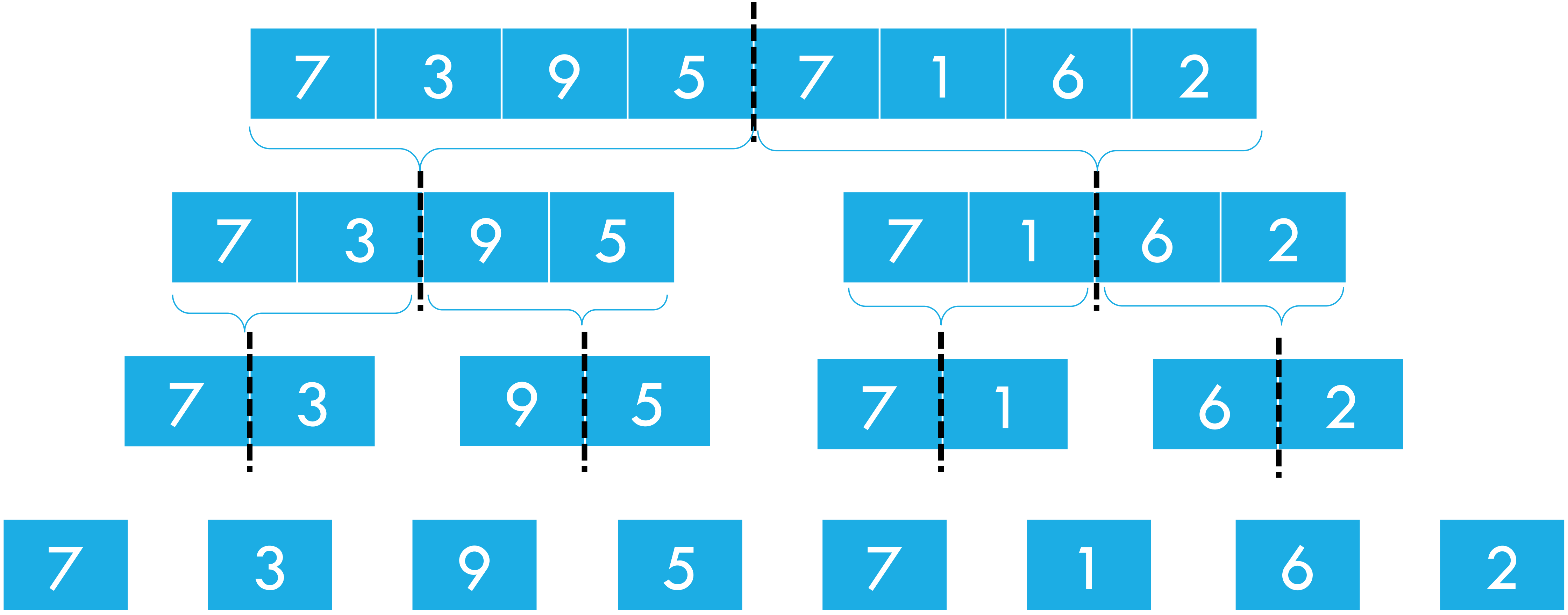
MERGESORT: RECURSE "DOWNWARDS"



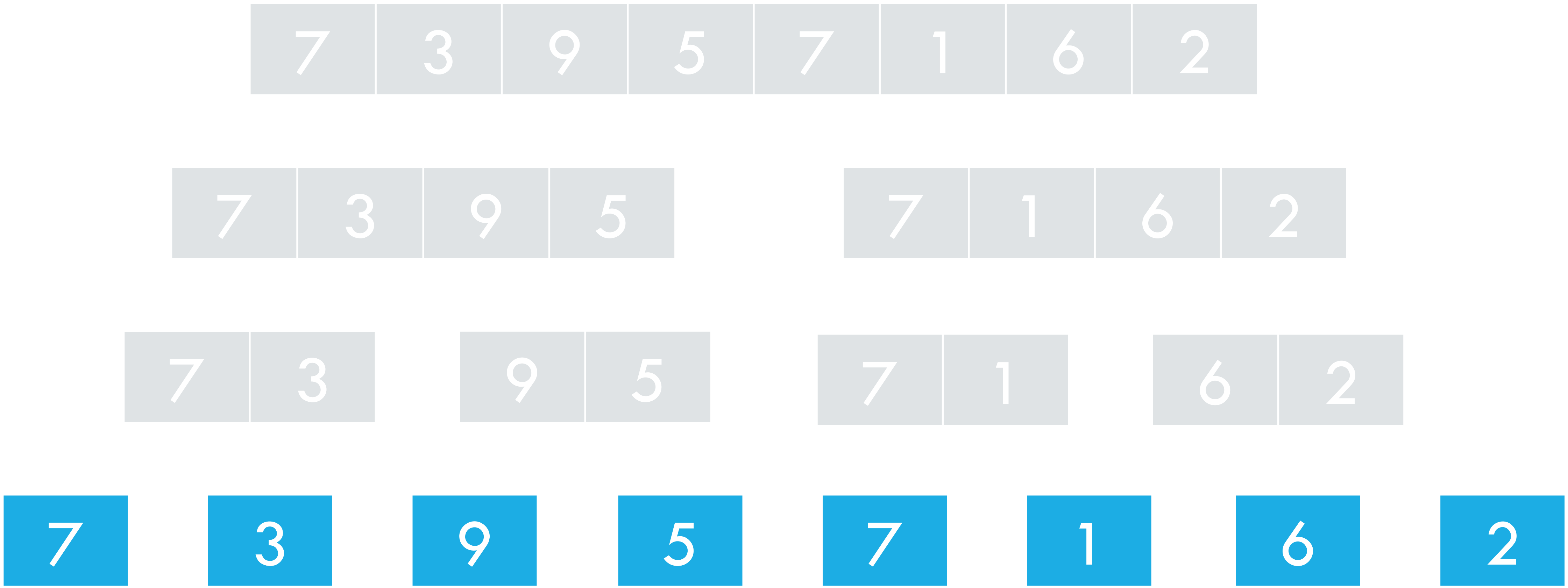
MERGESORT: RECURSE “DOWNWARDS”



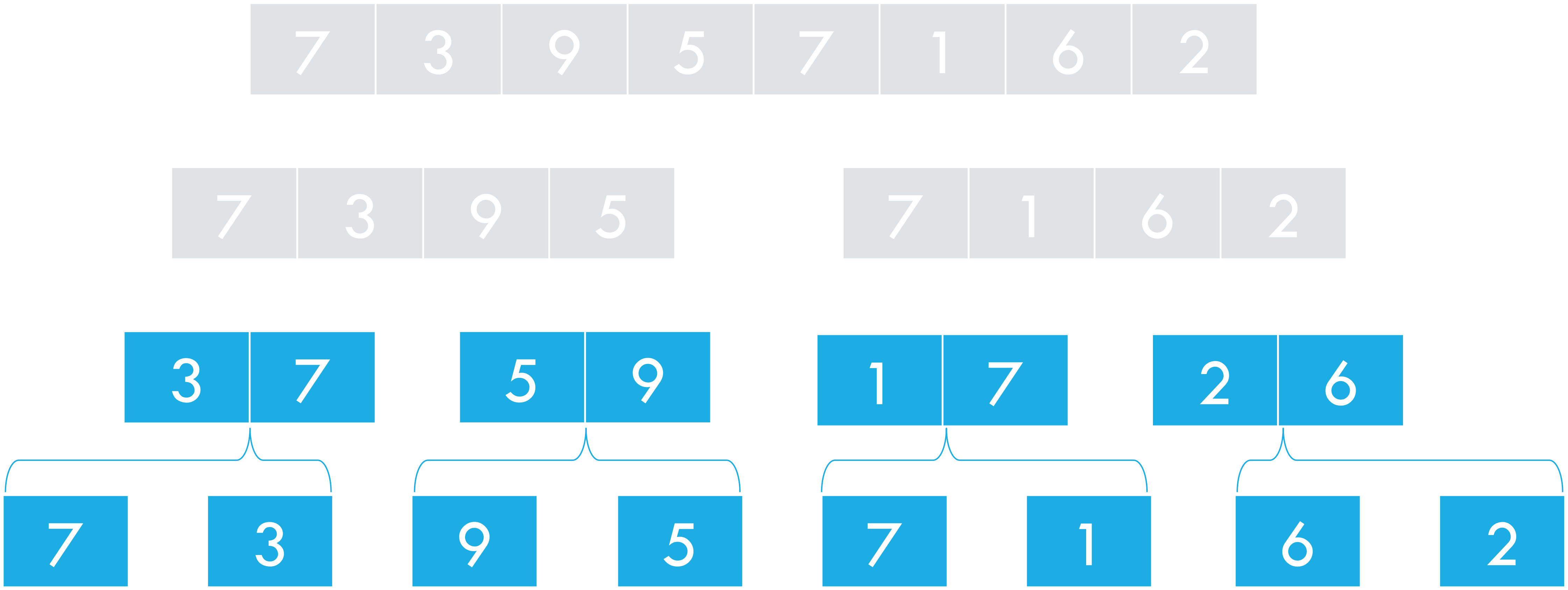
MERGESORT: RECURSE “DOWNWARDS”



