1	5	7	9	10	15

2	4	8	18

MERGE THESE TWO SORTED TO FORM A SINGLE SORTED ARRAY

1	5	7	9	10	15

COMPARE 1 & 2

2	4	8	18

1	5	7	9	10	15

2	4	8	18

1									

1	5	7	9	10	15

COMPARE 5 & 2

2	4	8	18

1									

1	5	7	9	10	15

COMARE 5 & 4

2	4	8	18

1	2								

1	5	7	9	10	15

COMARE 5 & 8

2	4	8	18

1	2	4							

1	5	7	9	10	15

2	4	8	18

1	2	4	5						

AND SO ON.......

1	5	7	9	10	15

2	4	8	18

1	2	4	5	7	8	9	10	15	18

CAN YOU WRITE THE PSEUDOCODE OF MERGE

```
MERGE (A, B)
{   C ← NEW ARRAY OF SIZE(A) + SIZE(B)
    a ← 1;
    b ← 1;
    c ← 1;

    WHILE( a ≤ SIZE(A) AND b ≤ SIZE(B))
    {     IF ( A[a] ≤ B[b])
          {   C[c] ← A[a];
              c ← c+1;
              a ← a+1;
          }
          ELSE
          {   C[c] ← B[b];
              c ← c+1;
              b ← b+1;
          }
    }
    WHILE ( a ≤ size(A))
    {  C[c] ← A[a];
       c ← c+1;
       a ← a+1;
    }
    WHILE (b ≤ size(B))
    {  C[c] ← B[b];
       c ← c+1;
       b ← b+1;
```

RUNNING TIME =

RUNNING TIME = $O(\text{size}(A) + \text{size}(B))$

CORRECTNESS :

RUNNING TIME = $O(\text{size}(A) + \text{size}(B))$

CORRECTNESS: USE INDUCTION.
SEE NOTES.

RUNNING TIME = $O(\text{size}(A) + \text{size}(B))$

CORRECTNESS : USE INDUCTION.
SEE NOTES.

OBSERVATION : THE TIME TO MERGE TWO SORTED ARRAY A & B is $O(\text{size}(A) + \text{size}(B))$.

A TECHNIQUE: DIVIDE AND CONQUER

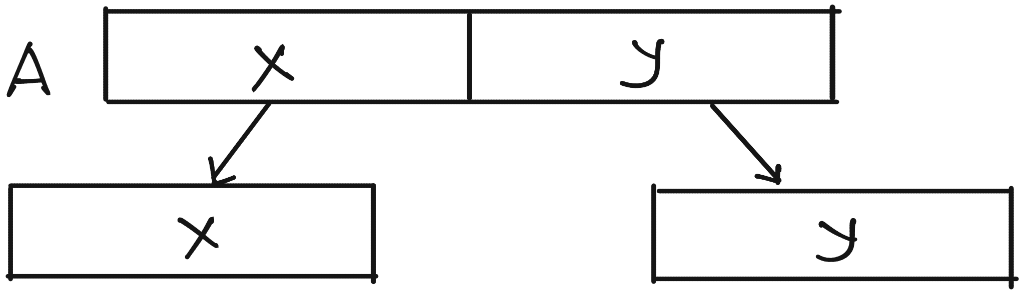
A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS

A

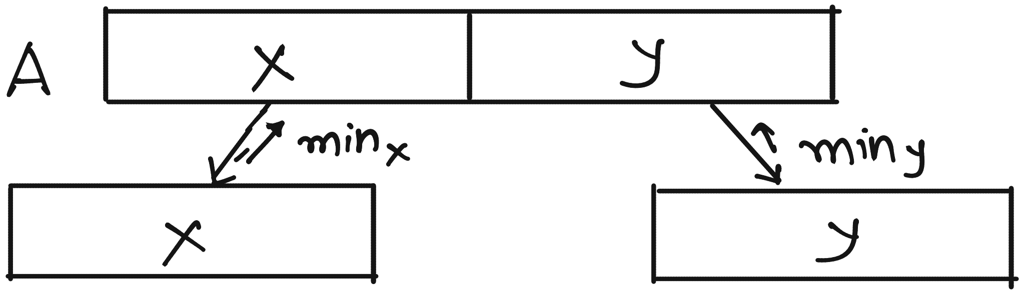
A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS



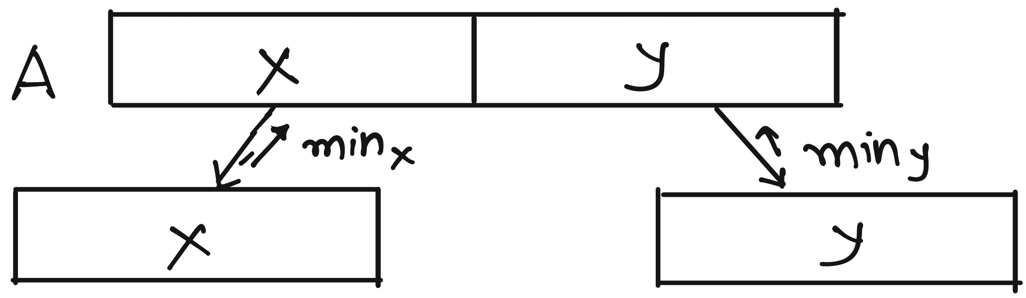
A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS



A TECHNIQUE: DIVIDE AND CONQUER

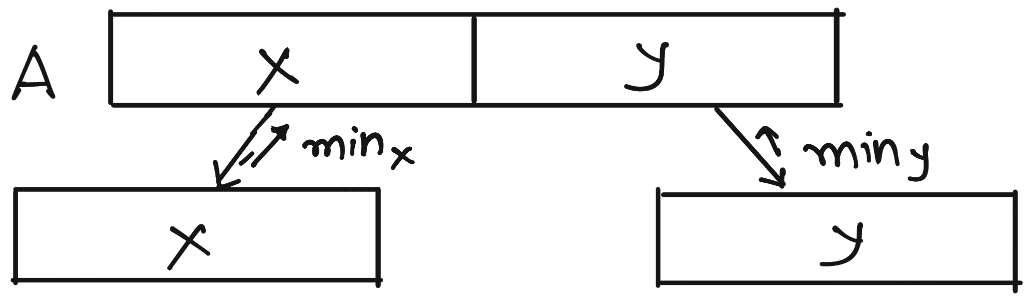
FIND MINIMUM OF n NUMBERS



A NICE PROPERTY OF THE PROBLEM: WE CAN COMBINE TWO SOLUTIONS IN REASONABLE TIME

A TECHNIQUE: DIVIDE AND CONQUER

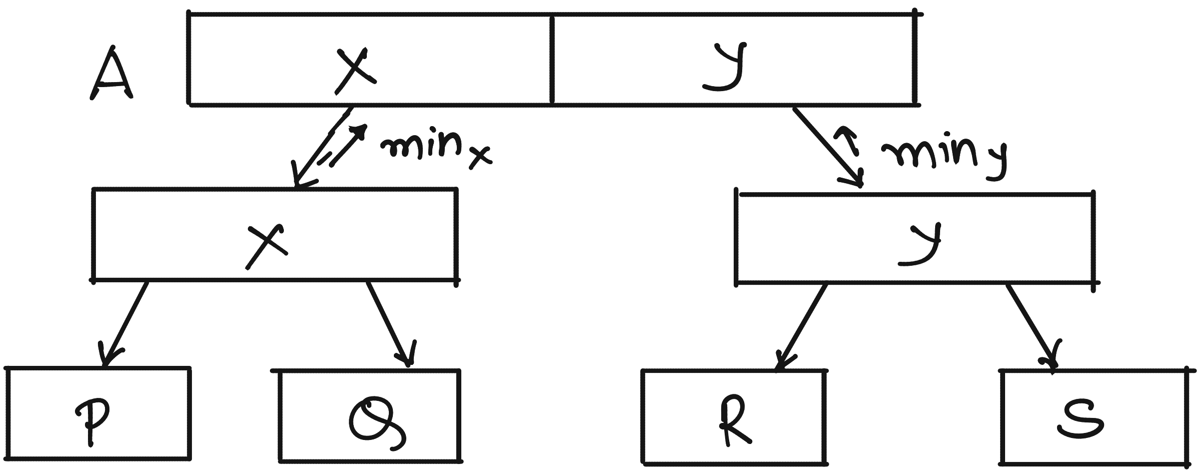
FIND MINIMUM OF n NUMBERS



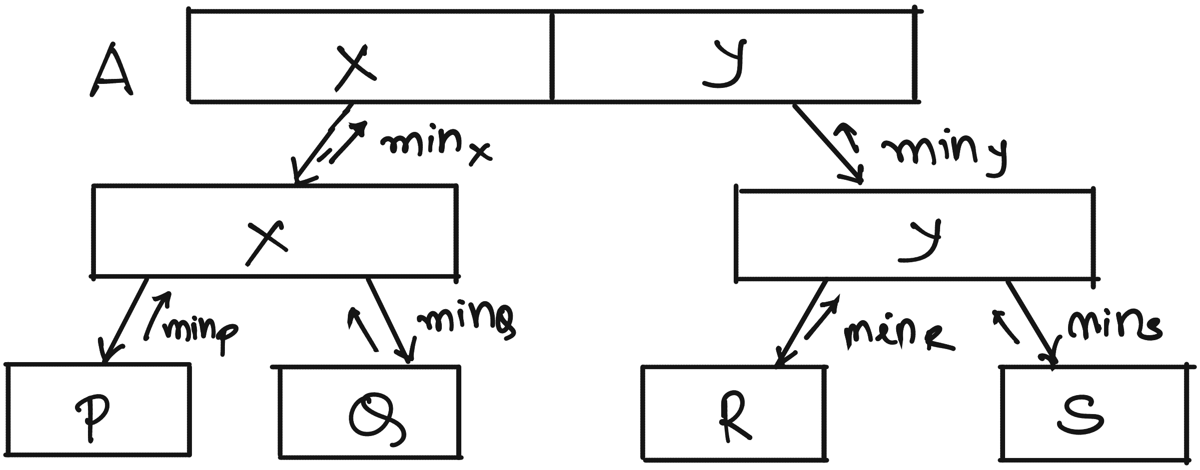
A NICE PROPERTY OF THE PROBLEM: WE CAN COMBINE TWO SOLUTIONS IN <u>REASONABLE</u> TIME

A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS

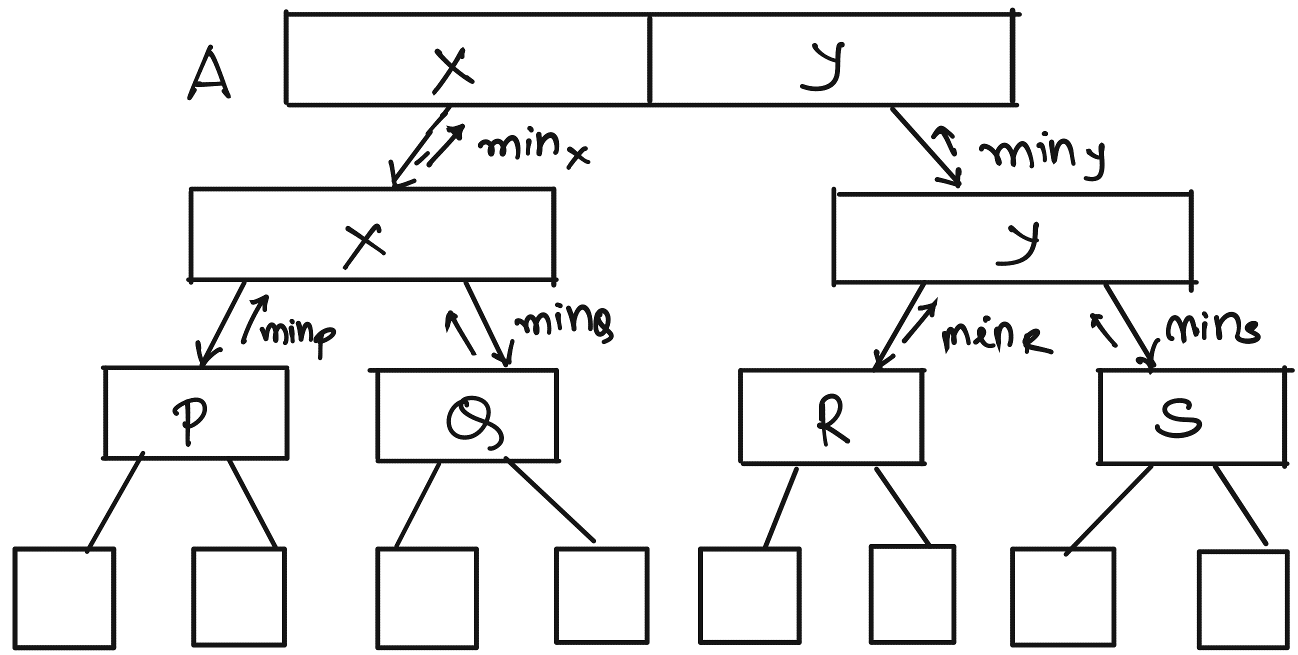


A TECHNIQUE: DIVIDE AND CONQUER
FIND MINIMUM OF n NUMBERS



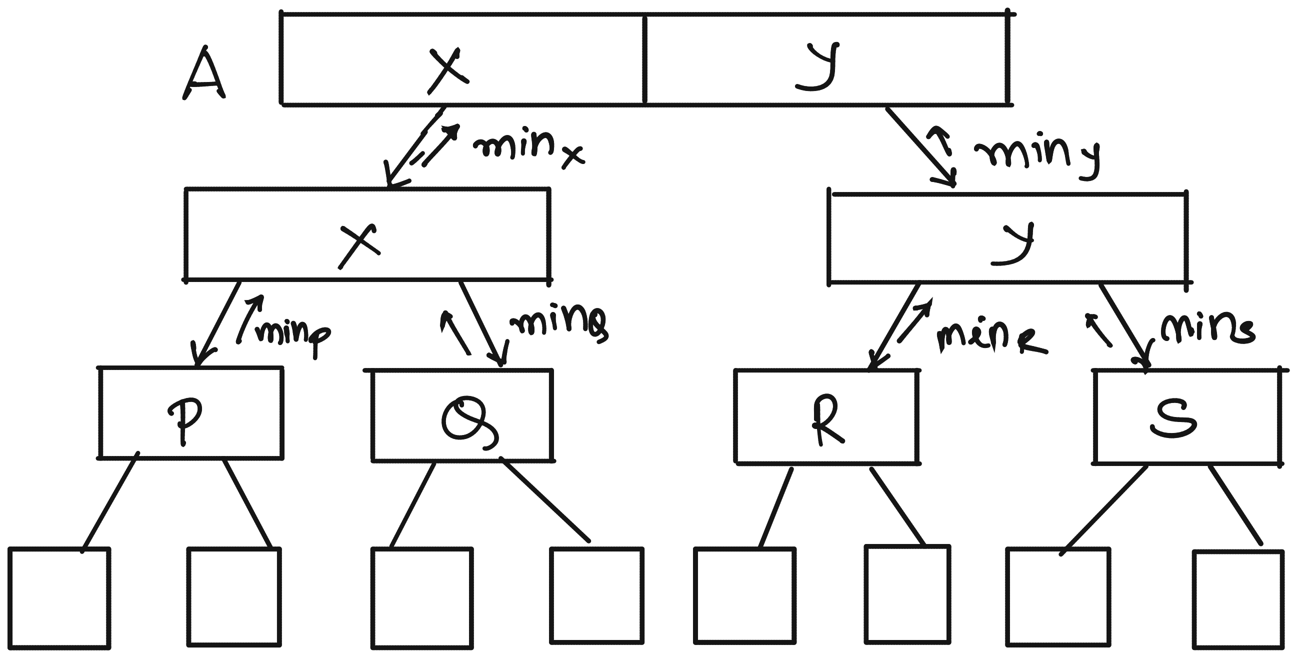
A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS



