COMP170
Discrete Mathematical Tools for Computer Science

Lecture 12 *Version 4: Last updated, Nov 3, 2005*

Discrete Math for Computer Science
K. Bogart, C. Stein and R.L. Drysdale
Section 4.3, pp. 157-167

Growth Rates of Solutions to Recurrences

- Divide and Conquer Algorithms
- Recursion Trees
- Three Different Behaviors

Divide and Conquer Algorithms

Divide and Conquer Algorithms

In the previous section we analyzed recurrences of the form

$$T(n) = \begin{cases} a & \text{if } n = b \\ c \cdot T(n-1) + d & \text{if } n > b \end{cases}$$

Divide and Conquer Algorithms

In the previous section we analyzed recurrences of the form

$$T(n) = \begin{cases} a & \text{if } n = b \\ c \cdot T(n-1) + d & \text{if } n > b \end{cases}$$

These corresponded to the analyses of recursive algorithms in which a problem of size n is solved by recursively solving a problem(s) of size $n-1$.

We will now look at recurrences that arise from recursive algorithms in which problems of size n are solved by recursively solving problems of size n/m, for some fixed m. These recurrences will be in the form

Divide and Conquer Algorithms

In the previous section we analyzed recurrences of the form

$$T(n) = \begin{cases} a & \text{if } n = b \\ c \cdot T(n-1) + d & \text{if } n > b \end{cases}$$

These corresponded to the analyses of recursive algorithms in which a problem of size n is solved by recursively solving a problem(s) of size $n-1$.

We will now look at recurrences that arise from recursive algorithms in which problems of size n are solved by recursively solving problems of size n/m, for some fixed m. These recurrences will be in the form

$$T(n) = \begin{cases} \text{something given} & \text{if } n \le b \\ c \cdot T(n/m) + d & \text{if } n > b \end{cases}$$

Divide and Conquer Algorithms

Our first example will be binary search.

Someone has a chosen a number x between 1 and n.

We need to discover x.

We are only allowed to ask two types of questions:

Divide and Conquer Algorithms

Our first example will be binary search.

Someone has a chosen a number x between 1 and n.

We need to discover x.

We are only allowed to ask two types of questions:

- Is x greater than k?
- Is x equal to k?

Divide and Conquer Algorithms

Our first example will be binary search.
Someone has a chosen a number x between 1 and n.
We need to discover x.

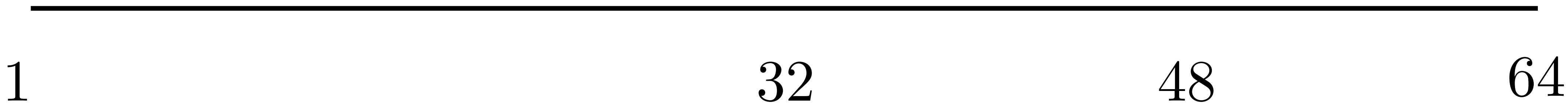
We are only allowed to ask two types of questions:

- Is x greater than k?
- Is x equal to k?

Our strategy will be to always ask greater than questions,
at each step halving our search range,
until the range only contains one number,
when we ask a final equal to question

Binary Search Example

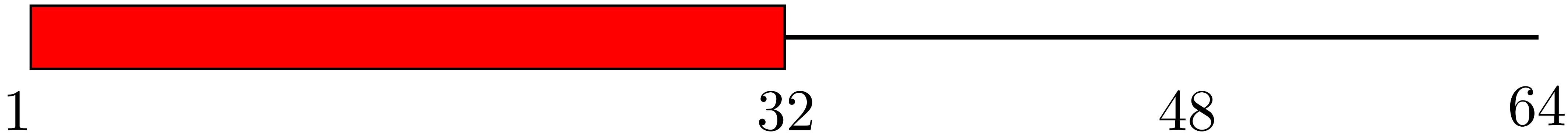
Binary Search Example



Binary Search Example

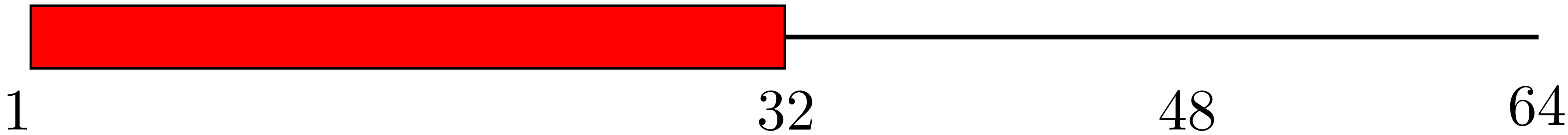
Is $x > 32$?

Binary Search Example



Is $x > 32$? Answer: Yes

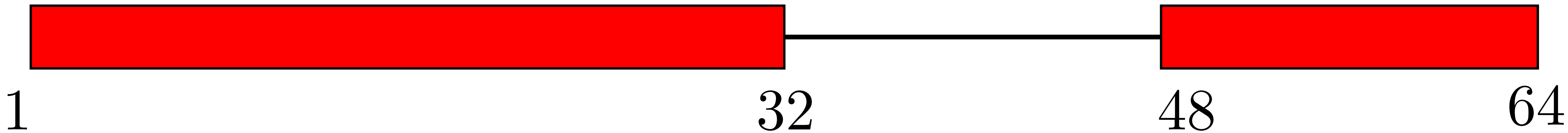
Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$?

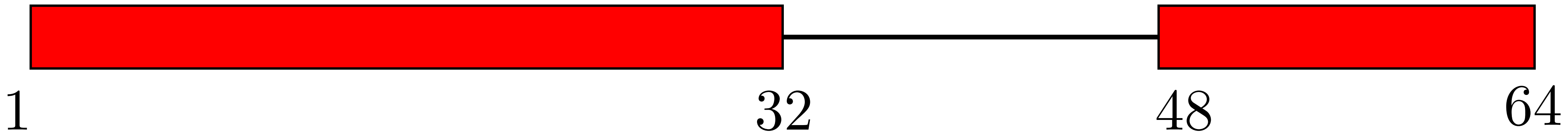
Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Binary Search Example

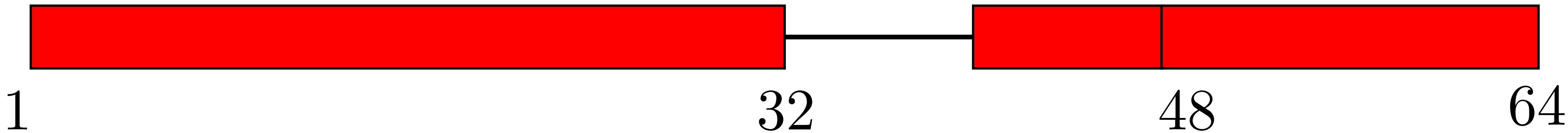


Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Is $x > 40$?

Binary Search Example

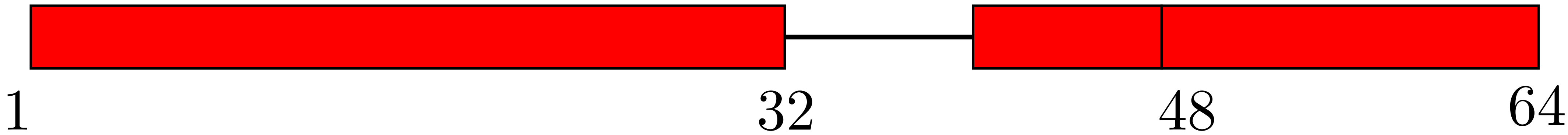


Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

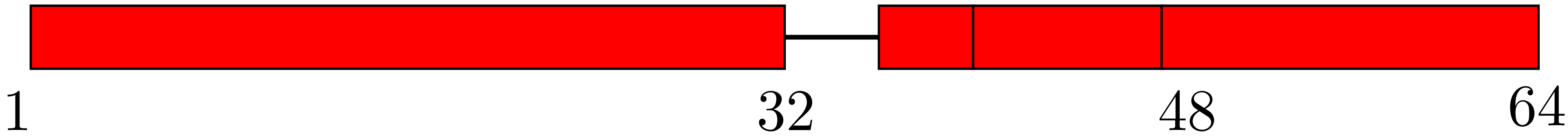
Is $x > 40$? Answer: No

Binary Search Example



Is $x > 32$? Answer: Yes
Is $x > 48$? Answer: No
Is $x > 40$? Answer: No
Is $x > 36$?

Binary Search Example



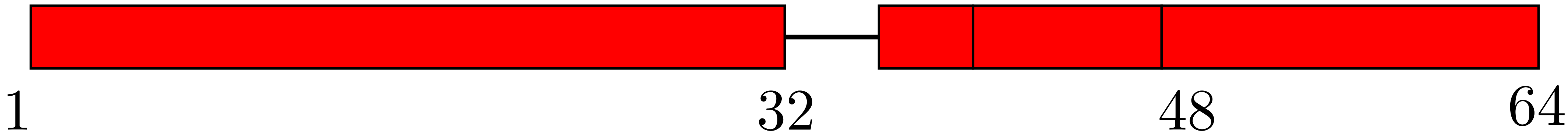
Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Binary Search Example



Is $x > 32$? Answer: Yes

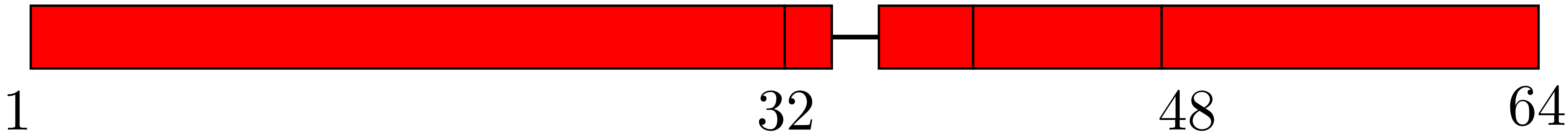
Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$?

Binary Search Example



Is $x > 32$? Answer: Yes

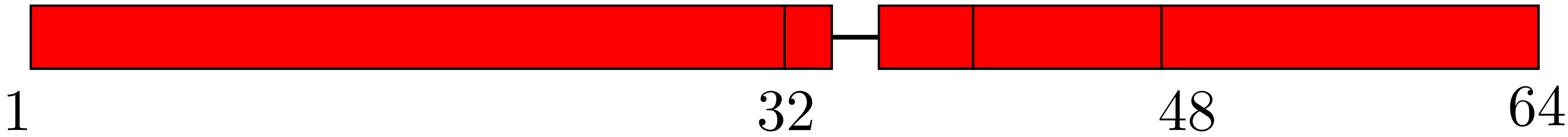
Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

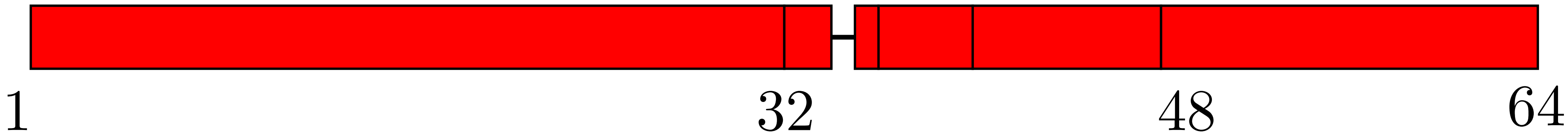
Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

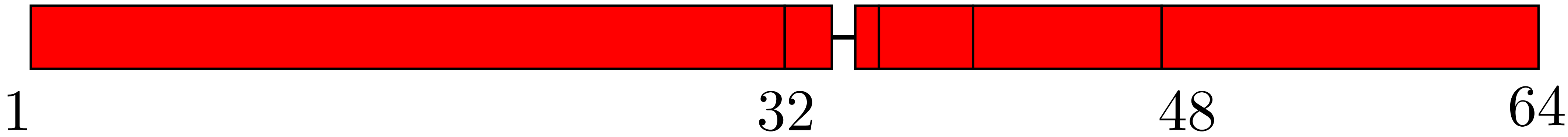
Is $x > 35$?

