# Recurrence Relations

- Can easily describe the runtime of recursive algorithms
- Can then be expressed in a closed form (not defined in terms of itself)

- Consider the linear search:

# Linear Search

- Recursively
- Look at an element (constant work, c), then search the remaining elements…

- T(n) = T( n-1 ) + c
- “The cost of searching n elements is the cost of looking at 1 element, plus the cost of searching n-1 elements”

# list of intermediates

| Result at $i^{th}$ unwinding | i |
|---|---|
| $T(n) = T(n-1) + 1c$ | 1 |
| $T(n) = T(n-2) + 2c$ | 2 |
| $T(n) = T(n-3) + 3c$ | 3 |
| $T(n) = T(n-4) + 4c$ | 4 |

# Linear Search (cont.)

- An expression for the kth unwinding:

  $T(n) = T(n-k) + kc$

# Linear Search (cont.)

- Let's decide to stop at T(0). When the list to search is empty, you're done…
- 0 is convenient, in this example…

  Let n-k = 0 => n=k

- Now, substitute n in everywhere for k:

  $T(n) = T(n-n) + nc$

  $T(n) = T(0) + nc = nc + c_0 = O(n)$

  ( T(0) is some constant, $c_0$ )

# Recurrence Relation

An expression which is defined in terms of **itself** (1) using **smaller** (2) **inputs**

$$T(n) = T(n-1) + 1 \ , T(1) = 1$$

$$T(n) = nT(n-1); \quad T(1) = 1$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n \ ; \quad T(2) = 1$$

Base Case

# Recurrence Relation

$$T(n) = T(n-1) + 1 \ , T(1) = 1$$

Base Case

If n =5 then

T(5) = T(4) + 1

T(4) = T(3) + 1

T(3) = T(2) + 1

T(2) = T(1) + 1

T(5) = T(4) + 1

= T(3) + 1 + 1

= T(2) + 1 + 1 +1

= T(1) + 1 + 1+ 1 +1

= 1 + 1 + 1+ 1 +1

=5

# Recurrence Relation

$$T(n) = nT(n-1)\ , T(1) = 1$$

Base Case

If n =5 then

T(5) = 5T(4)

T(4) = 4T(3)

T(3) = 3T(2)

T(2) = 2T(1)

T(5) = 5T(4)

= 5*4T(3)

= 5*4*3T(2)

= 5*4*3*2T(1)

=5*4*3*2*1

=5! = 120

# Recurrence Relation

$$T(n) = 2T(n/2) + n\,, T(1) = 1$$

Base Case

If n =16 then

T(16) = 2T(8) + 16

T(8) = 2T(4) + 8

T(4) = 2T(2) + 4

T(2) = 2T(1) + 2

T(16) = 2T(8) + 16

= 2[2T(4) + 8] +16

= $2^2$T(4) + 16 + 16

= $2^2$[2T(2) + 4] + 16 + 16

= $2^3$T(2) + 16 + 16 + 16

= $2^3$[2T(1) + 2] + 16 + 16 + 16

= $2^4$T(1) + 16 + 16 + 16 + 16

= 16 + 16 + 16 + 16 + 16

= 80

# Designing Recurrence Relation

**Problem :** Let us consider a piece of paper with 1 unit of thickness. If we fold it once the thickness is 2 units. If we fold it twice its thickness is 4 units. Design a recurrence relation to compute the thickness of paper after folding it n times.


folding 42 times will bring you to the Moon.

# Designing Recurrence Relation

**Problem :** Let us consider a piece of paper with 1 unit of thickness. If we fold it once the thickness is 2 units. If we fold it twice its thickness is 4 units. Design a recurrence relation to compute the thickness of paper after folding it n times.

**Solution:**

Let T(n) = The thickness of paper after folding n times

$T(1) = 2 = 2^1$ (Base Case)

$T(2) = 4 = 2 * 2 = 2T(1)$

$T(3) = 8 = 2 * 4 = 2T(2)$

.

.

.

.

$T(n) = 2T(n-1)$

$$T(n) = 2T(n-1) ; T(1) = 2$$

# Recurrence Relation for Bubble Sort

```
void bubbleSort(int *arr, int n)
{
  // TERMINATING CONDITION OF RECURSION
  if(n <= 1){ return; }
  // MOVE THE LARGEST ELEMENT AT THE END
  for(int j=0; j arr[j+1])
      swap(&arr[j], &arr[j+1]);
  }
  // CALL FUN RECURSIVELY TO SORT FIRST n-1
ELEMENTS
  bubbleSort(arr, n-1);
}
```

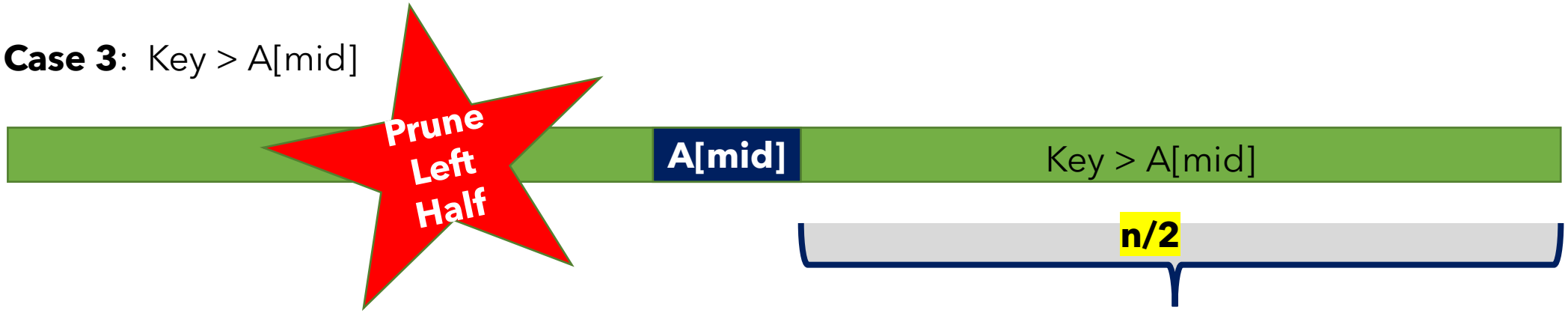
$$T(n) = T(n-1) + n; T(1) = 1$$

# Recurrence Relation for Binary Search

**Case 1**: Key = A[mid]


**Case 2**: Key < A[mid]


**Case 3**: Key > A[mid]




$$T(n) = T(n/2) + k; T(1) = 1$$

# Recurrence Relation for Merge Sort

*Alg.:* MERGE-SORT(A, p, r)

```
if p < r
  then q ← ⌊(p + r)/2⌋
       MERGE-SORT(A, p, q)
       MERGE-SORT(A, q + 1, r)
       MERGE(A, p, q, r)
```

**T(n)** = Time to sort n data items

**T(n/2)** = Time to sort left half (n/2 data items)

**T(n/2)** = Time to sort right half (n/2 data items)

**O(n)** = Time to merge left and right sorted half

Initial call: MERGE-SORT(A, 1, n)

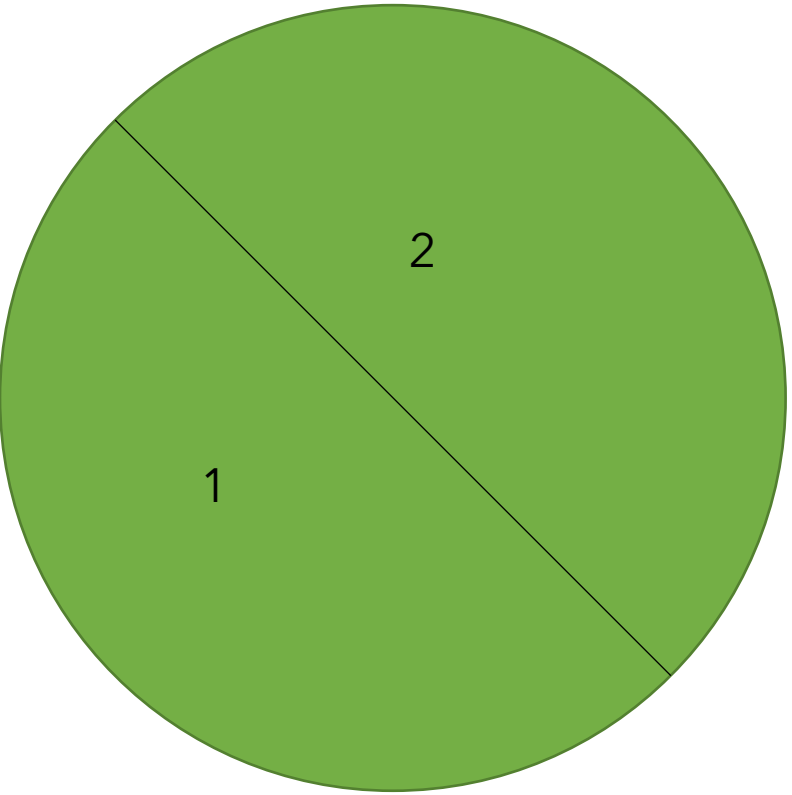
**T(n) = T(n/2) + T(n/2) + O(n)**

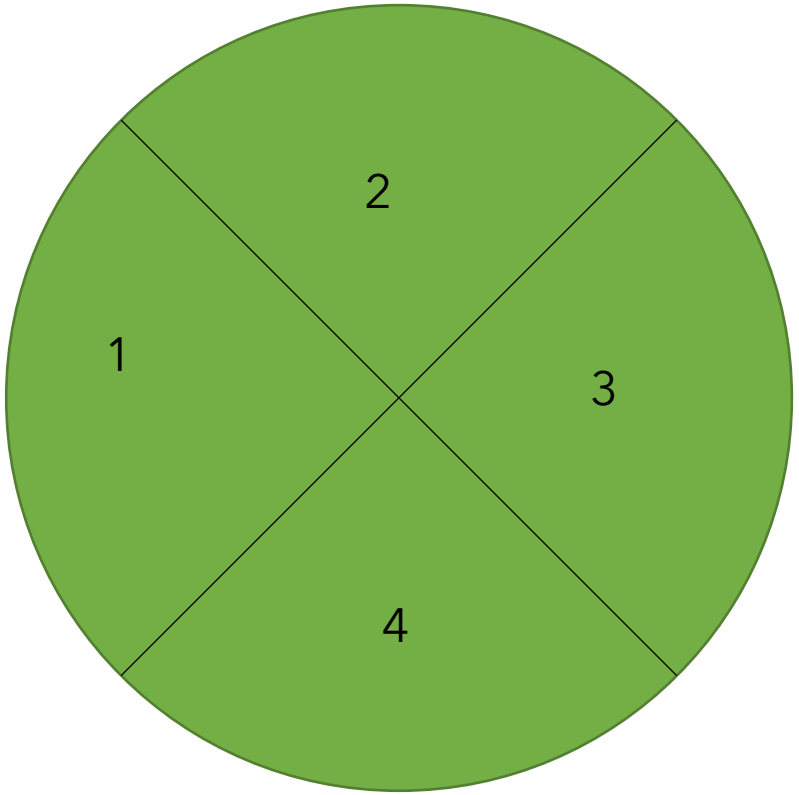
**T(n) = 2T(n/2) + O(n) ; T(1) =1**

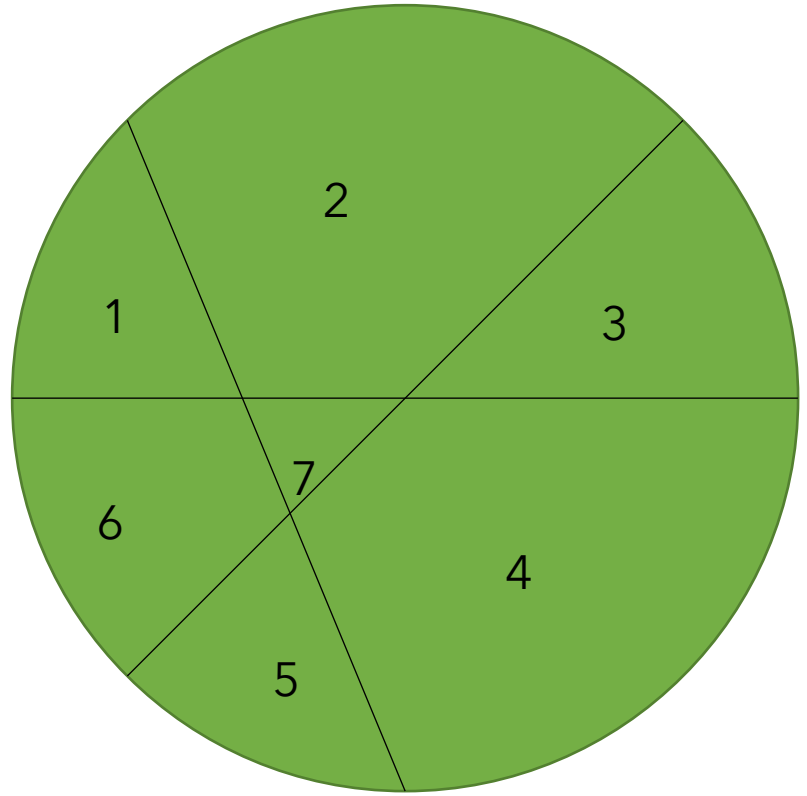
# Designing Recurrence Relation

**Problem :** **If we draw a chord in the circle maximum number of region we obtain is 2. Drawing 2 chord divides the circle into maximum 4 regions and so on. Design a recurrence relation to compute maximum number of regions in a circle after drawing n chords.**