A TECHNIQUE: DIVIDE AND CONQUER

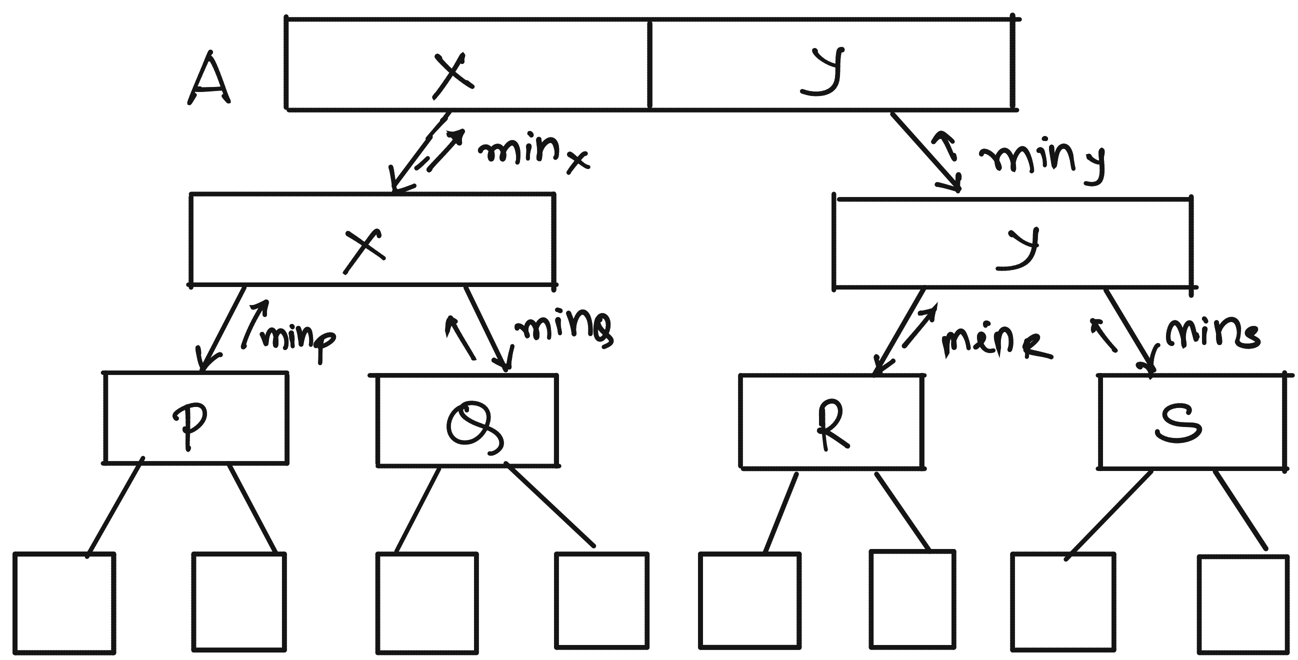
FIND MINIMUM OF n NUMBERS



YOU CANNOT DIVIDE FURTHER

A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS

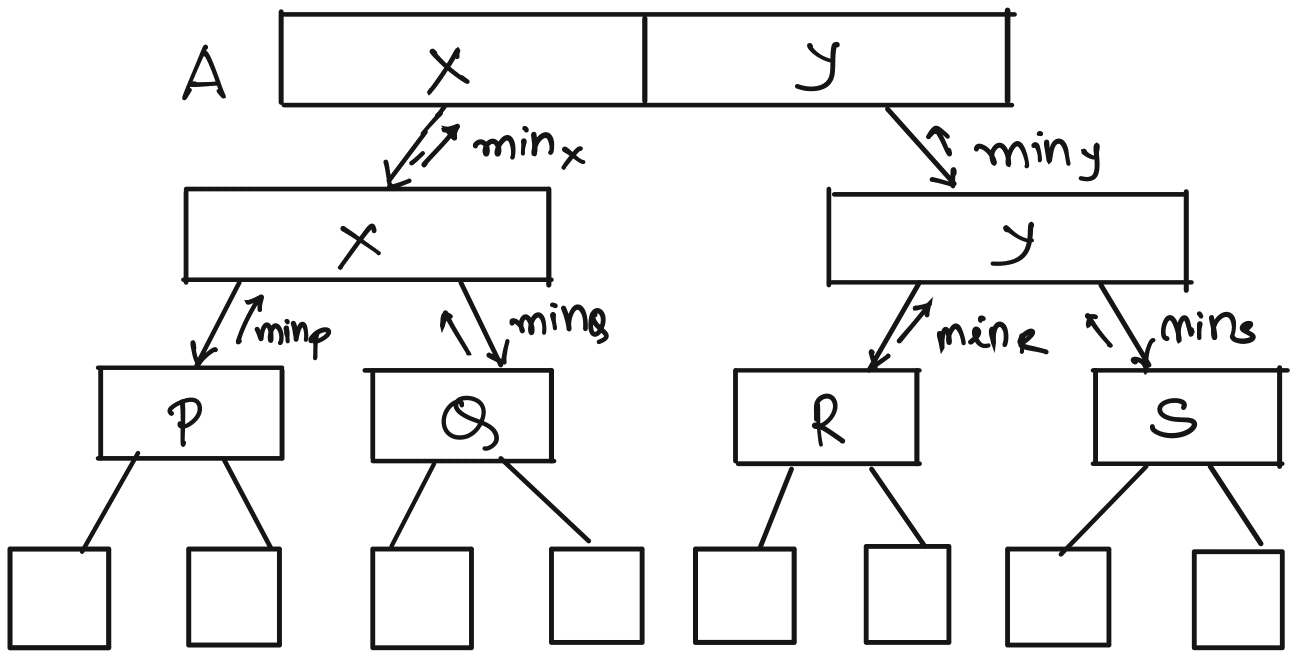


YOU CANNOT DIVIDE FURTHER

→ YOU HAVE REACHED BASE CASE

A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS



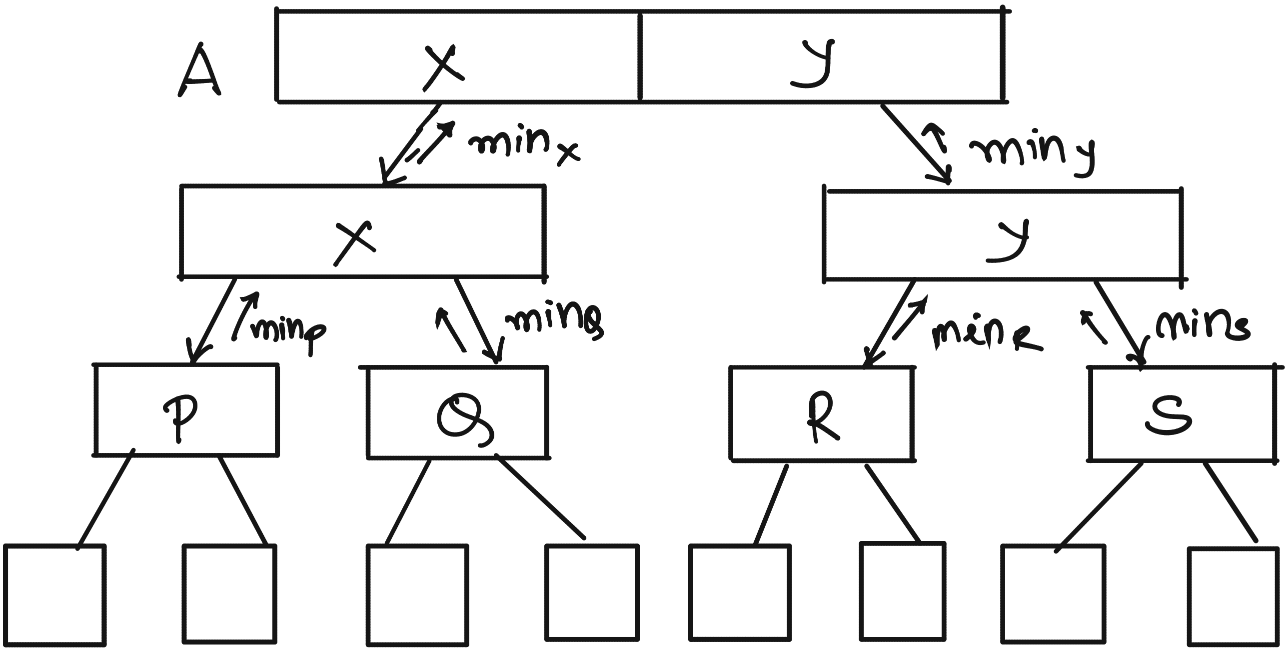
YOU CANNOT DIVIDE FURTHER

→ YOU HAVE REACHED BASE CASE

A NICE PROPERTY OF THE PROBLEM: WE CAN SOLVE THE BASE CASE IN REASONABLE TIME

A TECHNIQUE: DIVIDE AND CONQUER

FIND MINIMUM OF n NUMBERS



YOU CANNOT DIVIDE FURTHER

→ YOU HAVE REACHED BASE CASE

A NICE PROPERTY OF THE PROBLEM: WE CAN SOLVE THE BASE CASE IN REASONABLE TIME

FOR MIN PROBLEM: THE BASE CASE IS VERY EASY.

TWO IMPORTANT PROPERTIES OF PROBLEM.

1) WE CAN COMBINE MANY SUBPROBLEMS IN REASONABLE TIME

2) BASE CASE CAN BE SOLVED IN REASONABLE TIME

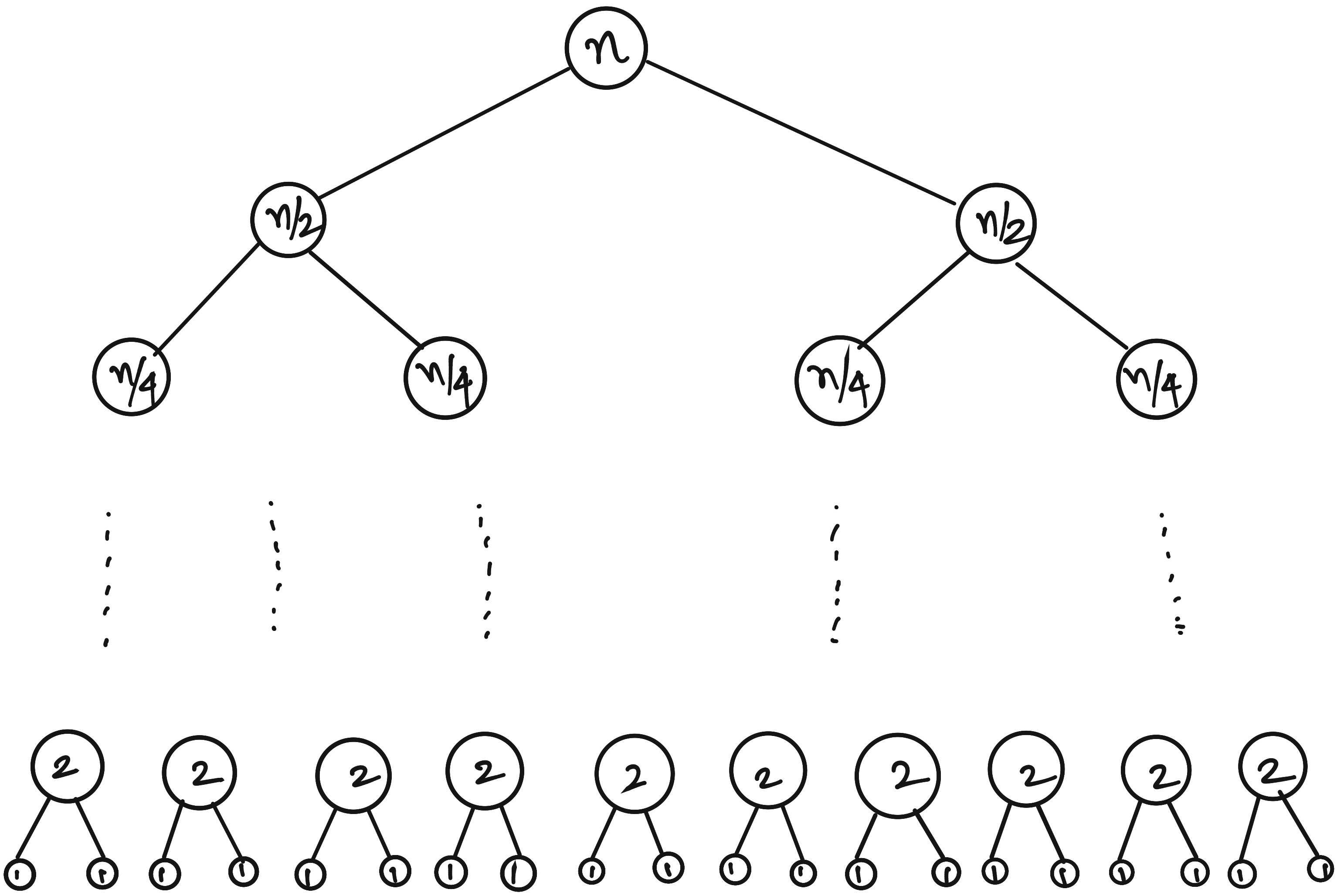
```
FINDMIN (A, LOW, HIGH)
{
    IF (                    )  ]
    {                          ] ← BASE CASE
    }                          ]
    ELSE
    {     MID ← ⌊(LOW+HIGH)/2⌋

        MIN1 ← FINDMIN (A, LOW, MID);
        MIN2 ← FINDMIN (A, MID+1, HIGH);
        RETURN MIN{ MIN1, MIN2 };
    }
}
```

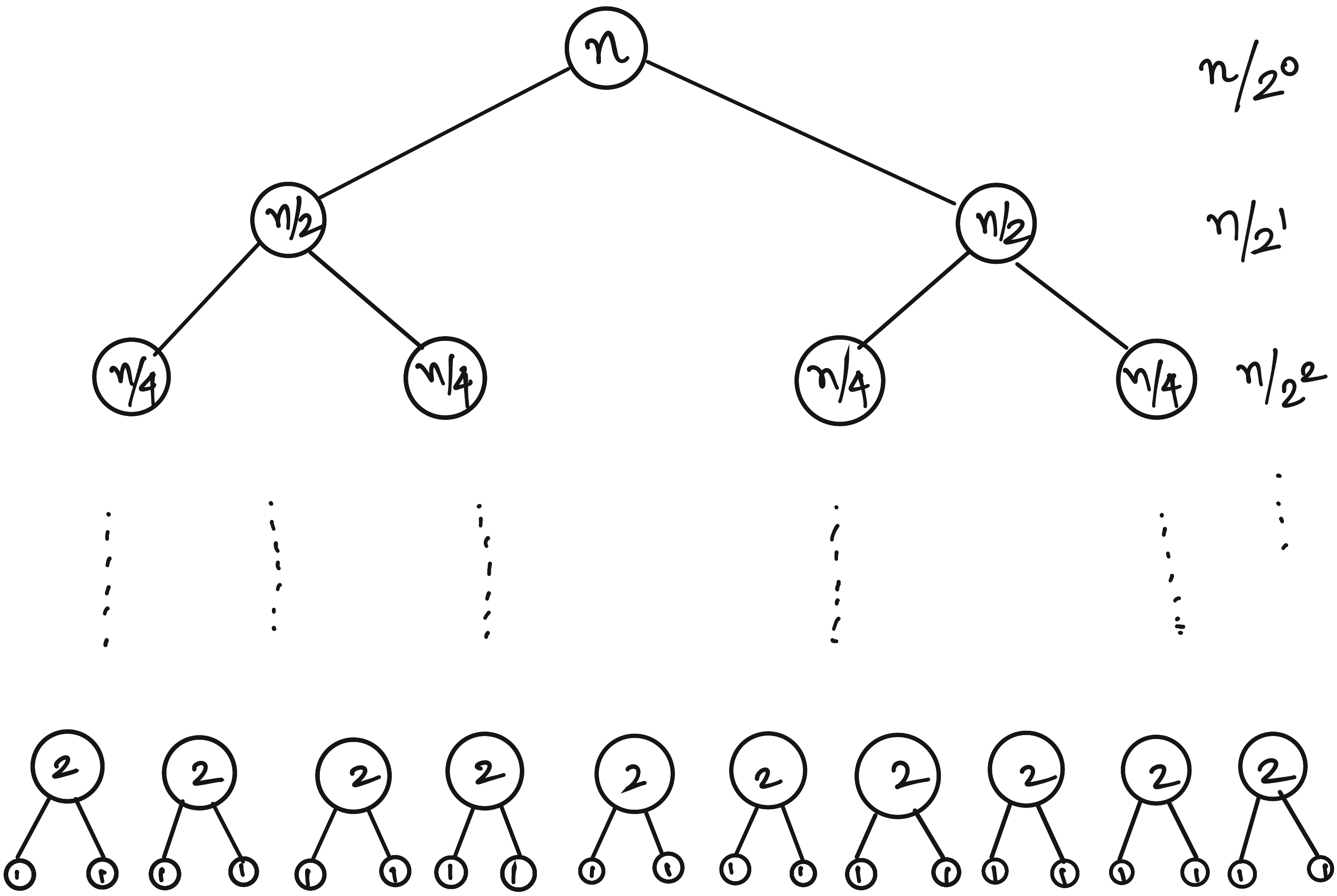
```
FINDMIN (A, LOW, HIGH)
{
    IF (LOW = HIGH)
    {   RETURN A[LOW];
    }                          ← BASE CASE
    ELSE
    {     MID ← ⌊(LOW+HIGH)/2⌋

          MIN1 ← FINDMIN (A, LOW, MID);
          MIN2 ← FINDMIN (A, MID+1, HIGH);
          RETURN MIN{MIN1, MIN2};
    }
}
```