Binary Search Example



Is $x > 32$?	Answer: Yes
Is $x > 48$?	Answer: No
Is $x > 40$?	Answer: No
Is $x > 36$?	Answer: No
Is $x > 34$?	Answer: Yes
Is $x > 35$?	Answer: No

Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

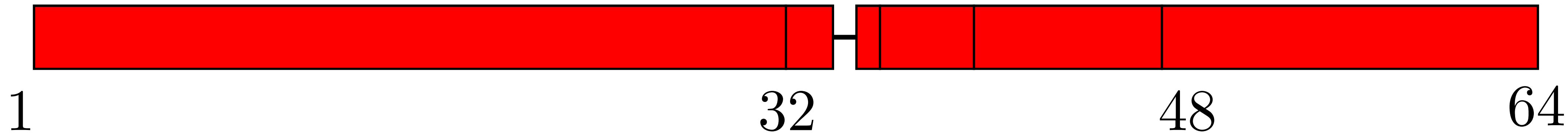
Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

Is $x > 35$? Answer: No

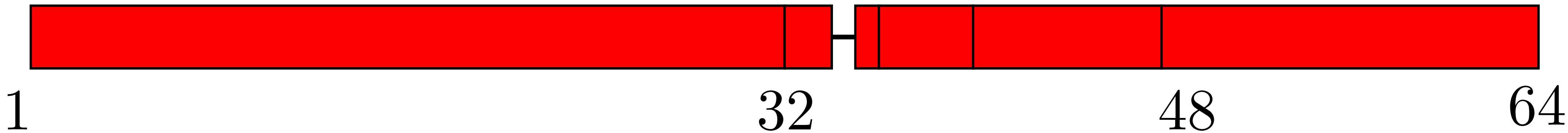
Is $x = 35$?

Binary Search Example



Is $x > 32$?	Answer: Yes
Is $x > 48$?	Answer: No
Is $x > 40$?	Answer: No
Is $x > 36$?	Answer: No
Is $x > 34$?	Answer: Yes
Is $x > 35$?	Answer: No
Is $x = 35$?	Answer: BINGO!

Binary Search Example



Is $x > 32?$ Answer: Yes

Is $x > 48?$ Answer: No

Is $x > 40?$ Answer: No

Is $x > 36?$ Answer: No

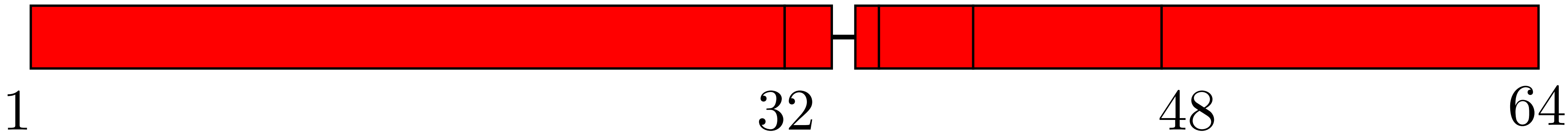
Is $x > 34?$ Answer: Yes

Is $x > 35?$ Answer: No

Is $x = 35?$ Answer: BINGO!

Method: Each guess reduces the problem to one in which the range is only **half** as big.

Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

Is $x > 35$? Answer: No

Is $x = 35$? Answer: BINGO!

Method: Each guess reduces the problem to one in which the range is only **half** as big.

This divides the original problem into one that is only half as big; we can now (recursively) conquer this smaller problem.

$T(n)$: number of questions in a binary search on $[1, n]$.

$T(n)$: number of questions in a binary search on $[1, n]$.

Assume: n is power of 2. Give recurrence for $T(n)$.

$T(n)$: number of questions in a binary search on $[1, n]$.

Assume: n is power of 2. Give recurrence for $T(n)$.

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

$T(n)$: number of questions in a binary search on $[1, n]$.

Assume: n is power of 2. Give recurrence for $T(n)$.

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

Number of questions needed for binary search on n items is:

$T(n)$: number of questions in a binary search on $[1, n]$.

Assume: n is power of 2. Give recurrence for $T(n)$.

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

Number of questions needed for binary search on n items is:

first step

$T(n)$: number of questions in a binary search on $[1, n]$.

Assume: n is power of 2. Give recurrence for $T(n)$.

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

Number of questions needed for binary search on n items is:

first step

plus

time to perform binary search on the remaining $n/2$ items.

$T(n)$: number of questions in a binary search on $[1, n]$.

Assume: n is power of 2. Give recurrence for $T(n)$.

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

Number of questions needed for binary search on n items is:

first step

plus

time to perform binary search on the remaining $n/2$ items.

Base case (1 item): $T(1) = 1$ to ask: "Is the number k?"

$T(n)$: number of questions in a binary search on $[1, n]$.

Assume: n is power of 2. Give recurrence for $T(n)$.

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

Number of questions needed for binary search on n items is:

first step

plus

time to perform binary search on the remaining $n/2$ items.

Base case (1 item): $T(1) = 1$ to ask: "Is the number k?"

Note: Our derivation that, when n is a power of 2, $T(n)$, the number of questions in a binary search on $[1, n]$, satisfies

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

was actually, implicitly, an inductive proof. This is similar to what we saw with the tower of Hanoi recurrence. We did not write out all the formal steps of the inductive proof, though.

$T(n)$: number of questions in a binary search on $[1, n]$.

$T(n)$: number of questions in a binary search on $[1, n]$.

If n is not power of 2 then $n/2$ not an integer, so what is $T(n/2)$?
Also, why should a greater than query
take same amount of time as equal to query?
More realistic time modeling is

$T(n)$: number of questions in a binary search on $[1, n]$.

If n is not power of 2 then $n/2$ not an integer, so what is $T(n/2)$?
Also, why should a greater than query
take same amount of time as equal to query?
More realistic time modeling is

$$T(n) = \begin{cases} T(\lceil n/2 \rceil) + C_1 & \text{if } n \geq 2, \\ C_2 & \text{if } n = 1, \end{cases}$$

where C_1 and C_2 are constants.

$T(n)$: number of questions in a binary search on $[1, n]$.

If n is not power of 2 then $n/2$ not an integer, so what is $T(n/2)$?
Also, why should a greater than query
take same amount of time as equal to query?
More realistic time modeling is

$$T(n) = \begin{cases} T(\lceil n/2 \rceil) + C_1 & \text{if } n \geq 2, \\ C_2 & \text{if } n = 1, \end{cases}$$

where C_1 and C_2 are constants.

$\lceil x \rceil$ is smallest integer larger than or equal to x,

$\lfloor x \rfloor$ is largest integer less than or equal to x.

$T(n)$: number of questions in a binary search on $[1, n]$.

If n is not power of 2 then $n/2$ not an integer, so what is $T(n/2)$?
Also, why should a greater than query
take same amount of time as equal to query?
More realistic time modeling is

$$T(n) = \begin{cases} T(\lceil n/2 \rceil) + C_1 & \text{if } n \geq 2, \\ C_2 & \text{if } n = 1, \end{cases}$$

where C_1 and C_2 are constants.

$\lceil x \rceil$ is smallest integer larger than or equal to x, $\lceil 3.5 \rceil = 4$ $\lceil 3 \rceil = 3$

$\lfloor x \rfloor$ is largest integer less than or equal to x. $\lfloor 3.5 \rfloor = 3$ $\lfloor 3 \rfloor = 3$

(*) $$T(n) = \begin{cases} T(\lceil n/2 \rceil) + C_1 & \text{if } n \geq 2, \\ C_2 & \text{if } n = 1, \end{cases}$$

In order to avoid complications we will (usually) assume that n is a power of 2 (or sometimes 3 or 4) and also often that constants such as C_1, C_2 are 1. This will let us replace a recurrence such as (*) by one such as (**).

(**) $$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

In practice, the solution of (*) will be very close to the solution of (**) (this can be proven mathematically) so, as in this class, we can restrict ourselves to (**) without losing much.

Growth Rates of Solutions to Recurrences

- Divide and Conquer Algorithms
- Recursion Trees
- Three Different Behaviors

Recursion Trees

Recursion Trees

Recursion Trees are a visual and conceptual representation of the process of iterating the recurrence.

Recursion Trees

Recursion Trees are a visual and conceptual representation of the process of iterating the recurrence.

Examples:

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

Recursion Trees

Recursion Trees are a visual and conceptual representation of the process of iterating the recurrence.

Examples:

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

To solve some problem of size n, we
(i) solve 2 subproblems of size $n/2$ and
(ii) do n units of additional work.

Recursion Trees

Recursion Trees are a visual and conceptual representation of the process of iterating the recurrence.

Examples:

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

To solve some problem of size n, we
(i) solve 2 subproblems of size $n/2$ and
(ii) do n units of additional work.

$$T(n) = T\left(\frac{n}{4}\right) + n^2$$

Recursion Trees

Recursion Trees are a visual and conceptual representation of the process of iterating the recurrence.

Examples:

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

To solve some problem of size n, we
(i) solve 2 subproblems of size $n/2$ and
(ii) do n units of additional work.

$$T(n) = T\left(\frac{n}{4}\right) + n^2$$

To solve some problem of size n, we
(i) solve 1 subproblem of size $n/4$ and
(ii) do n^2 units of additional work.

Recursion Trees

Recursion Trees are a visual and conceptual representation of the process of iterating the recurrence.

Examples:

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

To solve some problem of size n, we
(i) solve 2 subproblems of size $n/2$ and
(ii) do n units of additional work.

$$T(n) = T\left(\frac{n}{4}\right) + n^2$$

To solve some problem of size n, we
(i) solve 1 subproblem of size $n/4$ and
(ii) do n^2 units of additional work.

$$T(n) = 3T\left(n-1\right) + n$$

Recursion Trees

Recursion Trees are a visual and conceptual representation of the process of iterating the recurrence.

Examples:

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

To solve some problem of size n, we
(i) solve 2 subproblems of size $n/2$ and
(ii) do n units of additional work.

$$T(n) = T\left(\frac{n}{4}\right) + n^2$$

To solve some problem of size n, we
(i) solve 1 subproblem of size $n/4$ and
(ii) do n^2 units of additional work.

$$T(n) = 3T\left(n-1\right) + n$$

To solve some problem of size n, we
(ii) solve 3 subproblems of size $n-1$ and
(ii) do n units of additional work.

We will start off by examining the recurrence

$$(*) \qquad T(n) = 2T\left(\frac{n}{2}\right) + n$$

This corresponds to solving a problem of size n, by
(i) solving 2 subproblems of size $n/2$ and
(ii) doing n units of additional work.

We will start off by examining the recurrence

$$(*) \qquad T(n) = 2T\left(\frac{n}{2}\right) + n$$

This corresponds to solving a problem of size n, by
(i) solving 2 subproblems of size $n/2$ and
(ii) doing n units of additional work.

In your later "analysis of algorithms" class (COMP271), you will see that this is exactly how Mergesort, one of the most famous sorting algorithms, works.

We will start off by examining the recurrence

$$(*) \qquad T(n) = 2T\left(\frac{n}{2}\right) + n$$

This corresponds to solving a problem of size n, by
(i) solving 2 subproblems of size $n/2$ and
(ii) doing n units of additional work.

