

Divide and Conquer

CSCI 432

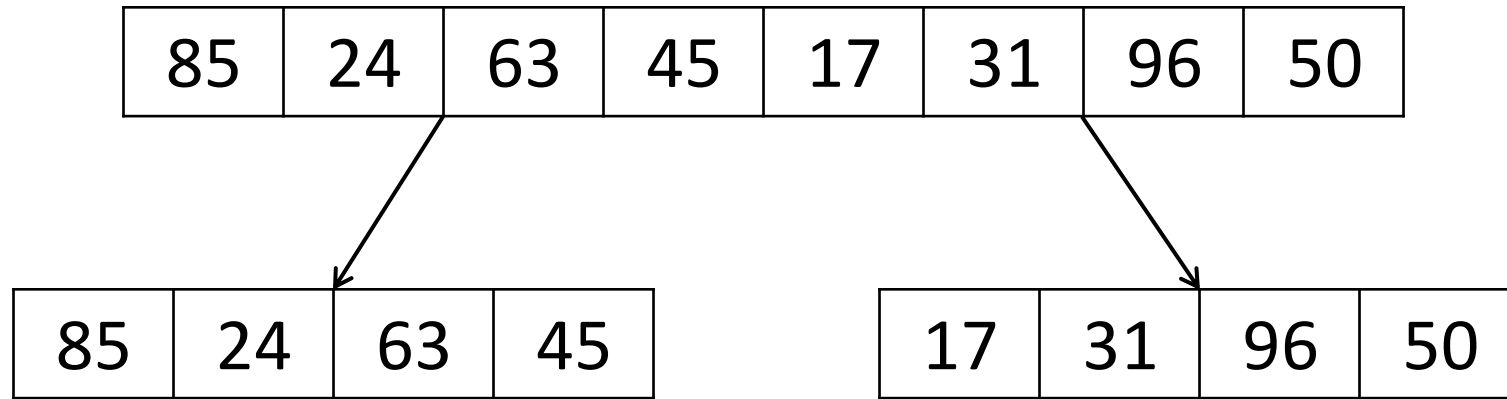
Divide and Conquer Battle Plan

1. Divide problem into subproblems that are smaller instances of the same problem.
2. Conquer the subproblems by solving them recursively.
3. Combine the solutions to the subproblems into the solution for the original problem.

Divide and Conquer – Merge Sort

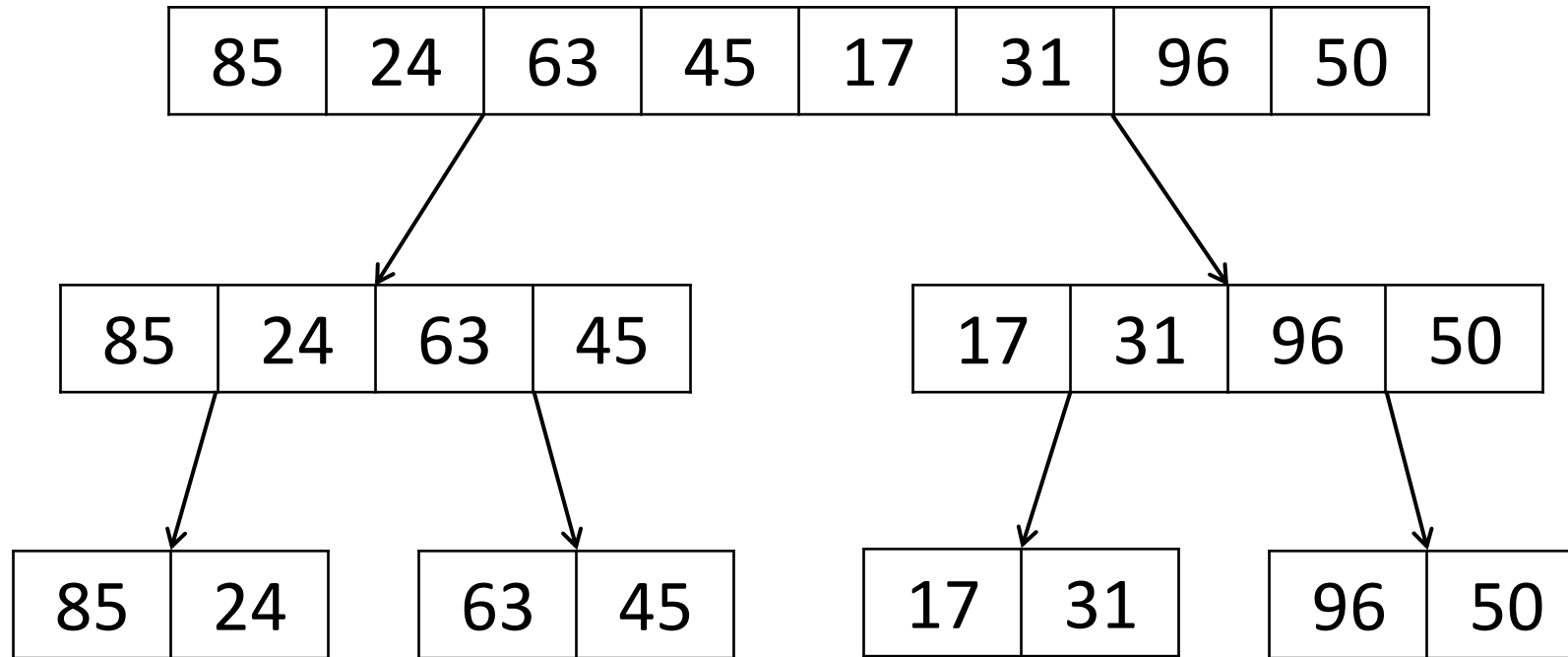
1. Divide array in half.
2. Sort sub arrays.
3. Merge into sorted array.