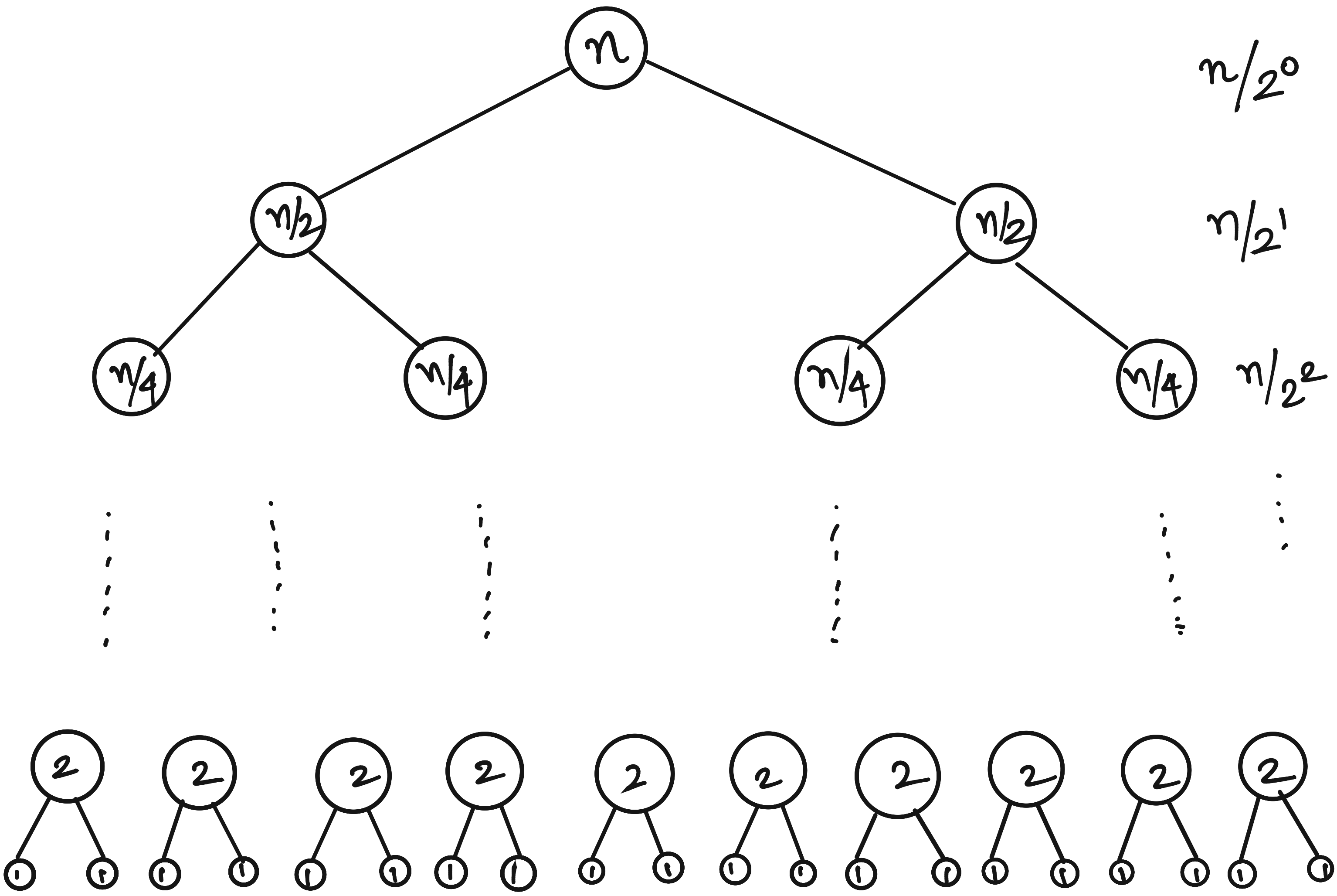
RUNNING TIME:



Q: WHAT IS THE HEIGHT OF THIS TREE?

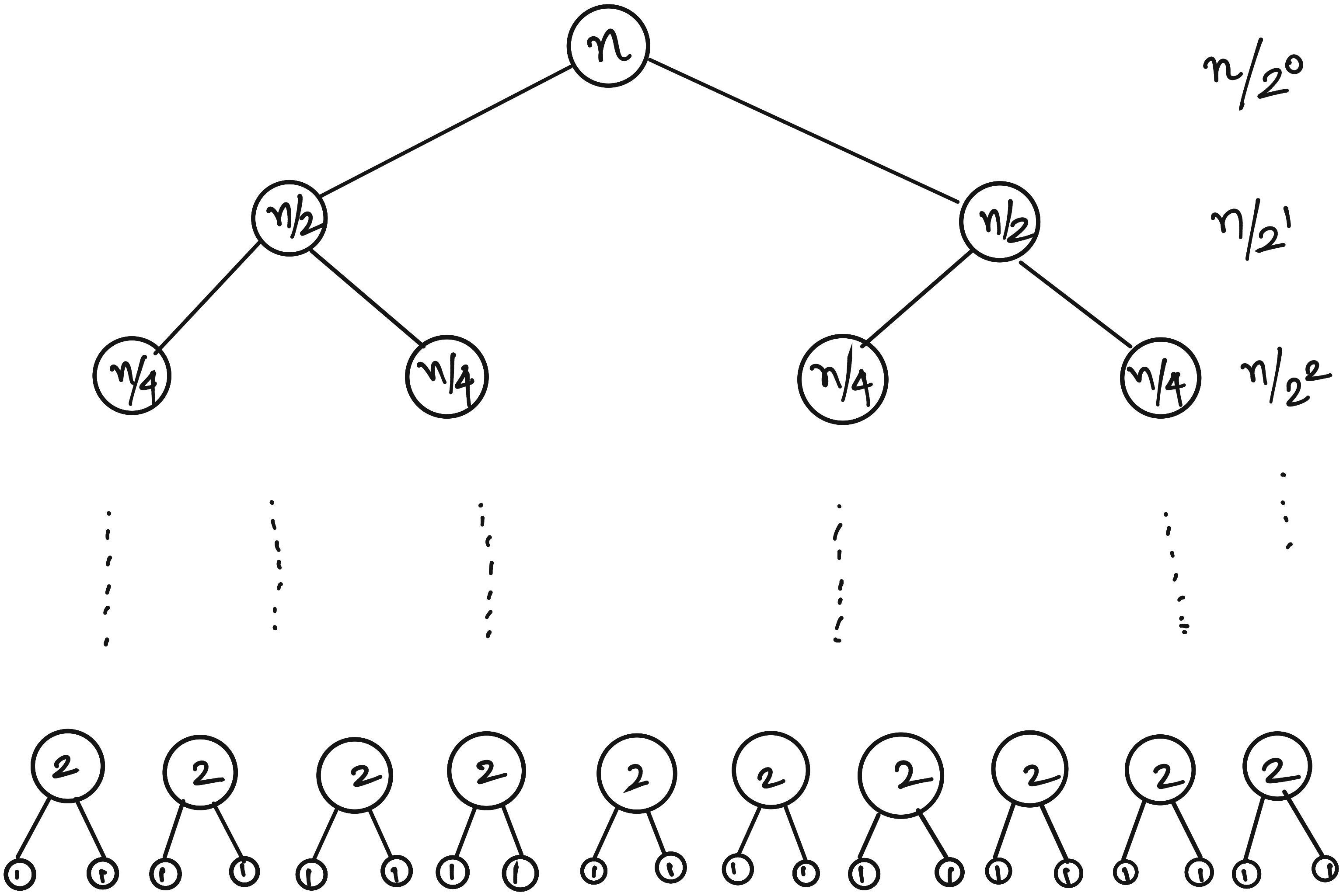


Q: WHAT IS THE HEIGHT OF THIS TREE?



Q: WHAT IS THE HEIGHT OF THIS TREE?

ASSUME THAT AT HEIGHT k, THE PROBLEM SIZE IS REDUCED TO 1.



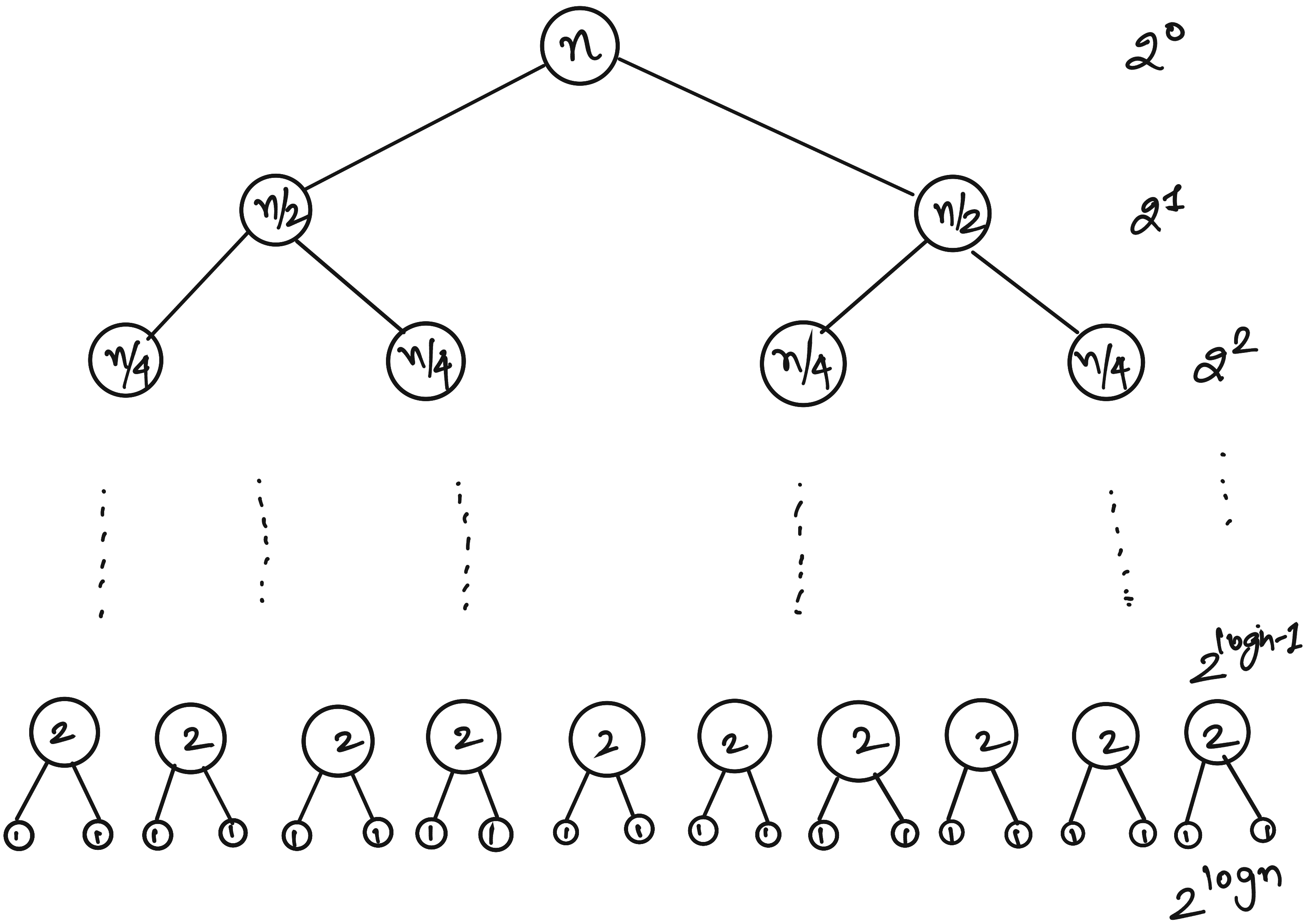
Q: WHAT IS THE HEIGHT OF THIS TREE?

ASSUME THAT AT HEIGHT k, THE PROBLEM SIZE IS REDUCED TO 1.