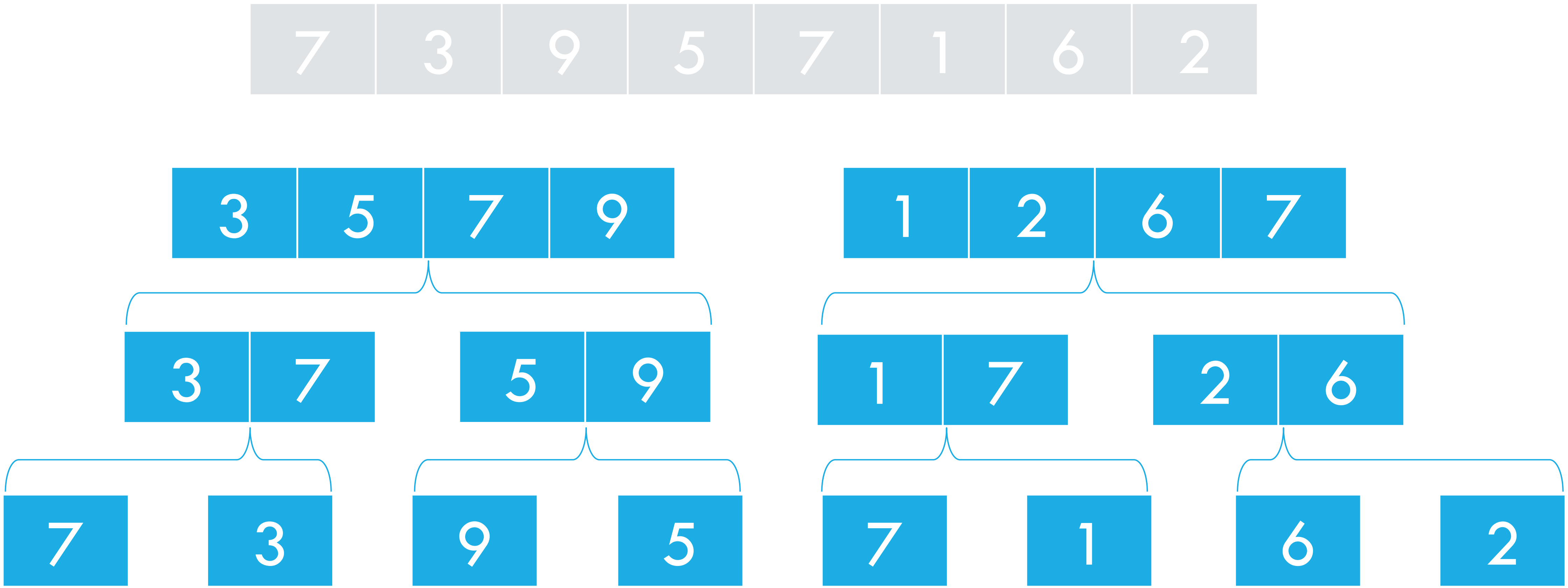
MERGESORT: MERGING “UPWARDS”



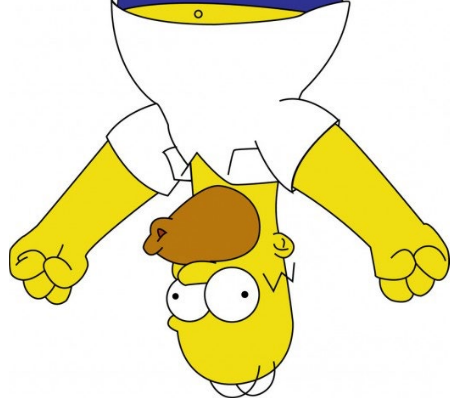
MERGESORT: MERGING "UPWARDS"

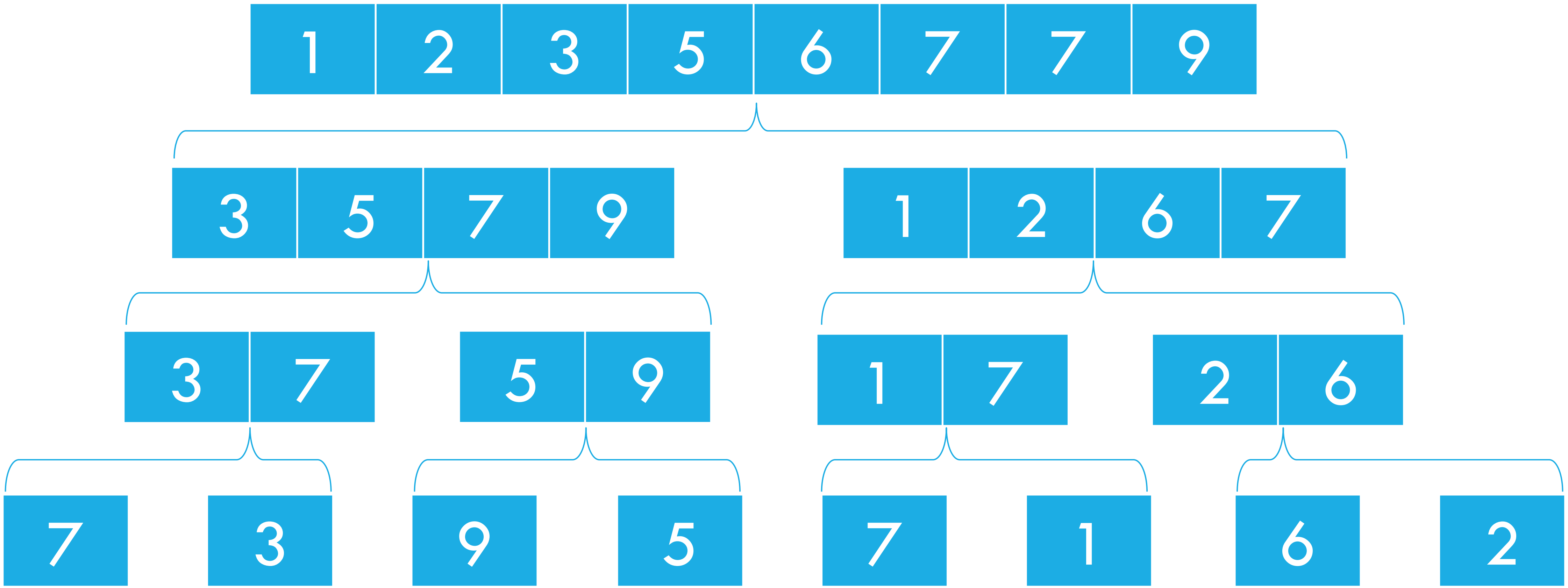


MERGESORT: MERGING “UPWARDS”



MERGESORT: MERGING “UPWARDS”





THE MERGE STEP

3	5	7	9

1	2	6	7

MERGE PSEUDOCODE

```
function merge(int low, int mid, int high)
    for i = low to high
        buffer[i] = A[i]
    i = low
    j = mid + 1
    k = low
    while i <= mid and j <= high
        if buffer[i] <= buffer[j]
            A[k] = buffer[i]
            i++
        else
            A[k] = buffer[j]
            j++
        k++

    while i <= middle
        A[k] = buffer[i]
        k++
        i++
```