Divide and Conquer – Merge Sort



1. Divide array in half.
2. Sort sub arrays.
3. Merge into sorted array.

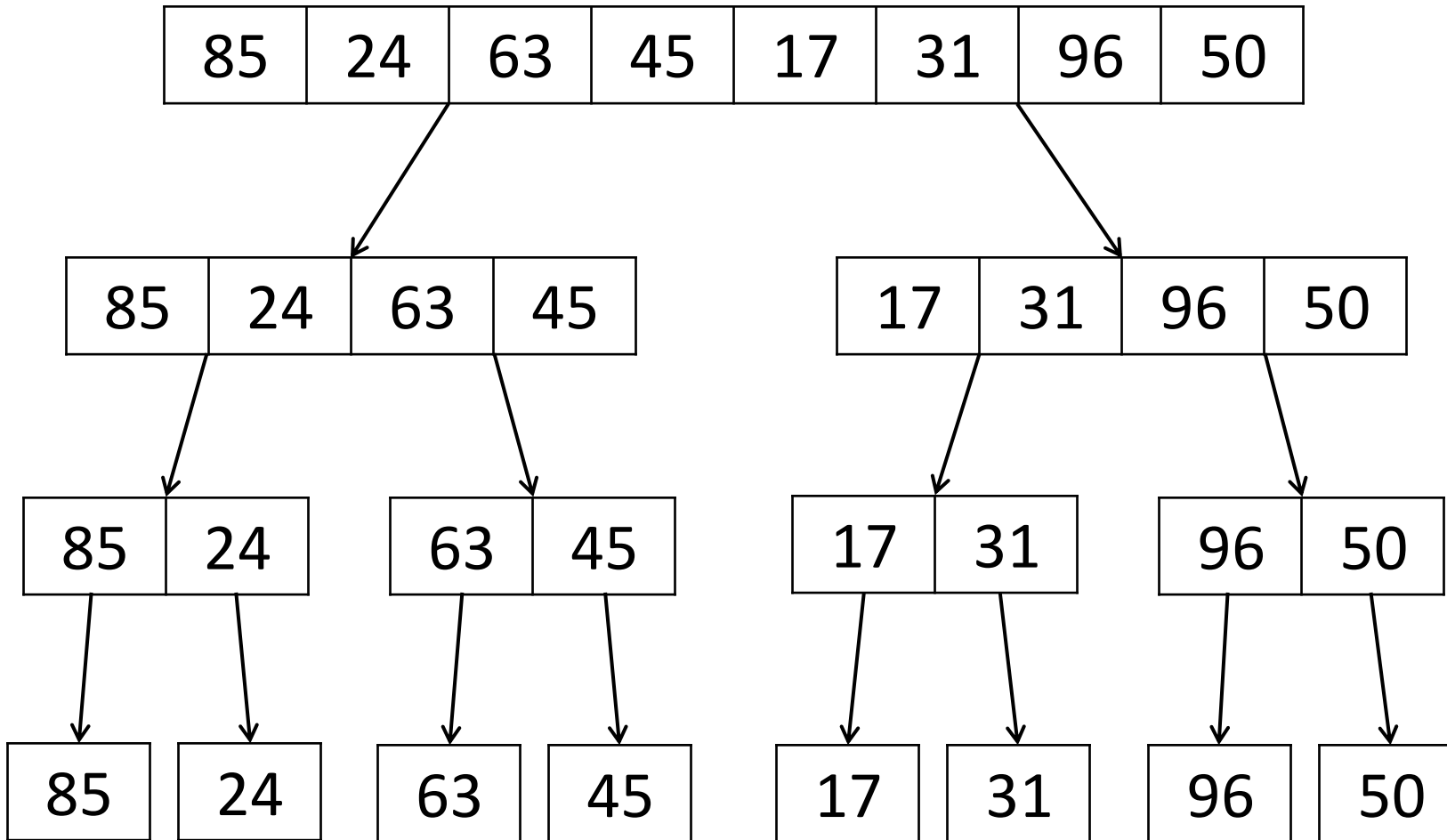
Divide and Conquer – Merge Sort



1. Divide array in half.
2. Sort sub arrays.
3. Merge into sorted array.

Divide and Conquer – Merge Sort

1. Divide array in half.
2. Sort sub arrays.
3. Merge into sorted array.