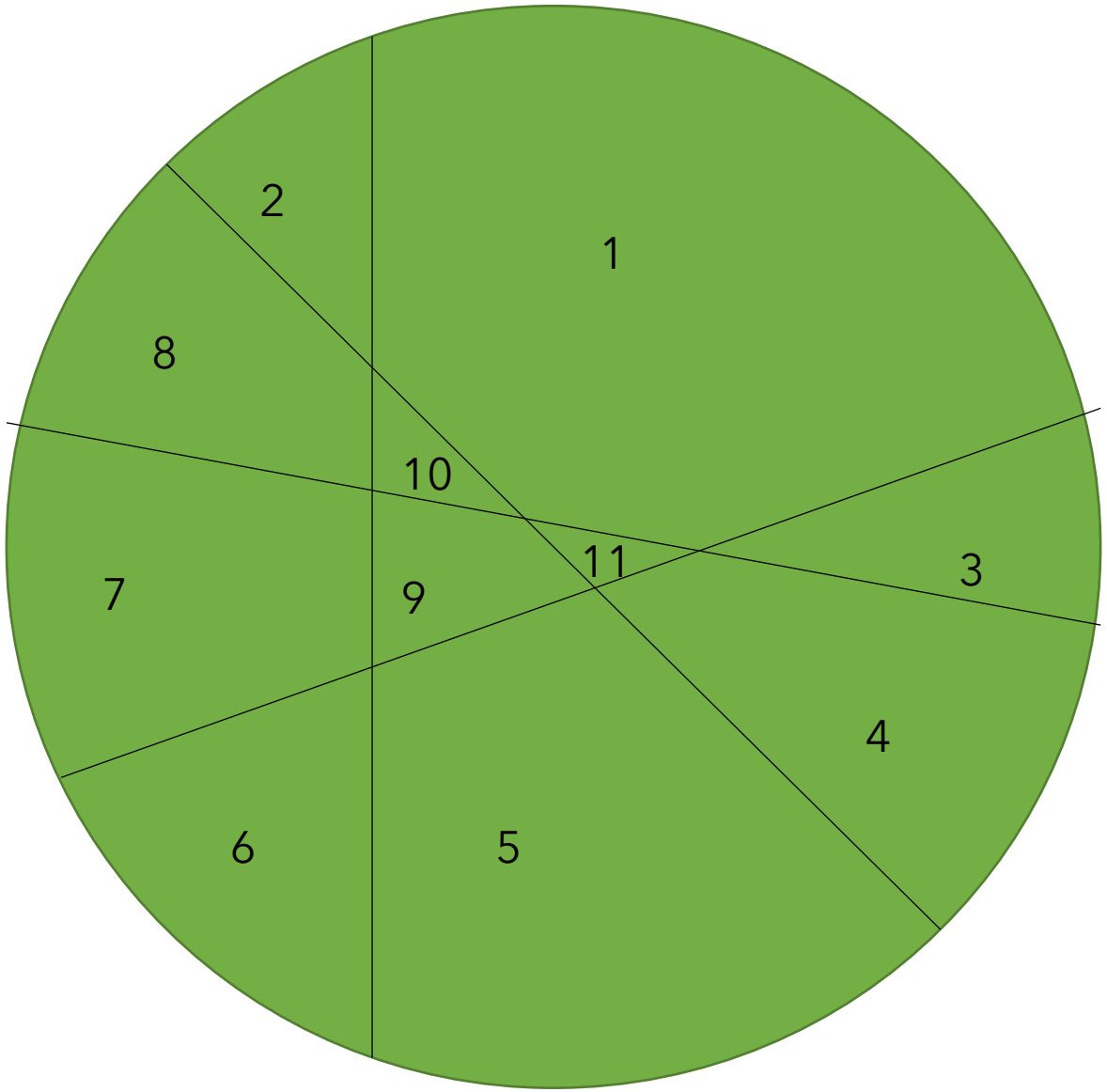
**Solution:**



# Recurrence Relation



# Designing Recurrence Relation

**T(n)** = **Max number of regions** obtained if we draw **n chords** in a circle

**Base Case**: T(1) = 2

**Recursive Case:**

**T(2) = 4 = T(1) + 2**
**T(3) = 7 = T(2) + 3**
**T(4) = 11 = T(3) + 4**
**.....**
**T(n) = T(n-1) + n**

$$T(n) = T(n-1) + n \; ; \; T(1) = 2$$

# Solving Recurrence Relation

1. Backward Substitution
2. Recursion tree method
3. Master Method

# Backward Substitution Method

**Problem :** Let us consider a piece of paper with 1 unit of thickness. If we fold it once the thickness is 2 units. If we fold it twice its thickness is 4 units. Design a recurrence relation to compute the thickness of paper after folding it n times.

**Solution:** Let T(n) = The thickness of the paper after folding n times

$T(n) = 2T(n-1)\ ;\ T(1) = 2$

$= 2[2T(n-2)]$ as $T(n-1) = 2T(n-2)$

$=2^2T(n-2)$

$=2^2[2T(n-3)]$ as $T(n-2) = 2T(n-3)$

$=2^3T(n-3)$

.

.

.

$=2^{?}T(1)$

$1 = n - (n-1)$

$=2^{n-1}T(n-(n-1))$

$=2^{n-1}T(1) = 2^{n-1}2 = 2^n = O(2^n)$

# Backward Substitution Method

**T(n)** = The **thickness** of the paper after folding **n times** = $2^n$


**T(42)** = The **thickness** of the paper after folding **42 times** = $2^{42}$ = **4 398 046.51** km

# Backward Substitution Method

**Problem :** If we draw a chord in the circle maximum number of region we obtain is 2. Drawing 2 chord divides the circle into maximum 4 regions and so on. Design a recurrence relation to compute maximum number of regions in a circle after drawing n chords.

**Solution:**

$T(n) = T(n-1) + n \; ; \; T(1) = 2$

$= T(n-2) + (n-1) + n$ as $T(n-1) = T(n-2) + (n-1)$

$= T(n-3) + (n-2) + (n-1) + n$ as $T(n-2) = T(n-3) + (n-2)$

$= T(n-4) + (n-3) + (n-2) + (n-1) + n$ as $T(n-3) = T(n-4) + (n-3)$

.

.

.

$= T(1) + 2 + 3 + 4 + \ldots\ldots\ldots\ldots + n$

$= 2 + 2 + 3 + 4 + 5 + \ldots\ldots\ldots\ldots + n$

$= 1 + 1 + 2 + 3 + 4 + 5 + \ldots\ldots\ldots\ldots + n$

$= 1 + \{1 + 2 + 3 + 4 + 5 + \ldots\ldots\ldots\ldots + n\}$

$= 1 + \frac{n(n+1)}{2} = O(n^2)$

**T(n)** = Max number of regions obtained after drawing n chords = $1 + \frac{n(n+1)}{2}$

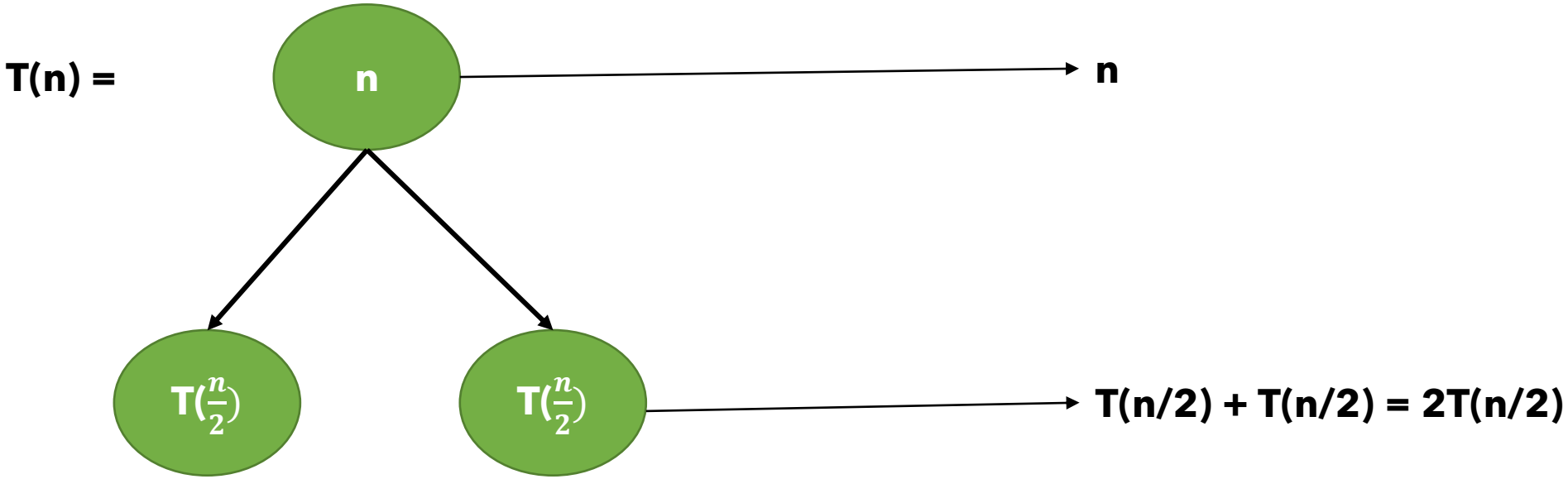
# Recursion Tree Method

**Steps:**

1. Draw the recursion Tree
2. Compute the cost at each level
3. Compute the overall cost

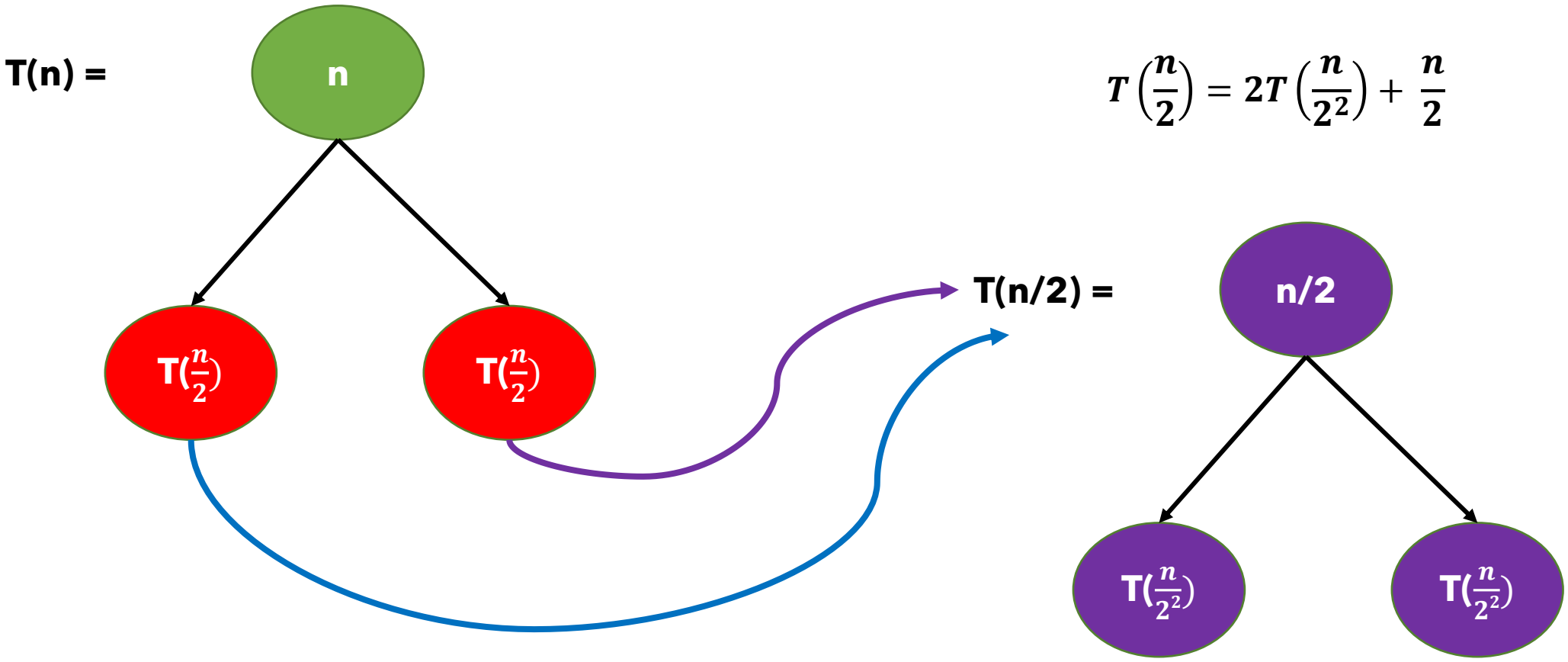
Represents the **solution** of recurrence relation