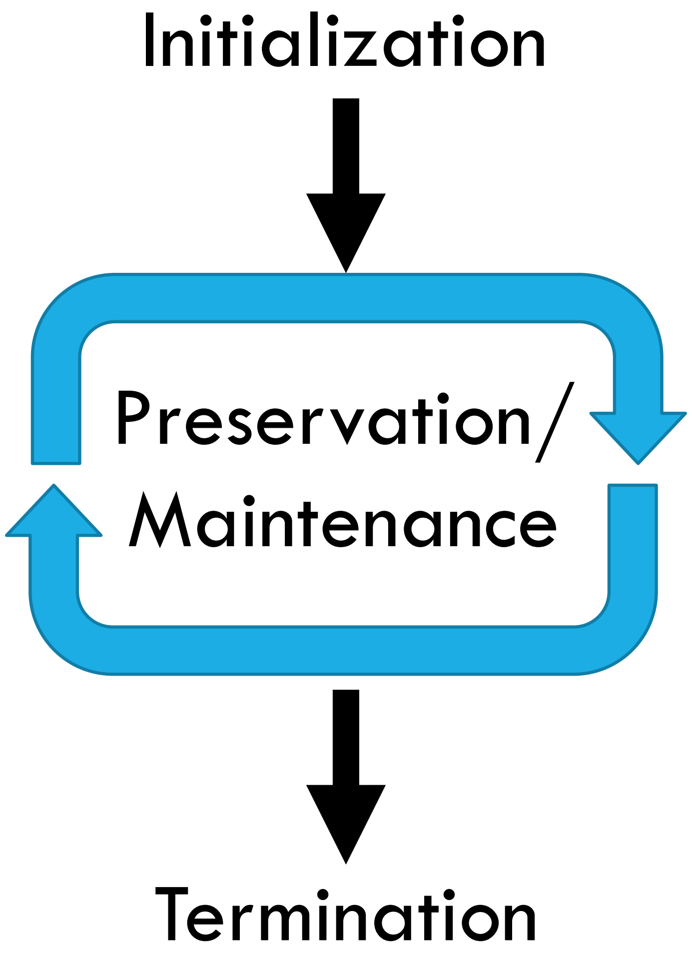
Initialization.

Iterate through helper

At each iteration, compare the elements in the two segments and copy over the smaller element

Copy over the remaining items from the left segment (if needed)

MERGE: CORRECT?




```
function merge(int low, int mid, int high)
    for i = low to high
        buffer[i] = A[i]
    i = low
    j = mid + 1
    k = low
    while i <= mid and j <= high
        if buffer[i] <= buffer[j]
            A[k] = buffer[i]
            i++
        else
            A[k] = buffer[j]
            j++
        k++

    while i <= middle
        A[k] = buffer[i]
        k++
        i++
```

MERGE: WHAT IS THE LOOP INVARIANT?

Behind here lies the answer

```
function merge(int low, int mid, int high)
    for i = low to high
        buffer[i] = A[i]
    i = low
    j = mid + 1
    k = low
    while i <= mid and j <= high
        if buffer[i] <= buffer[j]
            A[k] = buffer[i]
            i++
        else
            A[k] = buffer[j]
            j++
        k++

    while i <= middle
        A[k] = buffer[i]
        k++
        i++
```

MERGE: WHAT IS THE LOOP INVARIANT?

let L = buffer[low, ..., mid]
R = buffer[mid+1, ..., high]

A[low, k-1] contains the k-1 smallest elements of L and R in sorted order.

L[i] and R[j] contain the smallest items in their respective arrays not yet copied to A.

```
function merge(int low, int mid, int high)
    for i = low to high
        buffer[i] = A[i]
    i = low
    j = mid + 1
    k = low
    while i <= mid and j <= high
        if buffer[i] <= buffer[j]
            A[k] = buffer[i]
            i++
        else
            A[k] = buffer[j]
            j++
        k++

    while i <= middle
        A[k] = buffer[i]
        k++
        i++
```

MERGE: WHAT IS THE LOOP INVARIANT?

let L = buffer[low, ..., mid]
R = buffer[mid+1, ..., high]

A[low, k-1] contains the k-1 smallest elements of L and R in sorted order.

L[i] and R[j] contain the smallest items in their respective arrays not yet copied to A.

Verify the loop invariant holds at each iteration!

```
function merge(int low, int mid, int high)
    for i = low to high
        buffer[i] = A[i]
    i = low
    j = mid + 1
    k = low
    while i <= mid and j <= high
        if buffer[i] <= buffer[j]
            A[k] = buffer[i]
            i++
        else
            A[k] = buffer[j]
            j++
        k++

    while i <= middle
        A[k] = buffer[i]
        k++
        i++
```

Poll Everywhere

https://bit.ly/2LvG9bq

HOW LONG DOES MERGE TAKE?

What is the worst-case time complexity for merging?

A. $O(n \log n)$

B. $O(n)$

C. $O(n^2)$

D. $O(1)$

E. I wish you would stop asking these complexity questions!

Poll Everywhere

https://bit.ly/2LvG9bq

HOW LONG DOES MERGE TAKE?

What is the worst-case time complexity for merging?

A. $O(n \log n)$
B. $\boldsymbol{O(n)}$
C. $O(n^2)$
D. $O(1)$
E. I wish you would stop asking these complexity questions!

Poll Everywhere

https://bit.ly/2LvG9bq

MERGESORT: WHAT IS THE RUNNING TIME?

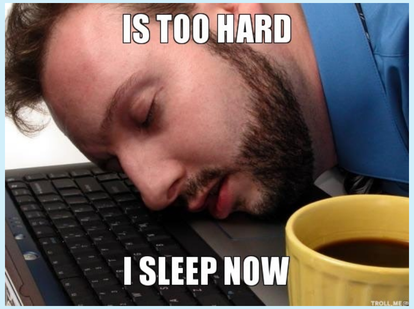
```
function MergeSort(A, low, high)
    if low < high
        mid = (high + low)/2
        MergeSort(A, low, mid)
        MergeSort(A, mid+1, high)
        Merge(A, low, mid, high)
```

What is the worst-case time complexity for mergesort?

A. $O(n \log n)$

B. $O(n)$

C. $O(n + \log n)$

D. 

Poll Everywhere

https://bit.ly/2LvG9bq

MERGESORT: WHAT IS THE RUNNING TIME?

```
function MergeSort(A, low, high)
    if low < high
        mid = (high + low)/2
        MergeSort(A, low, mid)
        MergeSort(A, mid+1, high)
        Merge(A, low, mid, high)
```

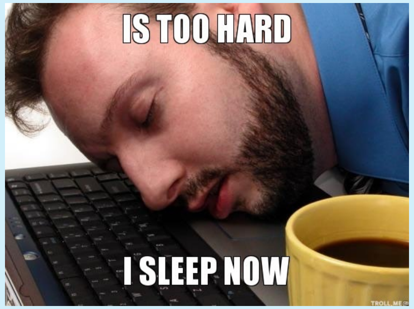
Why?

What is the worst-case time complexity for mergesort?

A. $\boldsymbol{O(n\ log n)}$

B. $O(n)$

C. $O(n + \log n)$

D. 

MERGESORT RUNNING TIME

```
function MergeSort(A, low, high)
  if low < high
    mid = (high + low)/2
    MergeSort(A, low, mid)
    MergeSort(A, mid+1, high)
    Merge(A, low, mid, high)
```

MERGESORT RUNNING TIME

```
function MergeSort(A, low, high)
  if low < high                          c1
    mid = (high + low)/2                 c2
    MergeSort(A, low, mid)               T(n/2)
    MergeSort(A, mid+1, high)            T(n/2)
    Merge(A, low, mid, high)             c3n
```