Divide and Conquer – Merge Sort

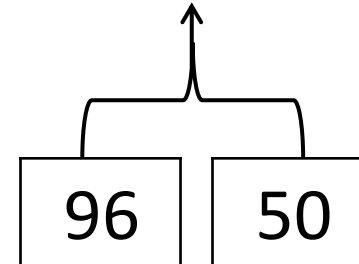
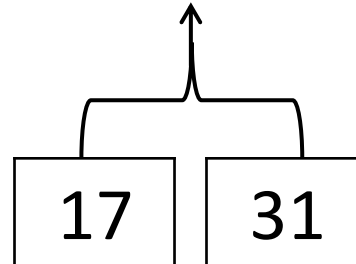
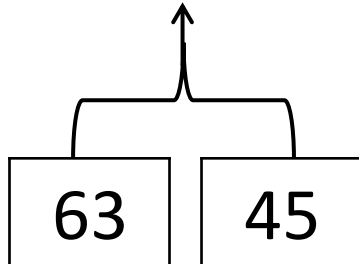
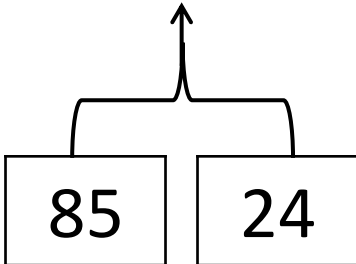
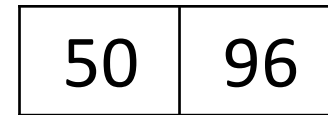
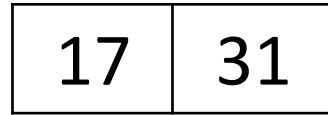
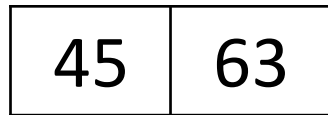
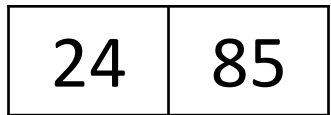


