Merge Sort

Lecture 2

Pseudocode for Merge

```
C = output [length = n]
A = 1st sorted array [n/2]
B = 2nd sorted array [n/2]
i = 1
j = 1
```

```
for k = 1 to n
        if A(i) ≤ B(j)
                C(k) = A(i)
                i++
        else
                C(k) = B(j)
                j++
end
```

Pseudocode for Merge (Running time)

```
C = output [length = n]
A = 1st sorted array [n/2]
B = 2nd sorted array [n/2]

i = 1   ⎫
j = 1   ⎭   2 operations
```

```
for k = 1 to n
    if A(i) ≤ B(j)
        C(k) = A(i)
        i++
    else
        C(k) = B(j)
        j++
end
```

Running Time of Merge

- running time of Merge on array of m numbers is

$$\leq 4m + 2$$

$$\leq 6m \qquad (since\ m\ > 1)$$

4n+2 is Θ(n)

$k_2 n \ \leq 4\text{n}+ 2 \leq k_1 n$

$k_2 \ \leq 4+ 2/\text{n} \leq k_1$

$4+ 2/\text{n} \leq k_1$
$k_1 = 6$

$4+ 2/\text{n} \geq k_2$
$k_2 = 4$

Running Time of Merge

$4n+2$ is $O(n^2)$

$4\mathrm{n}+ 2 \leq k_1 n^2$

$\frac{4}{\mathrm{n}}+ 2/n^2 \leq k_1$

$k_1 = 6$

Running Time of Merge

$4n+2$ is not $\Omega(n^2)$

$4n + 2 \geq k_1 n^2$

$\frac{4}{n} + 2/n^2 \geq k_1$

Merge Sort Running Time?

Key Question : running time of Merge Sort on array of n numbers ?

[running time = # of lines of code executed]

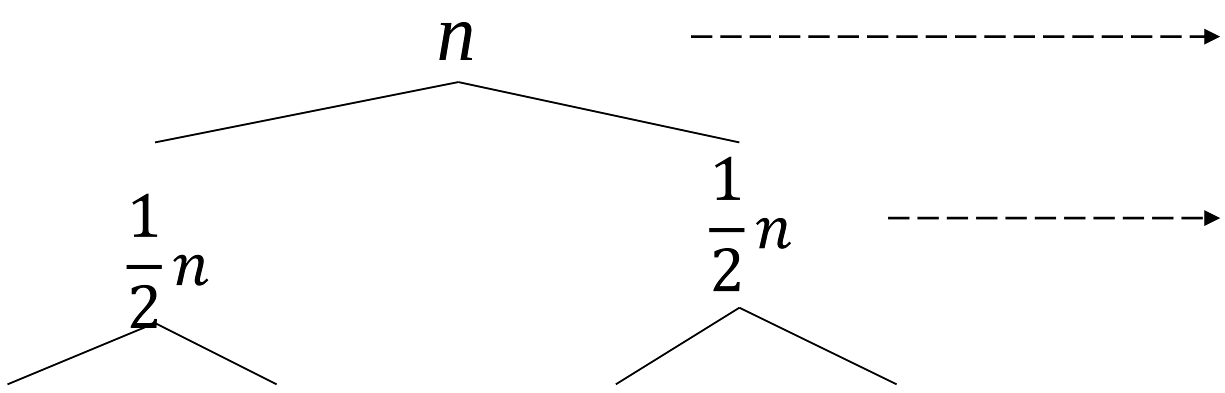
Merge Sort Running Time?