In your later "analysis of algorithms" class (COMP271), you will see that this is exactly how Mergesort, one of the most famous sorting algorithms, works.

We will now see how to "solve" (*), first by algebraically iterating the recurrence, and then by using a recusion tree (which is a visual method for iterating the recurrence).

Example: Algebraically iterating the recurrence

Example: Algebraically iterating the recurrence

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

Example: Algebraically iterating the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

Example: Algebraically iterating the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \quad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

Example: Algebraically iterating the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \quad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

Example: Algebraically iterating the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \quad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$\vdots \qquad\qquad \vdots$$

Example: Algebraically iterating the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \quad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$\vdots \qquad\qquad \vdots$$

$$= 2^{(\log_2 n)} T\left(\frac{n}{2^{(\log_2 n)}}\right) + (\log_2 n)\, n$$

Example: Algebraically iterating the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \quad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$\vdots \qquad\qquad \vdots$$

$$= 2^{(\log_2 n)} T\left(\frac{n}{2^{(\log_2 n)}}\right) + (\log_2 n)\, n$$

$$= nT(1) + n \log_2 n$$

Example: Algebraically iterating the recurrence

$$\boxed{T(n)} = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \quad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$\vdots \qquad\qquad \vdots$$

$$= 2^{(\log_2 n)} T\left(\frac{n}{2^{(\log_2 n)}}\right) + (\log_2 n)\, n$$

$$\boxed{= nT(1) + n \log_2 n}$$

Example: Visually “iterating” the recurrence

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

Example: Visually "iterating" the recurrence

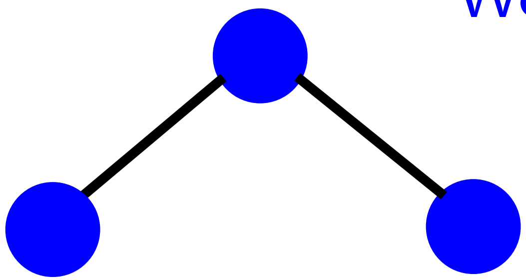
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

# of Probl	Probl Size	Work per Probl	Work per Level

Example: Visually "iterating" the recurrence

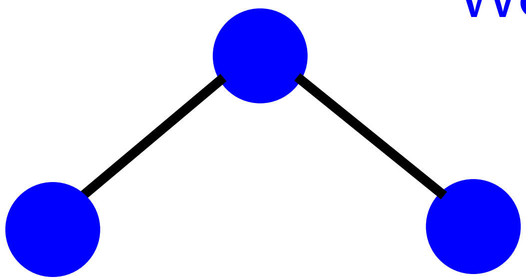
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$		



Example: Visually "iterating" the recurrence

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

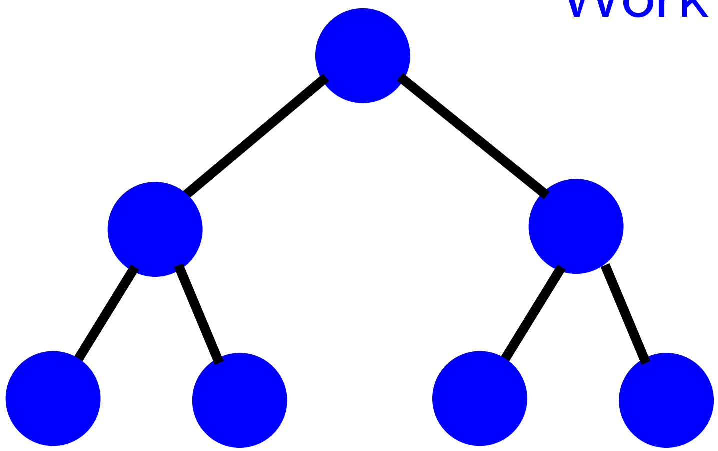


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$

Example: Visually "iterating" the recurrence

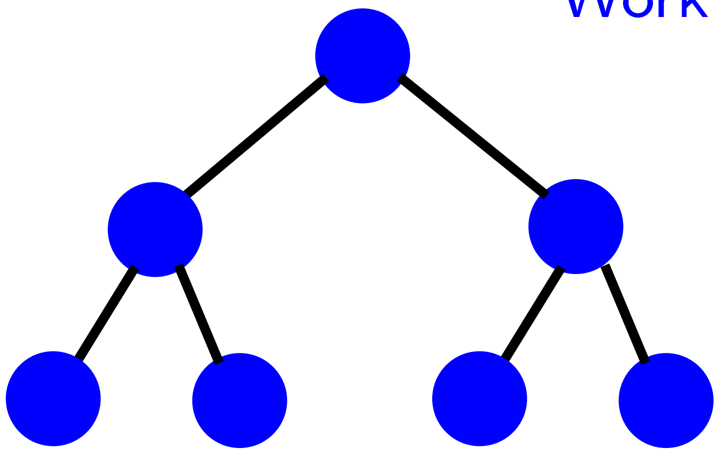
$$T(n) = 2T\left(\frac{n}{2}\right) + n = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n = 4T\left(\frac{n}{4}\right) + 2n$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$		



Example: Visually "iterating" the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \; = 4T\left(\frac{n}{4}\right) + 2n$$

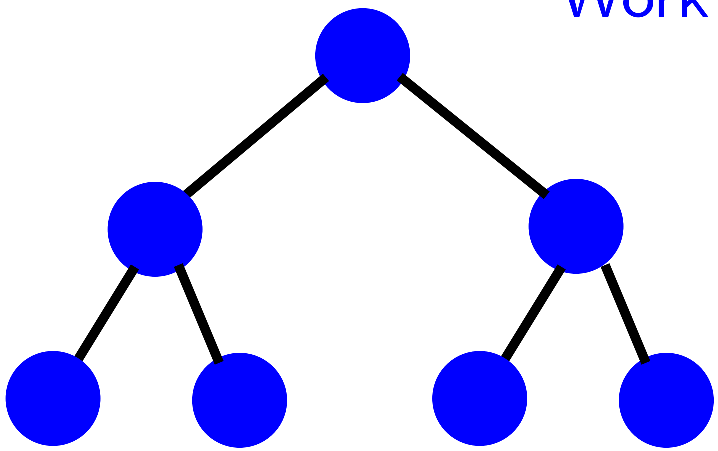


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$		

Example: Visually "iterating" the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \; = 4T\left(\frac{n}{4}\right) + 2n$$

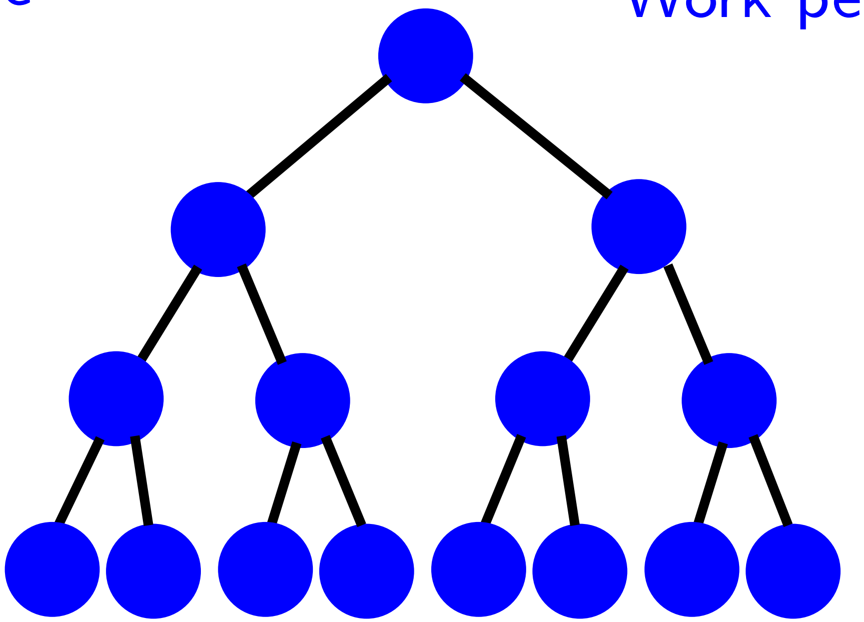
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$



Example: Visually "iterating" the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \;= 4T\left(\frac{n}{4}\right) + 2n \;= 8T\left(\frac{n}{4}\right) + 3n.$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$
$2^3 = 8$	$n/8$		



Recursion Tree Diagram

Recursion Tree Diagram

We need to determine four things for each level:

Recursion Tree Diagram

We need to determine four things for each level:

- the number of subproblems

Recursion Tree Diagram

We need to determine four things for each level:

- the number of subproblems
- the size of each subproblem

Recursion Tree Diagram

We need to determine four things for each level:

- the number of subproblems
- the size of each subproblem
- the amount of work done per subproblem

Recursion Tree Diagram

We need to determine four things for each level:

- the number of subproblems
- the size of each subproblem
- the amount of work done per subproblem
- the total work done at that level

Recursion Tree Diagram

We need to determine four things for each level:

- the number of subproblems
- the size of each subproblem
- the amount of work done per subproblem
- the total work done at that level

  At bottom level, amount of work comes from $T(1)$.

Recursion Tree Diagram

We need to determine four things for each level:

- the number of subproblems
- the size of each subproblem
- the amount of work done per subproblem
- the total work done at that level

  At bottom level, amount of work comes from $T(1)$.

We also need to figure out how many levels there are in the recursion tree.

Example 1

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

Example 1

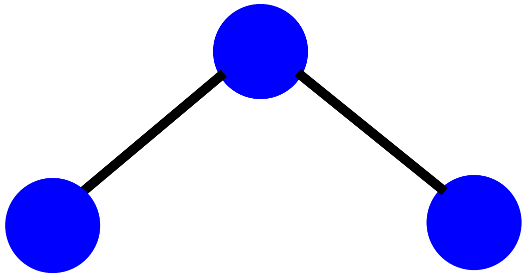
$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level

Example 1

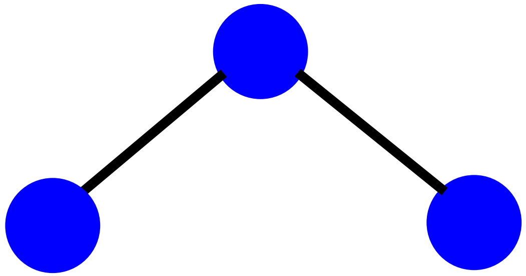
$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$		



Example 1

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

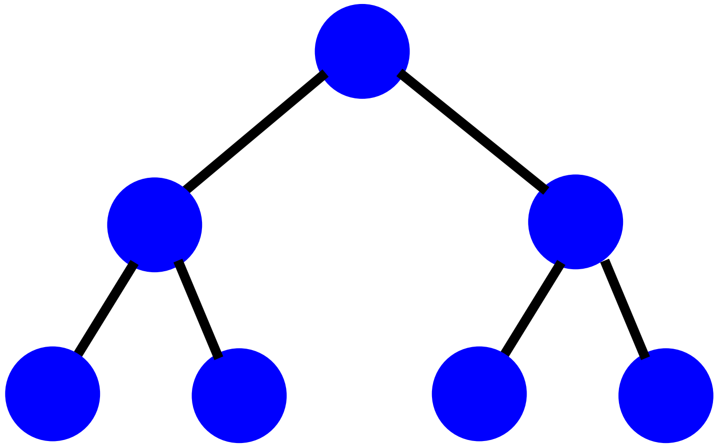


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$

Example 1

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

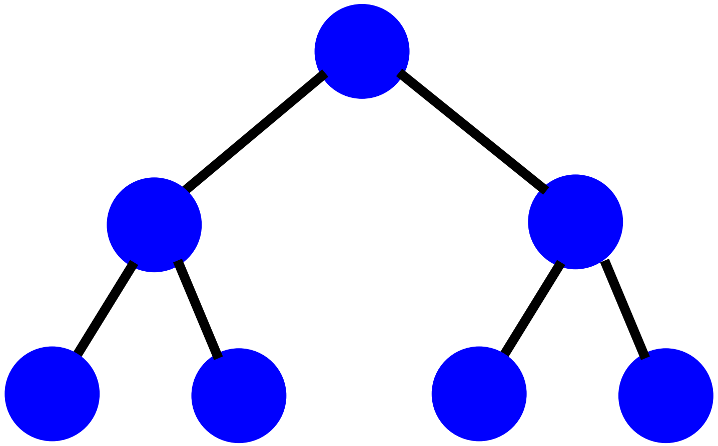
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$		