1. Divide array in half.



2. Sort sub arrays.

3. Merge into sorted array.



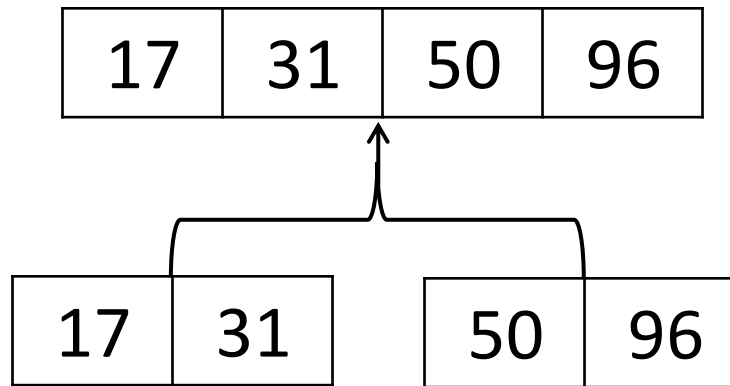
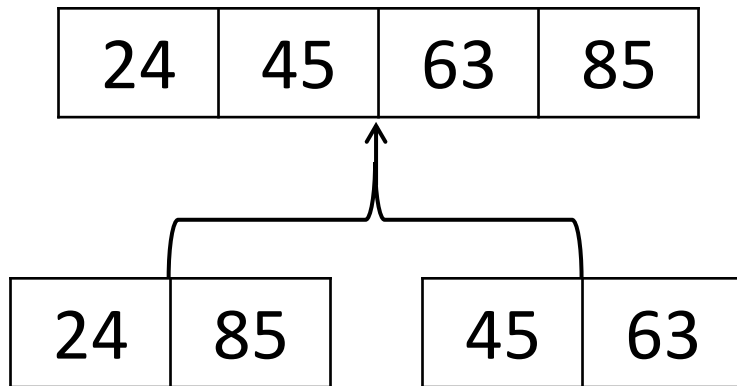
Divide and Conquer – Merge Sort



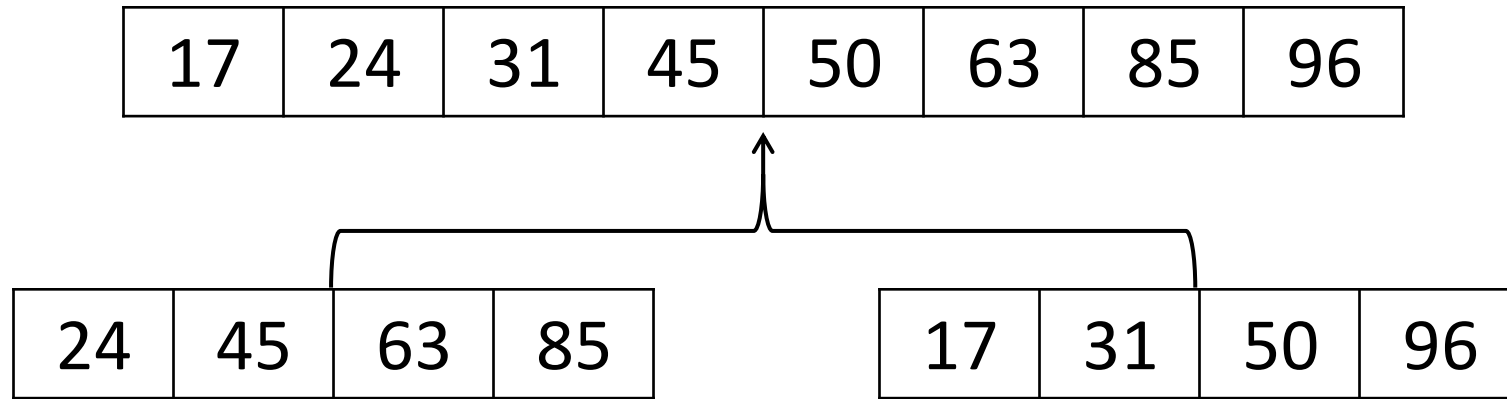
1. Divide array in half.