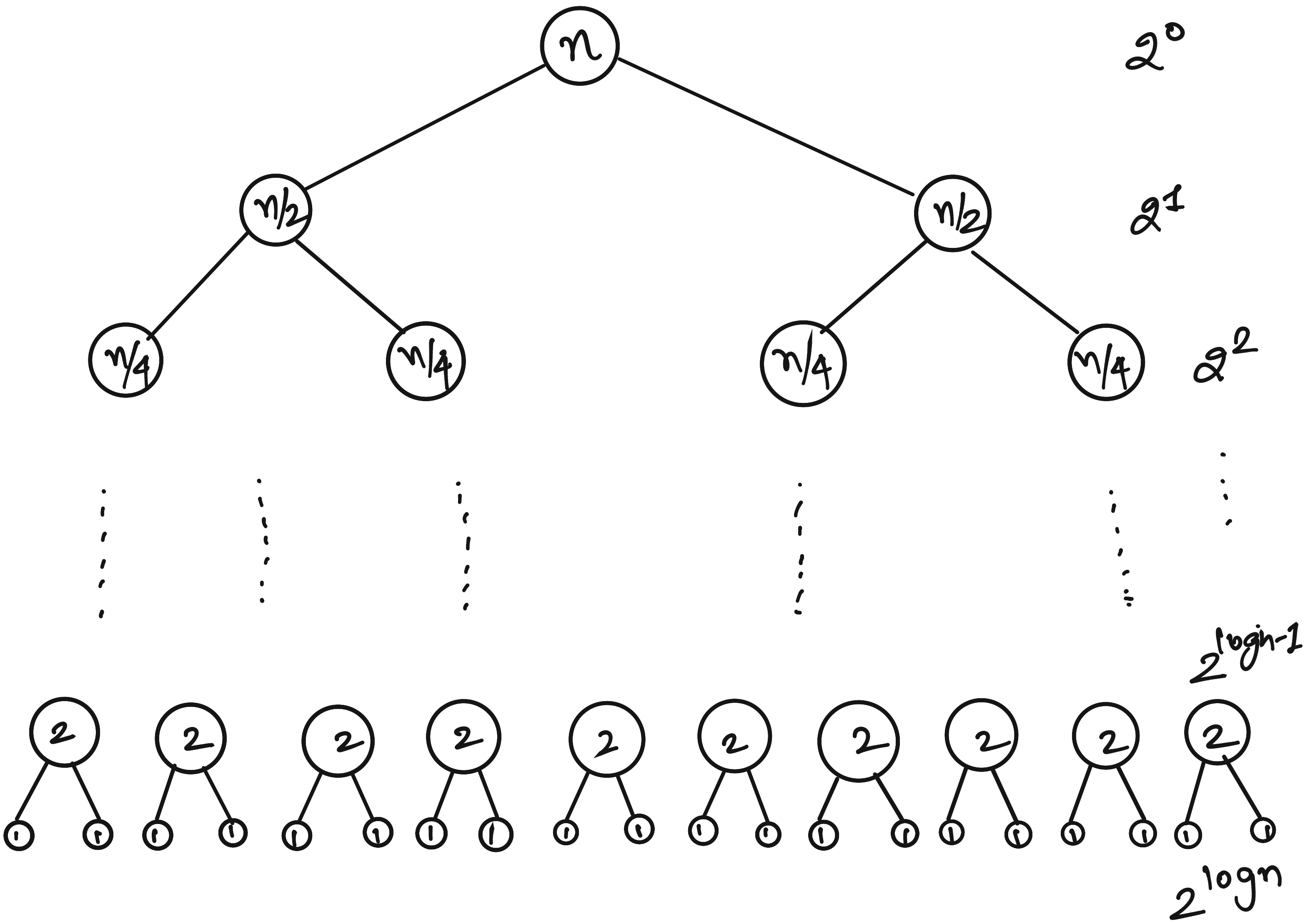
$$\frac{n}{2^k} = 1$$

$$\Rightarrow 2^k = n$$

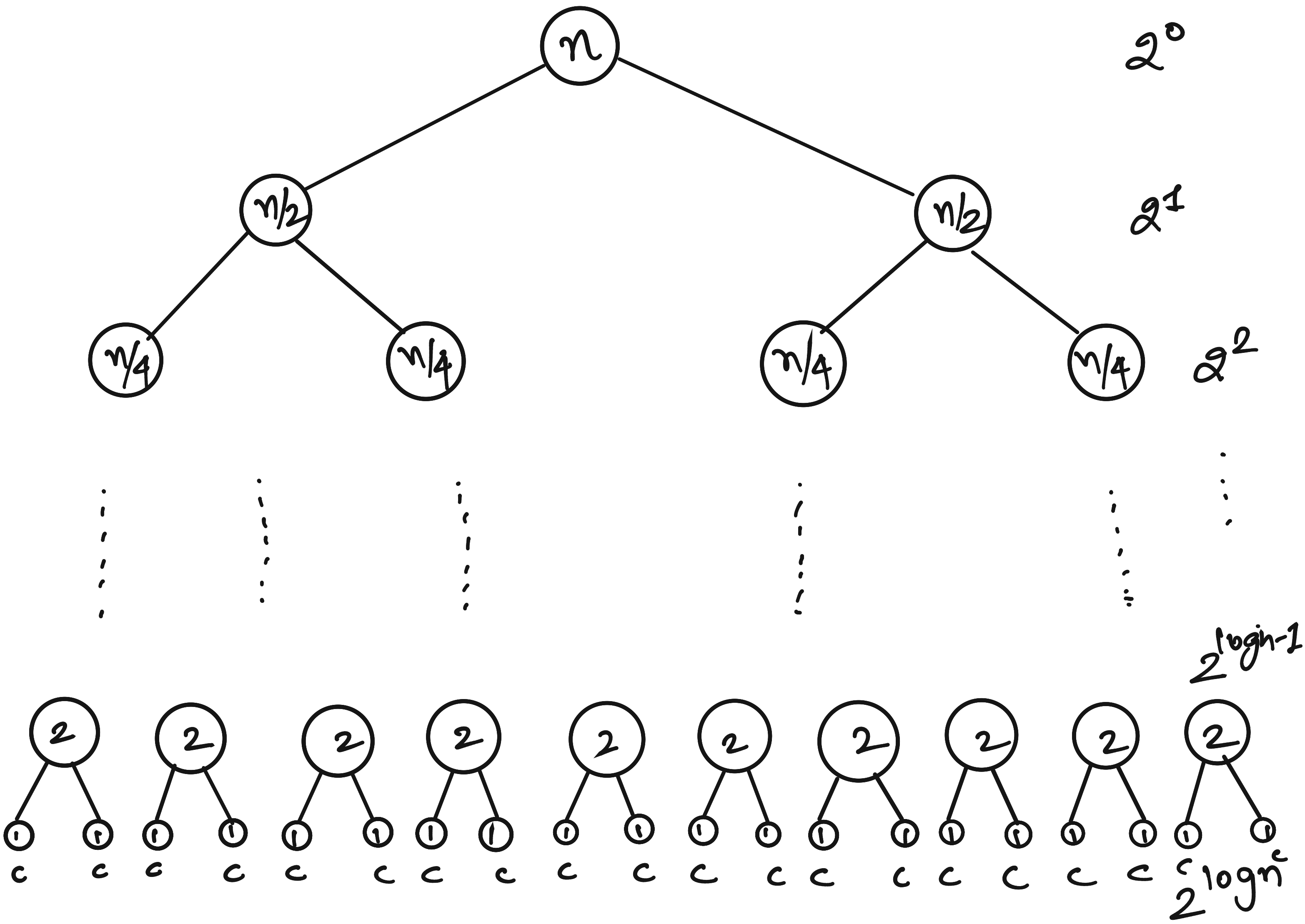
$$\Rightarrow k = \log n.$$



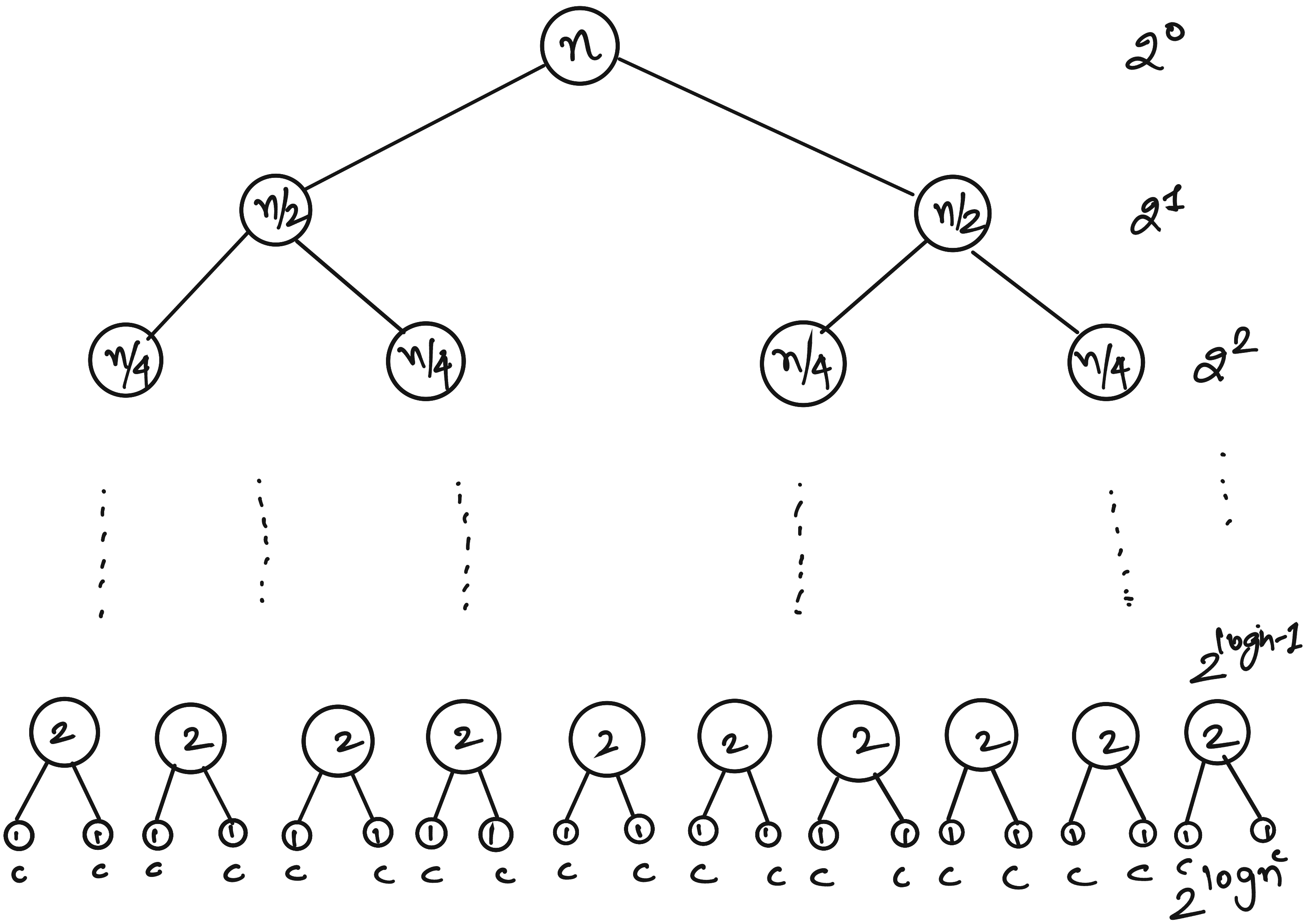
n
n/2
n/2
n/4
n/4
n/4
n/4
2
1
2^0
2^1
2^2
$2^{\log n-1}$
$2^{\log n}$



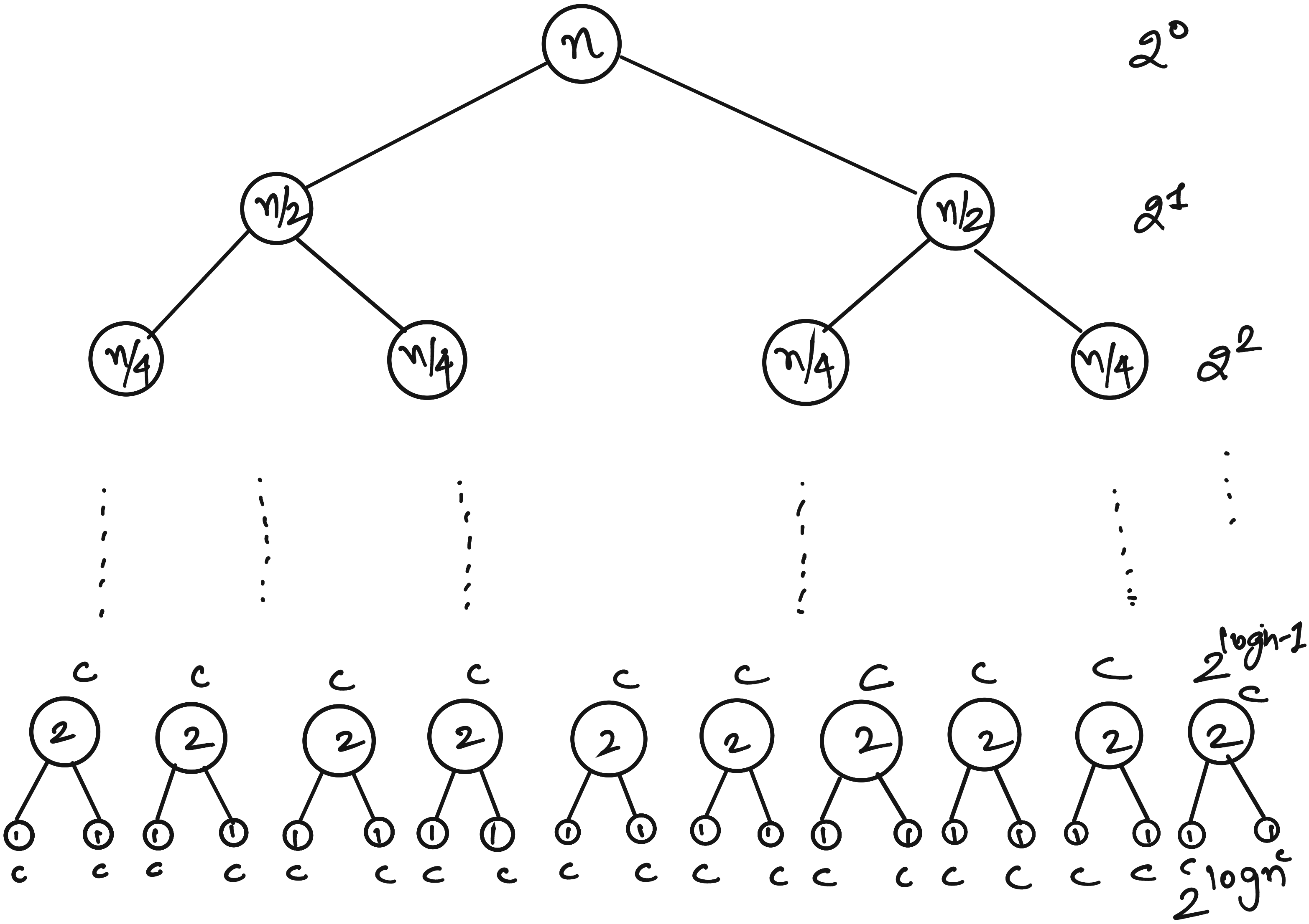
TIME REQUIRED FOR THE BASE CASE ??