$$T(n) = 2T\left(\frac{n}{2}\right) + n; \qquad T(1) = 1$$

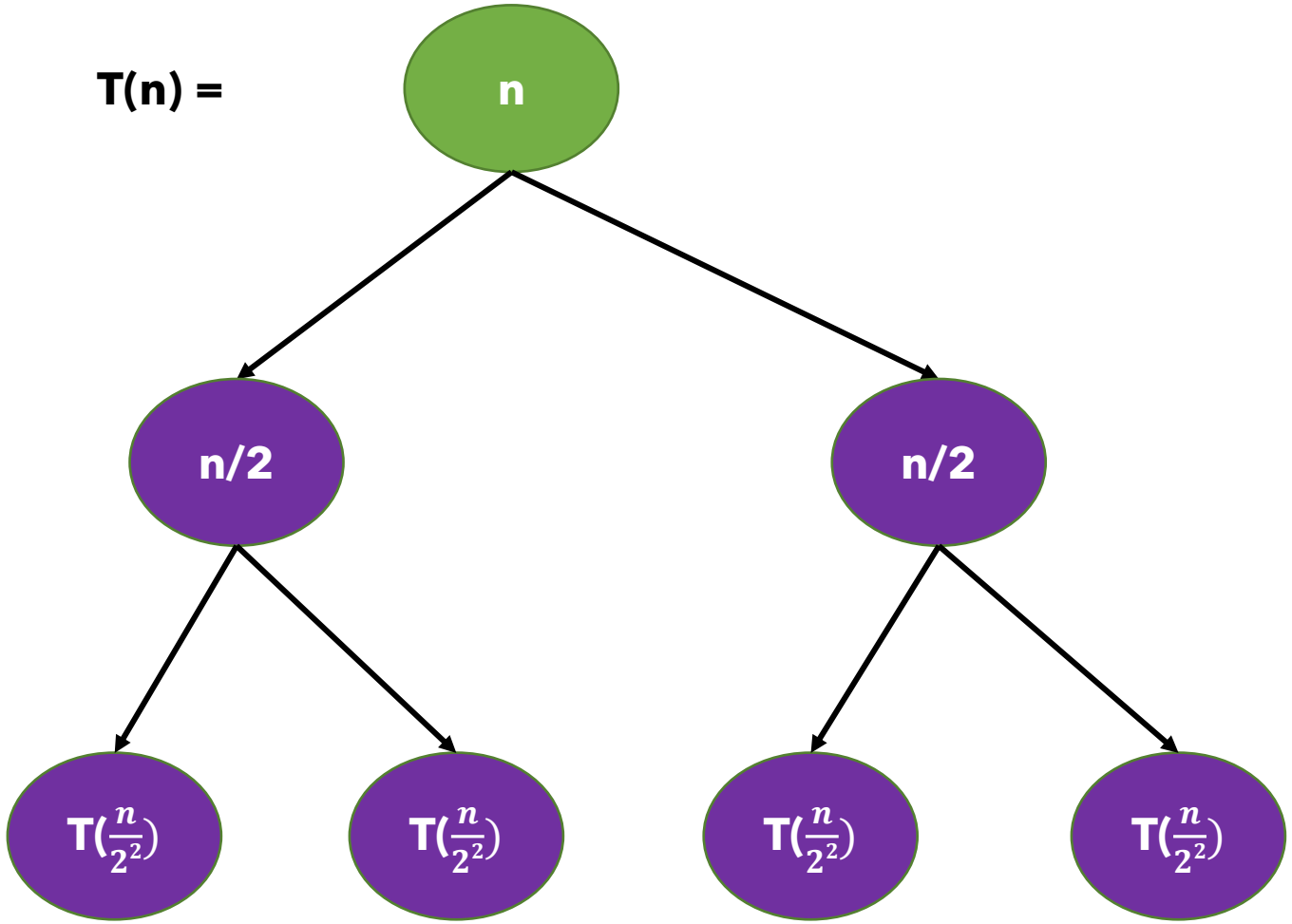



$$Total\ Cost = T(n) = 2T\left(\frac{n}{2}\right) + n$$

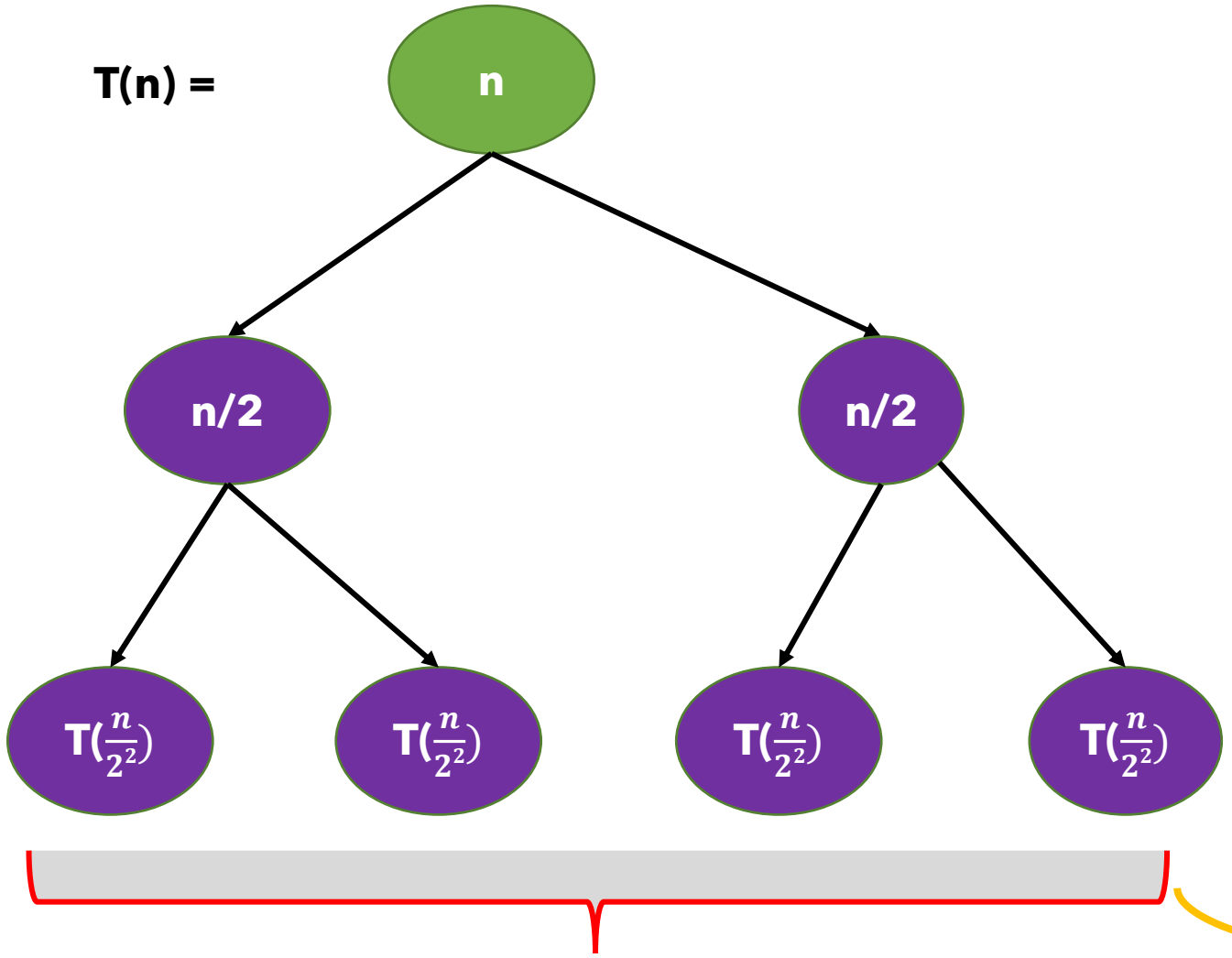
$$T(n) = 2T\left(\frac{n}{2}\right) + n; \qquad T(1) = 1$$



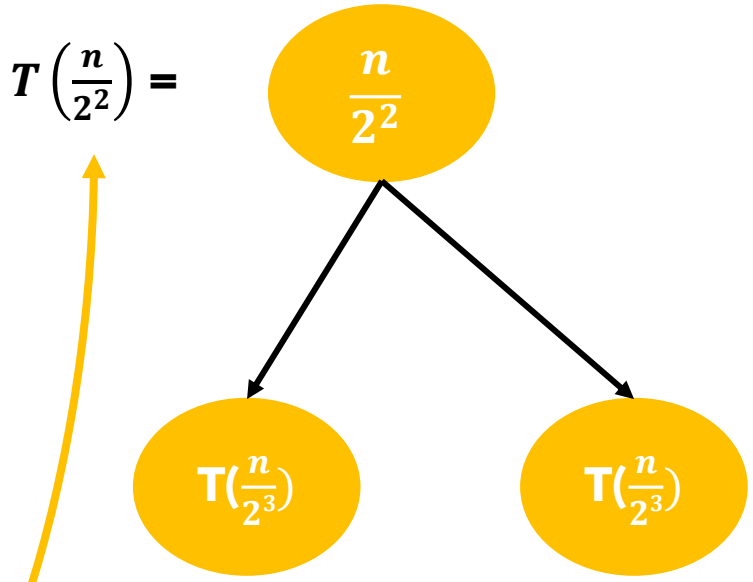
$$T(n) = 2T\left(\frac{n}{2}\right) + n; \qquad T(1) = 1$$

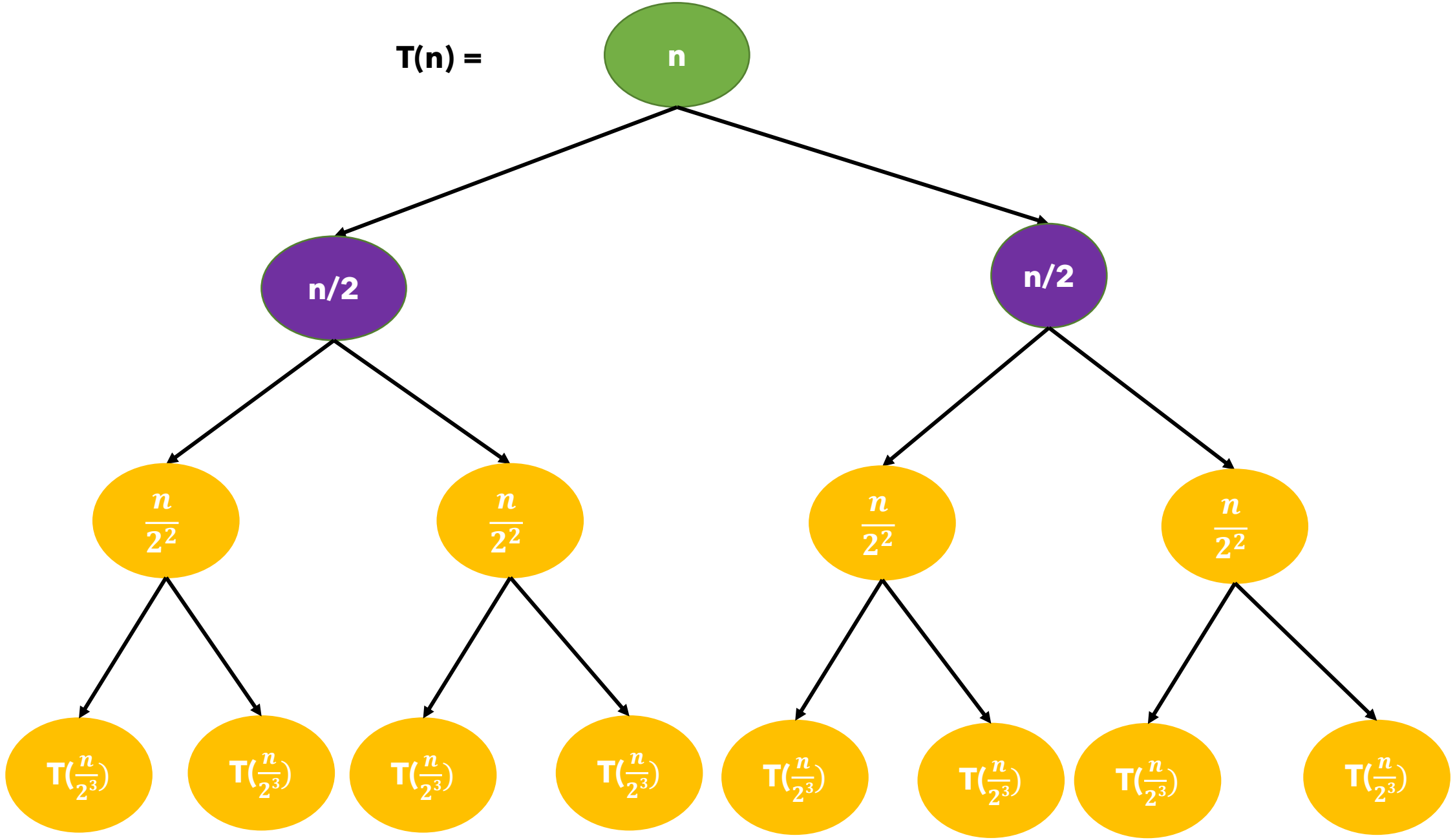


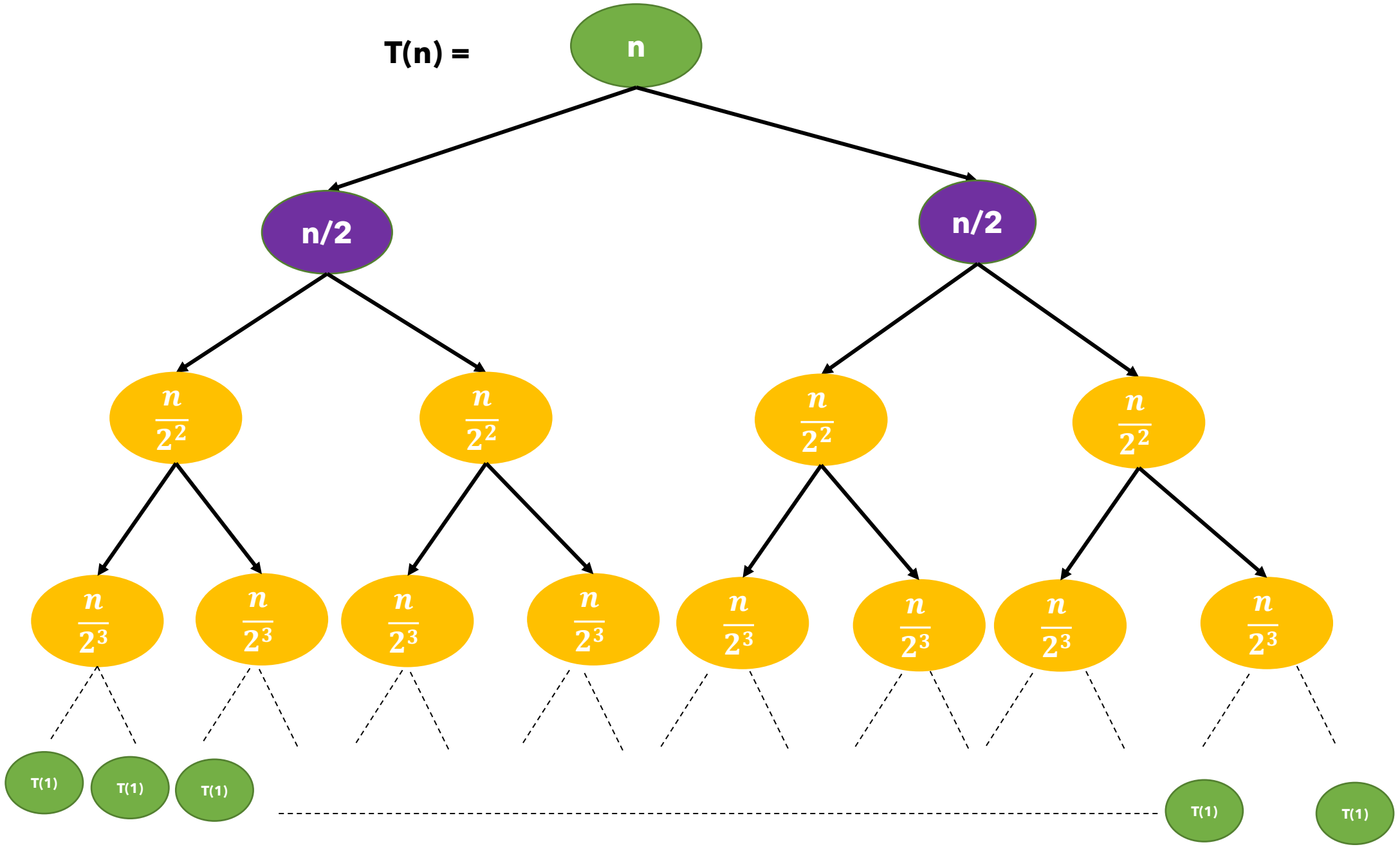
$$T(n) = 2T\left(\frac{n}{2}\right) + n; \qquad T(1) = 1$$