MERGESORT RUNNING TIME

```
function MergeSort(A, low, high)
   if low < high                          c1
      mid = (high + low)/2                c2
      MergeSort(A, low, mid)              T(n/2)
      MergeSort(A, mid+1, high)           T(n/2)
      Merge(A, low, mid, high)            c3n
```

$$\boxed{T(n) = 2T(n/2) + cn}$$

SUBSTITUTION METHOD

Make a Guess

Substitute into the Recurrence Relation

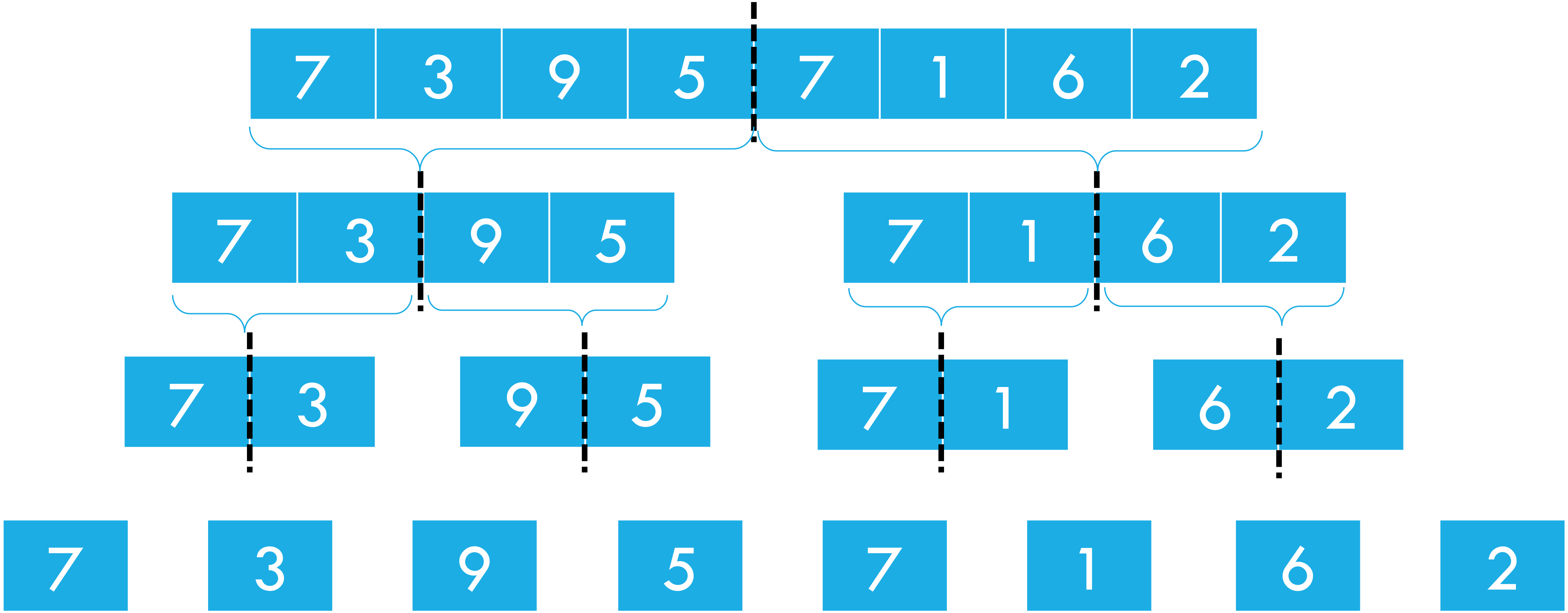
Prove using mathematical induction

Prove the base cases

$$T(n) = 2T(n/2) + cn$$

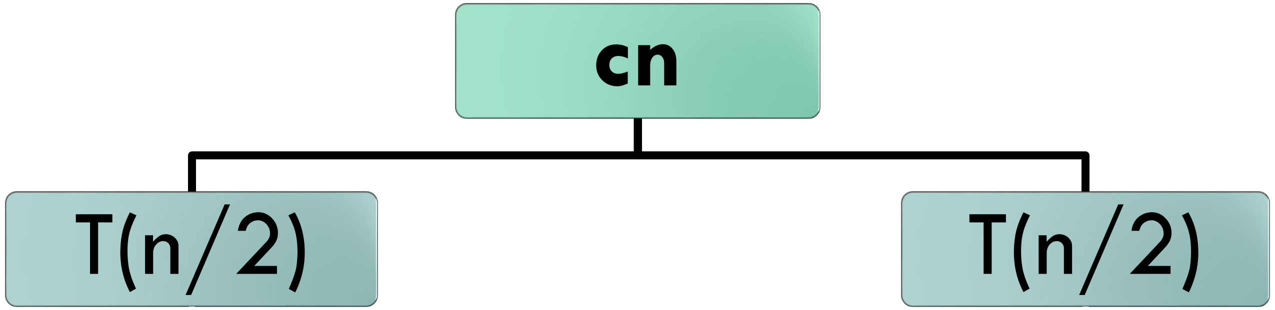
(you can fill in the missing steps here.
Hint: lg(a/b) = lg(a) – lg(b))

MERGESORT: RECURSE “DOWNWARDS”



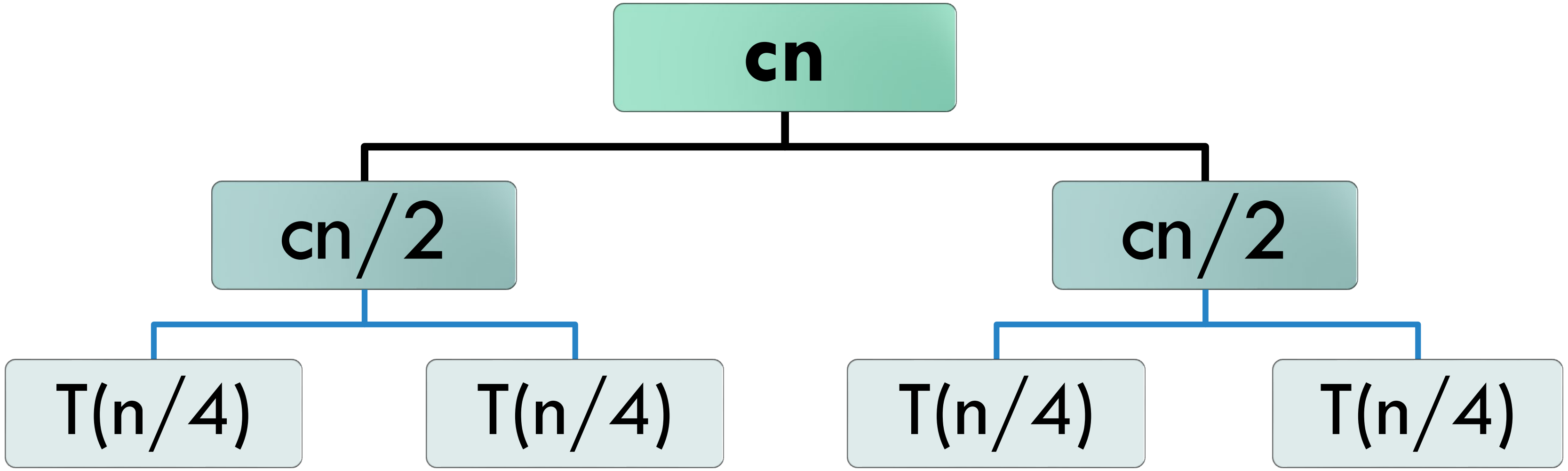
MERGESORT: RECURSE “DOWNWARDS”