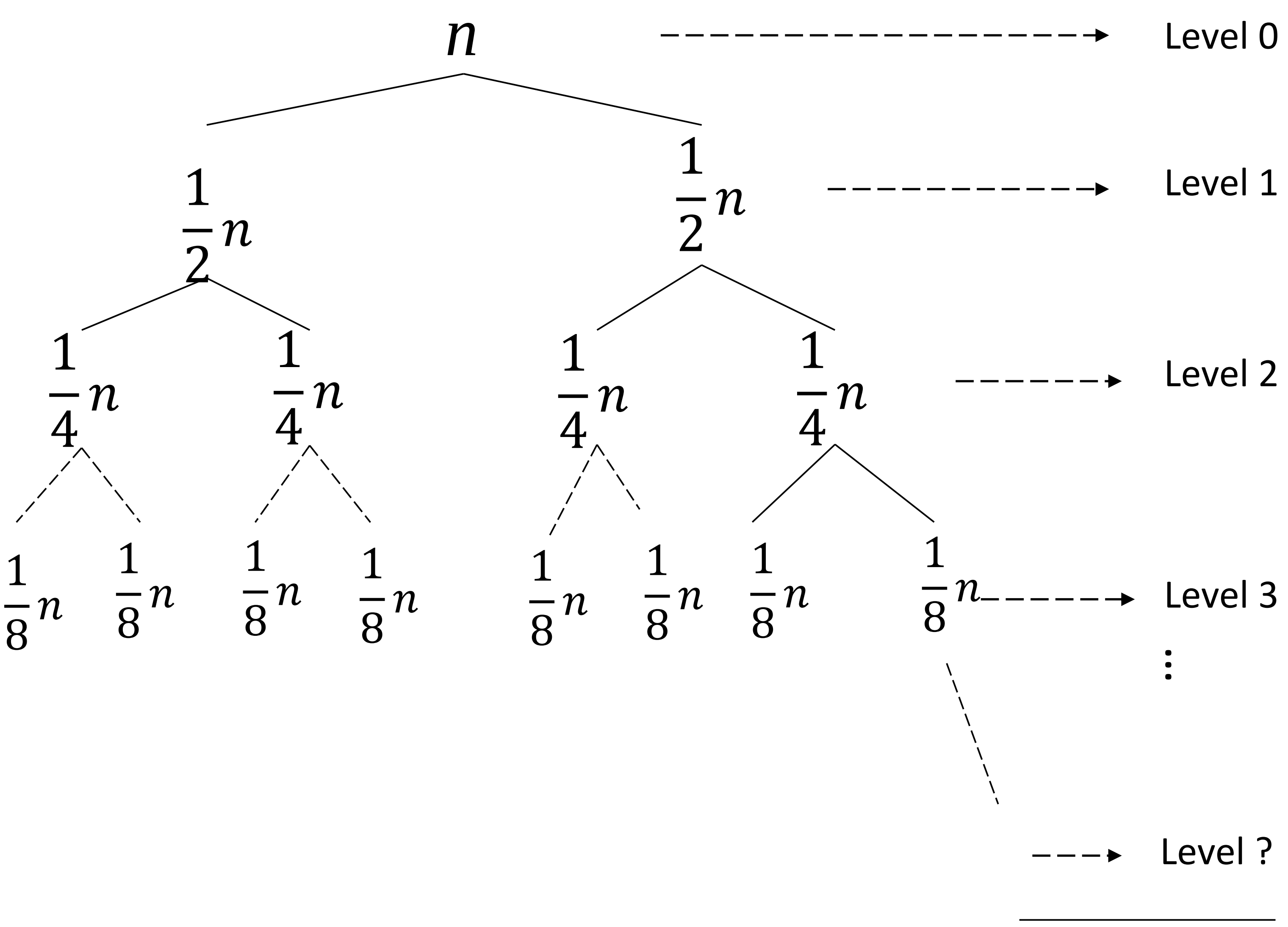
$$T(n) = T(n/2) + T(n/2) + \Theta(n)$$

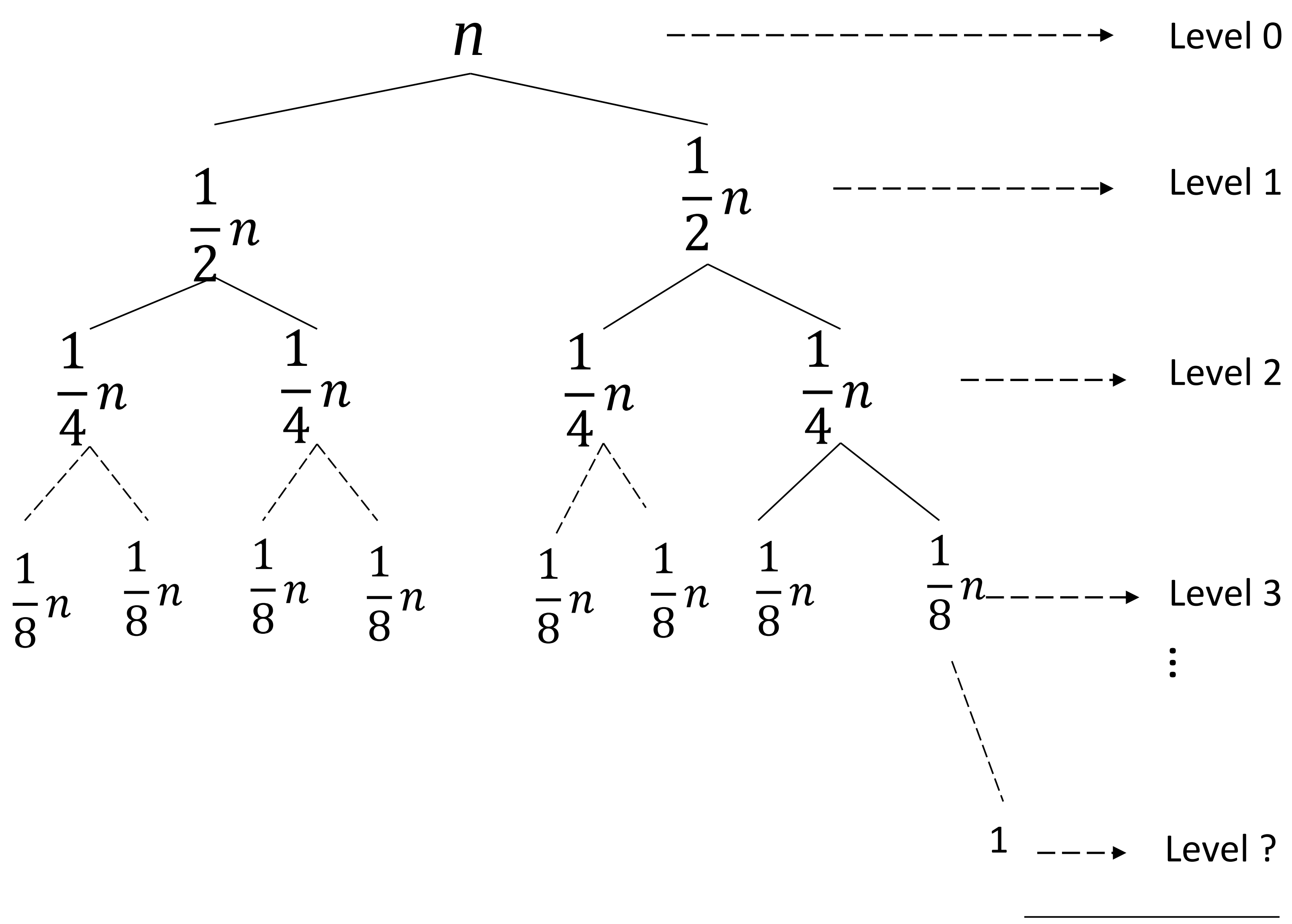
$$T(n) = 2T(n/2) + \Theta(n)$$

```
Mergesort(array,  p, q)
   {
      if (q – p = 1)
            return array
       m = (p+q)/2
       A = Mergesort(array, p, m)
       B = Mergesort(array, m+1, q)
       C = merge(A, B)
       return C
   }
```

T(n) = 2T(n/2) + Θ(n)




n
$\frac{1}{2}n$
$\frac{1}{2}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
Level 0
Level 1
Level 2
Level 3
⋮
Level ?



n
$\frac{1}{2}n$
$\frac{1}{2}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
1
Level 0
Level 1
Level 2
Level 3
⋮
Level ?