TIME REQUIRED FOR THE BASE CASE = C

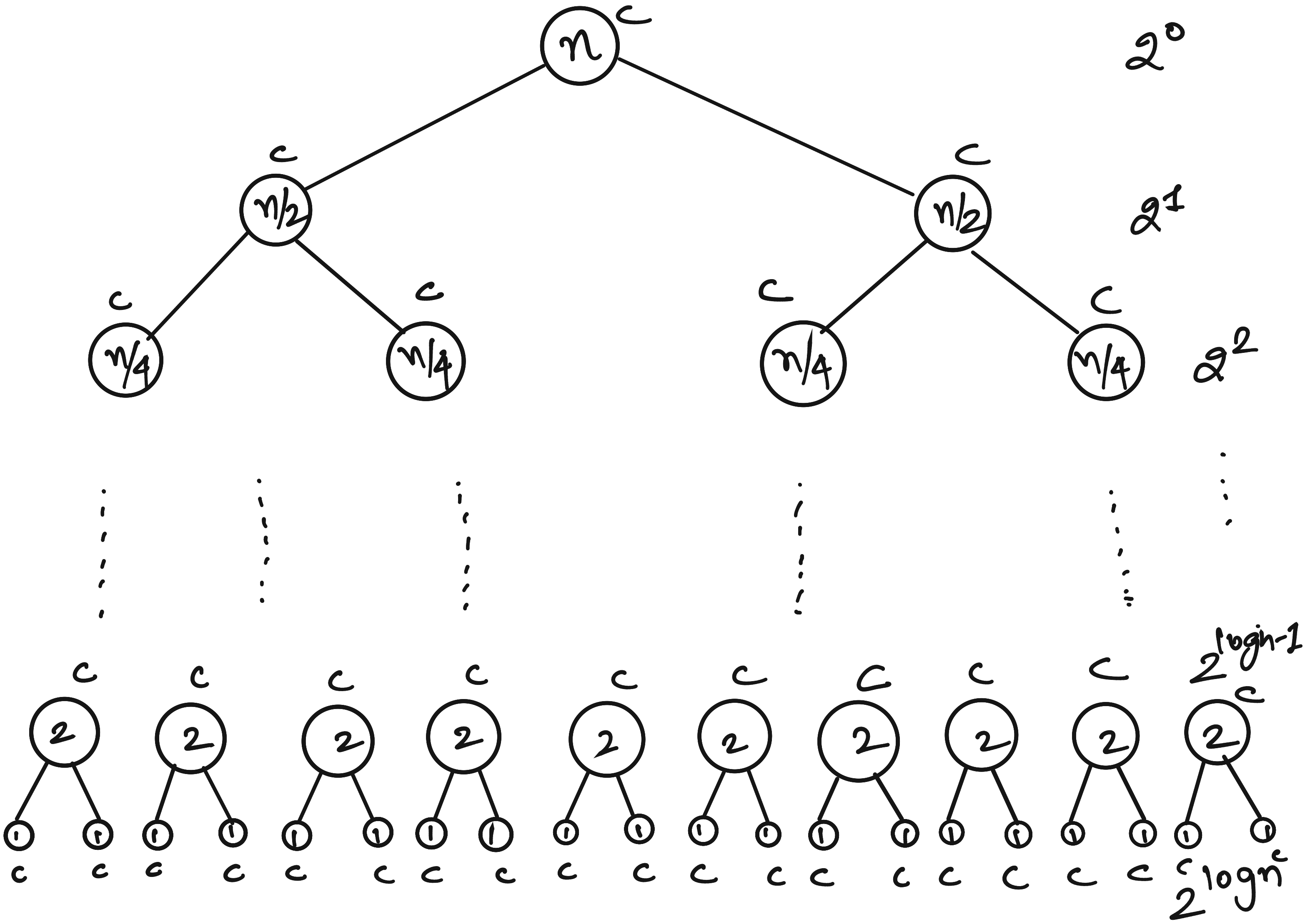


TIME REQUIRED FOR THE SECOND LAST LAYER



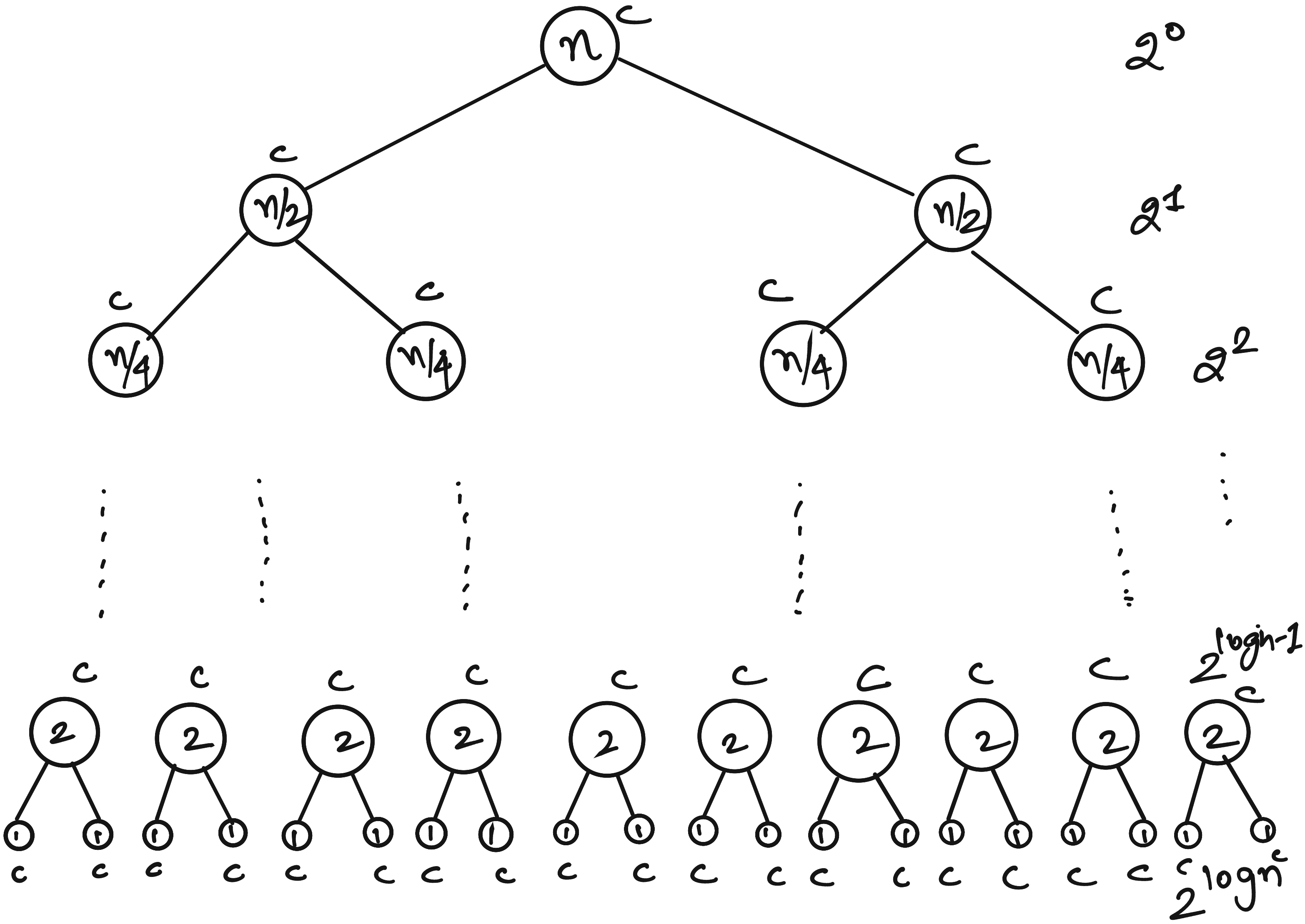
TIME REQUIRED FOR THE SECOND LAST LAYER

= C



INFACT THE TIME REQUIRED AT EACH NODE

$= c$



INFACT THE TIME REQUIRED AT EACH NODE $= c$

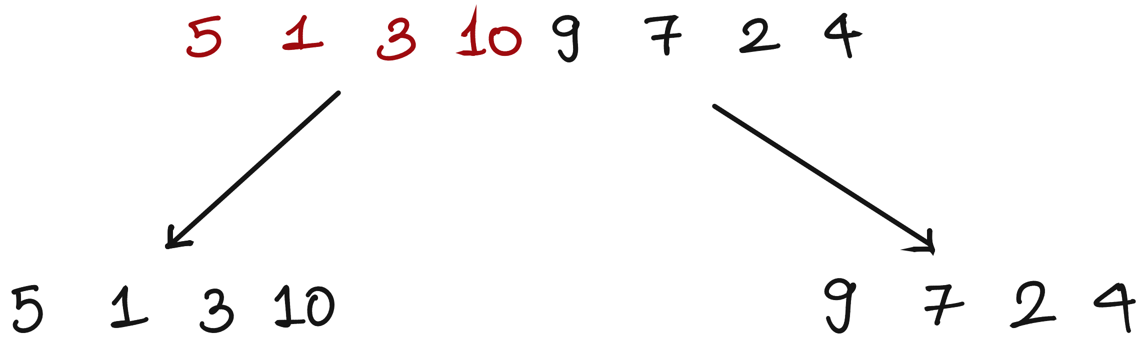
$$\text{TOTAL TIME} = c\left(2^0 + 2^1 + 2^2 + \dots + 2^{\log n}\right)$$

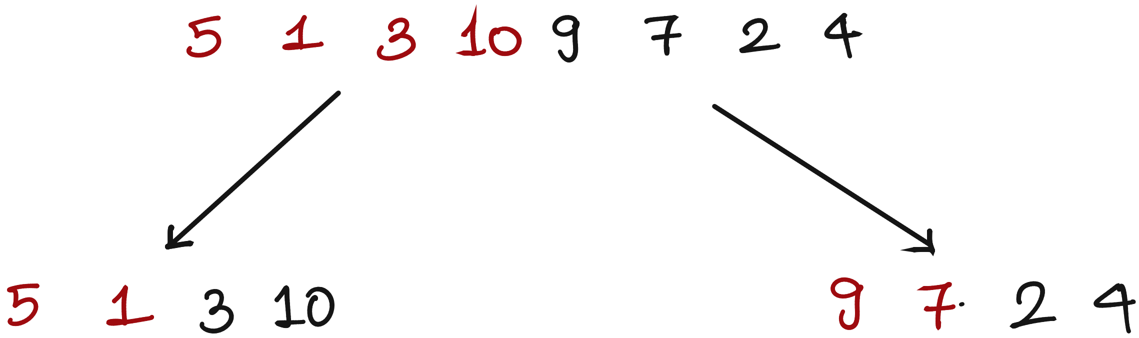
$$= \left(\frac{2^{\log n + 1} - 1}{2 - 1}\right) c$$