Example 1

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

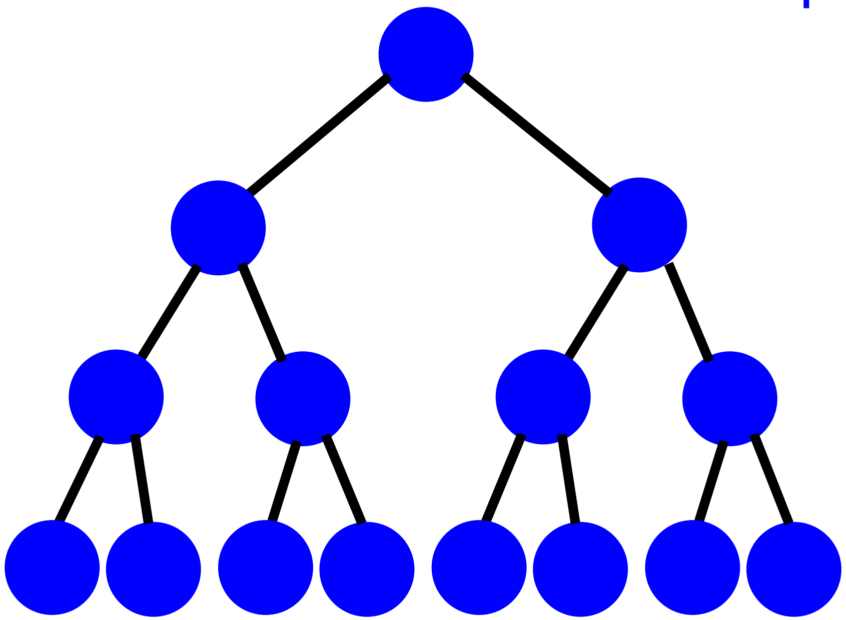
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$



Example 1

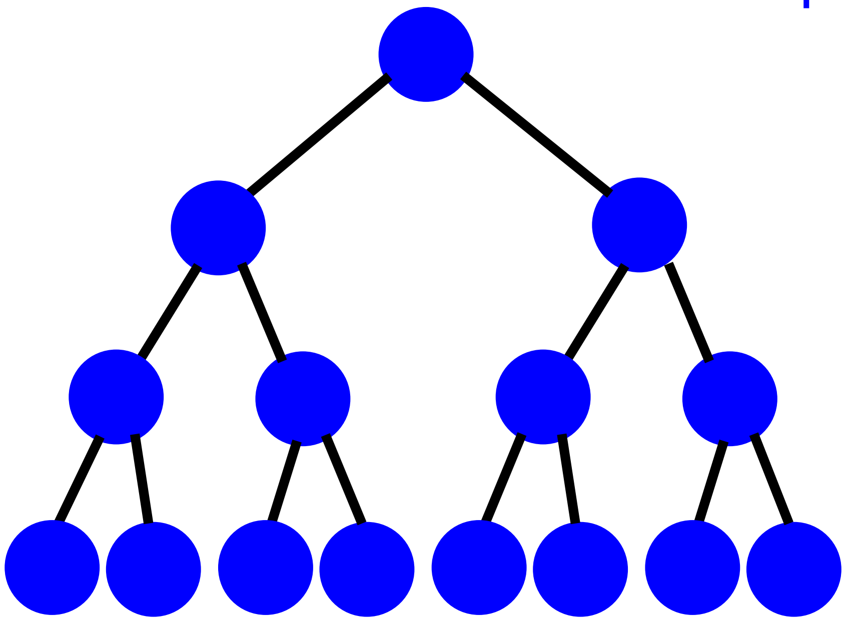
$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$
$2^3 = 8$	$n/8$		



Example 1

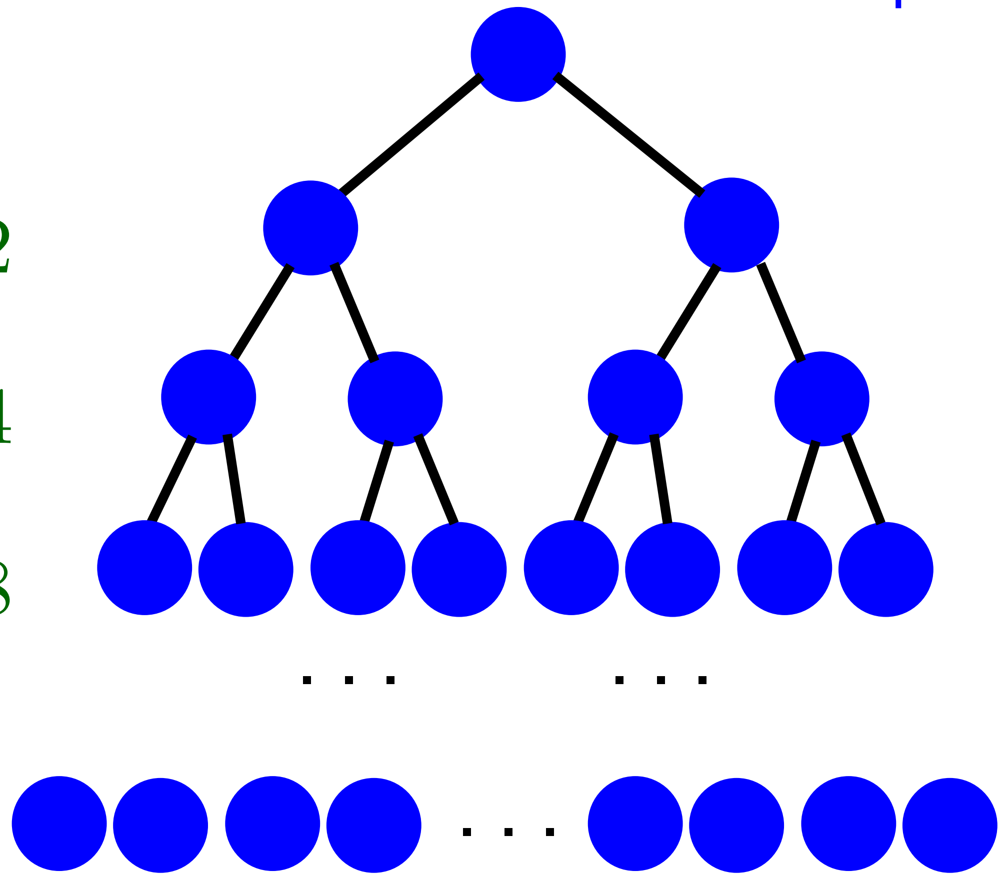
$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$
$2^3 = 8$	$n/8$	$n/8$	$8(n/8) = n$

Example 1

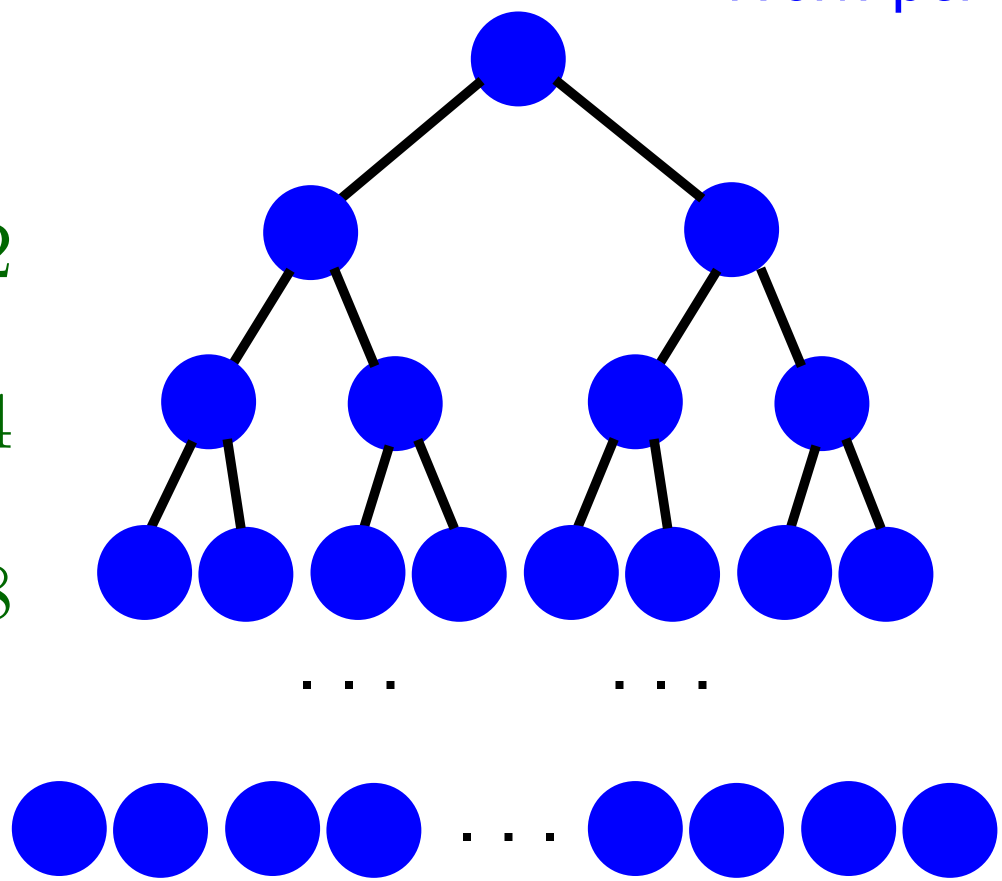
$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$
$2^3 = 8$	$n/8$	$n/8$	$8(n/8) = n$
$2^{\log n} = n$	1		