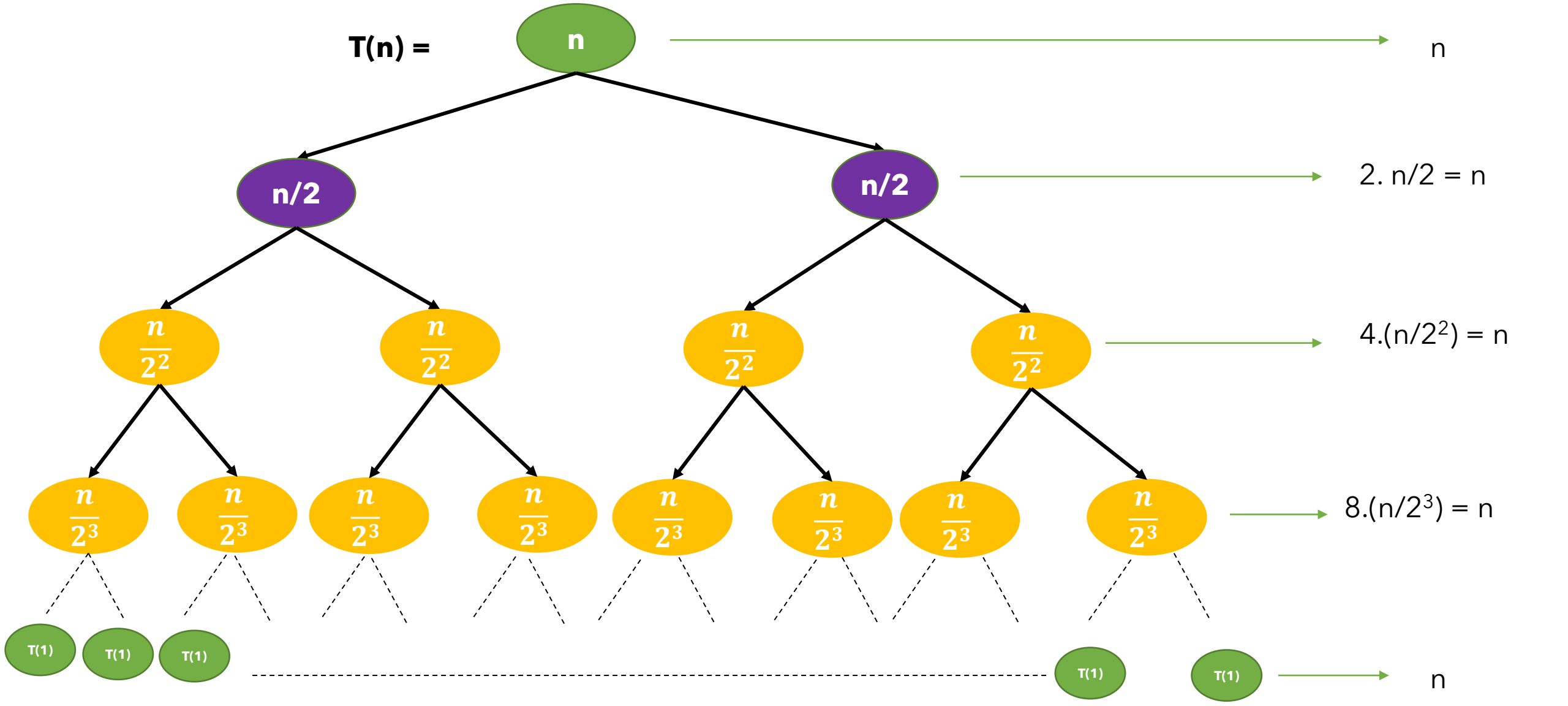



$$T\left(\frac{n}{2^2}\right) = 2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}$$



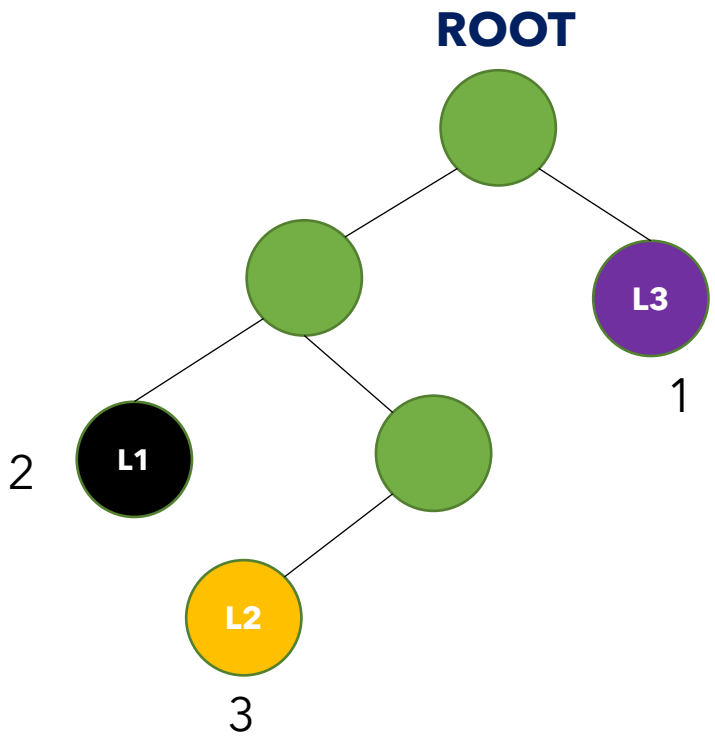

T(n) =
n
n/2
n/2
$\frac{n}{2^2}$
$\frac{n}{2^2}$
$\frac{n}{2^2}$
$\frac{n}{2^2}$
$T(\frac{n}{2^3})$
$T(\frac{n}{2^3})$
$T(\frac{n}{2^3})$
$T(\frac{n}{2^3})$
$T(\frac{n}{2^3})$
$T(\frac{n}{2^3})$
$T(\frac{n}{2^3})$
$T(\frac{n}{2^3})$


T(n) =
n
n/2
n/2
$\frac{n}{2^2}$
$\frac{n}{2^2}$
$\frac{n}{2^2}$
$\frac{n}{2^2}$
$\frac{n}{2^3}$
$\frac{n}{2^3}$
$\frac{n}{2^3}$
$\frac{n}{2^3}$
$\frac{n}{2^3}$
$\frac{n}{2^3}$
$\frac{n}{2^3}$
$\frac{n}{2^3}$
T(1)
T(1)
T(1)
T(1)
T(1)

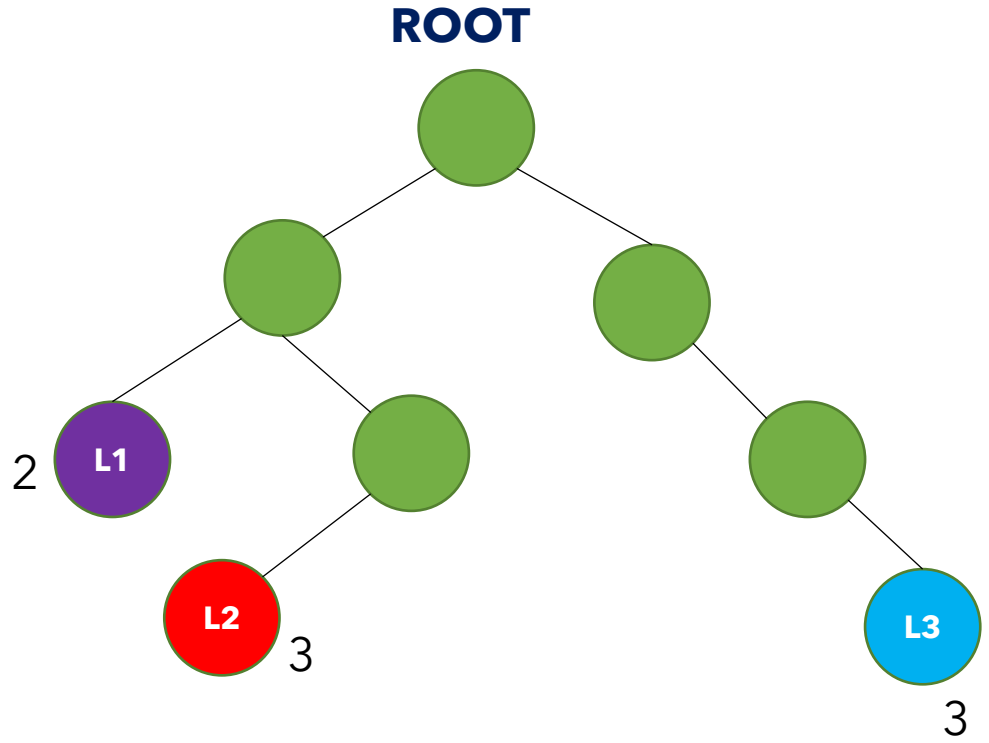



**Total Cost = T(n) = n + n + n + ..... + n ( Hight of tree times)**

# Height of the tree is maximum distance of root and leaf node in terms of number of edges




Height = 3

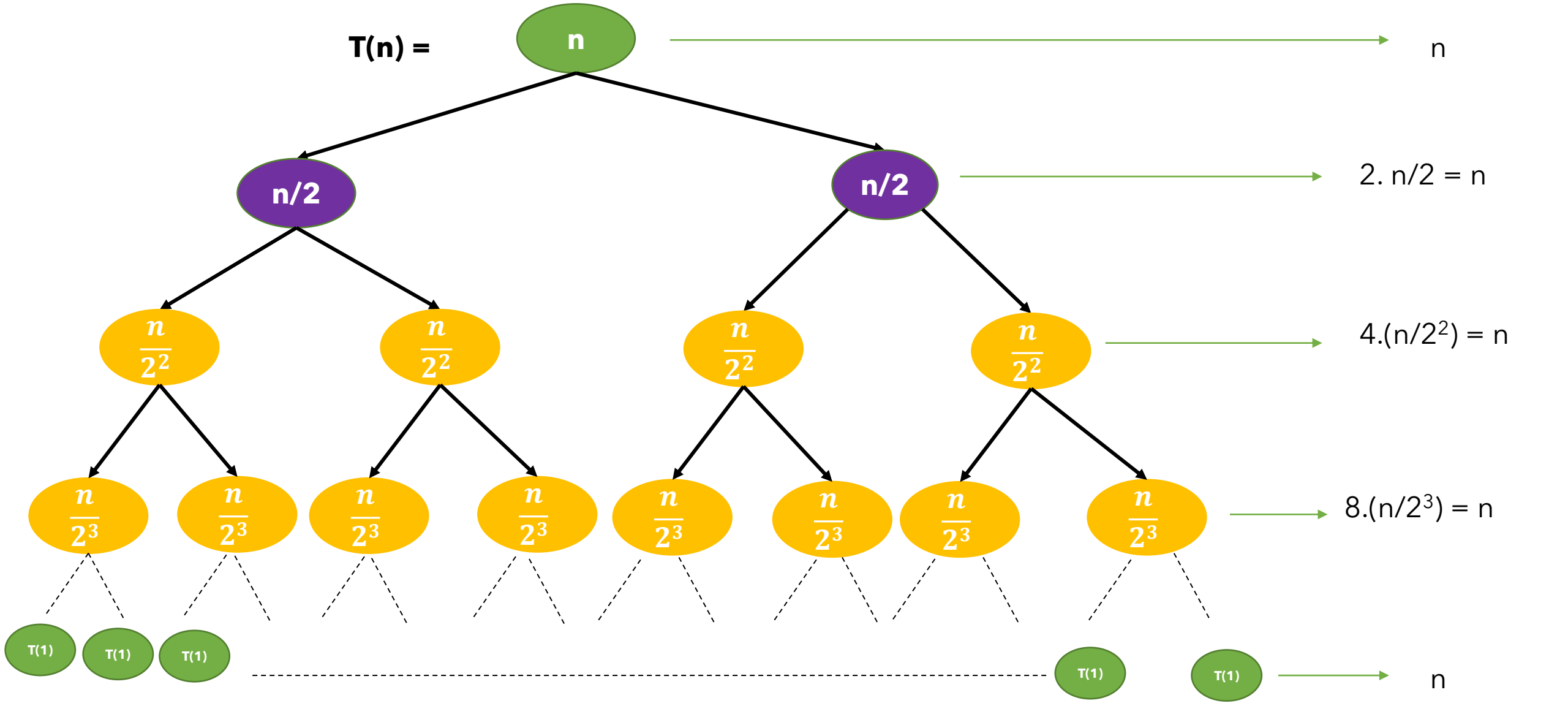



Height = 3

$$n \rightarrow \frac{n}{2} \rightarrow \frac{n}{2^2} \rightarrow \frac{n}{2^3} \rightarrow \cdots \ldots \ldots \ldots \rightarrow \frac{n}{2^i} = 1$$