$$T(n) = 2T(n/2) + cn$$



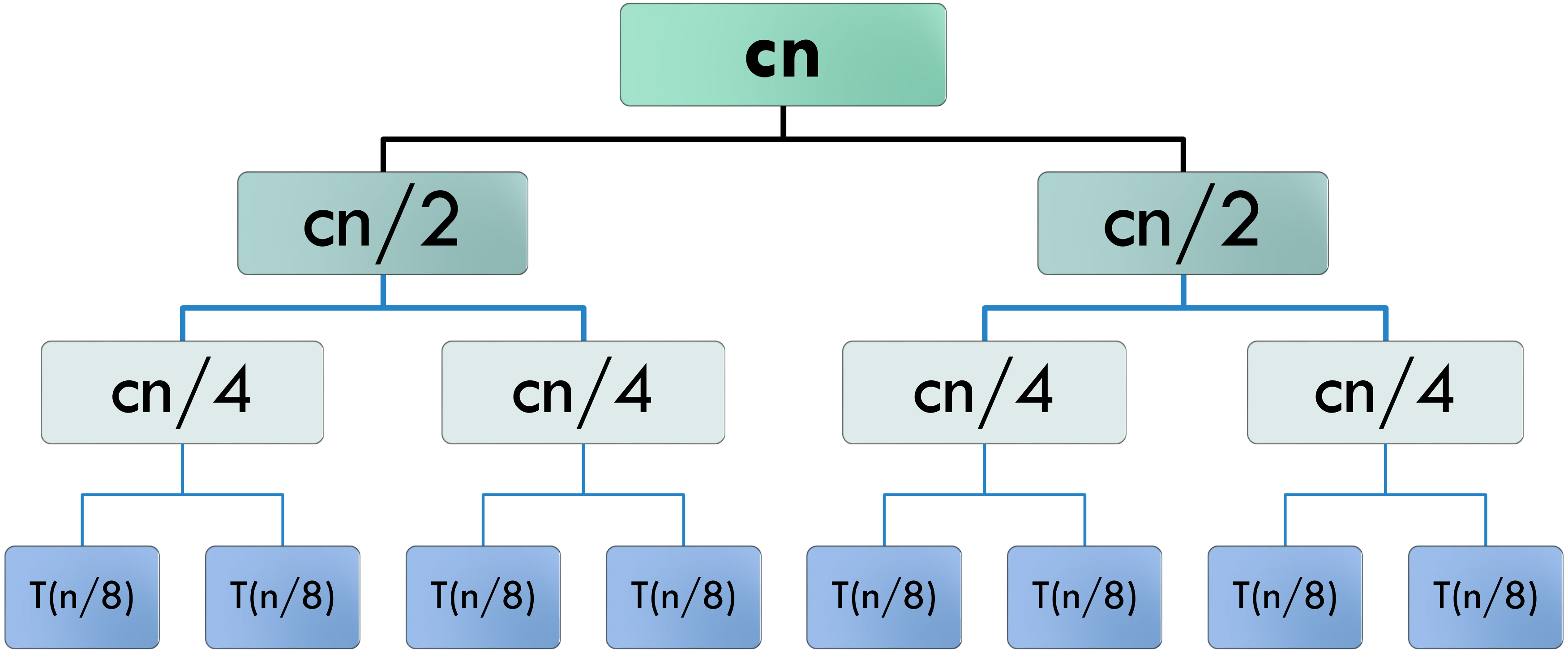
MERGESORT: RECURSE "DOWNWARDS"

$$T(n) = 2T(n/2) + cn$$



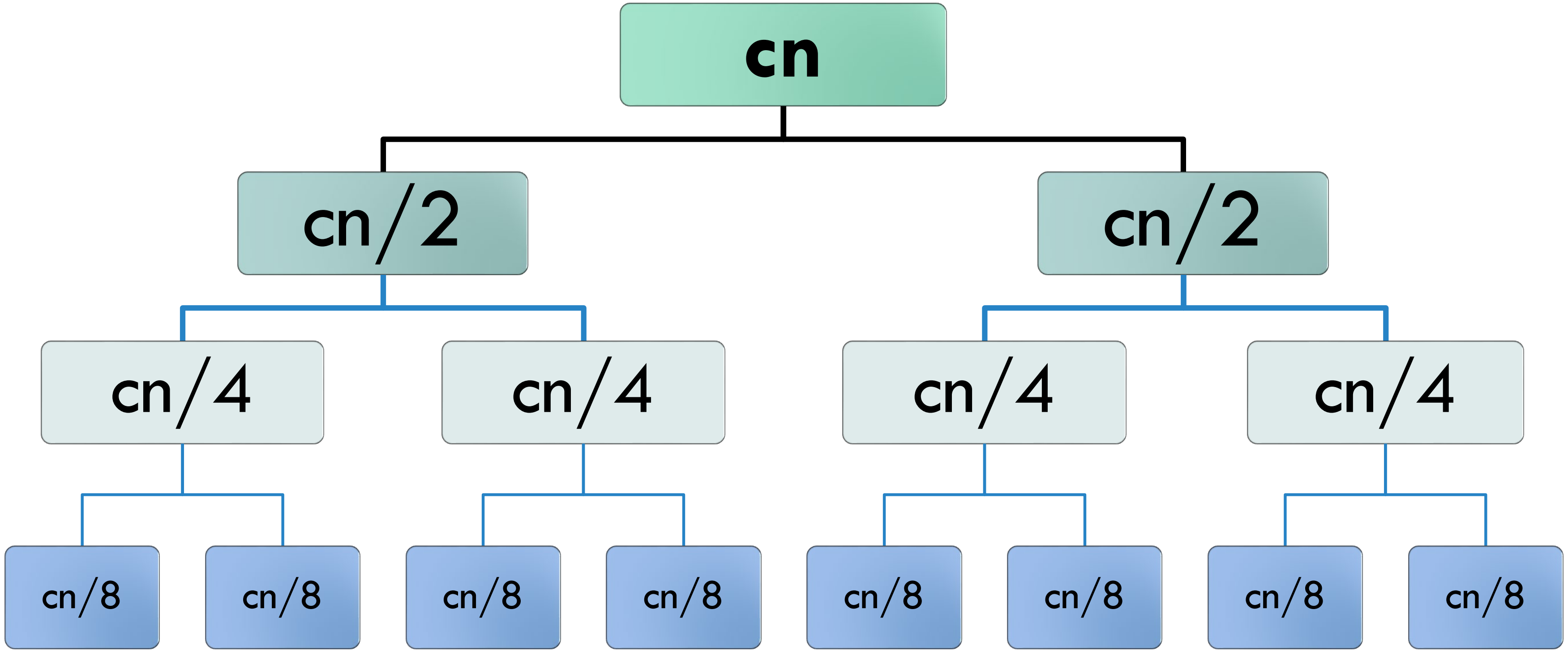
MERGESORT: RECURSE "DOWNWARDS"

$$T(n) = 2T(n/2) + cn$$



MERGESORT: RECURSE "DOWNWARDS"

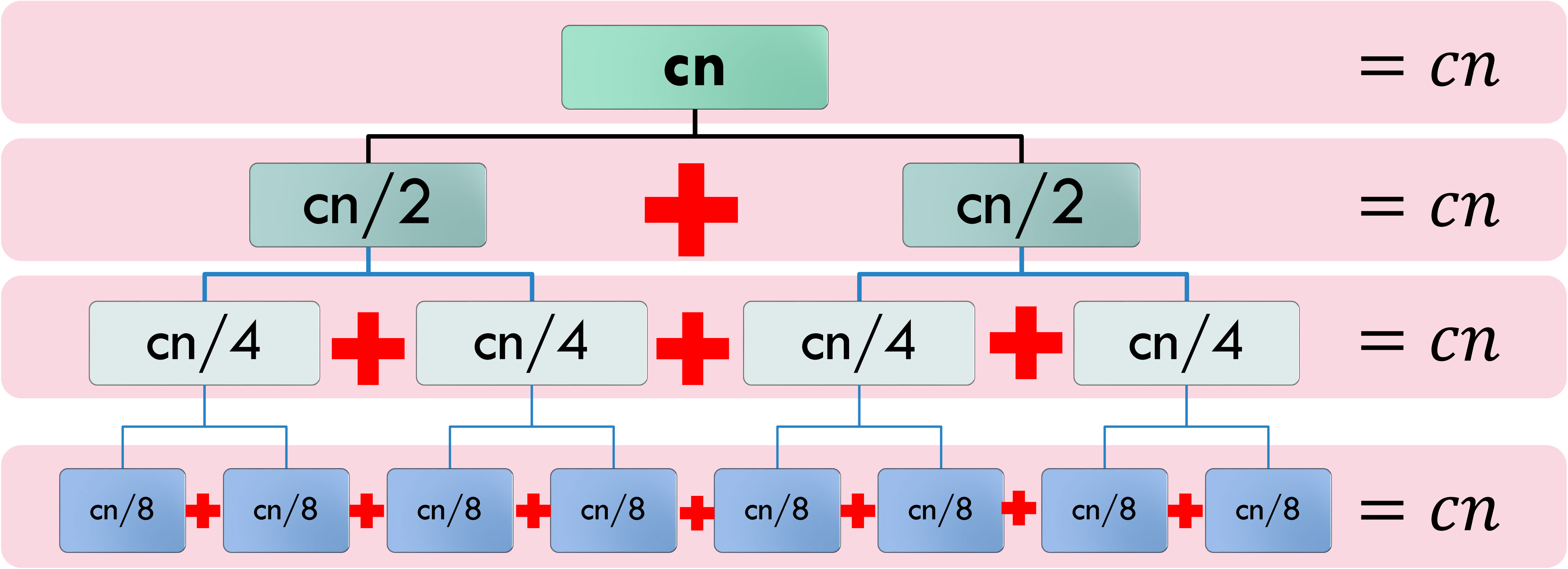
$$T(n) = 2T(n/2) + cn$$



MERGESORT: RECURSE "DOWNWARDS"

$$T(n) = 2T(n/2) + cn$$

how deep does it go?