2. Sort sub arrays.

3. Merge into sorted array.



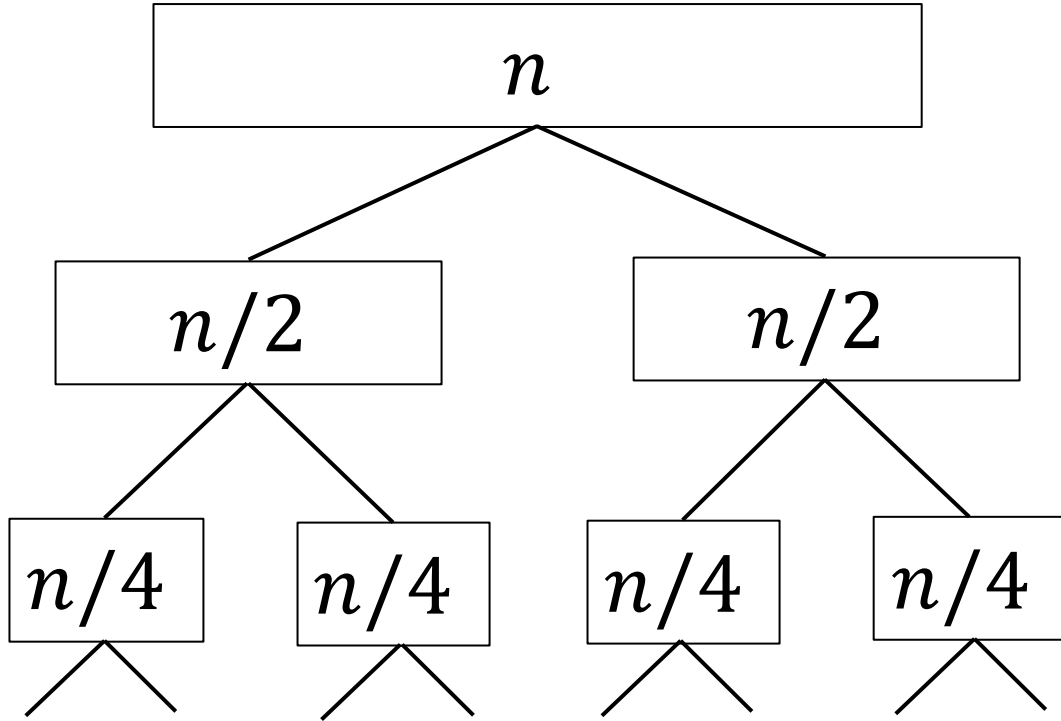
Divide and Conquer – Merge Sort



1. Divide array in half.
2. Sort sub arrays.
3. Merge into sorted array.

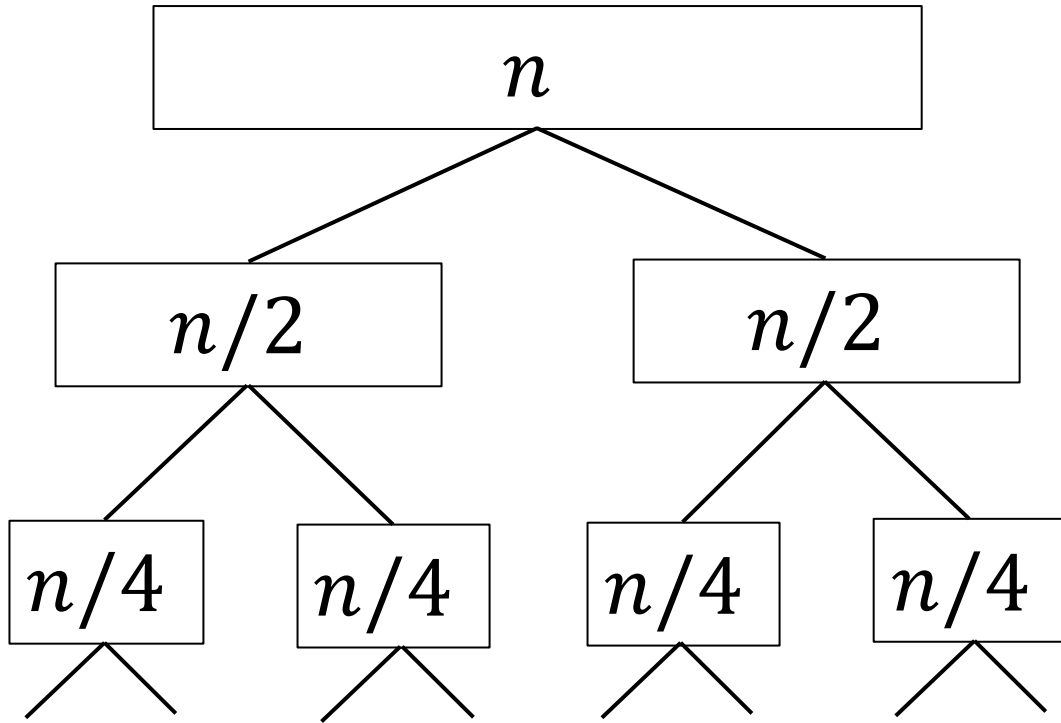
Recursion – Running Time

Recursion – Running Time