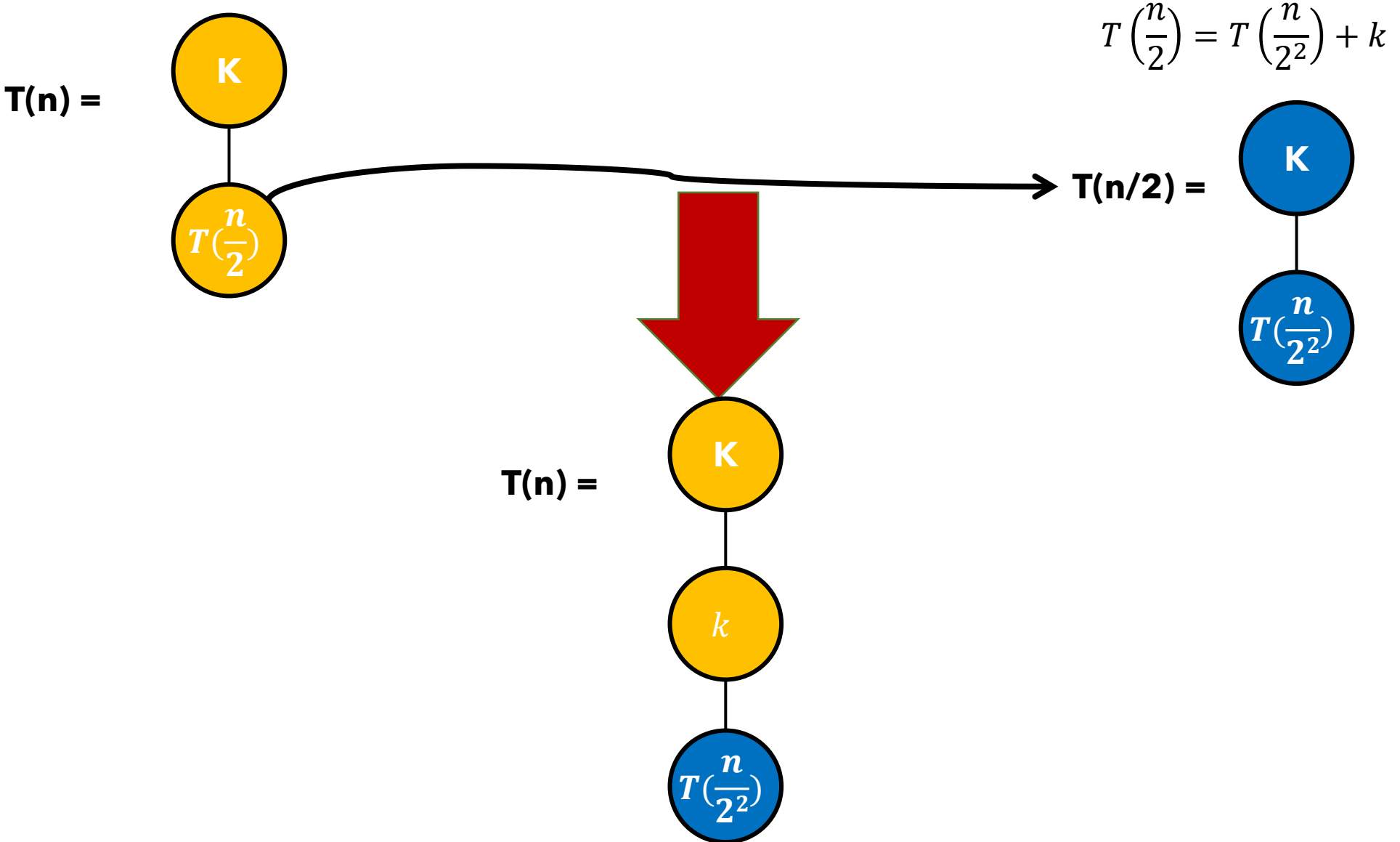
# i = Height of the recursion Tree

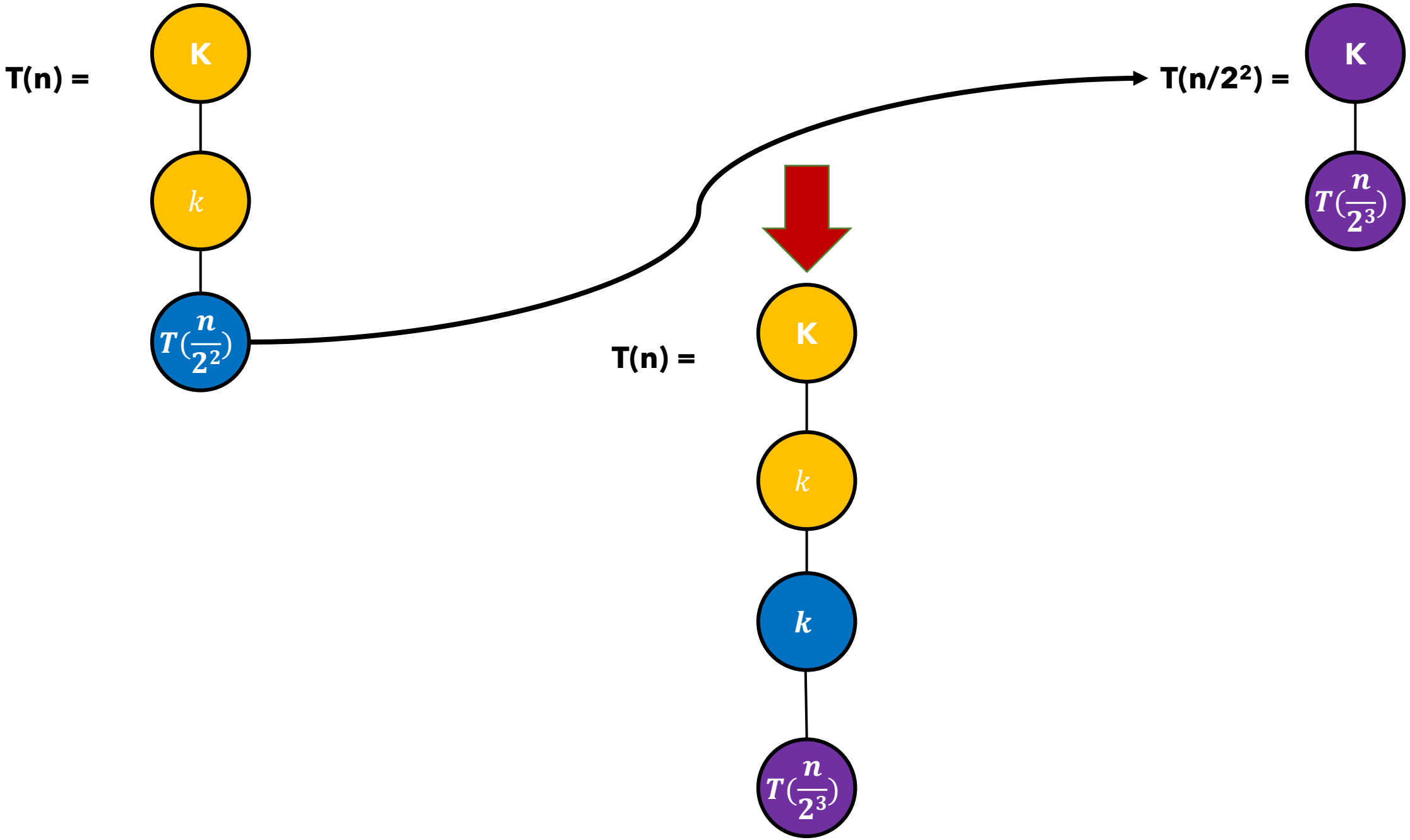
$$\frac{n}{2^i} = 1 \rightarrow n = 2^i \rightarrow i = \log_2 n$$



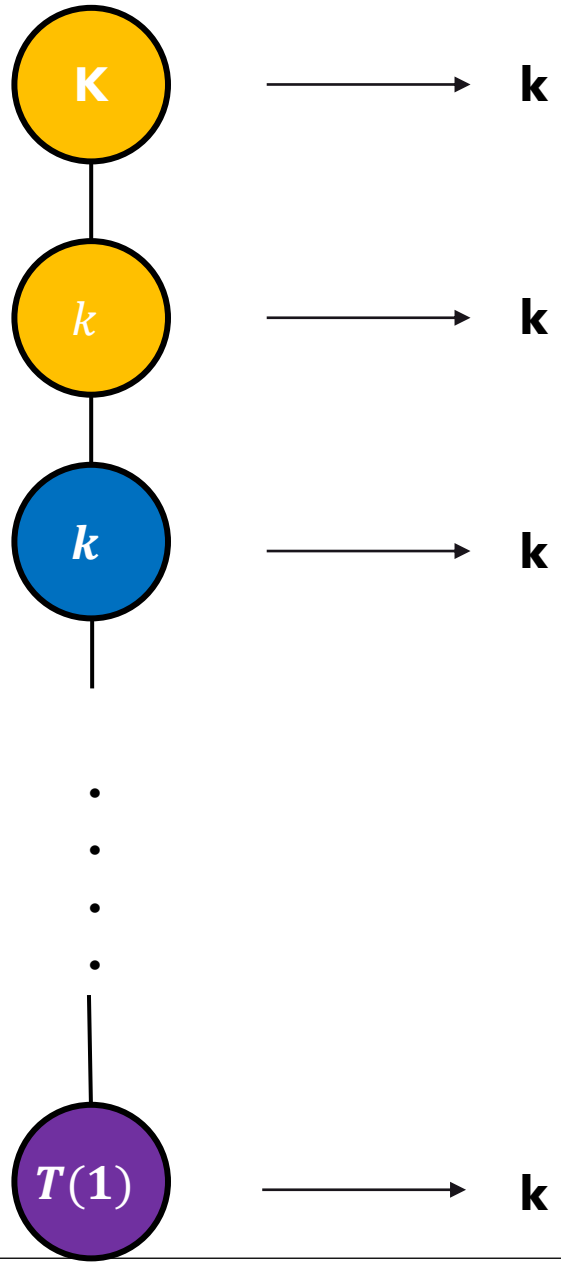

**Total Cost = T(n) = n + n + n + ..... + n ( logn times) = nlogn = O(nlogn)**

# $T(n) = T\left(\frac{n}{2}\right) + k\,, T(1) = 1\,, Where\ k\ is\ a\ constant,$




T(n) =
K
k
$T(\frac{n}{2^2})$
T(n/2²) =
K
$T(\frac{n}{2^3})$
T(n) =
K
k
k
$T(\frac{n}{2^3})$

T(n) =




k

k

k

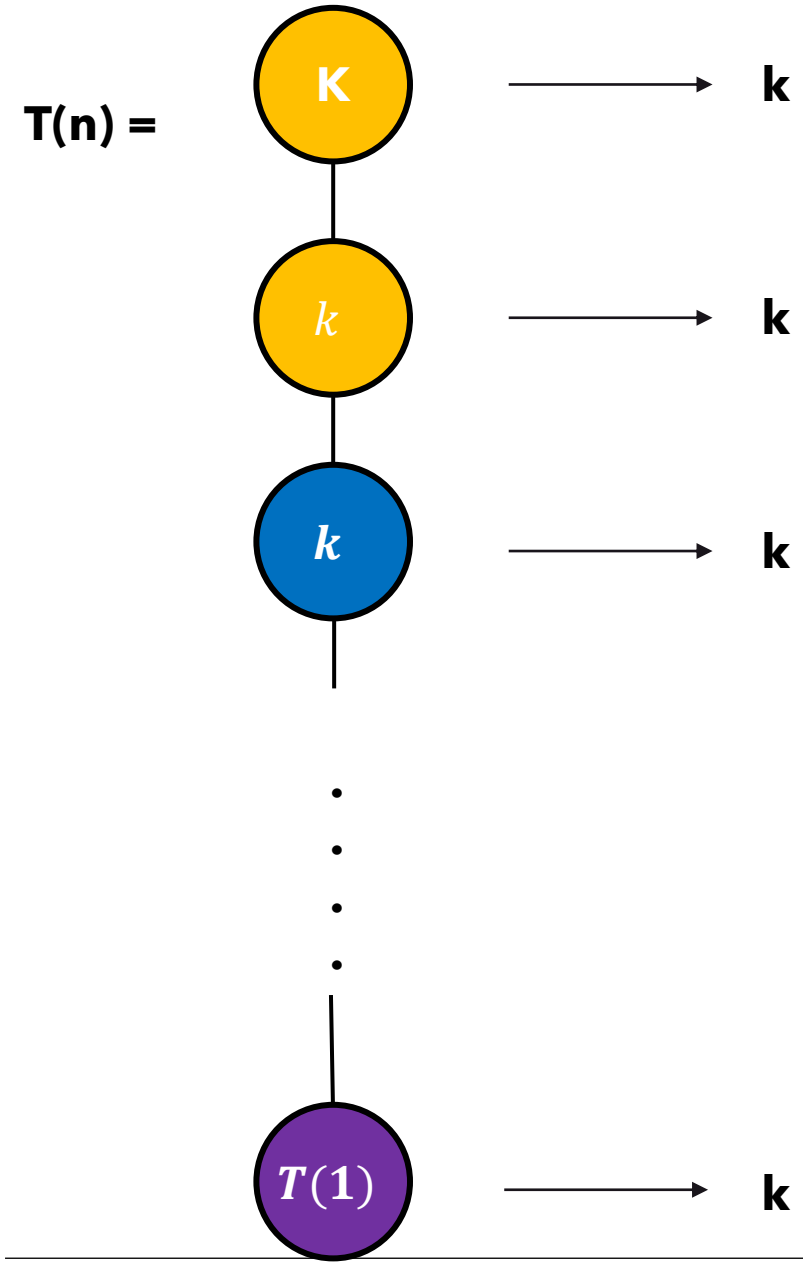
k

Total Cost = k + k + k+ …….+ k( height of the tree times)

$$n \rightarrow \frac{n}{2} \rightarrow \frac{n}{2^2} \rightarrow \frac{n}{2^3} \rightarrow \cdots \ldots \ldots \ldots \rightarrow \frac{n}{2^i} = 1$$