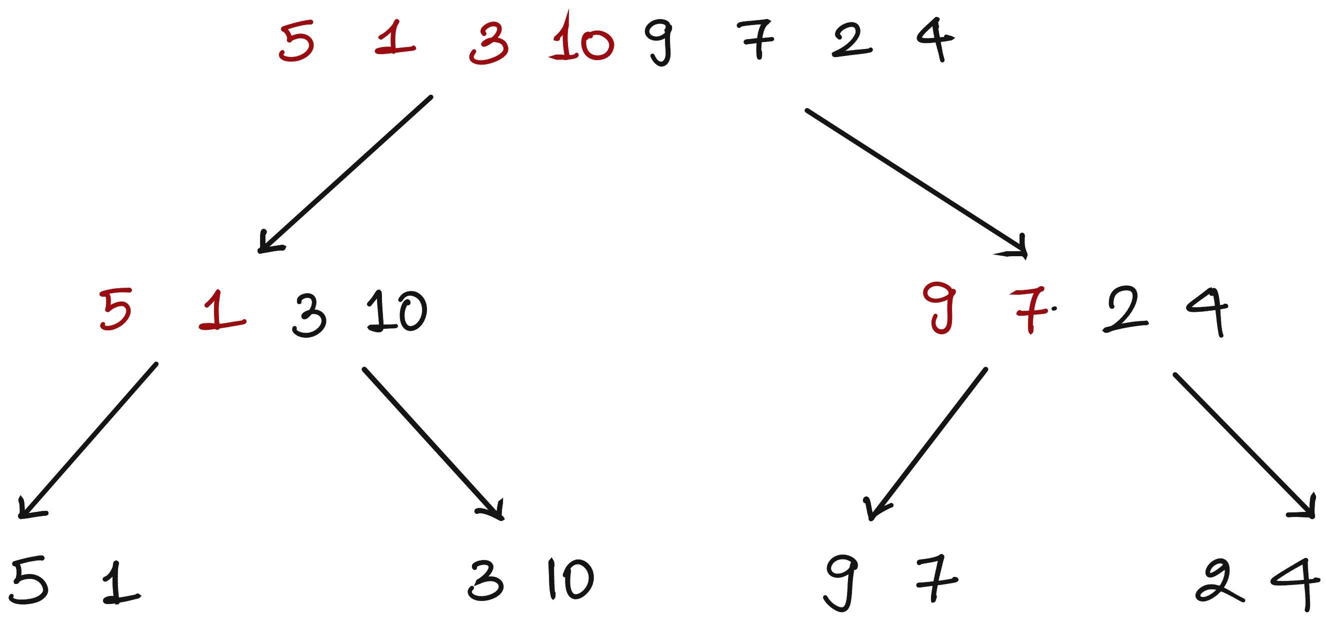
$$\leq 2nc$$

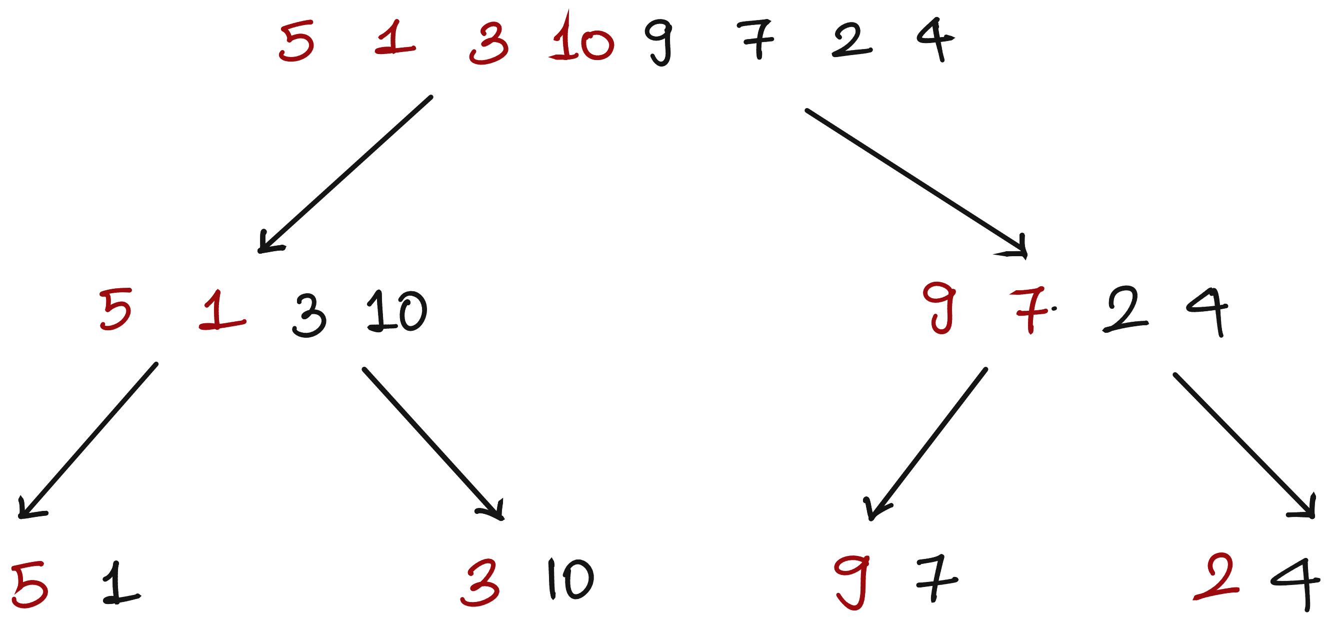
$$= O(n)$$

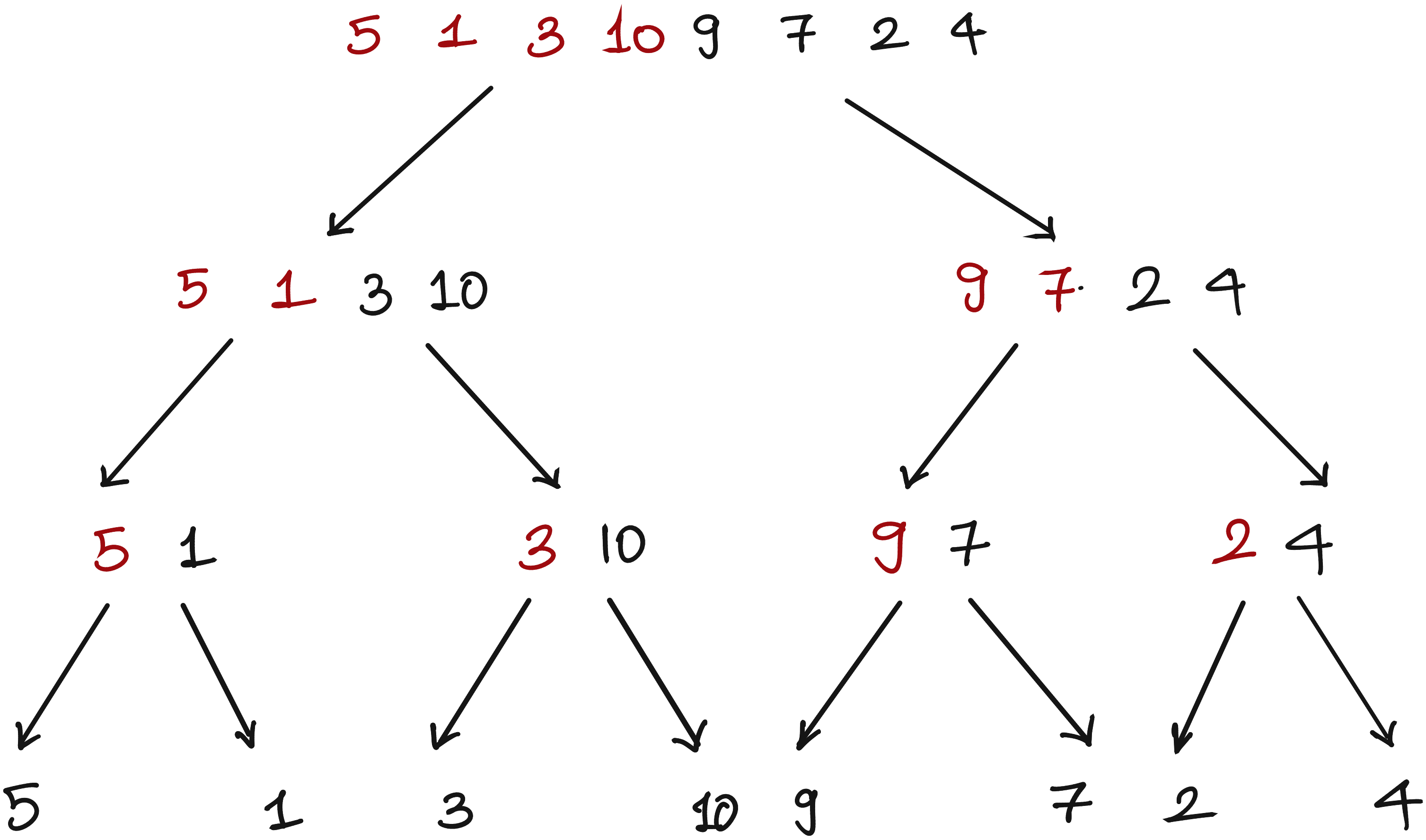
5 1 3 10 9 7 2 4

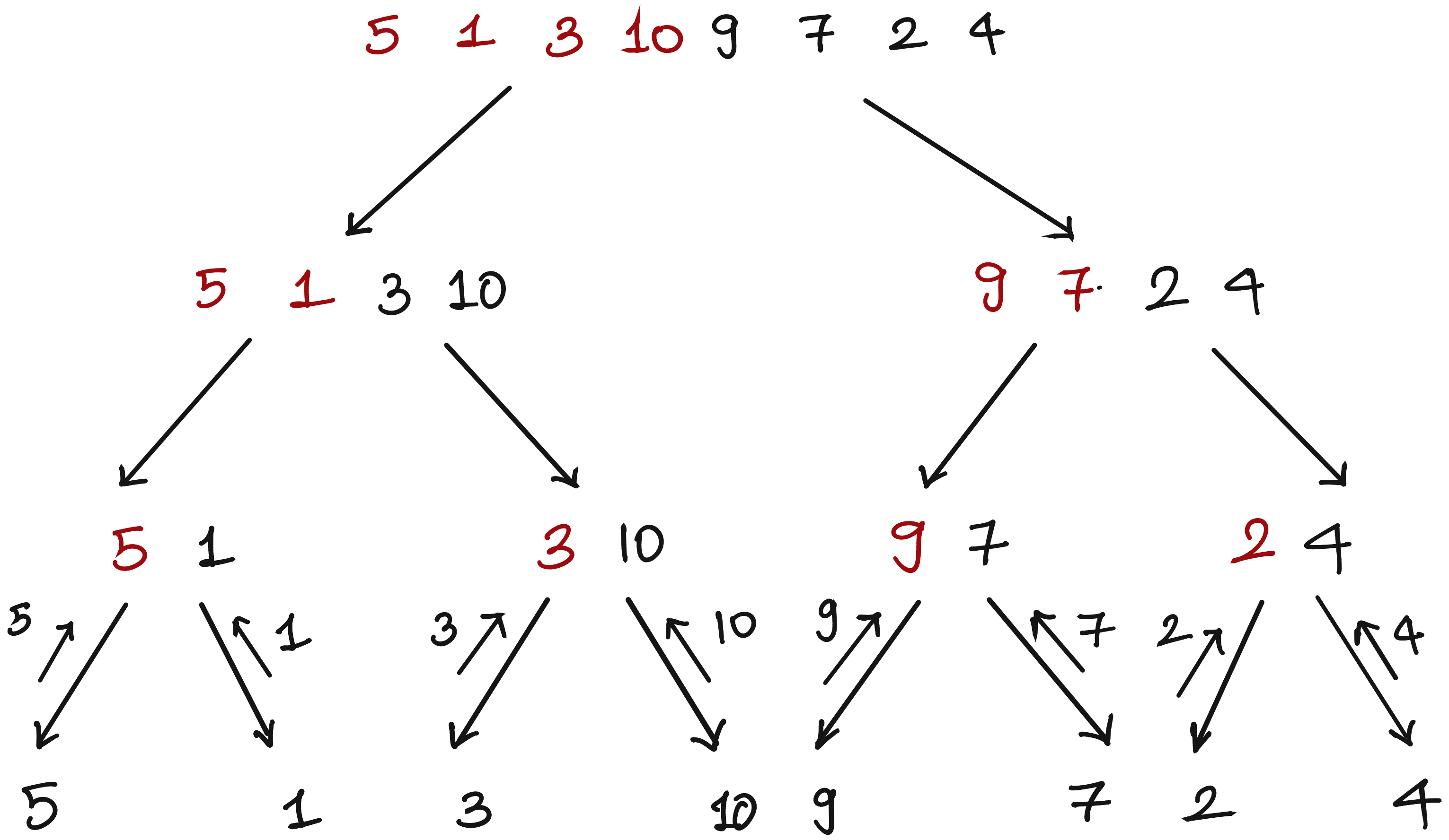

5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4

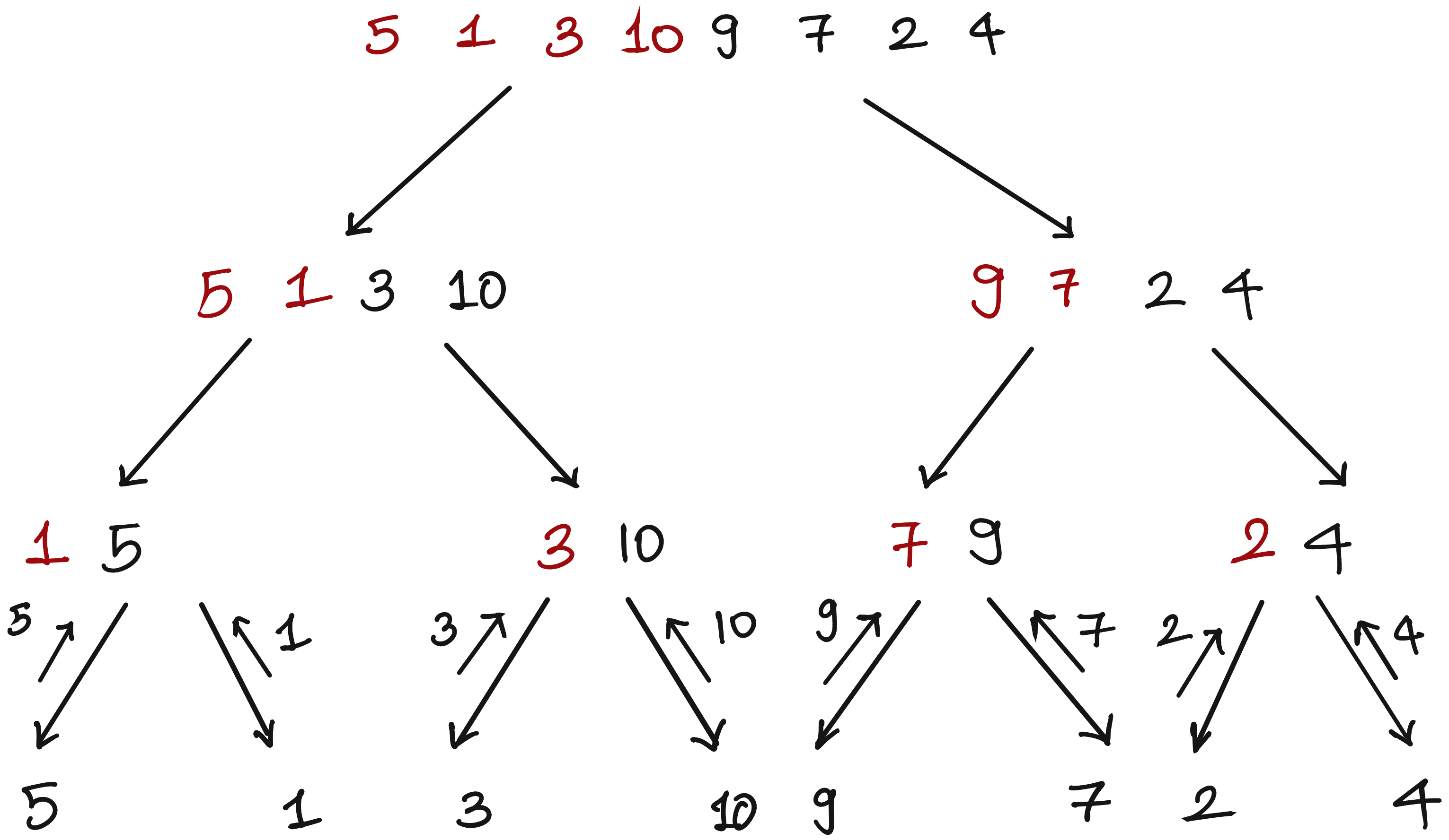

5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4


5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4
5 1
3 10
9 7
2 4

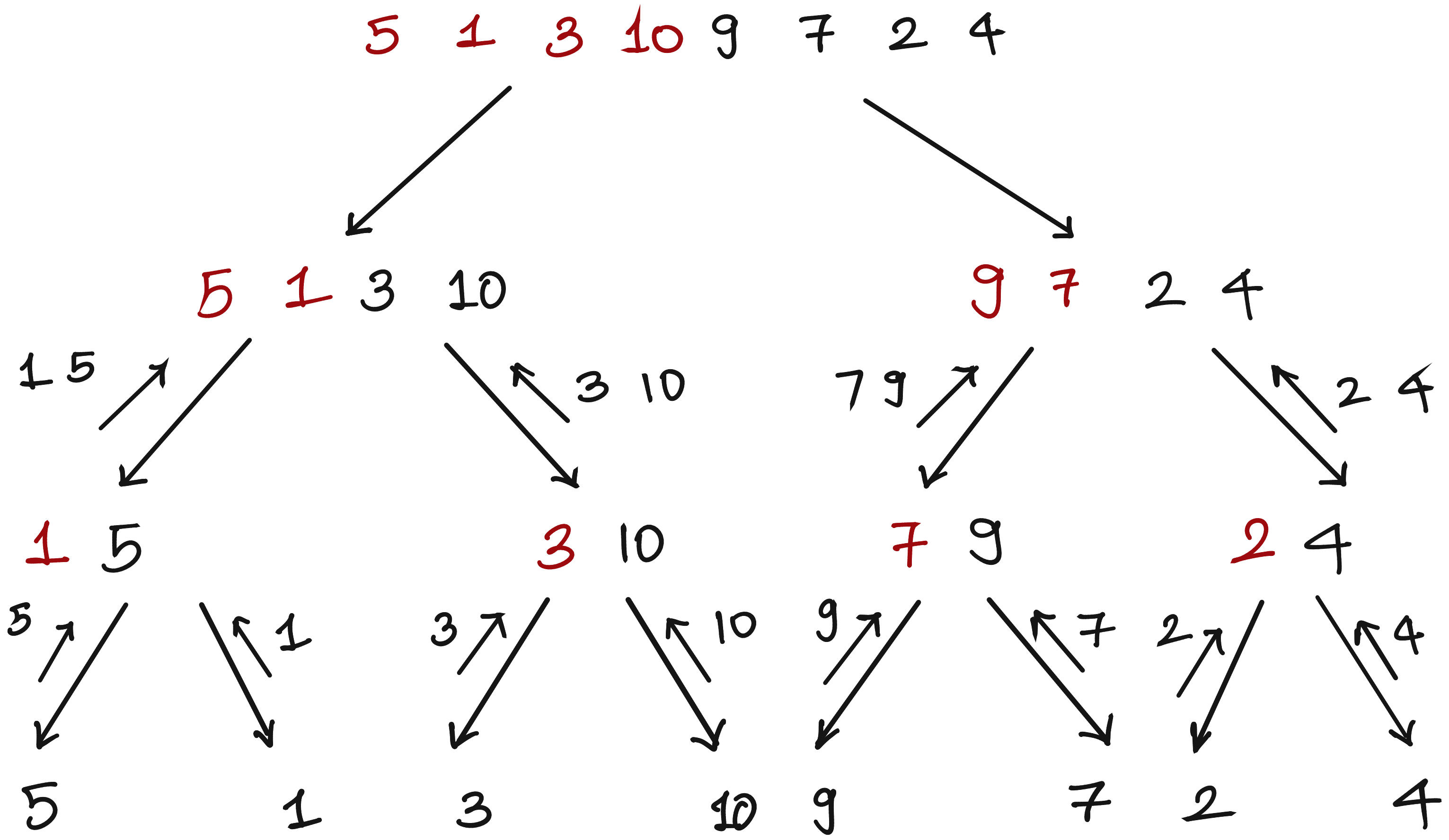

5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4
5 1
3 10
9 7
2 4

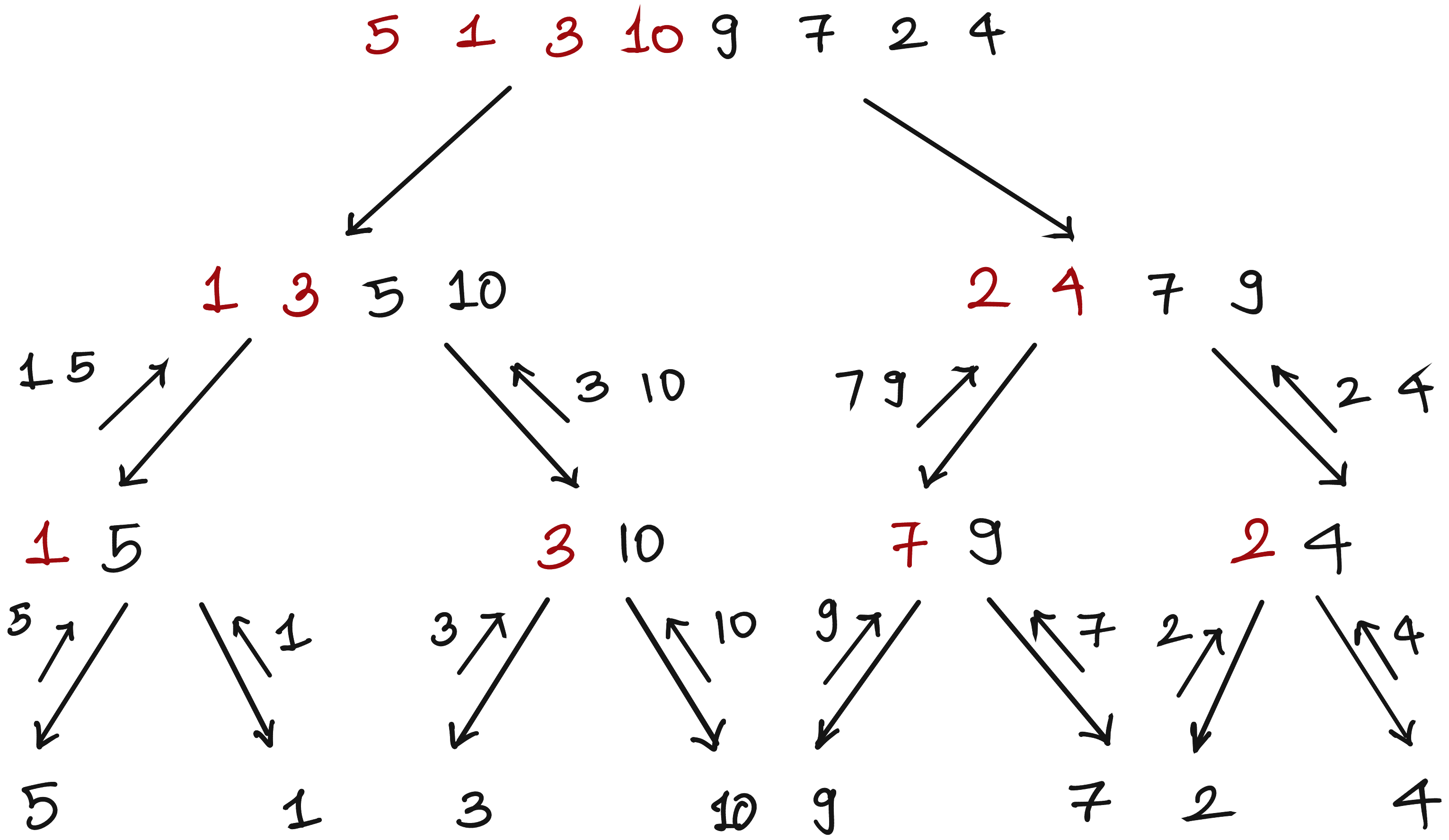

5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4
5 1
3 10
9 7
2 4
5
1
3
10
9
7
2
4