Often, determining running times for algorithms that use recursion benefits from using a recursion tree

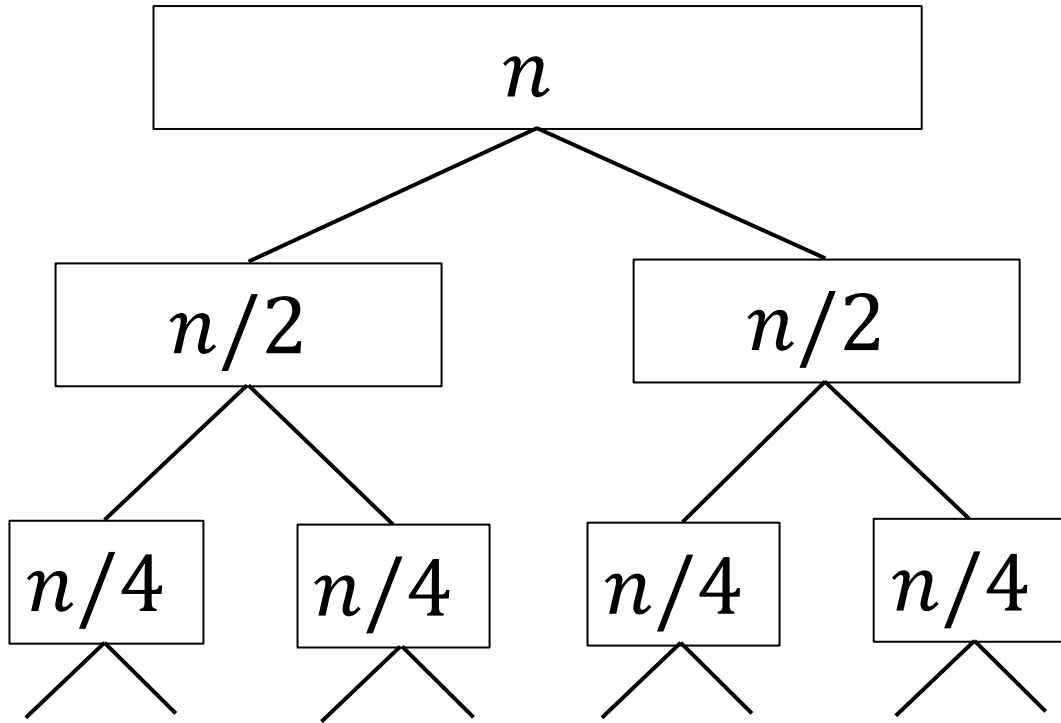
Recursion – Running Time



$$\text{Running Time} = \text{Amount of work per level} \times \text{Number of levels}$$

Often, determining running times for algorithms that use recursion benefits from using a recursion tree