# i = Height of the recursion Tree

$$\frac{n}{2^i} = 1 \rightarrow n = 2^i \rightarrow i = \log_2 n$$




T(n) = Total Cost = k + k + k+ .....+ k( logn times) = klogn = O(logn)

# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

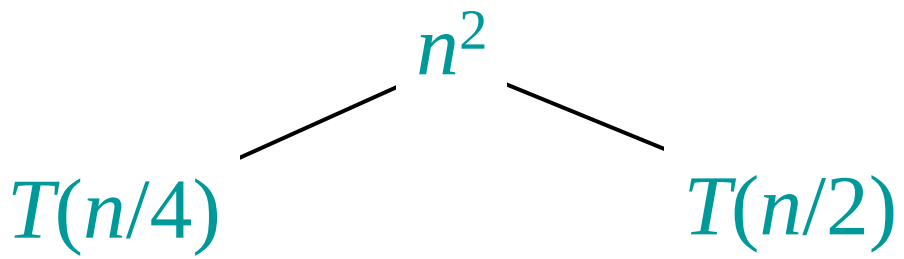
# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$T(n)$

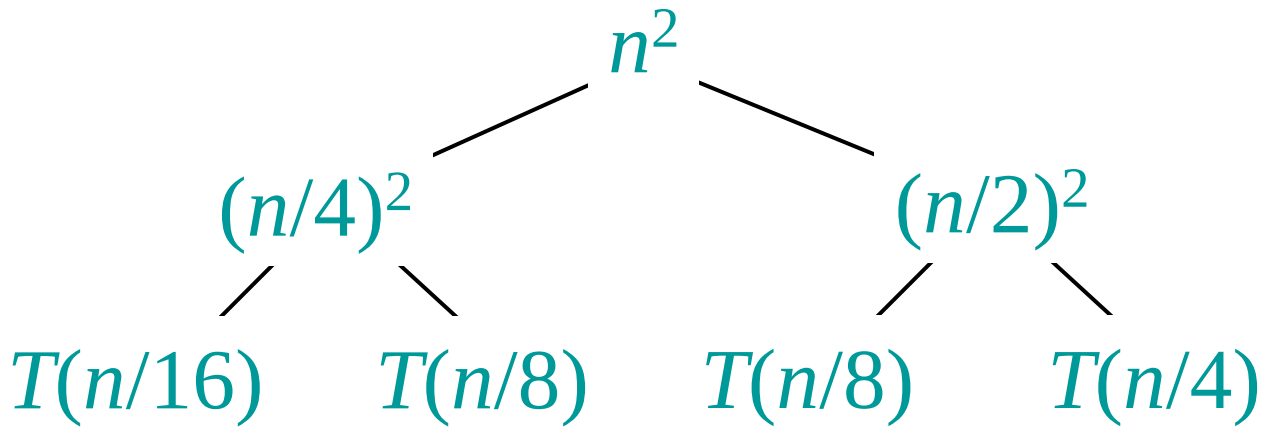
# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



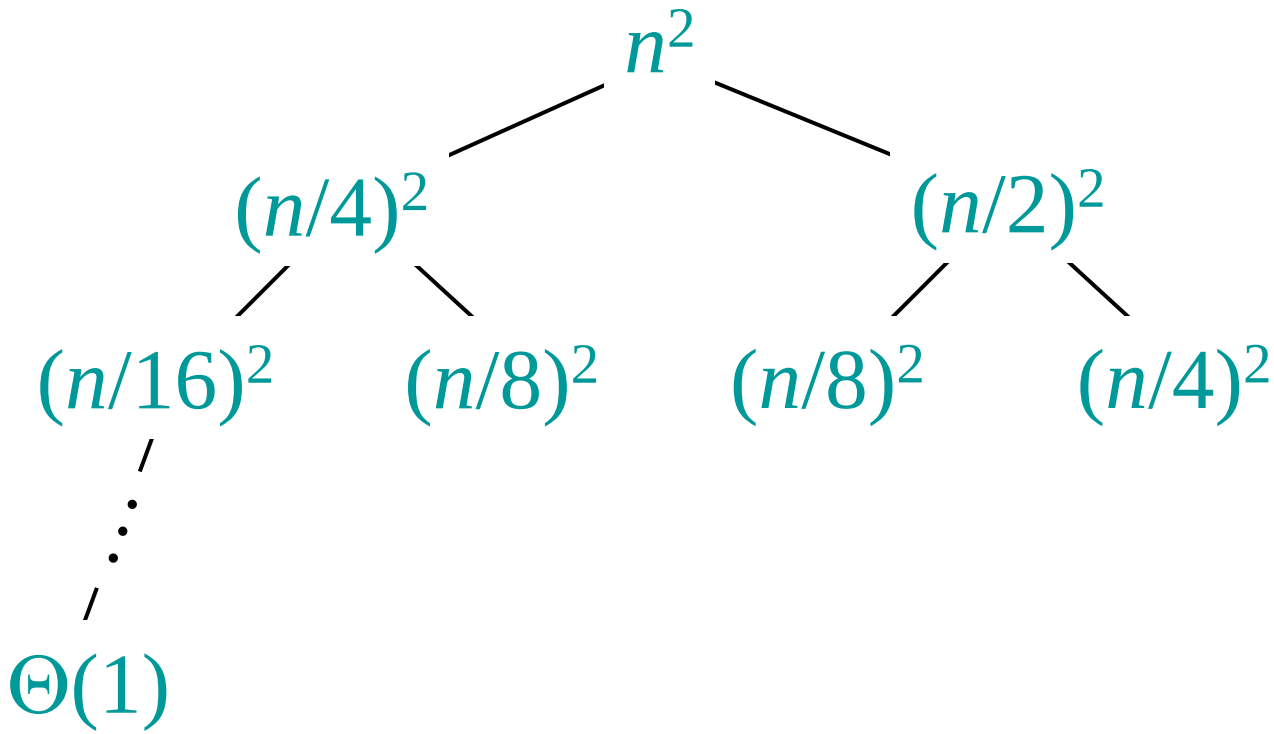
# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



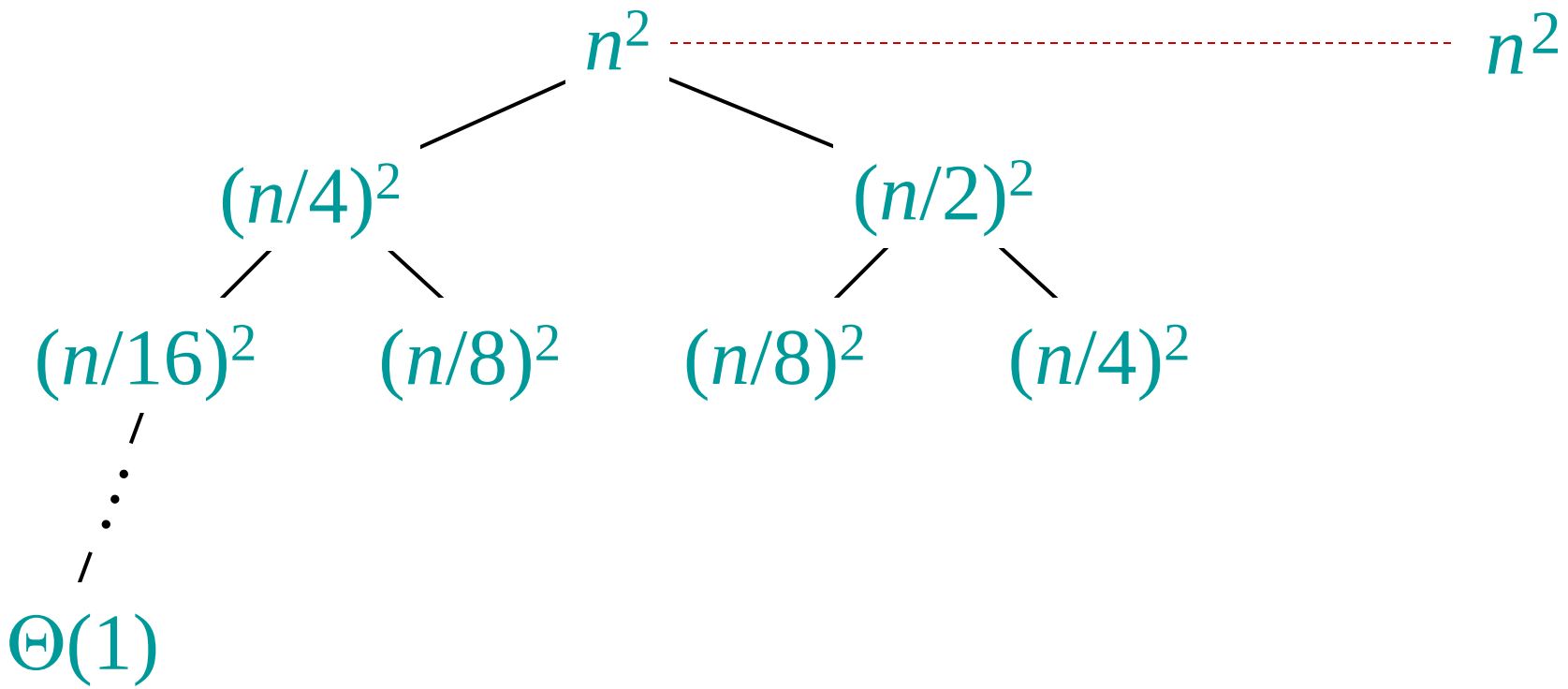
# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



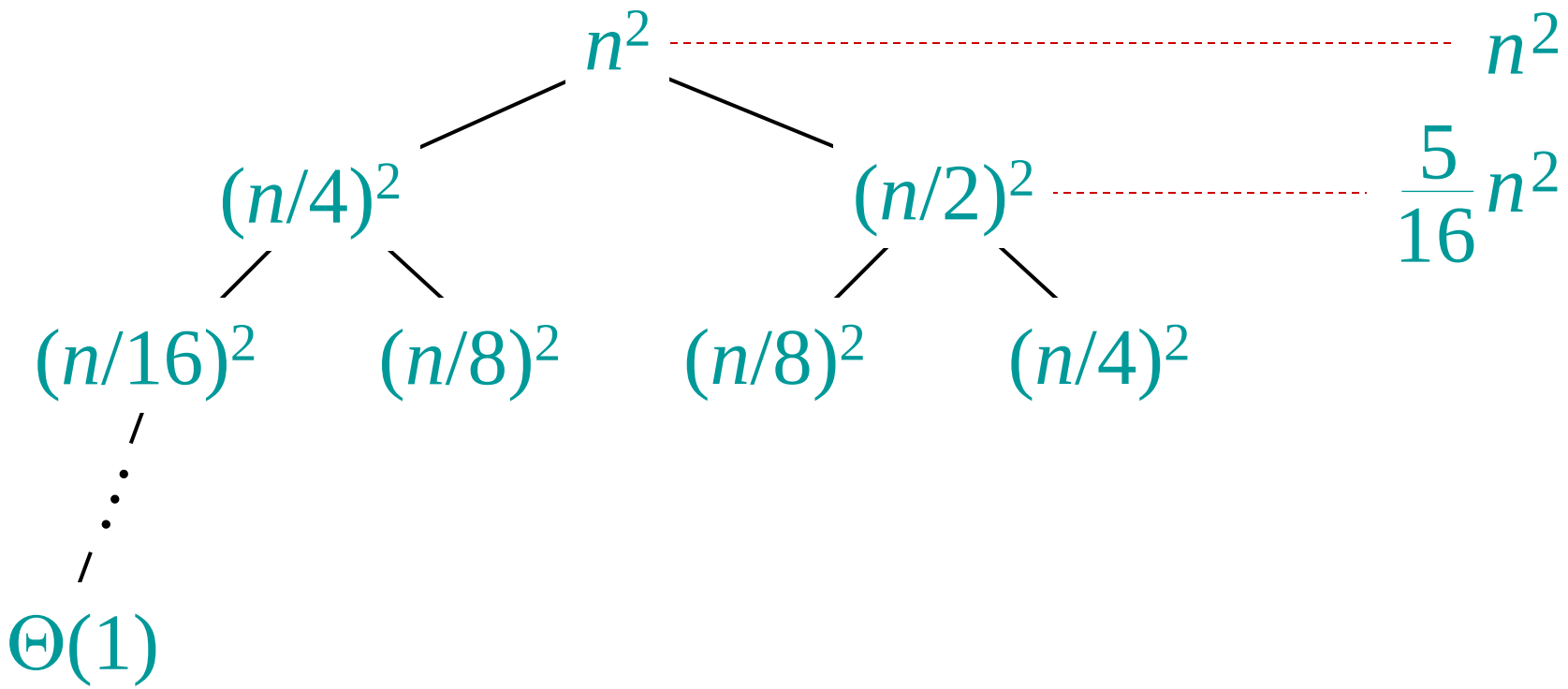
# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