MERGESORT: RECURSE "DOWNWARDS"

$$T(n) = 2T(n/2) + cn$$

Level	Number
0	1
1	2
2	4
3	8
4	16
…	…
h	*n*

$$n = 2^h$$

does this look familiar?

Poll Everywhere

https://bit.ly/2LvG9bq

n

TIME COMPLEXITY OF BINARY SEARCH

n	2	4	4	5	6	7	8	9	11	17	23	28

Iteration 1: n

Iteration 2: $n/2$

Iteration 3: $n/4$

…

Iteration k: $n/2\hat{}k \ = 1$

If $n/2^k \ = 1$, what is k in terms of n ?

A. $k = 2^n$

B. $k = 2/n$

C. $\boldsymbol{k = \log(n)}$

D. Am I in CS or Math?

MERGESORT: RECURSE "DOWNWARDS"

$$T(n) = 2T(n/2) + cn$$

Level	Number
0	1
1	2
2	4
3	8
4	16
…	…
h	*n*

$$n = 2^h$$

MERGESORT: RECURSE "DOWNWARDS"

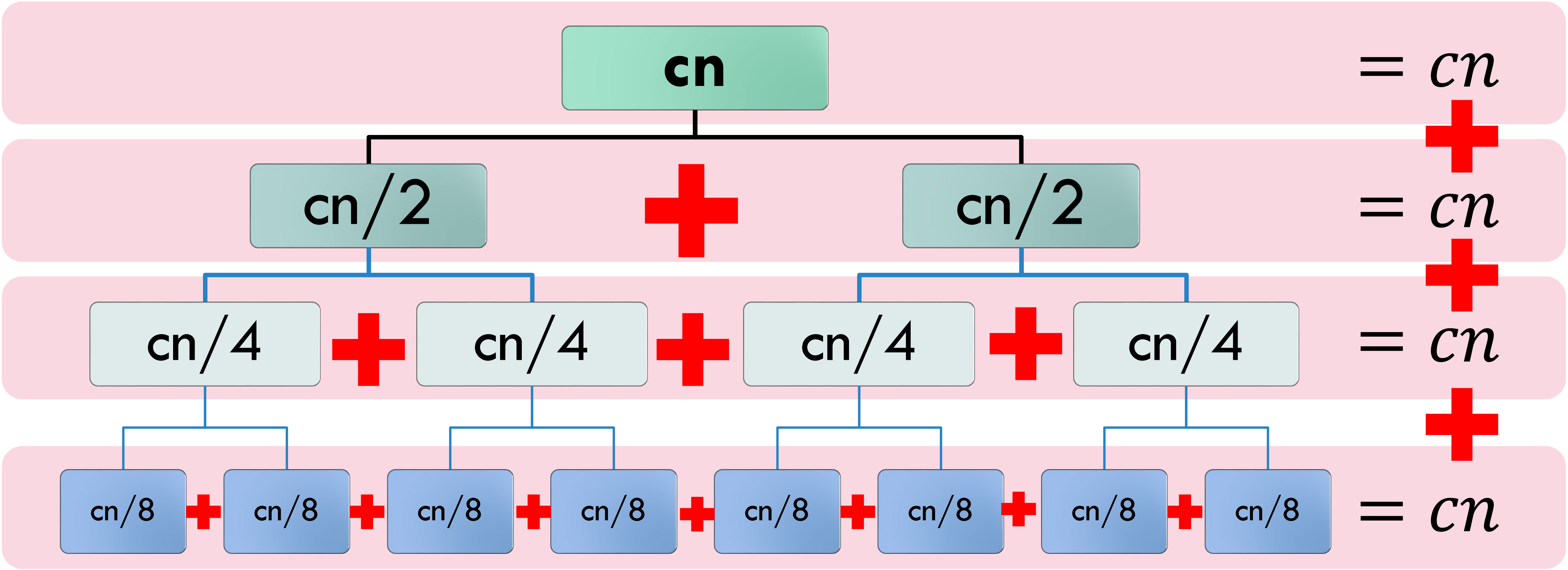
$$T(n) = 2T(n/2) + cn$$

Level	Number
0	1
1	2
2	4
3	8
4	16
…	…
h	*n*

$$n = 2^h$$
$$\log n = h$$
$$h = \log n$$

MERGESORT: RECURSE “DOWNWARDS”

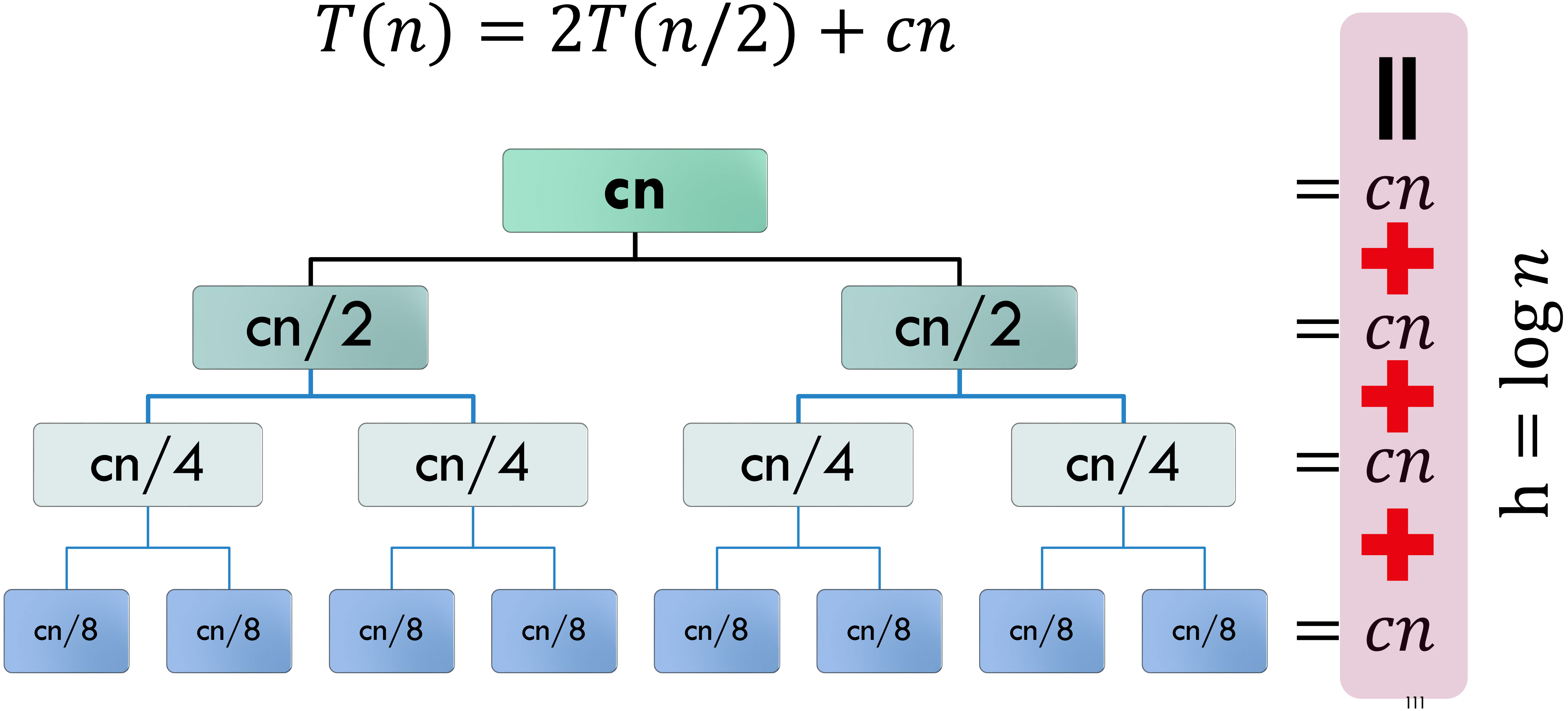
$$T(n) = 2T(n/2) + cn$$



MERGESORT: RECURSE “DOWNWARDS”