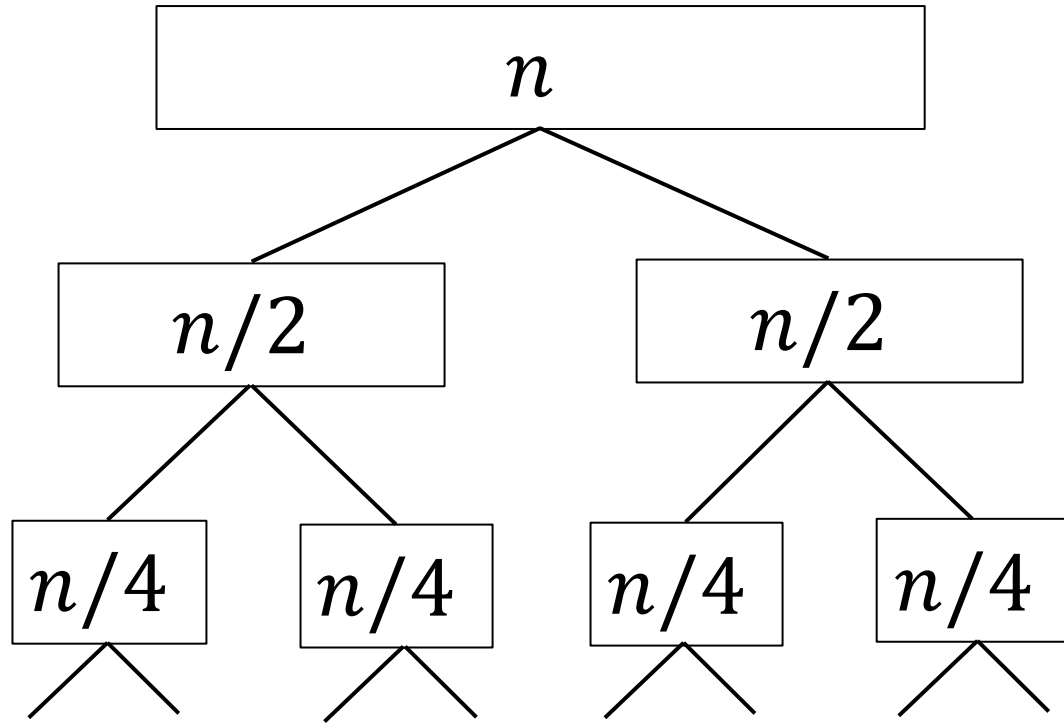
Recursion – Running Time



$$\text{Running Time} = \text{Amount of work per level} \times \text{Number of levels}$$

$O(n)$ 

Recursion – Running Time



Running Time = Amount of work per level $\times$ Number of levels

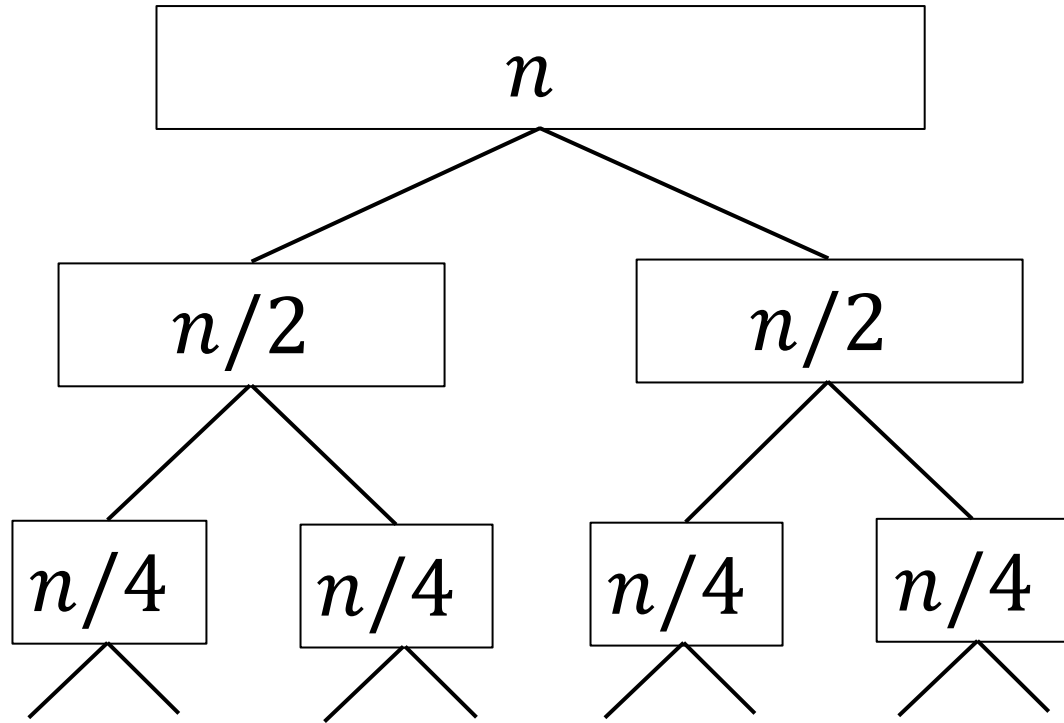
$O(n)$

In level k , there are 2^k arrays. At the bottom, there are n arrays.