Question

Roughly how many levels does this recursion tree have (as a function of n, the length of the input array)?

a) A constant number (independent of n).
$b)$ $\log_2 n$
$c)$ n
$d)$ $\sqrt{n}$

Question

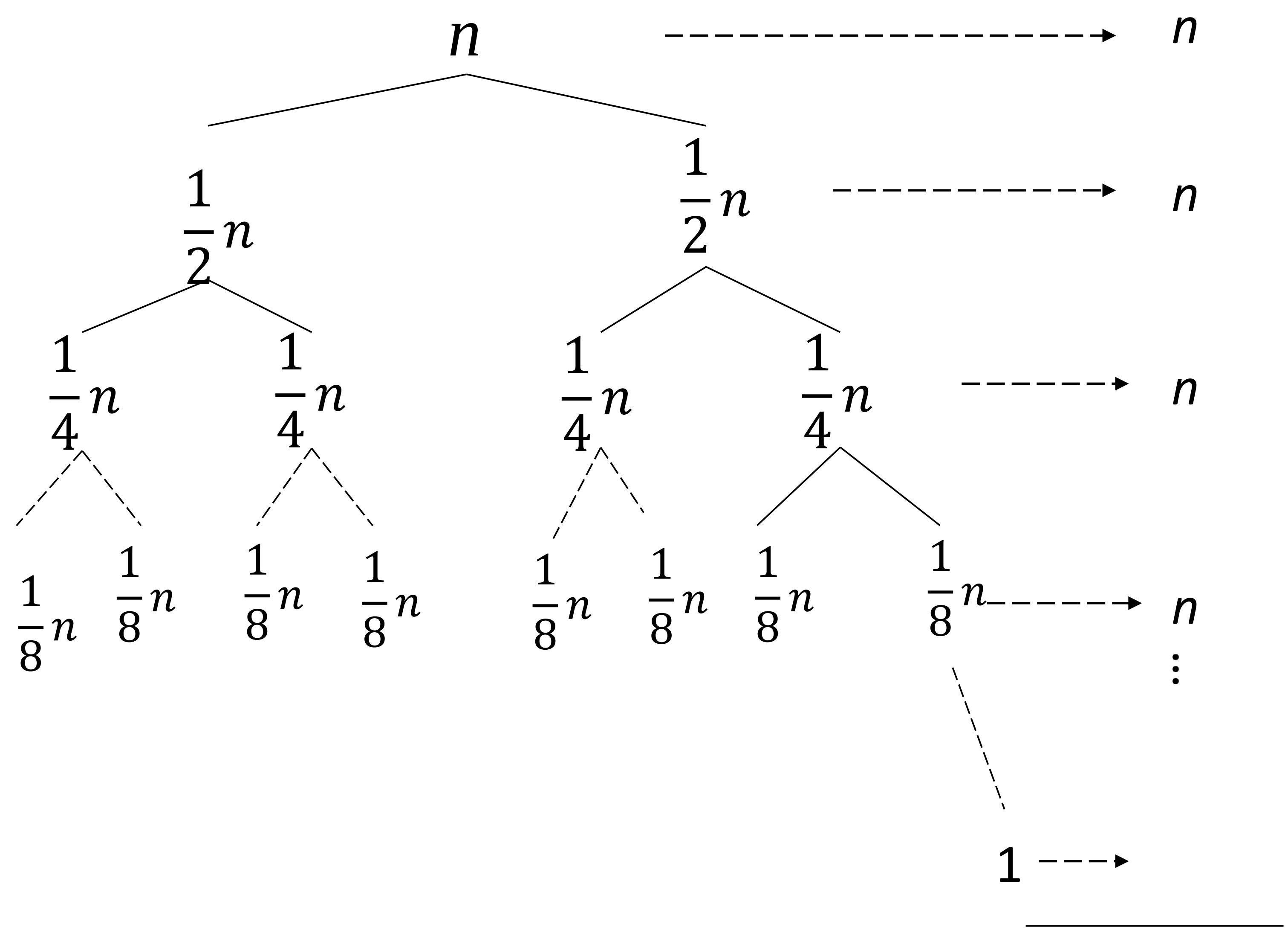
Roughly how many levels does this recursion tree have (as a function of n, the length of the input array)?