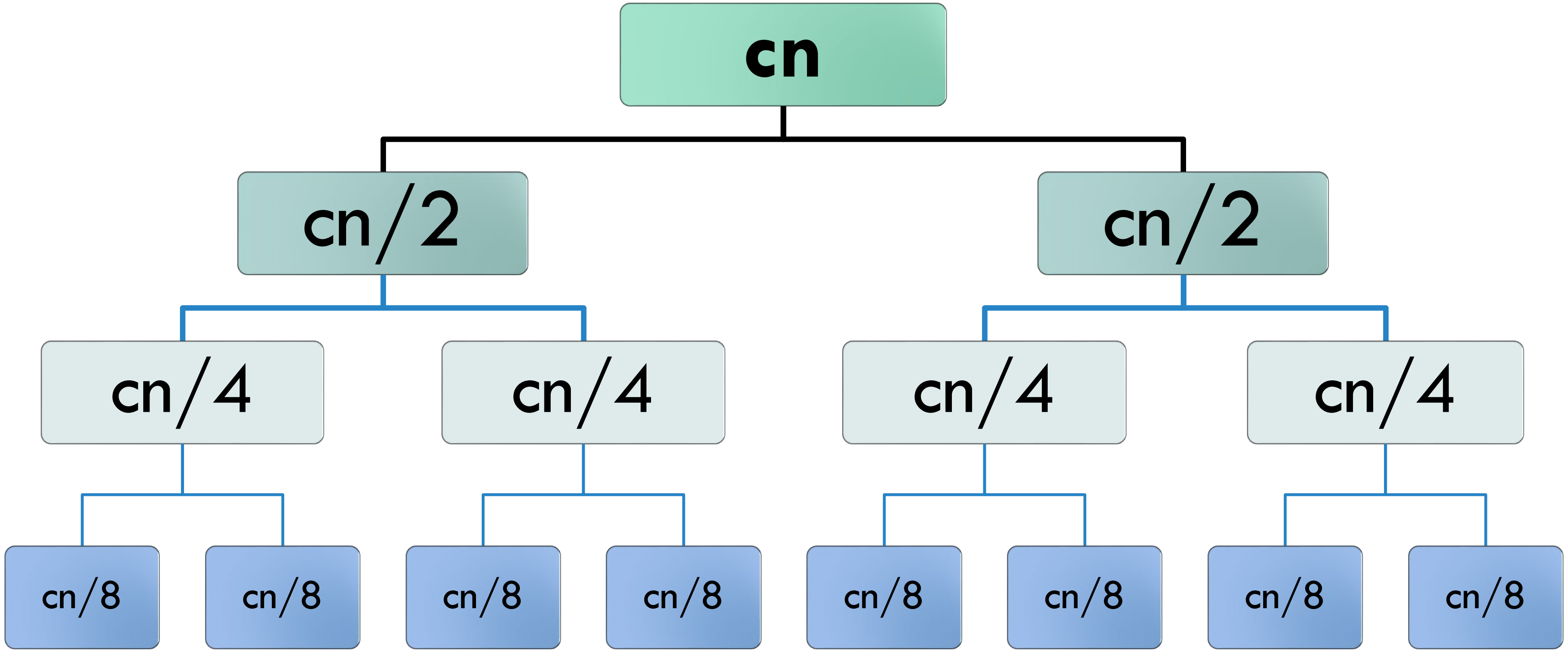
$$cn \log n$$

$$T(n) = 2T(n/2) + cn$$



MERGESORT: RECURSE "DOWNWARDS"

$$T(n) = 2T(n/2) + cn = O(\mathrm{n}\log n)$$



$cn\log n$

$= cn + cn + cn + cn$

$\mathrm{h} = \log n$

MERGESORT: WHAT IS THE RUNNING TIME?

Poll Everywhere

https://bit.ly/2LvG9bq

```
function MergeSort(A, low, high)
    if low < high
        mid = (high + low)/2
        MergeSort(A, low, mid)
        MergeSort(A, mid+1, high)
        Merge(A, low, mid, high)
```

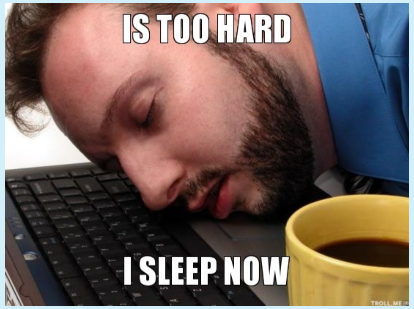
What is the worst-case time complexity for mergesort?

A. $\boldsymbol{O(n\ log n)}$

B. $O(n)$

C. $O(n + \log n)$

D.

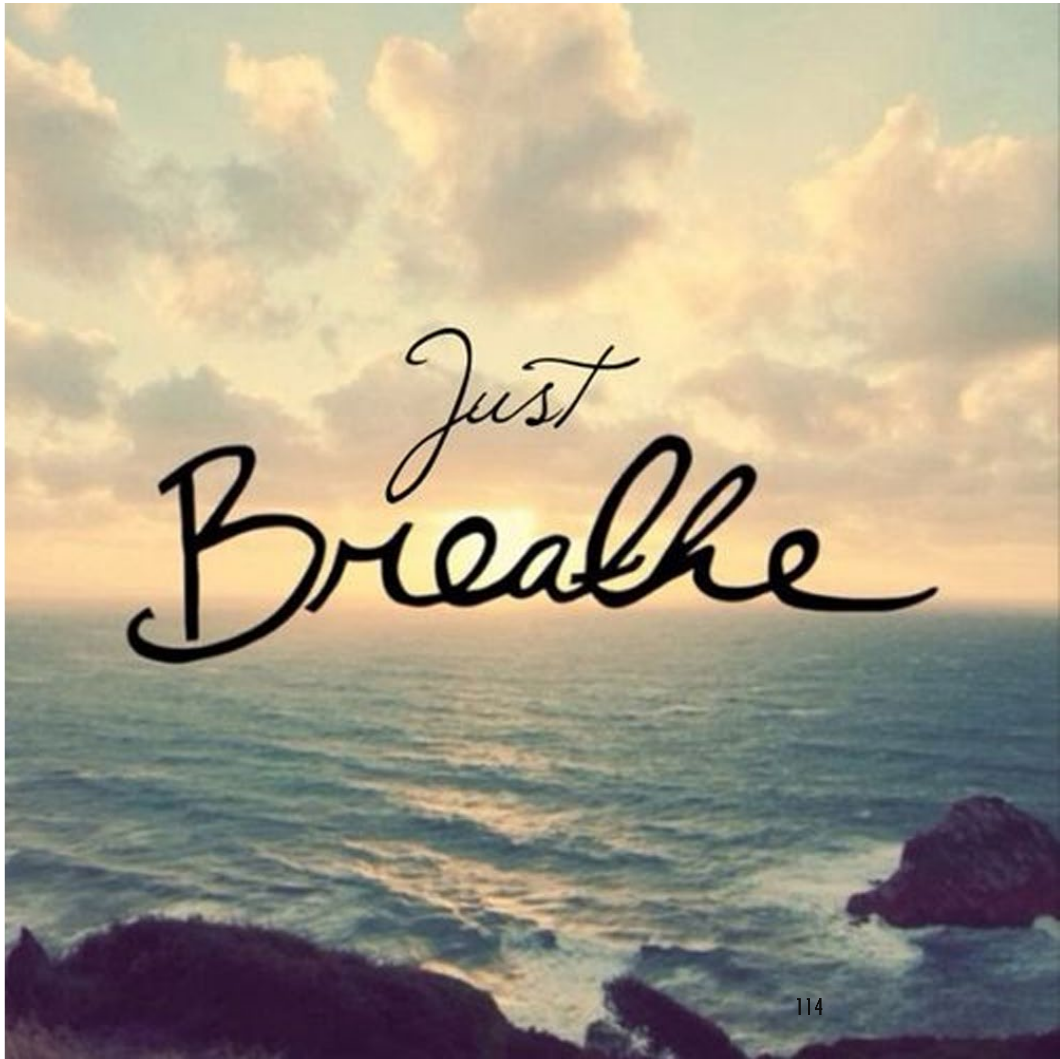


OK... LET'S TAKE A BREATHER

Mergesort is $O(n \log n)$

Derivation required:

- Thinking through the steps.
- Breaking down the recursive calls
- Summing up the costs