$$\Rightarrow \frac{n}{2^k} = 1$$

$$\Rightarrow k = \log(n).$$

Recursion – Running Time



Running Time = Amount of work per level $\times$ Number of levels

$O(n)$

In level k , there are 2^k arrays. At the bottom, there are n arrays.

$$\Rightarrow \frac{n}{2^k} = 1$$

$$\Rightarrow k = \log(n).$$

$O(n \log(n))$ running time

Recursion – Running Time

Recursive divide and conquer running time can be characterized as:

$$T(n) = aT(n/b) + D(n) + C(n)$$

Where, a – Number of subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

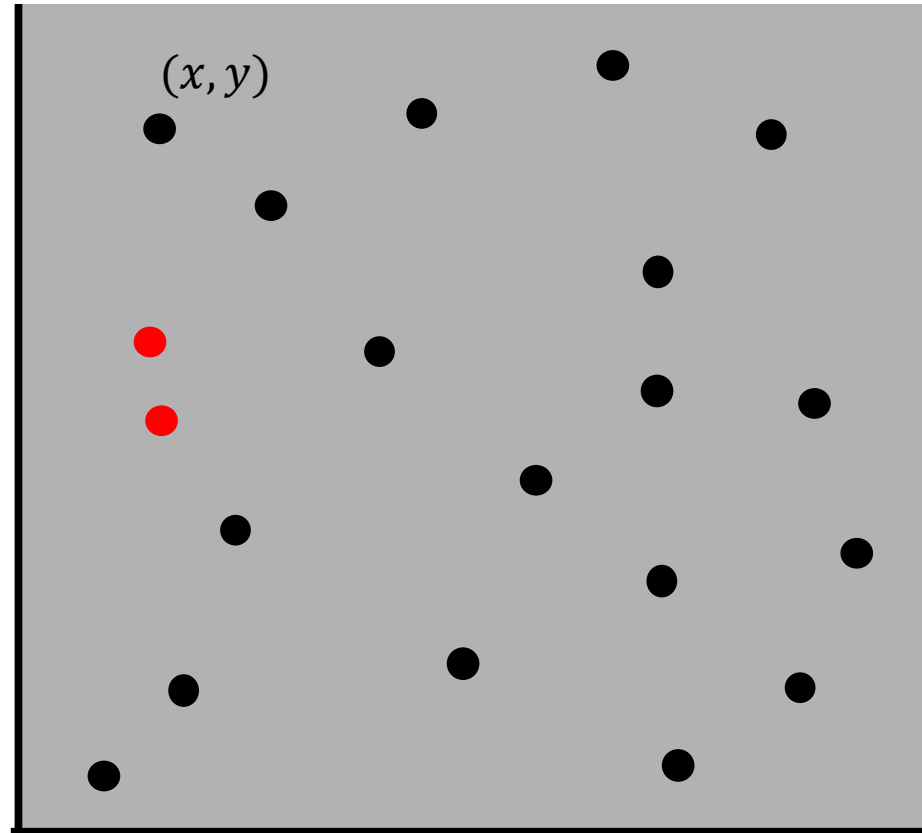
Master theorem: If $T(n) = aT(n/b) + O(n^d)$ for constants $a \geq 1, b > 1, d \geq 0$, then:

$$T(n) \in \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Divide and Conquer Battle Plan

1. Divide problem into subproblems that are smaller instances of the same problem.
2. Conquer the subproblems by solving them recursively.
3. Combine the solutions to the subproblems into the solution for the original problem.

Closest Pair Problem



Given n points, find a pair of points with the smallest distance between them.