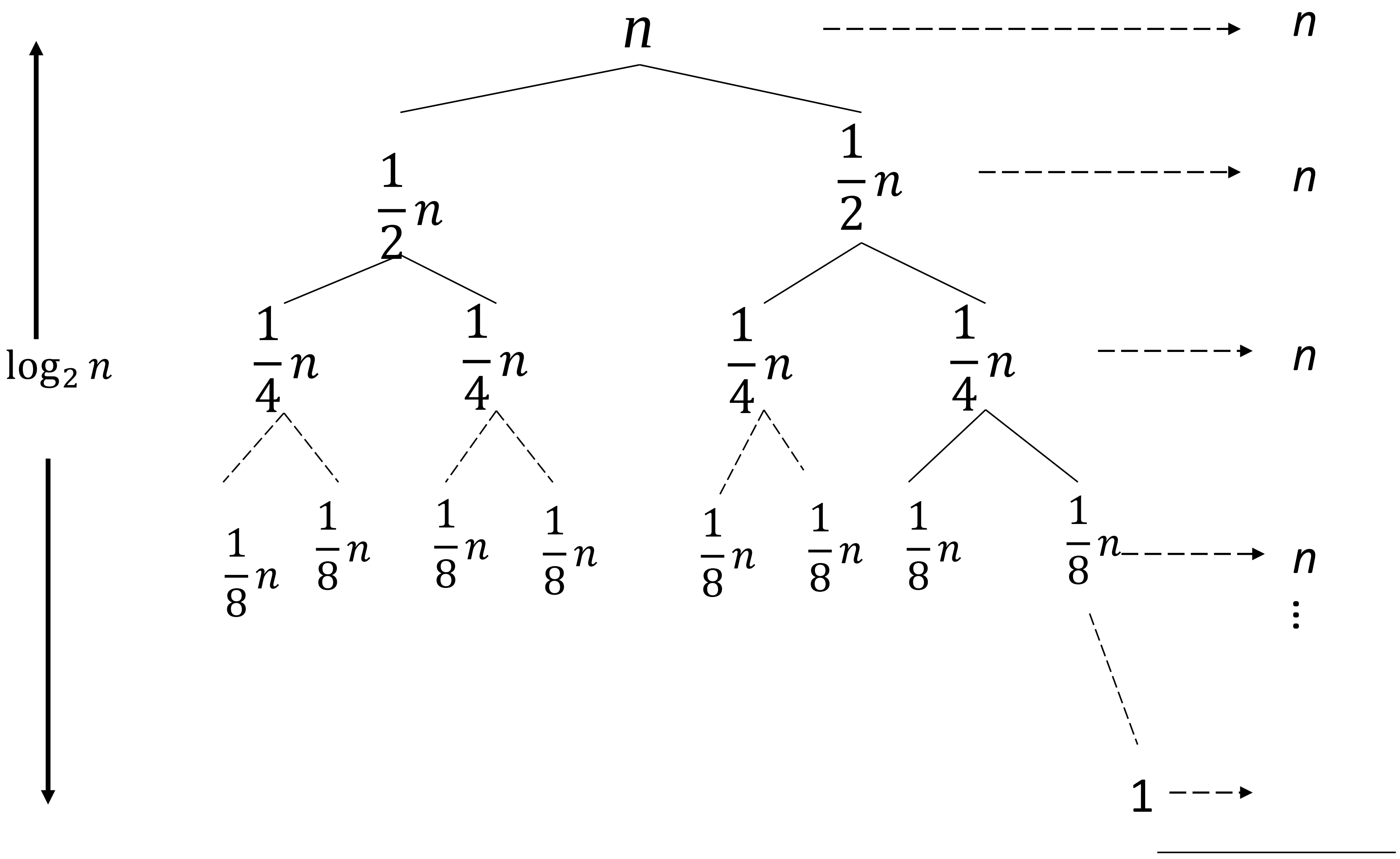
a) A constant number (independent of n).