Example 1

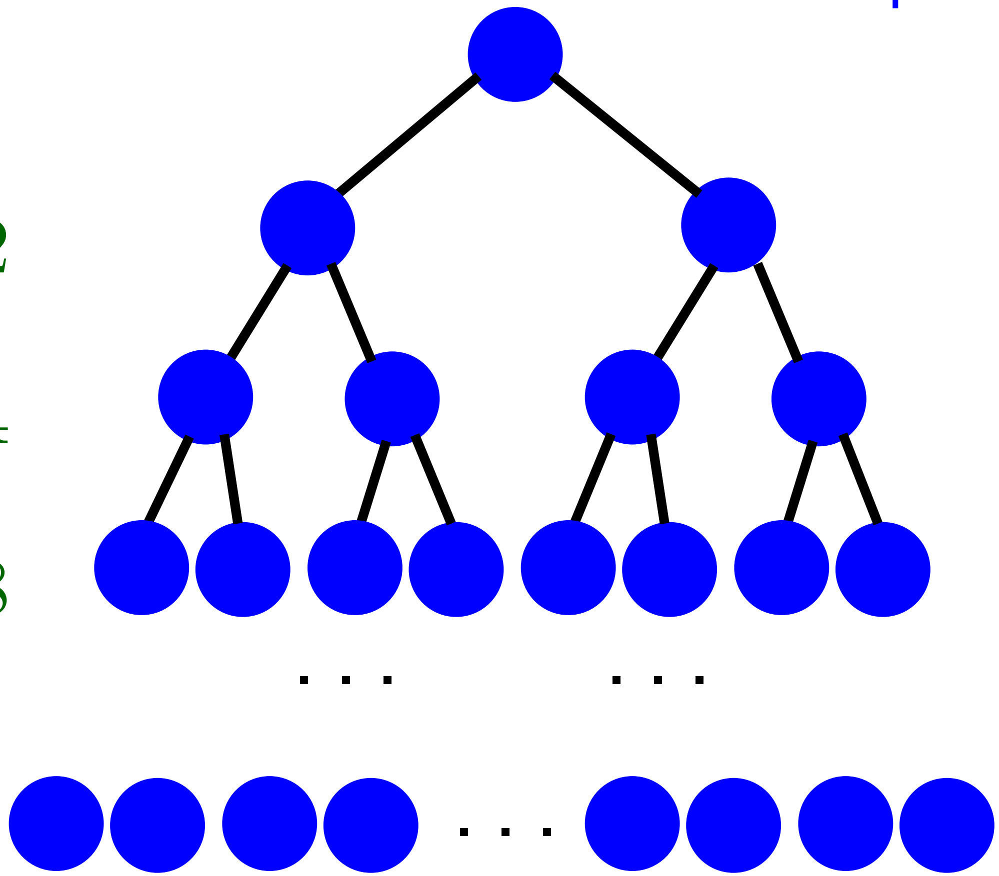
$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$
$2^3 = 8$	$n/8$	$n/8$	$8(n/8) = n$
$2^{\log n} = n$	1	$T(1)$	$nT(1)$

Example 1

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
2	$n/2$	$n/2$	$2(n/2) = n$
$2^2 = 4$	$n/4$	$n/4$	$4(n/4) = n$
$2^3 = 8$	$n/8$	$n/8$	$8(n/8) = n$
$2^{\log n} = n$	1	$T(1)$	$nT(1)$

$(1 + \log_2 n)$ levels $\Rightarrow$ total work $= n \log_2 n + nT(1)$

We just derived that the solution to

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

is $nT(1) + n \log_2 n$.

We derived this in two different, but similar, ways; first by algebraically iterating the recurrence. Second by drawing the recursion tree.

We just derived that the solution to

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n \geq 2, \\ T(1) & \text{if } n = 1. \end{cases}$$

is $nT(1) + n\log_2 n$.

We derived this in two different, but similar, ways; first by algebraically iterating the recurrence. Second by drawing the recursion tree.

Note: Technically, we still need to use induction to prove *that our solution is correct. Practically, we never explicitly perform this step, since it is obvious how the induction would work (the* $\boxed{\ldots}$ *in the algebraic iteration and the recursion tree, are really hiding an inductive step).*

Example 2

Example 2

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

Example 2

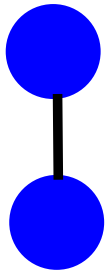
$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level

Example 2

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

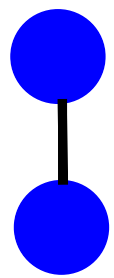
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$		



Example 2

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

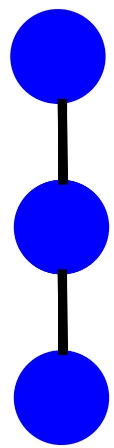
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1



Example 2

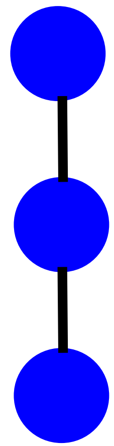
$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1
1	$n/4$		



Example 2

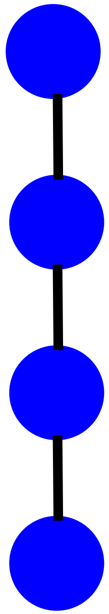
$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1
1	$n/4$	1	1

Example 2

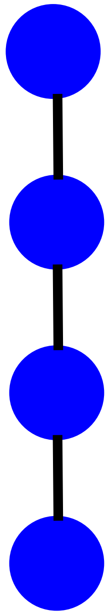
$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1
1	$n/4$	1	1
1	$n/8$		

Example 2

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

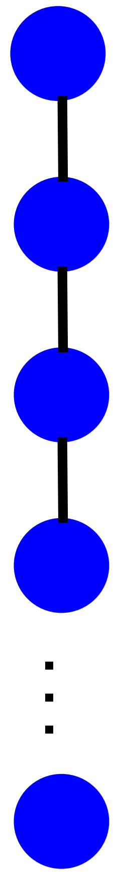


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1
1	$n/4$	1	1
1	$n/8$	1	1

Example 2

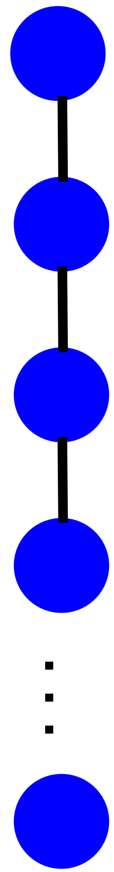
$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1
1	$n/4$	1	1
1	$n/8$	1	1
$\vdots$	$\vdots$		
1	$1 = n/2^{\log_2 n}$		



Example 2

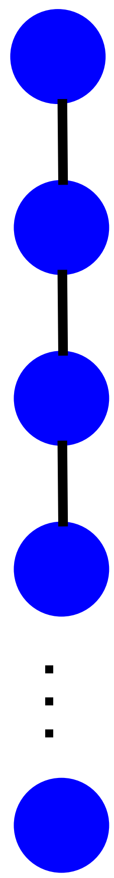
$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1
1	$n/4$	1	1
1	$n/8$	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	$1 = n/2^{\log_2 n}$	1	1

Example 2

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	1	1
1	$n/2$	1	1
1	$n/4$	1	1
1	$n/8$	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	$1 = n/2^{\log_2 n}$	1	1

$(1 + \log_2 n)$ levels $\Rightarrow$ total work $= 1 + \log_2 n$

Example 3

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

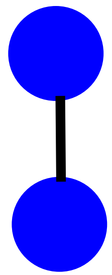
Example 3

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level

Example 3

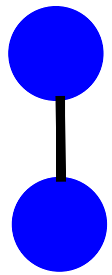
$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$		

Example 3

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

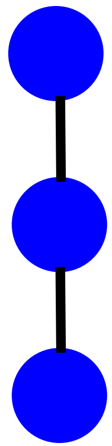


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$

Example 3

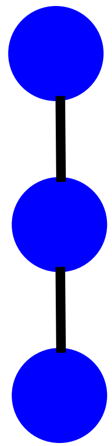
$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$
1	$n/4$		



Example 3

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

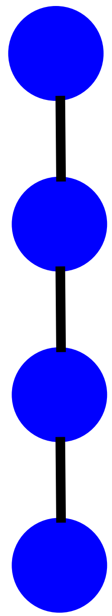


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$
1	$n/4$	$n/4$	$n/4$

Example 3

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

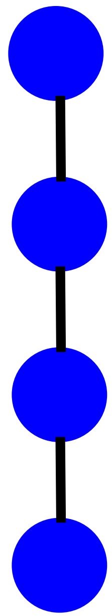
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$
1	$n/4$	$n/4$	$n/4$
1	$n/8$		



Example 3

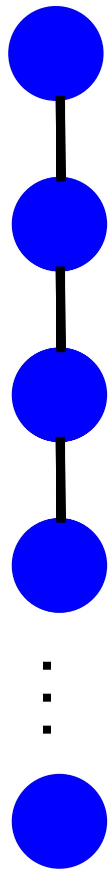
$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$
1	$n/4$	$n/4$	$n/4$
1	$n/8$	$n/8$	$n/8$



Example 3

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

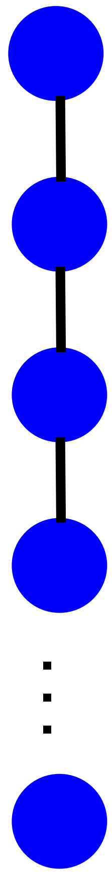


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$
1	$n/4$	$n/4$	$n/4$
1	$n/8$	$n/8$	$n/8$
$\vdots$	$\vdots$		
1	1		

Example 3

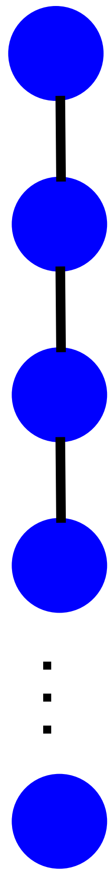
$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$
1	$n/4$	$n/4$	$n/4$
1	$n/8$	$n/8$	$n/8$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	1	1



Example 3

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
1	$n/2$	$n/2$	$n/2$
1	$n/4$	$n/4$	$n/4$
1	$n/8$	$n/8$	$n/8$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	1	1

$(1 + \log_2 n)$ levels. Total work $= n + \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1$

Total amount of work:

Total amount of work:

$$n + \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1$$

Total amount of work:

$$n + \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1$$

$$= n\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \left(\frac{1}{2}\right)^{\log n}\right)$$

Total amount of work:

$$n + \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1$$

$$= n\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \left(\frac{1}{2}\right)^{\log n}\right)$$

$$= n\sum_{i=0}^{\log_2 n}\left(\frac{1}{2}\right)^{i}$$

Total amount of work:

$$
\begin{aligned}
n + \frac{n}{2} + \frac{n}{4} + \cdots + 2 + 1 \\
&= n\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \left(\frac{1}{2}\right)^{\log n}\right) \\
&= n \sum_{i=0}^{\log_2 n} \left(\frac{1}{2}\right)^{i}
\end{aligned}
$$

Theorem 4.4 tells us that the value of the geometric series is $O(1)$ (in fact it is ≤ 2) so, the total amount of work done is $O(n)$.