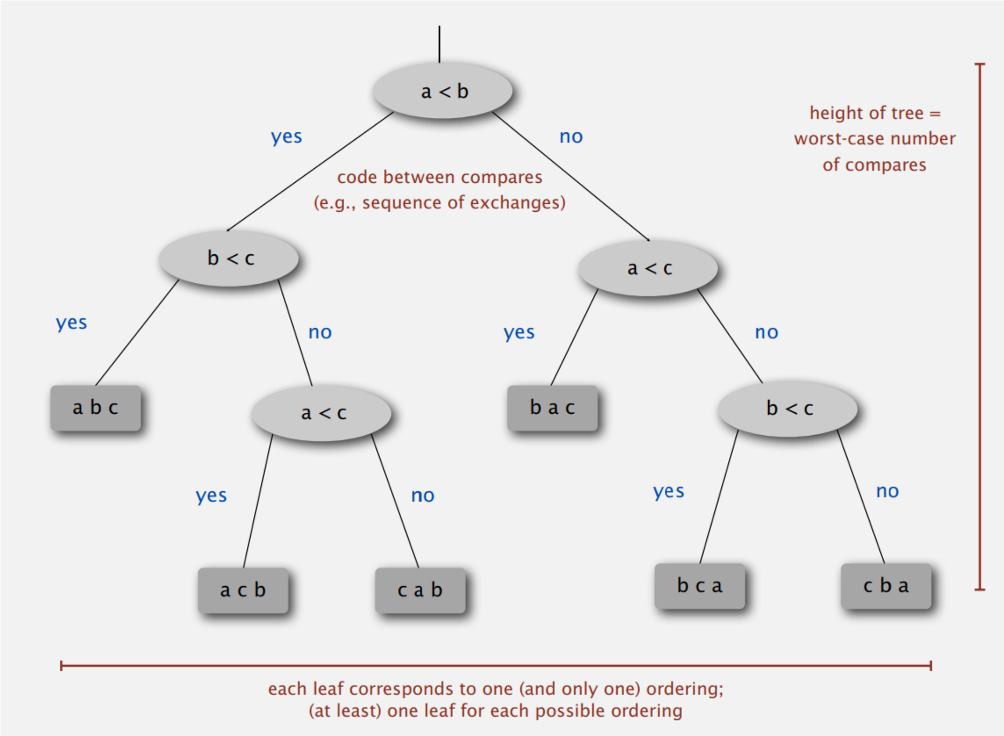
CAN WE DO BETTER THAN $O(n \log n)$ FOR A COMPARISON BASED SORT?

No

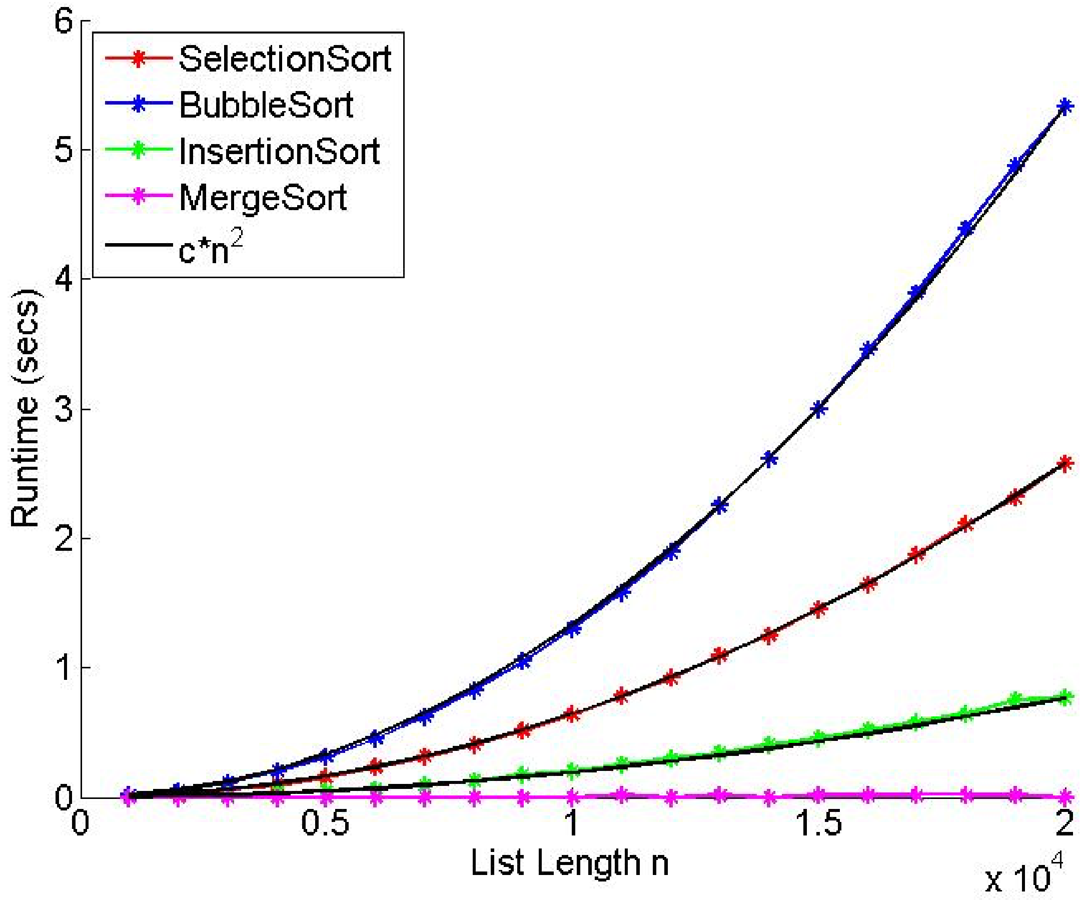
Proof: Chapter 2 of Sedgewick and Wayne or Chapter 8 of CLRS (or wait for CS3230)

Key Idea:

- Any sorting algorithm must output a permutation of the input
- There are $n!$ possible permutations.
- So, $n! \le l \le 2^h$ where l is the number of leaves, and h is the height of the binary tree.
- Take logarithms to show: $h \ge \log n! = \Omega(n \log n)$ using Stirling's approximation



REAL WORLD PERFORMANCE



[csc148 notes, http://www.cs.toronto.edu/~jepson/csc148/2007F/notes/sorting.html]

PROBLEM FIXED: YOUR SORT IS FAST!

We now have more than a million customers!

All of them are happily served!

You and Naruto get yet another promotion & a bonus!

LEARNING OUTCOMES

By the end of this session, you should be able to:

- Use the **divide and conquer strategy** for problem solving
- Explain the **binary search algorithm** and prove its correctness
- Explain the **mergesort algorithm** as an example of the divide and conquer strategy.
- Analyze and describe the **performance of mergesort and binary search** using $O(g(n))$ notation.

OTHER TAKE AWAYS

Divide & Conquer:

- Split
- Solve
- Combine

Don’t Panic when you face a complicated problem (e.g., recurrence relation)!

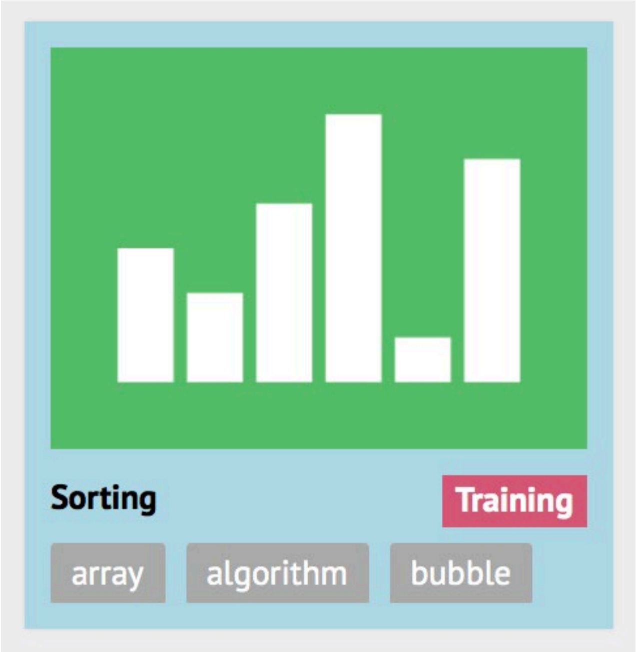
- Break down the problem and think through the cases.
- Build up a solution from the cases

This is also Divide & Conquer!

BEFORE NEXT LECTURE

Go to Visualgo.net and do the Sorting Module:

- https://visualgo.net/en/sorting
- Review: 11 (Quicksort)-12
- Optional: 13 onwards



QUESTIONS?