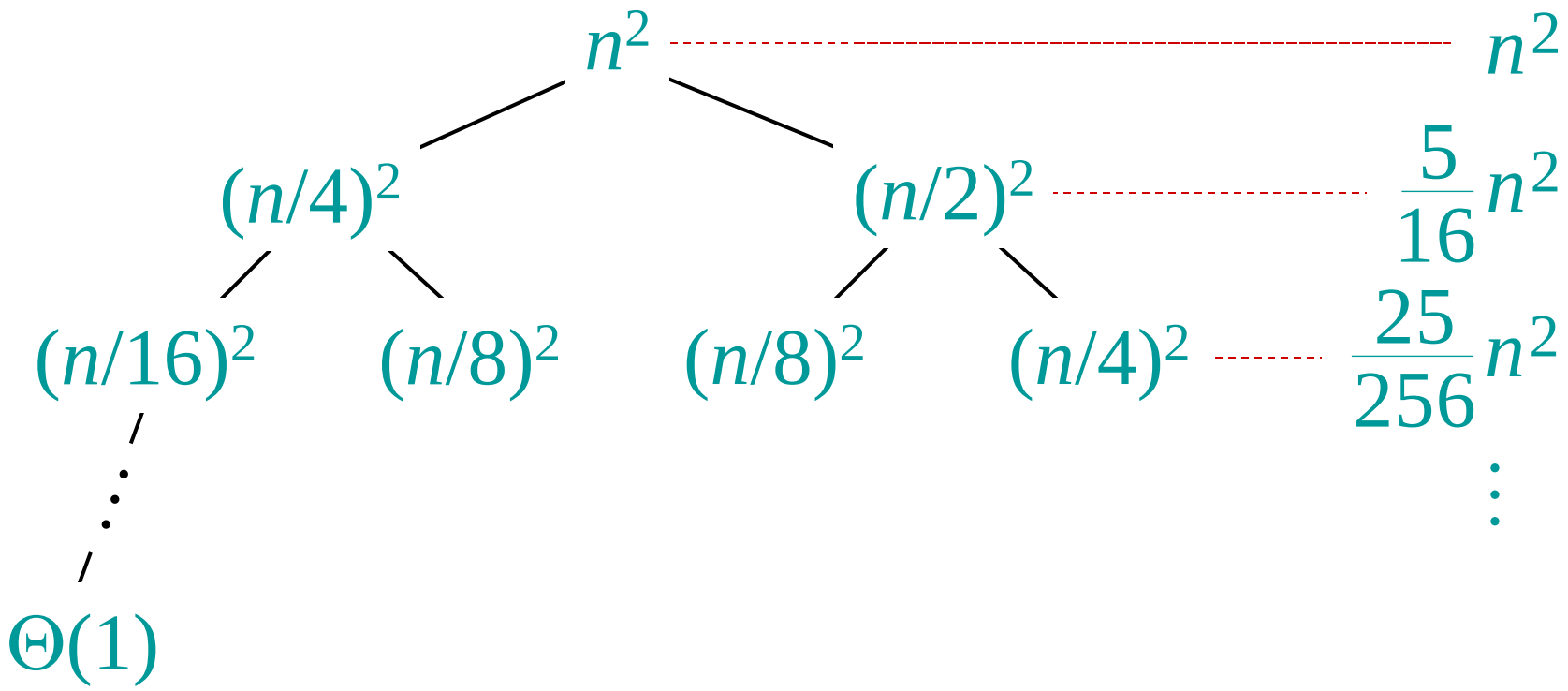
# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



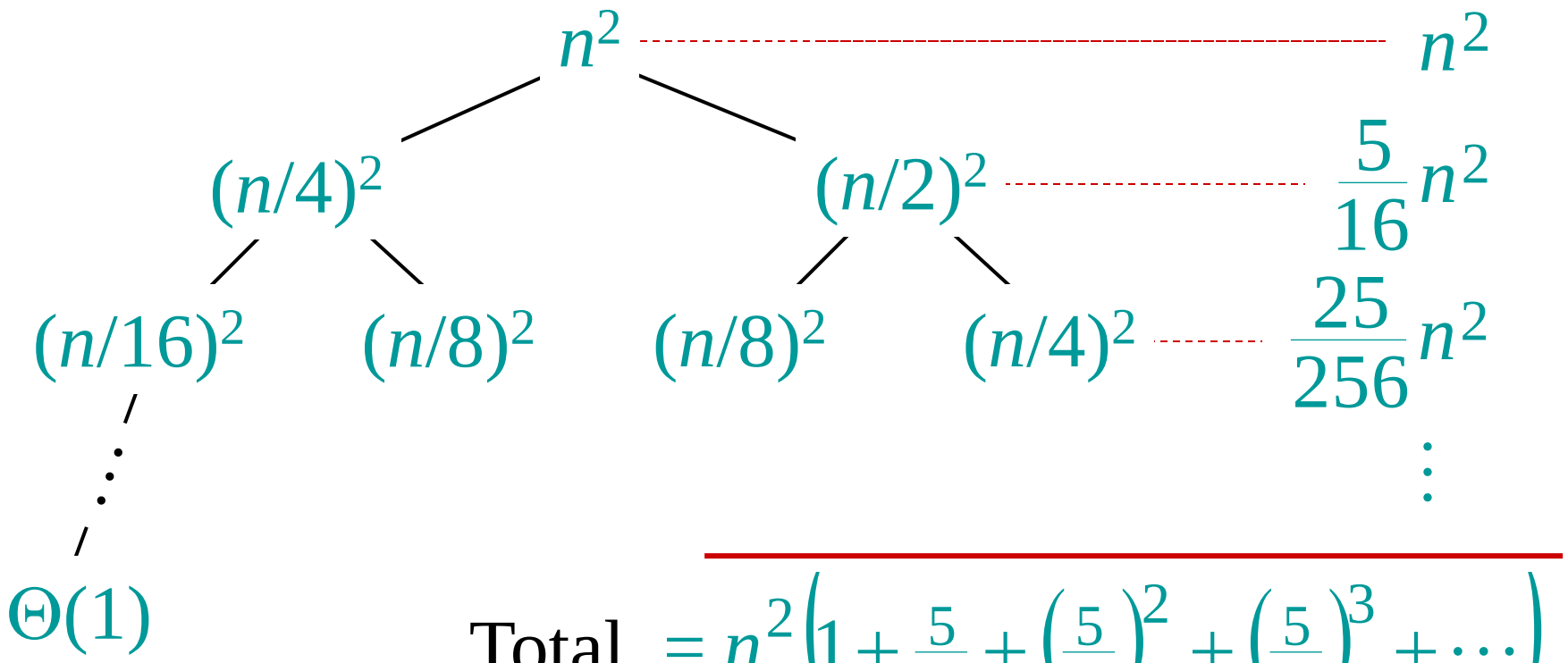
# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



# Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:




$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2)$$ *geometric series*

# Appendix: geometric series

$$1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x} \quad \text{for } x \neq 1$$

$$1 + x + x^2 + \cdots = \frac{1}{1 - x} \quad \text{for } |x| < 1$$

# Master Theorem

**The form of recurrence relation is as follow to apply master theorem**

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ $a \geq 1$ ; $b > 1$ and f(n) = O($n^k$)

**Case 1 :** $n^{\log_b a} > f(n)$ ⟹ $T(n) = \theta(n^{\log_b a})$

**Case 2 :** $n^{\log_b a} < f(n)$ ⟹ $T(n) = \theta(f(n))$

**Case 3 :** $n^{\log_b a} = f(n)$ ⟹ $T(n) = \theta(f(n)logn) = \theta\left(n^{\log_b a} logn\right)$

# Applying Master Theorem

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

a = 2 b = 2 f(n) = n

$$n^{\log_b a} = n^{\log_2 2} = n^1 = n = f(n)$$

Case 3

$$T(n) = \theta(f(n)\log n) = \theta(n\log n)$$

# Applying Master Theorem

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

a = 2 b = 2 f(n) = $n^2$

Case 2

$$n^{\log_b a} = n^{\log_2 2} = n^1 = n < n^2 = f(n)$$

$$T(n) = \theta\big(f(n)\big) = \theta(n^2)$$

# Applying Master Theorem

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

a = $2^n$

Can'nt apply a is not constant

b = 2

f(n) = $n^2$

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

Can'nt apply a < 1

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

Can'nt apply b < 1

# Applying Master Theorem

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

a = 4 b = 2 f(n) = n

Case 1

$$n^{\log_b a} = n^{\log_2 4} = n^2 > n = f(n)$$

$$T(n) = \theta\left(n^{\log_b a}\right) = \theta(n^2)$$

# Applying Master Theorem

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

a = 4 b = 2 f(n) = $n^2$

$$n^{\log_b a} = n^{\log_2 4} = n^2 = n^2 = f(n)$$

Case 3

$$T(n) = \theta(f(n)logn) = \theta(n^2logn)$$

# Applying Master Theorem

$$T(n) = 3T\left(\frac{n}{2}\right) + n$$

a = 3 b = 2 f(n) = n

$$n^{\log_b a} = n^{\log_2 3} \quad > n \quad = f(n)$$

Case 1

$$T(n) = \theta\left(n^{\log_b a}\right) = \theta\left(n^{\log_2 3}\right)$$

# Applying Master Theorem

$$T(n) = 7T\left(\frac{n}{3}\right) + n^3$$

a = 7 b = 3 f(n) = $n^3$

$$n^{\log_b a} = n^{\log_3 7} < n^3 = f(n)$$

Case 2

$$T(n) = \theta(f(n)) = \theta(n^3)$$

# Basic Master Theorem

The master method applies to recurrences of the form:

$$T(n) = a\,T(n/b) + f(n)\,,$$

where $a >= 1$, $b > 1$, and $f$ is asymptotically positive.

# Basic Master Theorem

$T(n) = a\,T(n/b) + f(n)$

To use the basic master theorem, you will need to memorize three cases:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then

$$T(n) = \theta(n^{\log_b a}).$$

2. If $f(n) = \theta(n^{\log_b a})$ then

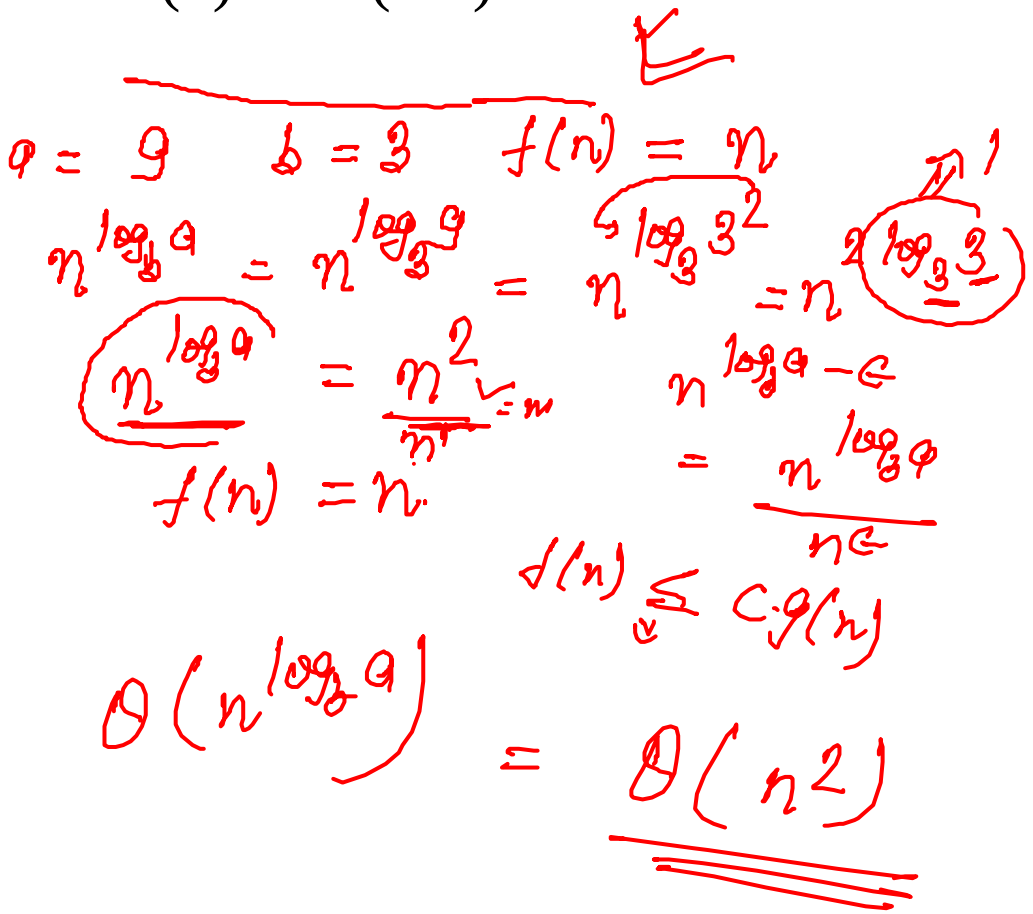
$$T(n) = \theta(n^{\log_b a} \lg n).$$

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $a\left(f\left(\frac{n}{b}\right)\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then

$$T(n) = \theta f(n)$$

# Examples

$T(n) = 9T(n/3) + n$




$T(n) = a\,T(n/b) + f(n)$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then
   $T(n) = \Theta(n^{\log_b a})$.