b) $\log_2 n$ (correct answer)

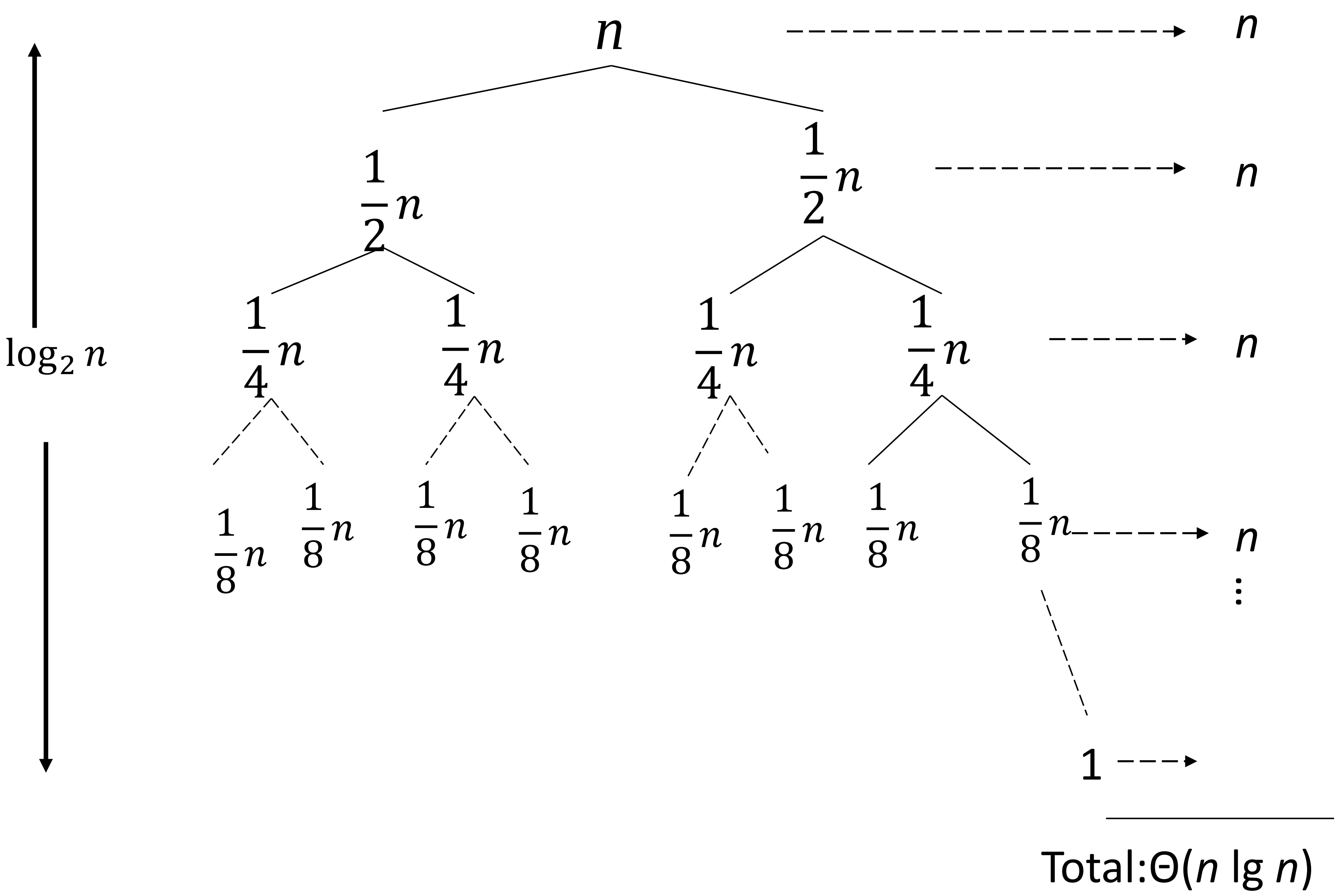
c) n

d) $\sqrt{n}$


n
$\frac{1}{2}n$ $\frac{1}{2}n$
$\frac{1}{4}n$ $\frac{1}{4}n$ $\frac{1}{4}n$ $\frac{1}{4}n$
$\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$
1
n
n
n
n
$\vdots$



n
$\frac{1}{2}n$
$\frac{1}{2}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{4}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
$\frac{1}{8}n$
1
$\log_2 n$
n
n
n
n
$\vdots$



n
$\frac{1}{2}n$ $\frac{1}{2}n$
$\frac{1}{4}n$ $\frac{1}{4}n$ $\frac{1}{4}n$ $\frac{1}{4}n$
$\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$ $\frac{1}{8}n$
1
$\log_2 n$
n
n
n
n
Total:Θ(n lg n)

What is the pattern ? Fill in the blanks in the following statement: at each level j = 0,1,2,.., $\log_2 n$, there are <blank> subproblems, each of size <blank>.

$a)$ 2^j and 2^j, respectively

$b)$ $\frac{n}{2^j}$ and $\frac{n}{2^j}$, respectively

$c)$ 2^j and $\frac{n}{2^j}$, respectively

$d)$ $\frac{n}{2^j}$ $and 2^j$, respectively

What is the pattern ? Fill in the blanks in the following statement: at each level j = 0,1,2,.., $\log_2 n$, there are <blank> subproblems, each of size <blank>.

$a)$ 2^j and 2^j, respectively