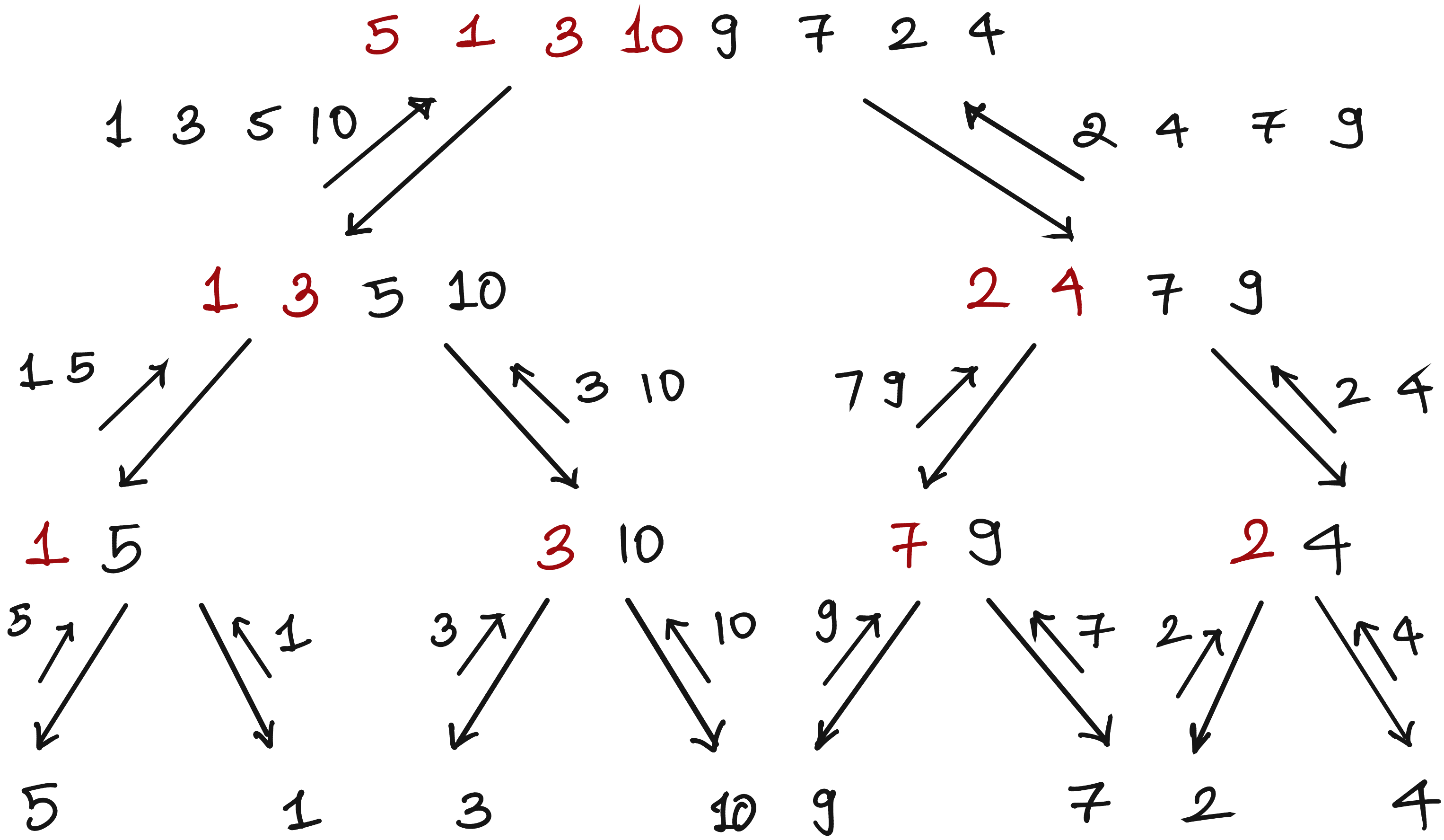

5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4
5 1
3 10
9 7
2 4
5
1
3
10
9
7
2
4
5 1 3 10 9 7 2 4



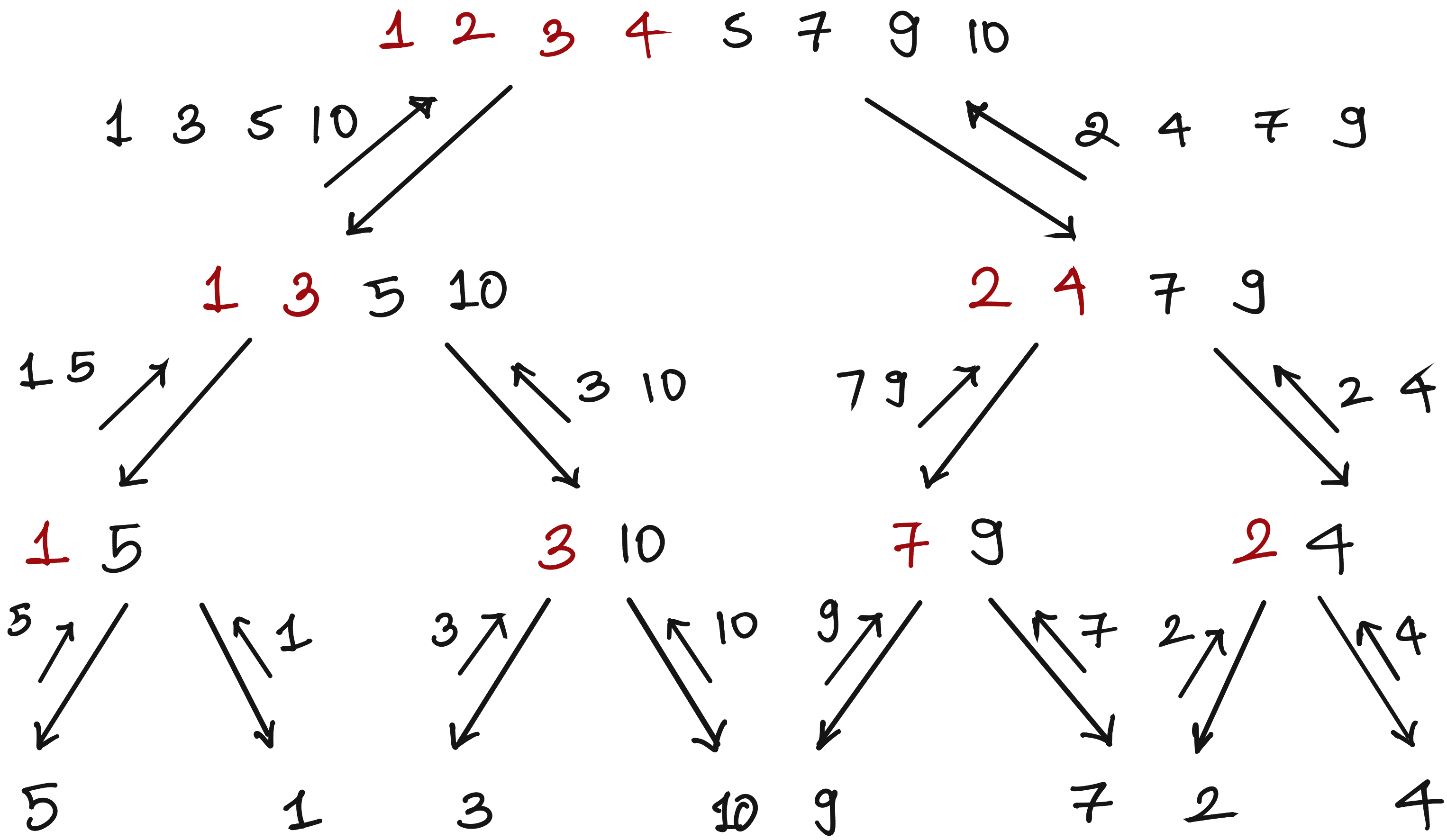
5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4
1 5
3 10
7 9
2 4
5 1 3 10 9 7 2 4

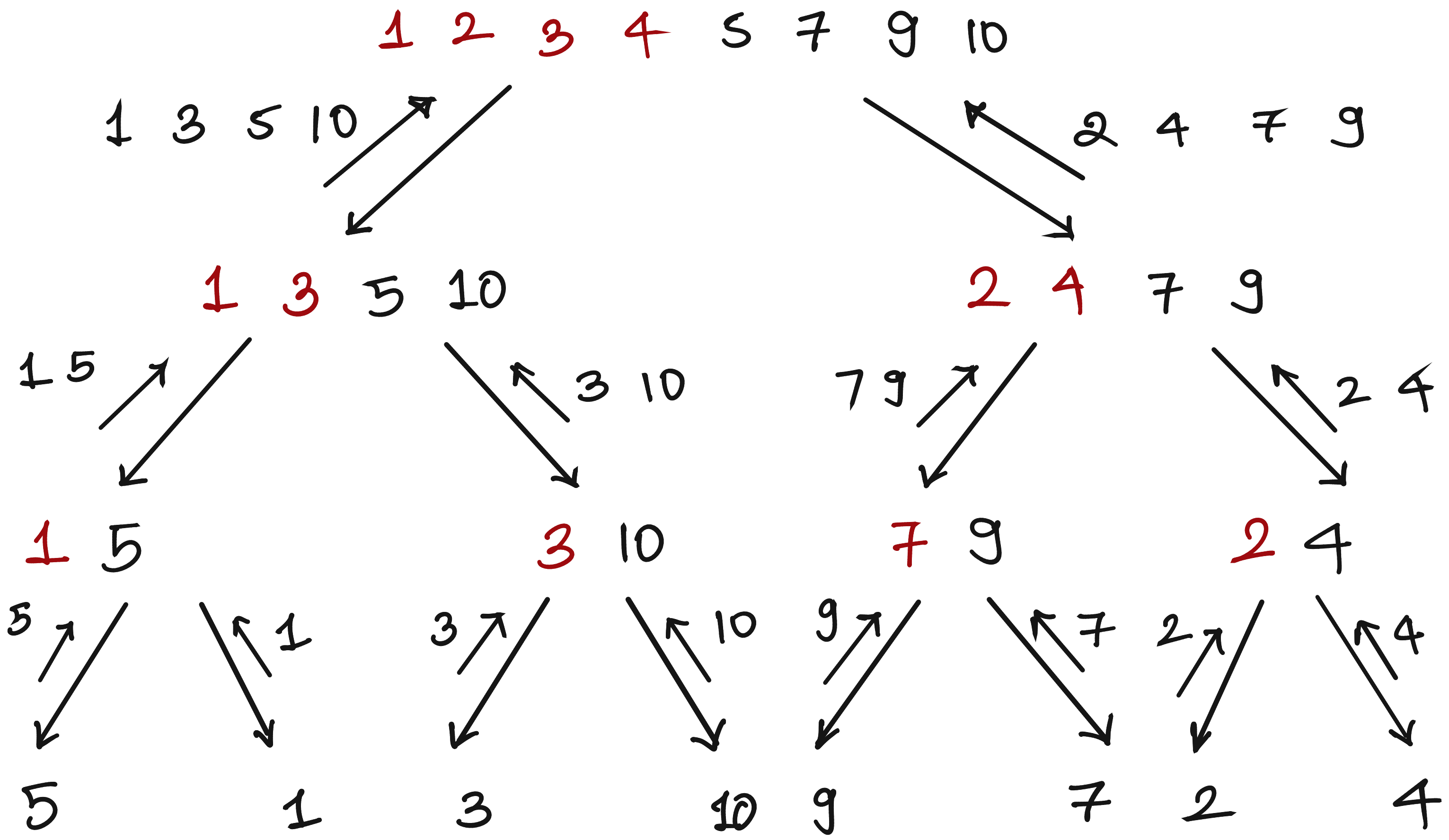

5 1 3 10 9 7 2 4
5 1 3 10
9 7 2 4
1 5
3 10
7 9
2 4
1 5
3 10
7 9
2 4
5
1
3
10
9
7
2
4
5
1
3
10
9
7
2
4


5 1 3 10 9 7 2 4
1 3 5 10
2 4 7 9
1 5
3 10
7 9
2 4
1 5
3 10
7 9
2 4
5
1
3
10
9
7
2
4
5
1
3
10
9
7
2
4



5 1 3 10 9 7 2 4
1 3 5 10
2 4 7 9
1 3 5 10
2 4 7 9
1 5
3 10
7 9
2 4
1 5
3 10
7 9
2 4
5
1
3
10
9
7
2
4
5
1
3
10
9
7
2
4


1 2 3 4 5 7 9 10
1 3 5 10
2 4 7 9
1 3 5 10
2 4 7 9
1 5
3 10
7 9
2 4
1 5
3 10
7 9
2 4
5
1
3
10
9
7
2
4
5
1
3
10
9
7
2
4




```
MERGESORT ( A[1...n])
{   if ( size(A) = 1)
        RETURN A;
    B ← MERGESORT (A[1....n/2]);
    C ← MERGESORT ( A[n/2+1... n]);
    D ← MERGE(B,C);
    RETURN D;
}
```

CORRECTNESS

CORRECTNESS

THE PATTERN LIES IN THE ALGORITHM CODE

CORRECTNESS

THE PATTERN LIES IN THE ALGORITHM CODE

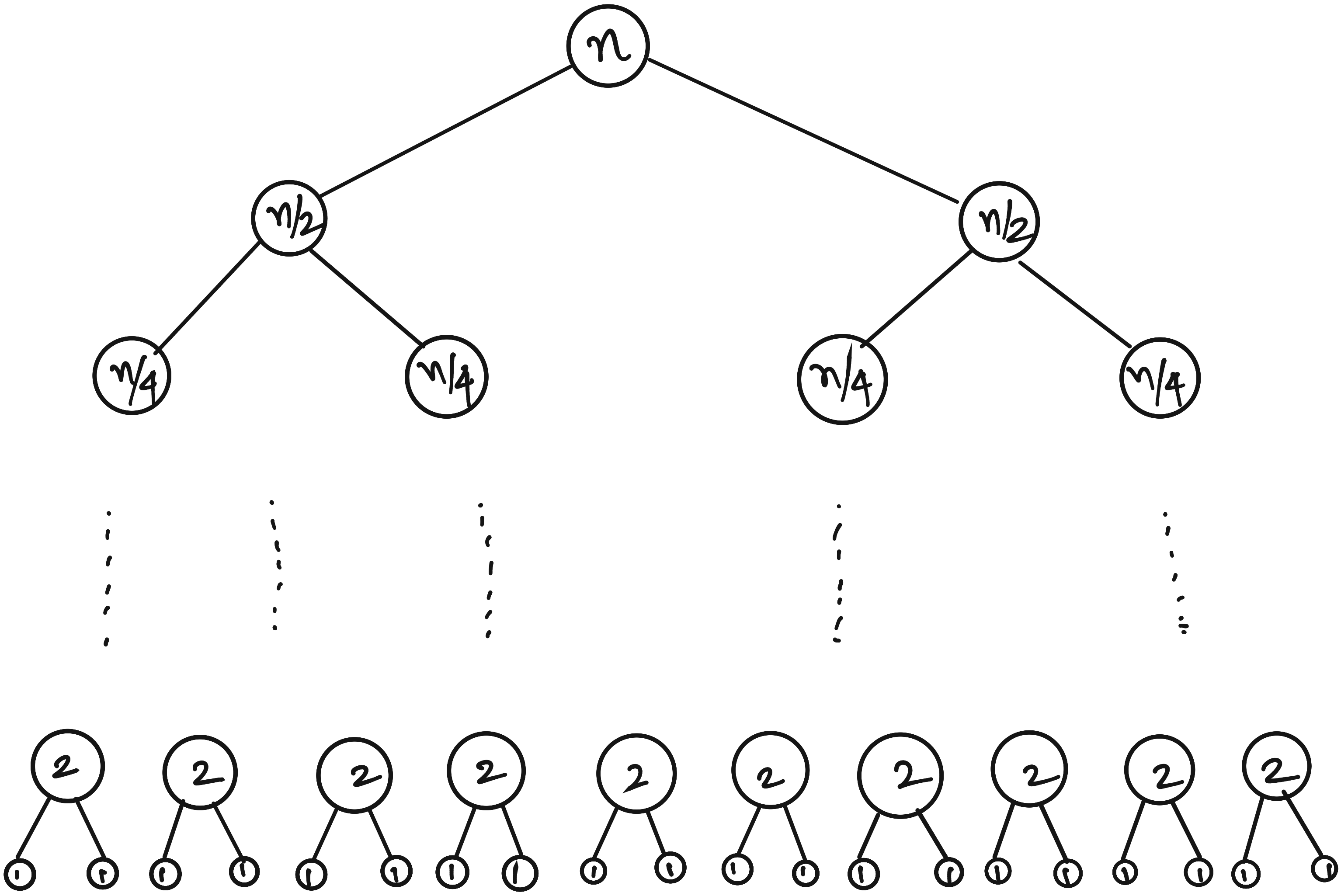
LEMMA : MERGESORT CORRECTLY SORTS $\underbrace{n \text{ NUMBERS}}_{\text{INDUCTION ON } n}$