2. If $f(n) = \Theta(n^{\log_b a})$ then
   $T(n) = \Theta(n^{\log_b a} \lg n)$.

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $a\left(f\left(\frac{n}{b}\right)\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then
   $T(n) = \Theta f(n)$

# Examples

$T(n) = T(2n/3) + 1$

$a = 1 \quad b = 3/2 \quad f(n) = 1$

$n^{\log_b a} = n^{\log_{1.5} 1} = n^0 = 1$

$f(n) \leq c \cdot g(n)$

$f(n) \geq c \cdot g(n)$

$\theta(1 \cdot \log n) = \theta(\log n)$

$T(n) = a\,T(n/b) + f(n)$

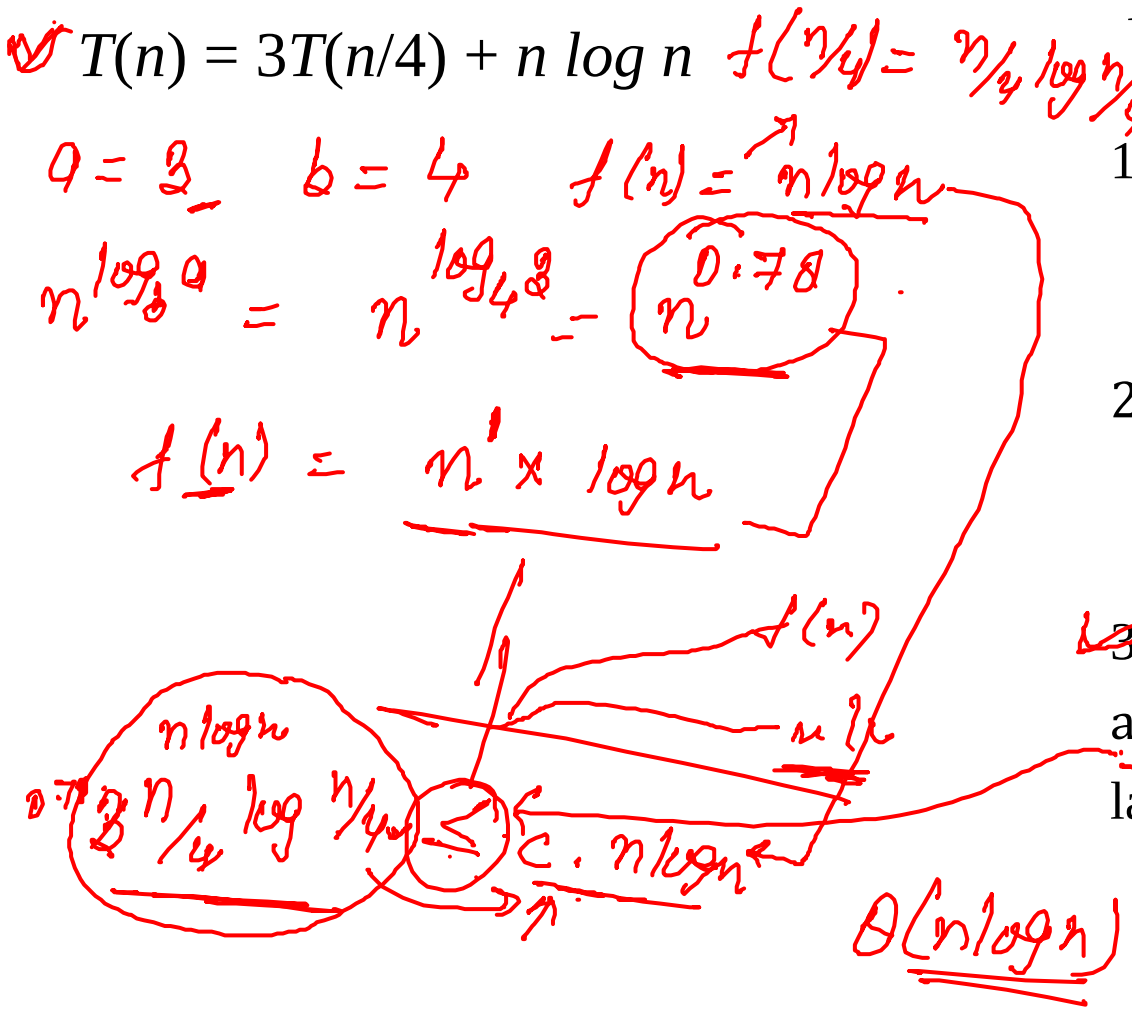
1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then
   $T(n) = \theta(n^{\log_b a})$.

2. If $f(n) = \theta(n^{\log_b a})$ then
   $T(n) = \theta(n^{\log_b a} \lg n)$.

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $a\left(f\left(\frac{n}{b}\right)\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then
   $T(n) = \theta f(n)$

# Examples

$T(n) = 3T(n/4) + n \log n$




$T(n) = a\,T(n/b) + f(n)$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then
   $T(n) = \theta(n^{\log_b a})$.

2. If $f(n) = \theta(n^{\log_b a})$ then
   $T(n) = \theta(n^{\log_b a} \lg n)$.

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $a\left(f\left(\frac{n}{b}\right)\right) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then
   $T(n) = \theta f(n)$

# Examples (Failed Case)

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

$T(n) = a\,T(n/b) + f(n)$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then
   $T(n) = \theta(n^{\log_b a})$.

2. If $f(n) = \theta(n^{\log_b a})$ then
   $T(n) = \theta(n^{\log_b a} \lg n)$.

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $a\left(f\left(\frac{n}{b}\right)\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then
   $T(n) = \theta f(n)$

# Examples (Failed Case)

$T(n) = 4T(n/2) + n^2/\lg n$

$a = 4 \quad b = 2 \quad f(n)$

$n^{\log_b a} = n^{\log_2 4}$

$= n^2$

$n^2/\log n$

$n^2 \times \log n$

$T(n) = a\,T(n/b) + f(n)$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then

   $T(n) = \theta(n^{\log_b a})$.

2. If $f(n) = \theta(n^{\log_b a})$ then

   $T(n) = \theta(n^{\log_b a} \lg n)$.

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $a\left(f\left(\frac{n}{b}\right)\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then

   $T(n) = \theta f(n)$

# Extended Version of Master Theorem Cases

$T(n) = aT(n/b) + \theta(n^k log^p n)$ a > 0, b > 1, k ≥ 0, p is any real number

To use the master method, you will need to memorize three cases:

Case 1: if $a > b^k$; $T(n) = \theta(n^{log_b a})$

Case 2: if $a = b^k$; and

- $p > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $p = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $p < -1$, then $T(n) = \theta\left(n^{\log_b a}\right)$

Case 3: if $a < b^k$; and

- $p \geq 0$, then $T(n) = \theta\left(n^k \log^p n\right)$
- $p < 0$, then $T(n) = \theta(n^k)$

# Example

$$T(n) = 9T\left(\frac{n}{3}\right) + n^1$$

$a = 9 \quad b = 3 \quad k = 1 \quad p = 0$

$b^k = 3^1 = 3$

$a = 9 > b^k = 3$

$$T(n) = \theta\left(n^{\log_3 9}\right)$$

$$= \theta\left(n^{\log_3 3^2}\right)$$

$$= \theta\left(n^{2\log_3 3}\right)$$

$$= \theta(n^2)$$

$$T(n) = aT(n/b) + \theta(n^k \log^p n)$$

Case 1: if $a > b^k$; $T(n) = \theta(n^{\log_b a})$

Case 2: if $a = b^k$; and

- $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $P = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $P < -1$, then $T(n) = \theta(n^{\log_b a})$

Case 3: if $a < b^k$; and

- $P \geq 0$, then $T(n) = \theta(n^k \log^p n)$
- $P < 0$, then $T(n) = \theta(n^k)$

# Example

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

$n^k \log^p n$

$n^0 \log^0 n$

$a = 1, \quad b = 3/2$

$k = 0 \qquad p = 0$

$b^k = (3/2)^0 = 1$

$a = 1 \quad = \quad b^k = 1$

$T(n) = \theta(n^{\log_b a} \log^{p+1} n)$

$= \theta(n^{\log_{3/2} 1} \log^{0+1} n)$

$= \theta(n^0 \log n)$

$= \theta(\log n)$

Case 1: if $a > b^k$; $T(n) = \theta(n^{log_b a})$

Case 2: if $a = b^k$; and

- $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $P = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $P < -1$, then $T(n) = \theta(n^{\log_b a})$

Case 3: if $a < b^k$; and

- $P \geq 0$, then $T(n) = \theta(n^k \log^p n)$
- $P < 0$, then $T(n) = \theta(n^k)$

# Example

$T(n) = 3T\left(\frac{n}{4}\right) + nlogn$

$n^k \log^p n$

$n^1 \log^1 n$

$a = 3 \quad b = 4 \quad k = 1 \quad p = 1$

$b^k = 4^1 = 4$

$a = 3 < b^k = 4$

$p \geq 0$

$1 \geq 0$

$\theta(n^k \log^p n)$

$= \theta(n^1 \log^1 n)$

$= \theta(n \log n)$

Case 1: if $a > b^k$; $T(n) = \theta(n^{log_b a})$

Case 2: if $a = b^k$; and

- $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $P = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $P < -1$, then $T(n) = \theta(n^{\log_b a})$

Case 3: if $a < b^k$; and

- $P \geq 0$, then $T(n) = \theta(n^k \log^p n)$
- $P < 0$, then $T(n) = \theta(n^k)$

# Example

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

$a = 2 \quad b = 2 \quad k = 1 \quad p = 1$

$b^k = 2^1 = 2$

$a = 2 \quad = \quad b^k = 2$

$\theta\left(n^{\log_b a} \log^{p+1} n\right)$

$\theta\left(n^{\log_2 2} \log^{1+1} n\right)$

$\theta\left(n \log^2 n\right)$

Case 1: if $a > b^k$; $T(n) = \theta(n^{log_b a})$

Case 2: if $a = b^k$; and

- $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $P = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $P < -1$, then $T(n) = \theta\left(n^{\log_b a}\right)$

Case 3: if $a < b^k$; and

- $P \geq 0$, then $T(n) = \theta\left(n^k \log^p n\right)$
- $P < 0$, then $T(n) = \theta(n^k)$

# Example

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

$a = 2 \quad b = 2 \quad k = 1 \quad p = 0$

$b^k = 2^1 = 2$

$a = 2 = b^k = 2$

$\theta(n^{\log_2 2} \log^{0+1} n)$

$= \theta(n \log n)$

Case 1: if $a > b^k$; $T(n) = \theta(n^{log_b a})$

Case 2: if $a = b^k$; and

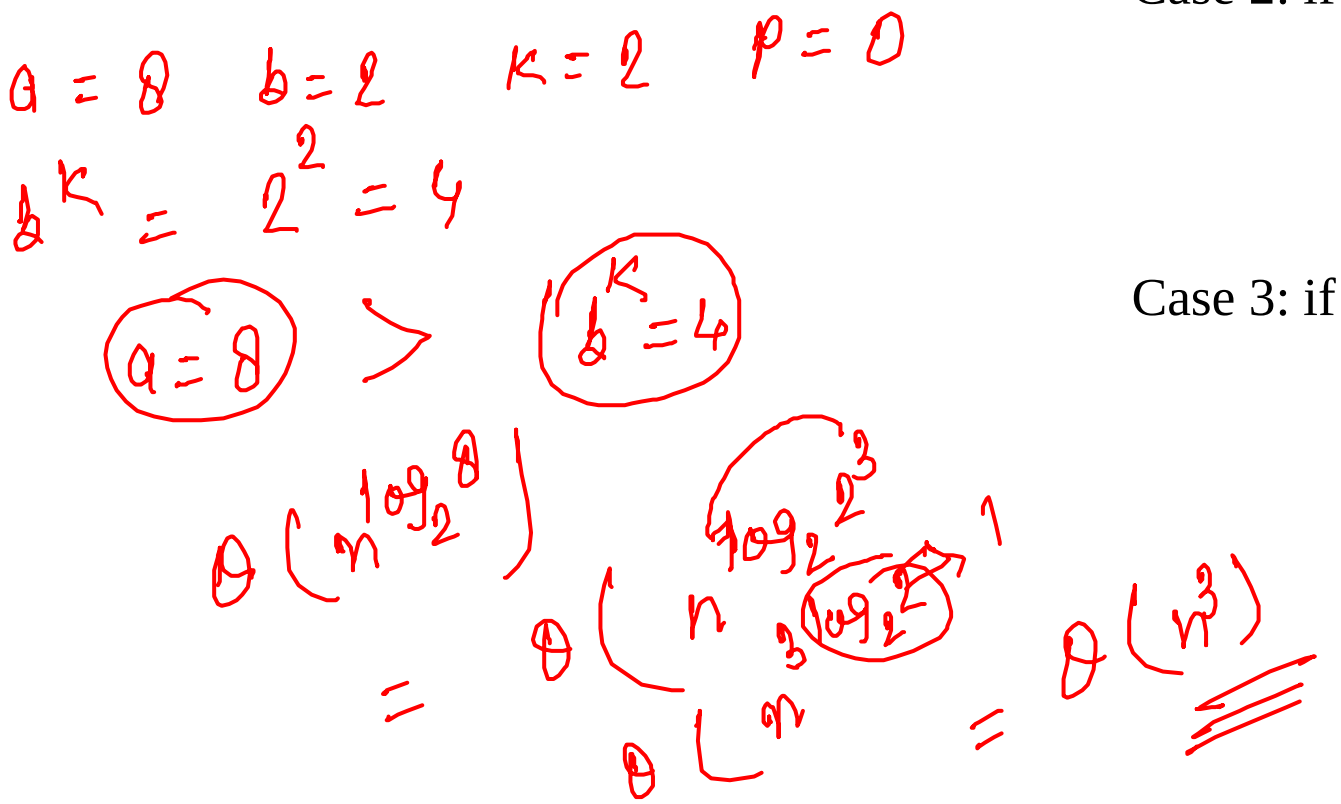
- $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $P = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $P < -1$, then $T(n) = \theta\left(n^{\log_b a}\right)$

Case 3: if $a < b^k$; and

- $P \geq 0$, then $T(n) = \theta\left(n^k \log^p n\right)$
- $P < 0$, then $T(n) = \theta(n^k)$

# Example

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




Case 1: if $a > b^k$; $T(n) = \theta(n^{log_b a})$

Case 2: if $a = b^k$; and

- $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $P = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $P < -1$, then $T(n) = \theta(n^{\log_b a})$

Case 3: if $a < b^k$; and

- $P \geq 0$, then $T(n) = \theta(n^k \log^p n)$
- $P < 0$, then $T(n) = \theta(n^k)$

# Example

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$a = 7 \quad b = 2 \quad k = 2 \quad p = 0$

$b^k = 2^2 = 4$

$a = 7 > b^k = 4$

$\theta(n^{\log_2 7}) = \theta(n^{2.8...})$

Case 1: if $a > b^k$; $T(n) = \theta(n^{log_b a})$

Case 2: if $a = b^k$; and

- $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$
- $P = -1$, then $T(n) = \theta(n^{\log_b a} \log\log n)$
- $P < -1$, then $T(n) = \theta(n^{\log_b a})$

Case 3: if $a < b^k$; and

- $P \geq 0$, then $T(n) = \theta(n^k \log^p n)$
- $P < 0$, then $T(n) = \theta(n^k)$

# Extension of Master Theorem

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

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

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