$b)$ $\frac{n}{2^j}$ and $\frac{n}{2^j}$, respectively

$c)$ 2^j and $\frac{n}{2^j}$, respectively

$d)$ $\frac{n}{2^j}$ $and 2^j$, respectively

Proof of claim (assuming n = power of 2) :

At each level j=0,1,2,.., $\log_2 n$,

Total # of operations at level j = 0,1,2,...,$\log_2 n$

$$\leq 2^j * 6\left(\frac{n}{2^j}\right) = 6n$$

# of level-j subproblems

Size of level-j subproblem

Work per level – j subproblem

$\boxed{\text{Total}}$

$$6n(\log_2 n + 1)$$

Work per level

# of levels

Running Time of Merge Sort

- For every input array of n numbers, Merge Sort produces a sorted output array and uses at most $6n\log_2 n + 6n$ operations.

- Prove $6n \log_2 n + 6n$ is Θ(n lg n).
- $k_2 n \log_2 n \leq 6n \log_2 n + 6n \leq k_1 n \log_2 n$
-

- Prove $6n \log_2 n + 6n$ is Θ(n lg n).
- $k_2 n \log_2 n \leq 6n \log_2 n + 6n \leq k_1 n \log_2 n$

- $k_2 \leq 6 + 6/\log_2 n \leq k_1$

- k_1 =

- k_2 =

- Prove $6n \log_2 n \ + 6n$ is O(n^2).
- $6n \log_2 n \ + 6n \leq \quad k_1$ n^2
- $\frac{6 \log_2 n}{n}$ + $\frac{6}{n} \leq \quad k_1$
- k_1 =

- Prove $6n\log_2 n + 6n$ is Θ(n^2).
- $k_2 n^2 \leq 6n\log_2 n + 6n \leq k_1 n^2$
- $k_2 \leq \frac{6\log_2 n}{n} + 6/n \leq k_1$

T(n) = T(n/3) +T(2n/3) + Θ(n)