Example 4

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$

Example 4

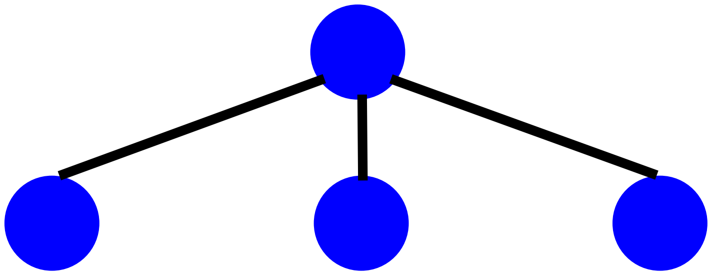
assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$

Example 4

assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$



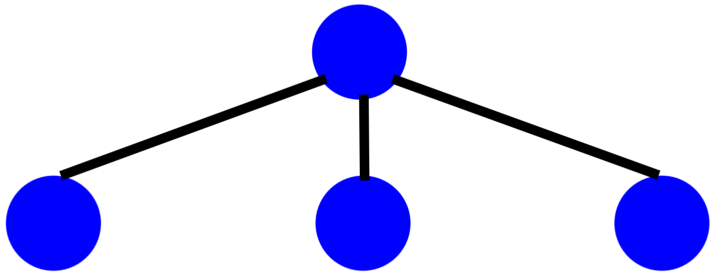
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
3	$n/3$		

Example 4

assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$

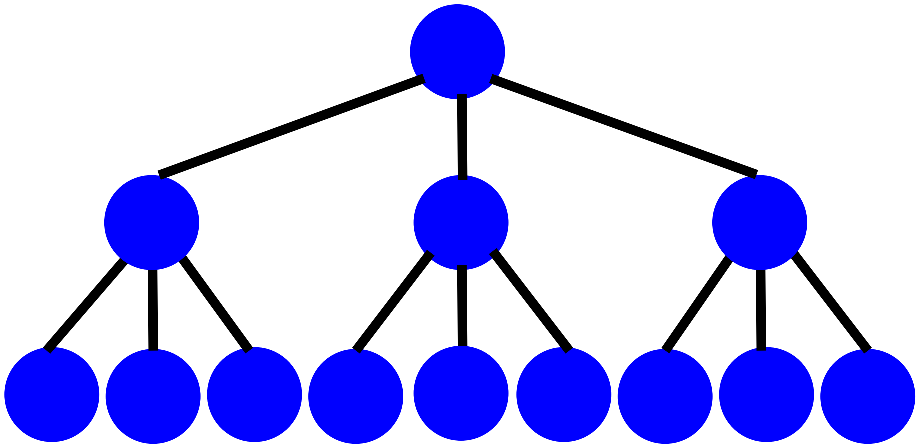
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
3	$n/3$	$n/3$	$3(n/3) = n$



Example 4

assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$

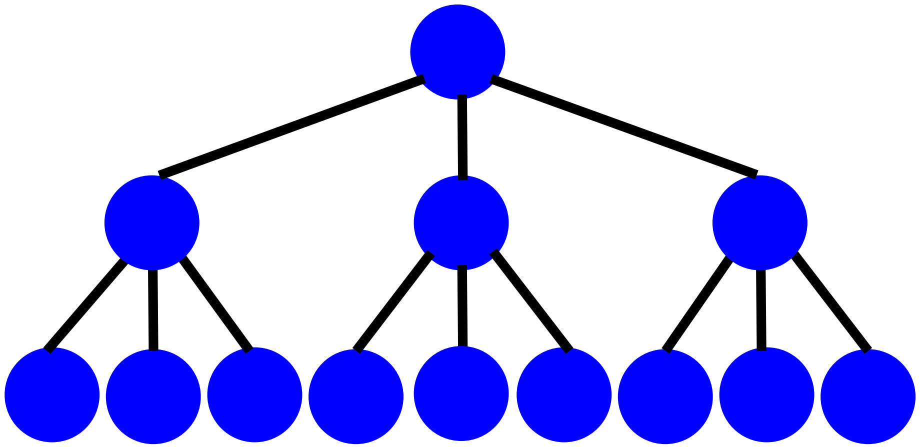


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
3	$n/3$	$n/3$	$3(n/3) = n$
$3^2 = 9$	$n/9$		

Example 4

assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$

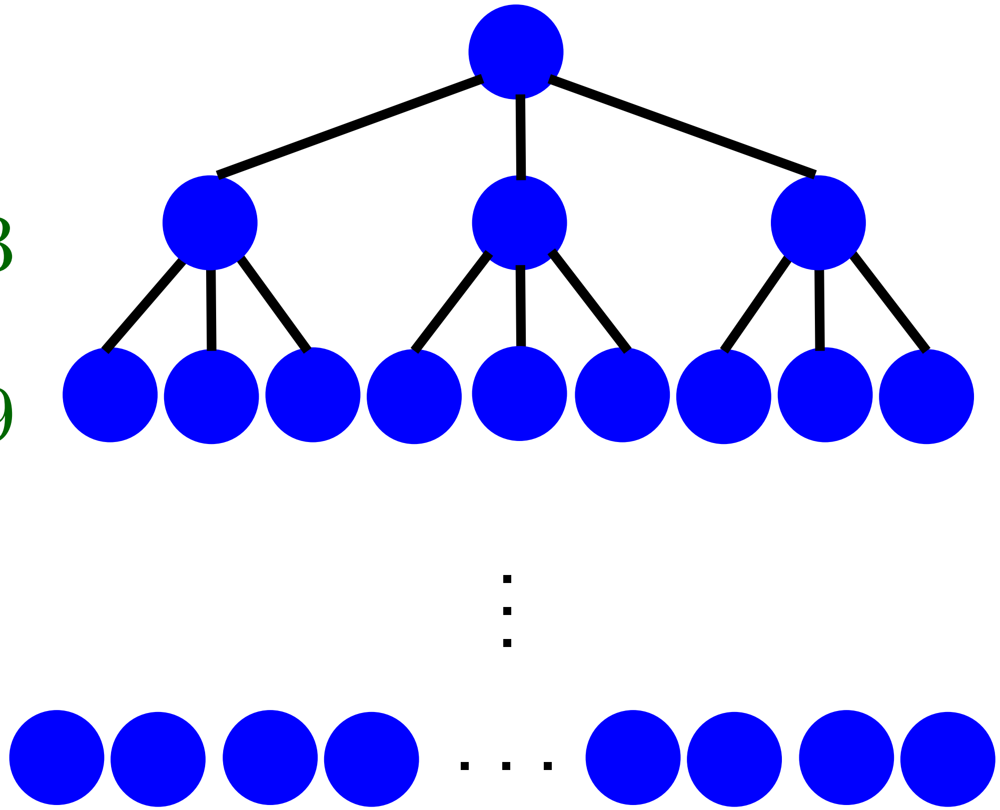


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
3	$n/3$	$n/3$	$3(n/3) = n$
$3^2 = 9$	$n/9$	$n/9$	$9(n/9) = n$

Example 4

assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$



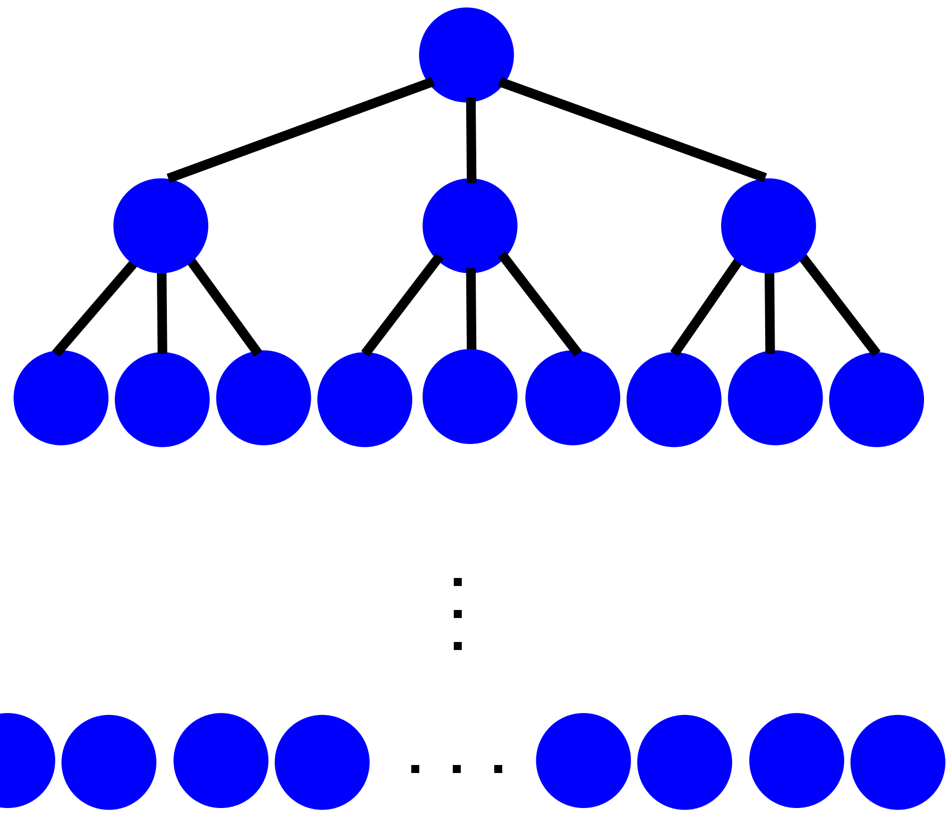
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
3	$n/3$	$n/3$	$3(n/3) = n$
$3^2 = 9$	$n/9$	$n/9$	$9(n/9) = n$
$\vdots$	$\vdots$		
$3^{\log_3 n} = n$	1		

Example 4

assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$

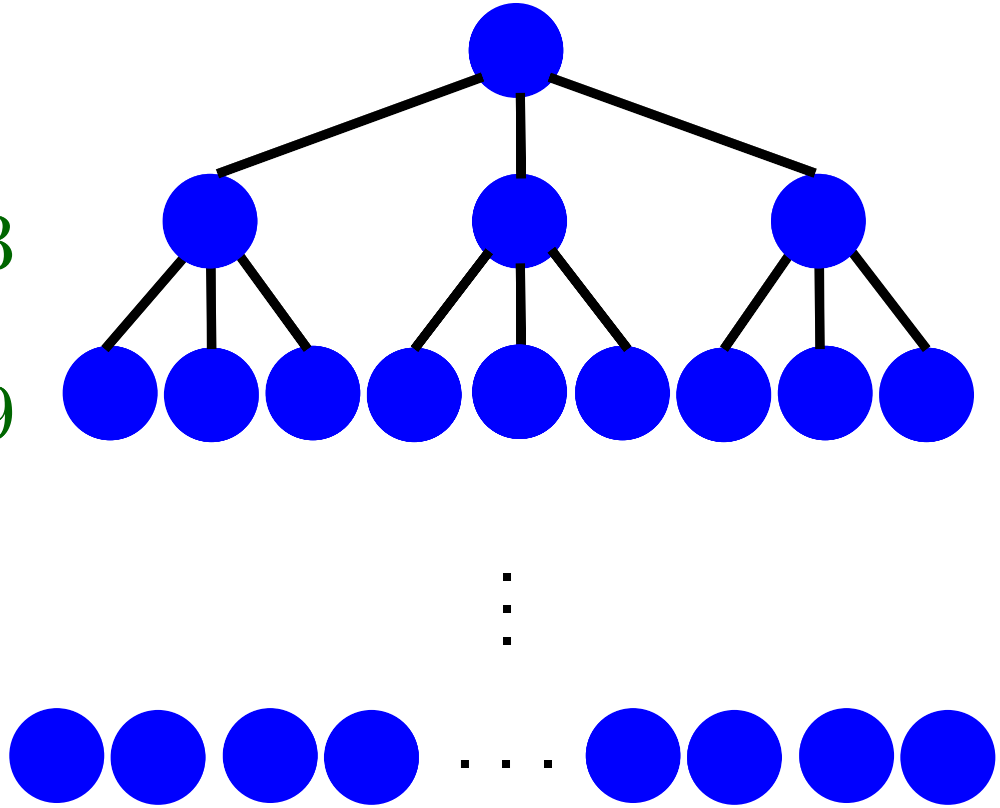
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
3	$n/3$	$n/3$	$3(n/3) = n$
$3^2 = 9$	$n/9$	$n/9$	$9(n/9) = n$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$3^{\log_3 n} = n$	1	1	$n(1) = n$



Example 4

assume n *is power of* 3

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3, \\ 1 & \text{if } n < 3. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
3	$n/3$	$n/3$	$3(n/3) = n$
$3^2 = 9$	$n/9$	$n/9$	$9(n/9) = n$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$3^{\log_3 n} = n$	1	1	$n(1) = n$

$(1 + \log_3 n)$ levels $\Rightarrow$ total work $= n(1 + \log_3 n)$

Example 5

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

Example 5

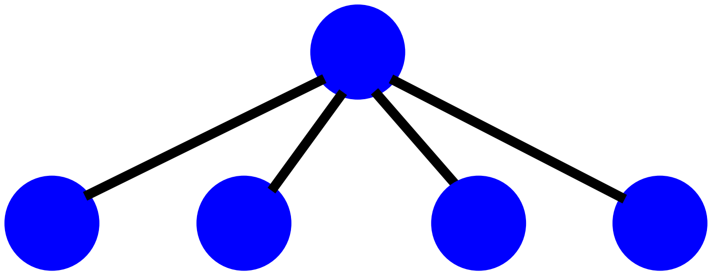
$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level

Example 5

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

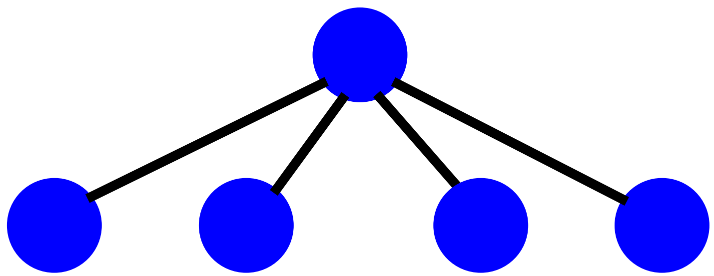
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
4	$n/2$		



Example 5

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

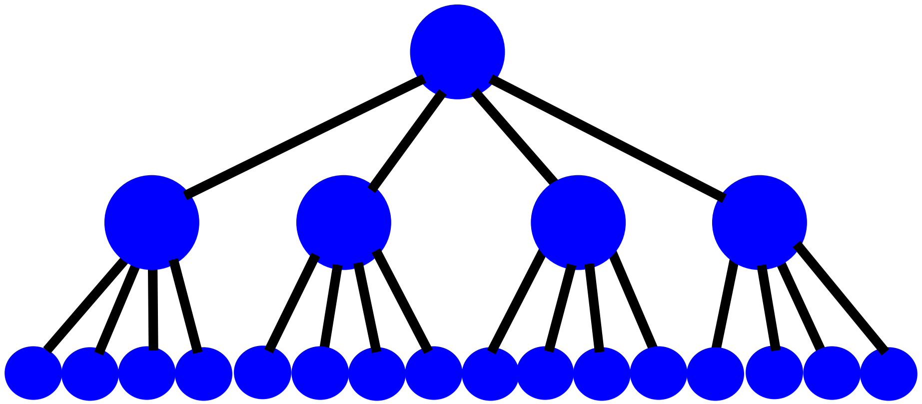
# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
4	$n/2$	$n/2$	$4(n/2) = 2n$



Example 5

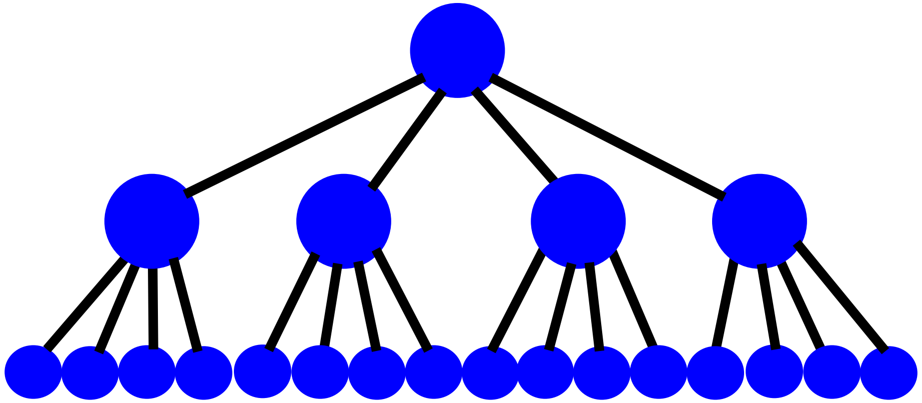
$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
4	$n/2$	$n/2$	$4(n/2) = 2n$
$4^2 = 16$	$n/4$		



Example 5

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

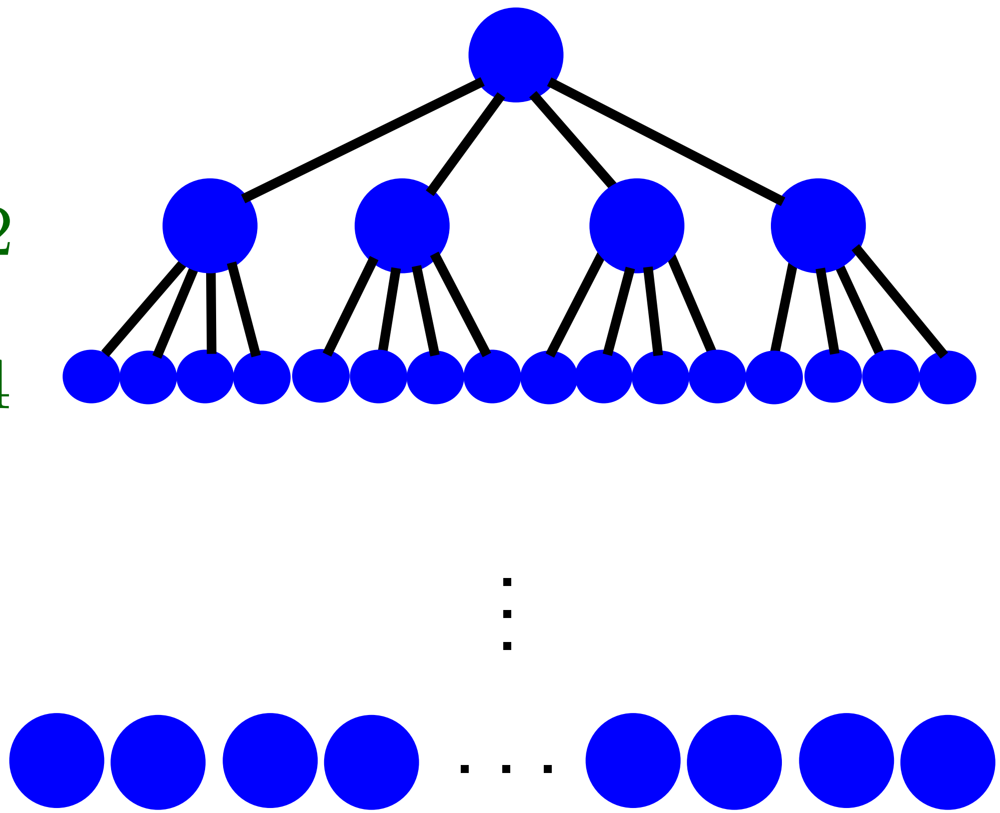


# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
4	$n/2$	$n/2$	$4(n/2) = 2n$
$4^2 = 16$	$n/4$	$n/4$	$16(n/4) = 4n$

Example 5

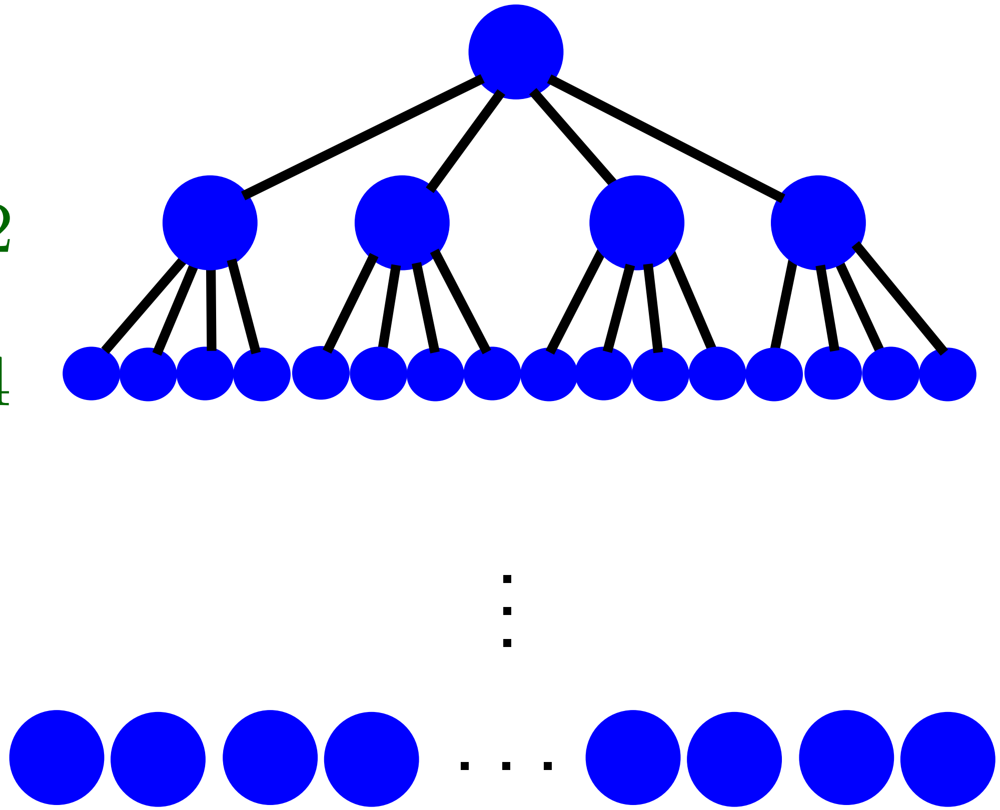
$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
4	$n/2$	$n/2$	$4(n/2) = 2n$
$4^2 = 16$	$n/4$	$n/4$	$16(n/4) = 4n$
$\vdots$	$\vdots$		
$4^{\log_2 n} = n^2$	1		



Example 5

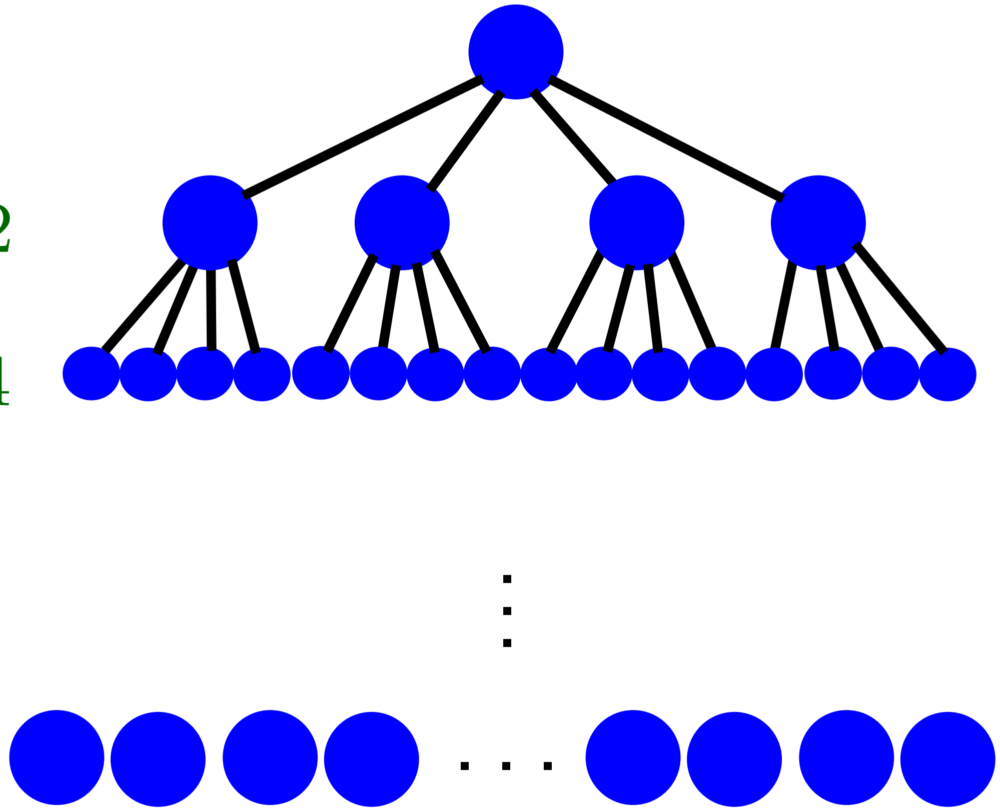
$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
4	$n/2$	$n/2$	$4(n/2) = 2n$
$4^2 = 16$	$n/4$	$n/4$	$16(n/4) = 4n$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$4^{\log_2 n} = n^2$	1	1	$n^2 \cdot 1 = n^2$

Example 5

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$



# of Probl	Probl Size	Work per Probl	Work per Level
1	n	n	n
4	$n/2$	$n/2$	$4(n/2) = 2n$
$4^2 = 16$	$n/4$	$n/4$	$16(n/4) = 4n$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$4^{\log_2 n} = n^2$	1	1	$n^2 \cdot 1 = n^2$

total work $= n + 2n + 4n + \cdots + 2^{\log_2 n} n$

Total work is

$$n + 2n + 4n + \cdots + 2^{\log_2 n} n$$

Total work is

$$n + 2n + 4n + \cdots + 2^{\log_2 n} n$$

$$= n \left(1 + 2 + 4 + \cdots 2^{\log_2 n}\right)$$

Total work is

$$
\begin{aligned}
n + 2n + 4n + \cdots + 2^{\log_2 n} n \\
&= n\left(1 + 2 + 4 + \cdots 2^{\log_2 n}\right) \\
&= n \sum_{i=0}^{\log_2 n} 2^i
\end{aligned}
$$

Total work is

$$n + 2n + 4n + \cdots + 2^{\log_2 n} n$$

$$= n\left(1 + 2 + 4 + \cdots 2^{\log_2 n}\right)$$

$$= n \sum_{i=0}^{\log_2 n} 2^i$$

$$= n\left(2^{1+\log_2 n} - 1\right)$$

Total work is

$$
\begin{aligned}
n + 2n + 4n + \cdots + 2^{\log_2 n} n \\
&= n\left(1 + 2 + 4 + \cdots 2^{\log_2 n}\right) \\
&= n \sum_{i=0}^{\log_2 n} 2^i \\
&= n\left(2^{1+\log_2 n} - 1\right) \\
&= 2n^2 - n
\end{aligned}
$$

Growth Rates of Solutions to Recurrences

- Divide and Conquer Algorithms
- Recursion Trees
- Three Different Behaviors

Three Different Behaviors

Three Different Behaviors

Compare the recursion tree diagrams for the recurrences

Three Different Behaviors

Compare the recursion tree diagrams for the recurrences

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

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

$$T(n) = 4T(n/2) + n$$

Three Different Behaviors

Compare the recursion tree diagrams for the recurrences

$$T(n) = 2T(n/2) + n$$
$$T(n) = T(n/2) + n$$
$$T(n) = 4T(n/2) + n$$

- all three trees have depth $1 + \log_2 n$

Three Different Behaviors

Compare the recursion tree diagrams for the recurrences

$$T(n) = 2T(n/2) + n$$
$$T(n) = T(n/2) + n$$
$$T(n) = 4T(n/2) + n$$

- all three trees have depth $1 + \log_2 n$
- in each case, the size of each subproblem is half the size of the parent problem

Three Different Behaviors

Compare the recursion tree diagrams for the recurrences

$$T(n) = 2T(n/2) + n$$
$$T(n) = T(n/2) + n$$
$$T(n) = 4T(n/2) + n$$

- all three trees have depth $1 + \log_2 n$

- in each case, the size of each subproblem is
  half the size of the parent problem

- differ in the amount of work done per level

Lemma 4.7:
Suppose that we have a recurrence of the form

$$T(n) = aT\left(\frac{n}{2}\right) + n,$$

where a is a positive integer and $T(1)$ is nonnegative.
Then we have the following big Θ bounds on the solution:

1. If $a < 2$, then $T(n) = \Theta(n)$.

2. If $a = 2$, then $T(n) = \Theta(n \log n)$.

3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$.

Lemma 4.7:

Suppose that we have a recurrence of the form

$$T(n) = aT\left(\frac{n}{2}\right) + n,$$

where a is a positive integer and $T(1)$ is nonnegative.

Then we have the following big Θ bounds on the solution:

1. If $a < 2$, then $T(n) = \Theta(n)$.
2. If $a = 2$, then $T(n) = \Theta(n \log n)$.
3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$.

Proof:

We already proved Case 1 when $a = 1$ in Example 2.

Lemma 4.7:

Suppose that we have a recurrence of the form

$$T(n) = aT\left(\frac{n}{2}\right) + n,$$

where a is a positive integer and $T(1)$ is nonnegative.

Then we have the following big Θ bounds on the solution:

1. If $a < 2$, then $T(n) = \Theta(n)$.
2. If $a = 2$, then $T(n) = \Theta(n \log n)$.
3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$.

Proof:

We already proved Case 1 when $a = 1$ in Example 2.

(will not prove it for $1 < a < 2$)

Lemma 4.7:
Suppose that we have a recurrence of the form

$$T(n) = aT\left(\frac{n}{2}\right) + n,$$

where a is a positive integer and $T(1)$ is nonnegative.
Then we have the following big Θ bounds on the solution:

1. If $a < 2$, then $T(n) = \Theta(n)$.
2. If $a = 2$, then $T(n) = \Theta(n \log n)$.
3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$.

Proof:

We already proved Case 1 when $a = 1$ in Example 2.
(will not prove it for $1 < a < 2$*)*

We already proved Case 2 in Example 1.

Lemma 4.7:
Suppose that we have a recurrence of the form

$$T(n) = aT\left(\frac{n}{2}\right) + n,$$

where a is a positive integer and $T(1)$ is nonnegative.
Then we have the following big Θ bounds on the solution:

1. If $a < 2$, then $T(n) = \Theta(n)$.
2. If $a = 2$, then $T(n) = \Theta(n \log n)$.
3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$.

Proof:

We already proved Case 1 when $a = 1$ in Example 2.
(will not prove it for $1 < a < 2$)

We already proved Case 2 in Example 1.

We will now prove Case 3.

$T(n) = aT\left(\frac{n}{2}\right) + n$ where $a > 2$. Assume n is a power of a.

$T(n) = aT\left(\frac{n}{2}\right) + n$ where $a > 2$. Assume n is a power of a.

At Level i, there are a^i nodes,
each corresponding to a problem of size $n/2^i$.

$T(n) = aT\left(\frac{n}{2}\right) + n$ where $a > 2$. Assume n is a power of a.

At Level i, there are a^i nodes,
each corresponding to a problem of size $n/2^i$.

$\Rightarrow$ Total work at nonbottom Level i is $a^i(n/2^i) = n(a/2)^i$.

$T(n) = aT\left(\frac{n}{2}\right) + n$ where $a > 2$. Assume n is a power of a.

At Level i, there are a^i nodes,
each corresponding to a problem of size $n/2^i$.

$\Rightarrow$ Total work at nonbottom Level i is $a^i(n/2^i) = n(a/2)^i$.

Summing over the $1 + \log_2 n$ levels, we get

$$a^{\log_2 n} T(1) \quad + \quad n \sum_{i=0}^{(\log_2 n)-1} \left(\frac{a}{2}\right)^i .$$

$T(n) = aT\left(\frac{n}{2}\right) + n$ where $a > 2$. Assume n is a power of a.

At Level i, there are a^i nodes,
each corresponding to a problem of size $n/2^i$.

$\Rightarrow$ Total work at nonbottom Level i is $a^i(n/2^i) = n(a/2)^i$.

Summing over the $1 + \log_2 n$ levels, we get

$$\underbrace{a^{\log_2 n} T(1)}_{\text{Work at bottom level}} \quad + \quad \underbrace{n \sum_{i=0}^{(\log_2 n)-1} \left(\frac{a}{2}\right)^i}_{\text{Work on non-bottom levels}}.$$

Total work is

$$a^{\log_2 n} T(1) + n \sum_{i=0}^{(\log_2 n)-1} \left(\frac{a}{2}\right)^i.$$

This sum is a geometric series.

Because $a/2 \neq 1$, Theorem 4.4 tells us that the sum will be big Θ of the largest term.

Because $a > 2$, the largest term in this case is clearly the last one, namely, $n(a/2)^{(\log_2 n)-1}$.

n times the largest term in the geometric series is

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}}$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{n}$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{n} = \frac{2}{a}\cdot a^{\log_2 n}$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{n} = \frac{2}{a} \cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{n} = \frac{2}{a}\cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

so the total work done is

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{n} = \frac{2}{a} \cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

so the total work done is

$$a^{\log_2 n} T(1)$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{n} = \frac{2}{a} \cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

so the total work done is

$$\underbrace{a^{\log_2 n} T(1)}_{\Theta\left(n^{\log_2 a}\right)}$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{n} = \frac{2}{a}\cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

so the total work done is

$$\underbrace{a^{\log_2 n}T(1)}_{\Theta\left(n^{\log_2 a}\right)} + n\sum_{i=0}^{(\log_2 n)-1}\left(\frac{a}{2}\right)^i$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{n} = \frac{2}{a} \cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

so the total work done is

$$\underbrace{a^{\log_2 n} T(1)}_{\Theta\left(n^{\log_2 a}\right)} + \underbrace{n \sum_{i=0}^{(\log_2 n)-1} \left(\frac{a}{2}\right)^i}_{\Theta\left(n^{\log_2 a}\right)}$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{n} = \frac{2}{a}\cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

so the total work done is

$$\underbrace{a^{\log_2 n}T(1)}_{\Theta\left(n^{\log_2 a}\right)} + \underbrace{n\sum_{i=0}^{(\log_2 n)-1}\left(\frac{a}{2}\right)^i}_{\Theta\left(n^{\log_2 a}\right)} = \Theta\left(n^{\log_2 a}\right)$$

n times the largest term in the geometric series is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a}\cdot\frac{n\cdot a^{\log_2 n}}{n} = \frac{2}{a}\cdot a^{\log_2 n}$$

Now notice that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

so the total work done is

$$\underbrace{a^{\log_2 n}T(1)}_{\Theta\left(n^{\log_2 a}\right)} + \underbrace{n\sum_{i=0}^{(\log_2 n)-1}\left(\frac{a}{2}\right)^i}_{\Theta\left(n^{\log_2 a}\right)} = \Theta\left(n^{\log_2 a}\right)$$

and we are done!

As an example of Case 3 consider

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

As an example of Case 3 consider

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

$a = 4$ so the Theorem says that

$$T(n) = \Theta\left(n^{\log_2 a}\right) = \Theta\left(n^{\log_2 4}\right) = \Theta(n^2)$$

As an example of Case 3 consider

$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

$a = 4$ so the Theorem says that

$$T(n) = \Theta\left(n^{\log_2 a}\right) = \Theta\left(n^{\log_2 4}\right) = \Theta(n^2)$$

This matches with the exact answer of $2n^2 - n$, which we already derived in Example 5.