CORRECTNESS

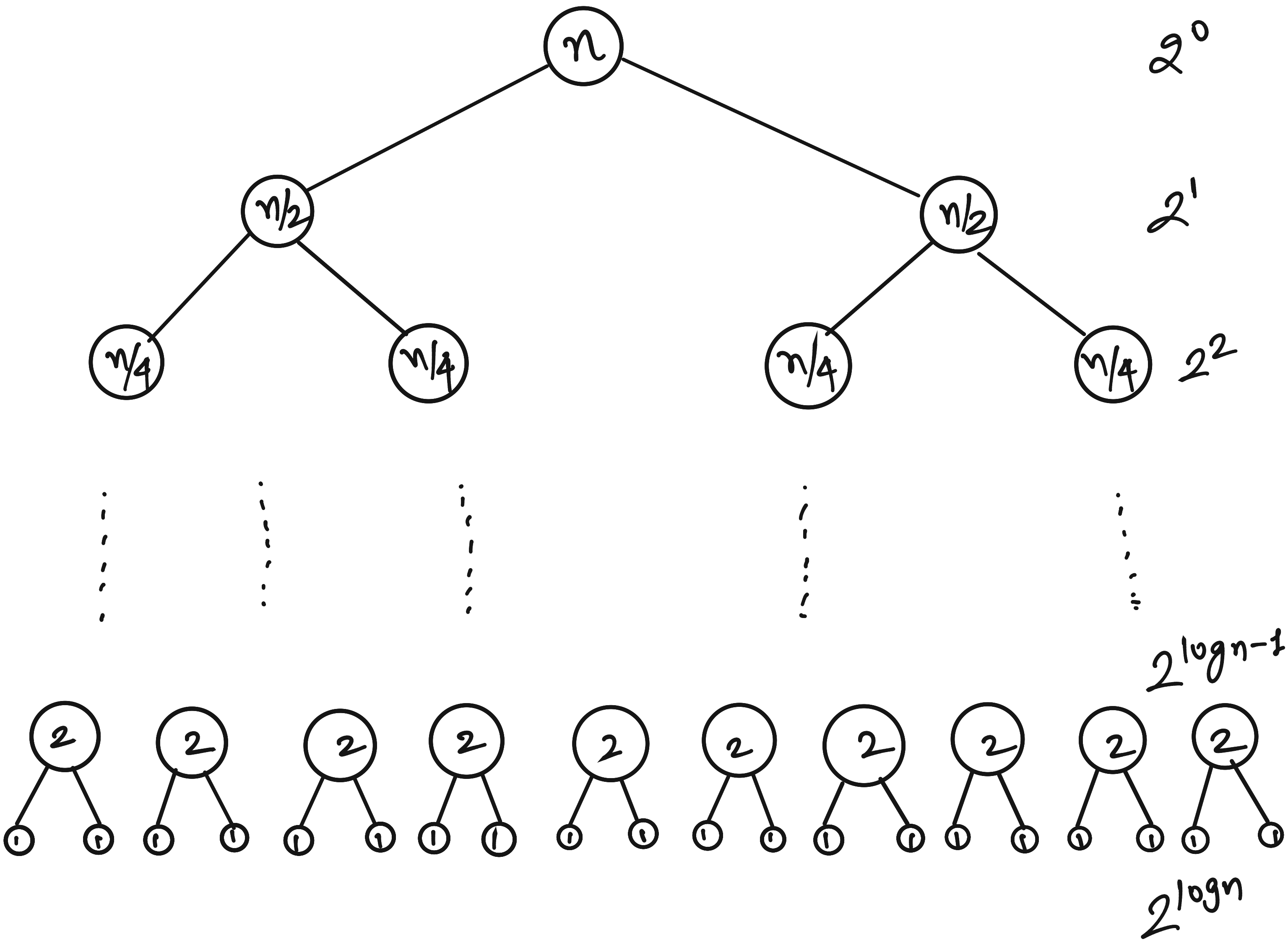
THE PATTERN LIES IN THE ALGORITHM CODE

LEMMA: MERGESORT CORRECTLY SORTS $\underbrace{n \text{ NUMBERS}}_{\text{INDUCTION ON } n}$

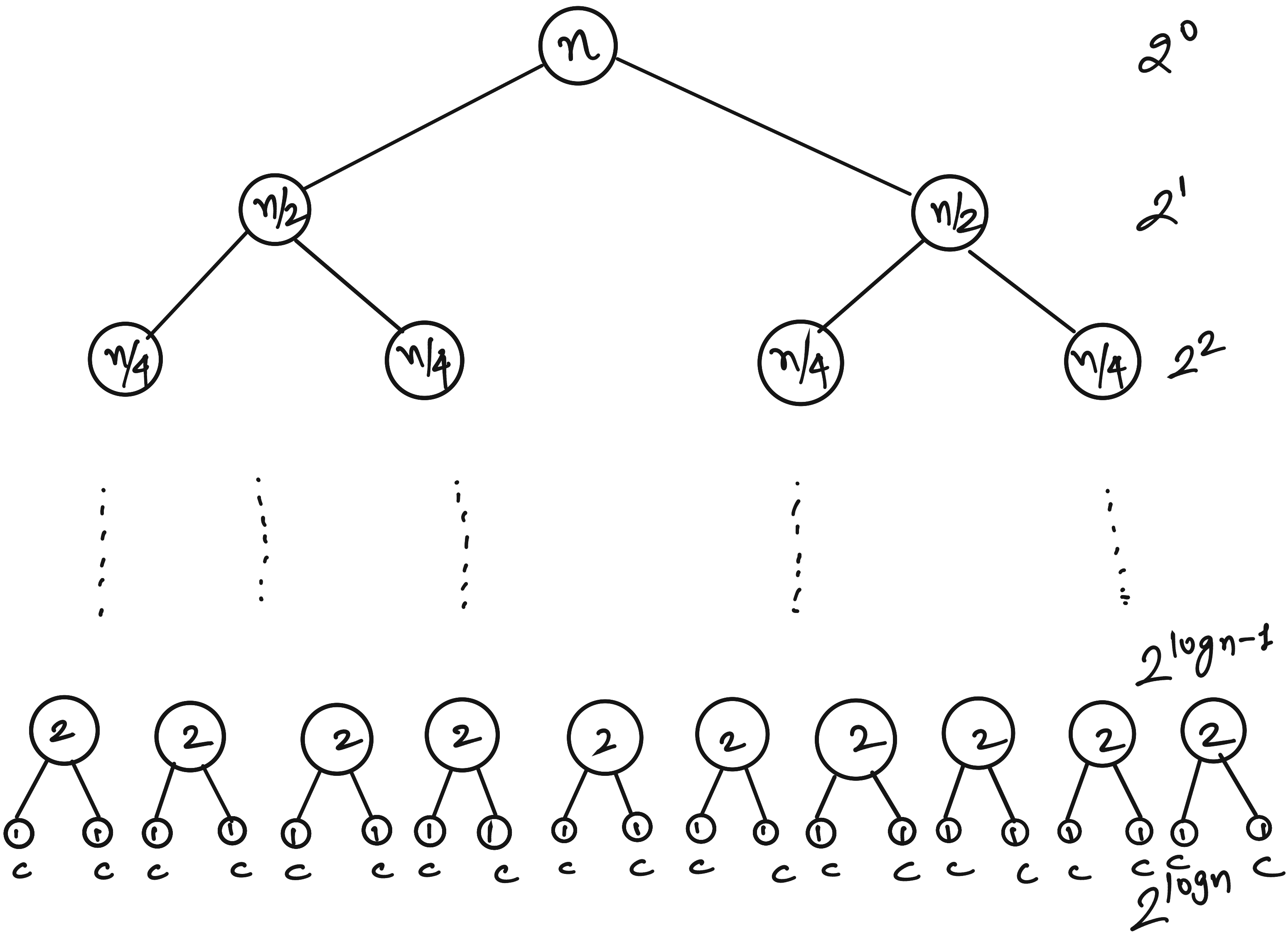
RUNNING TIME:



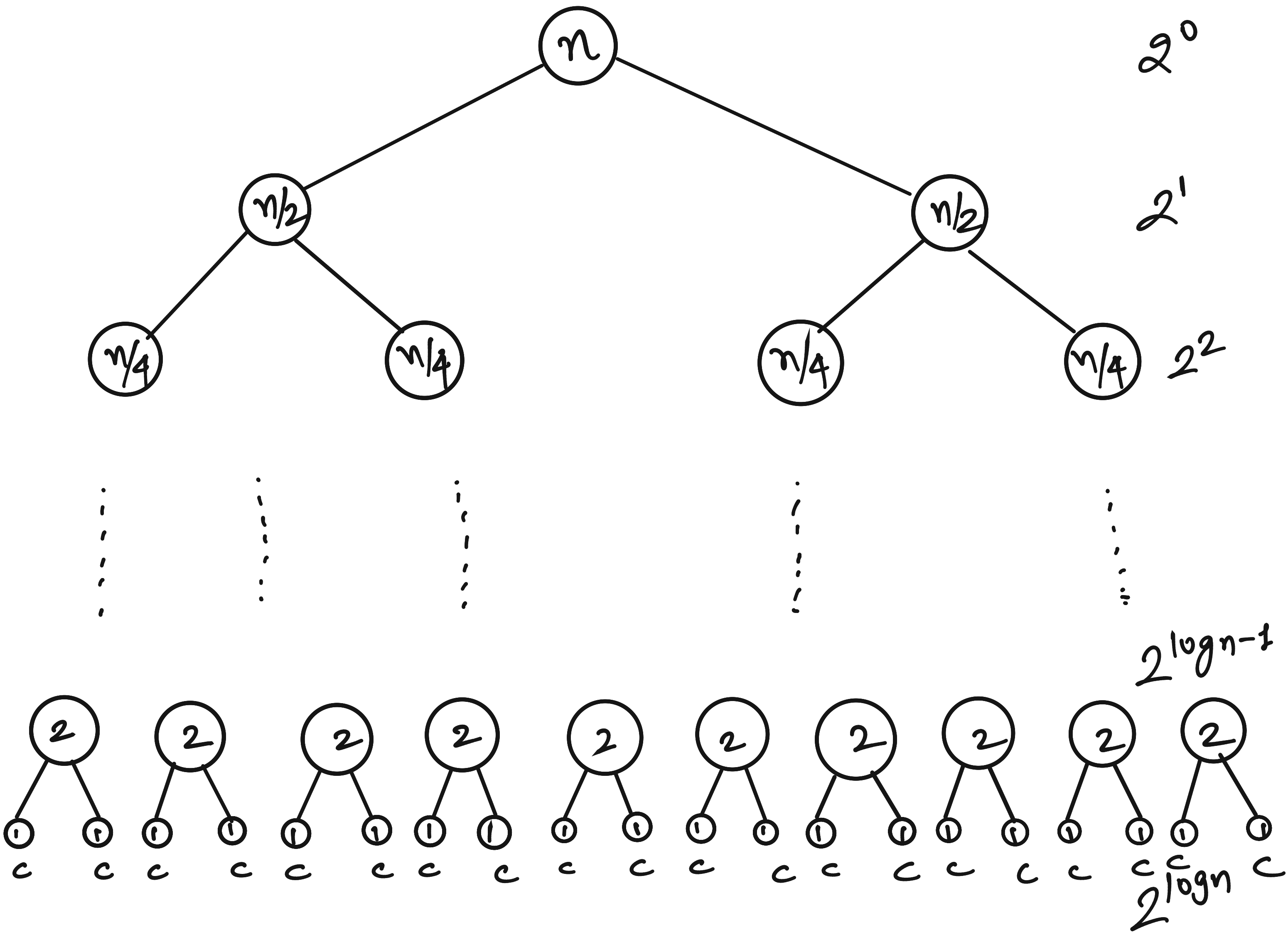
Q: WHAT IS THE HEIGHT OF THIS TREE ?



TIME TAKEN FOR BASE CASE

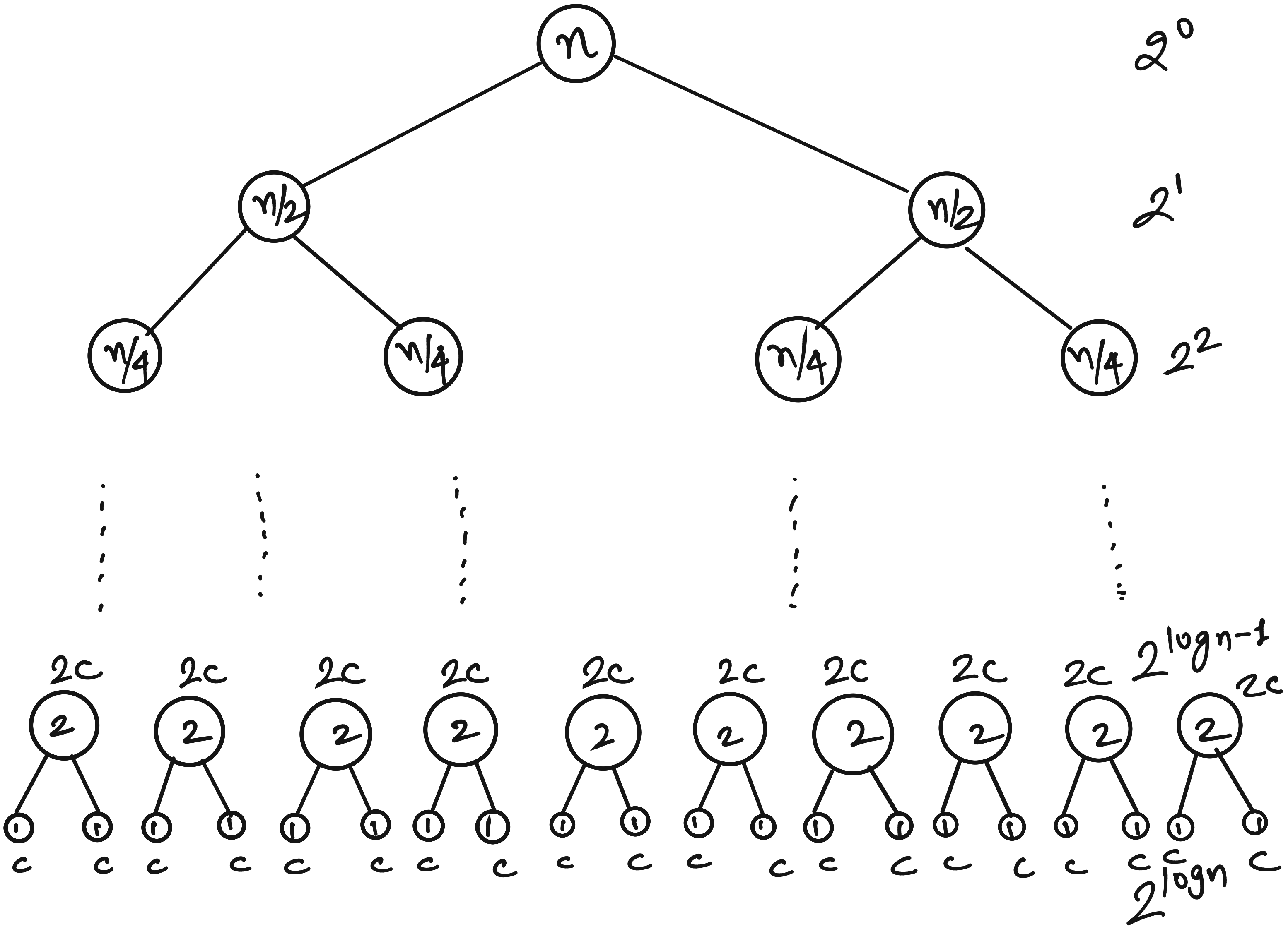


TIME TAKEN FOR BASE CASE = c



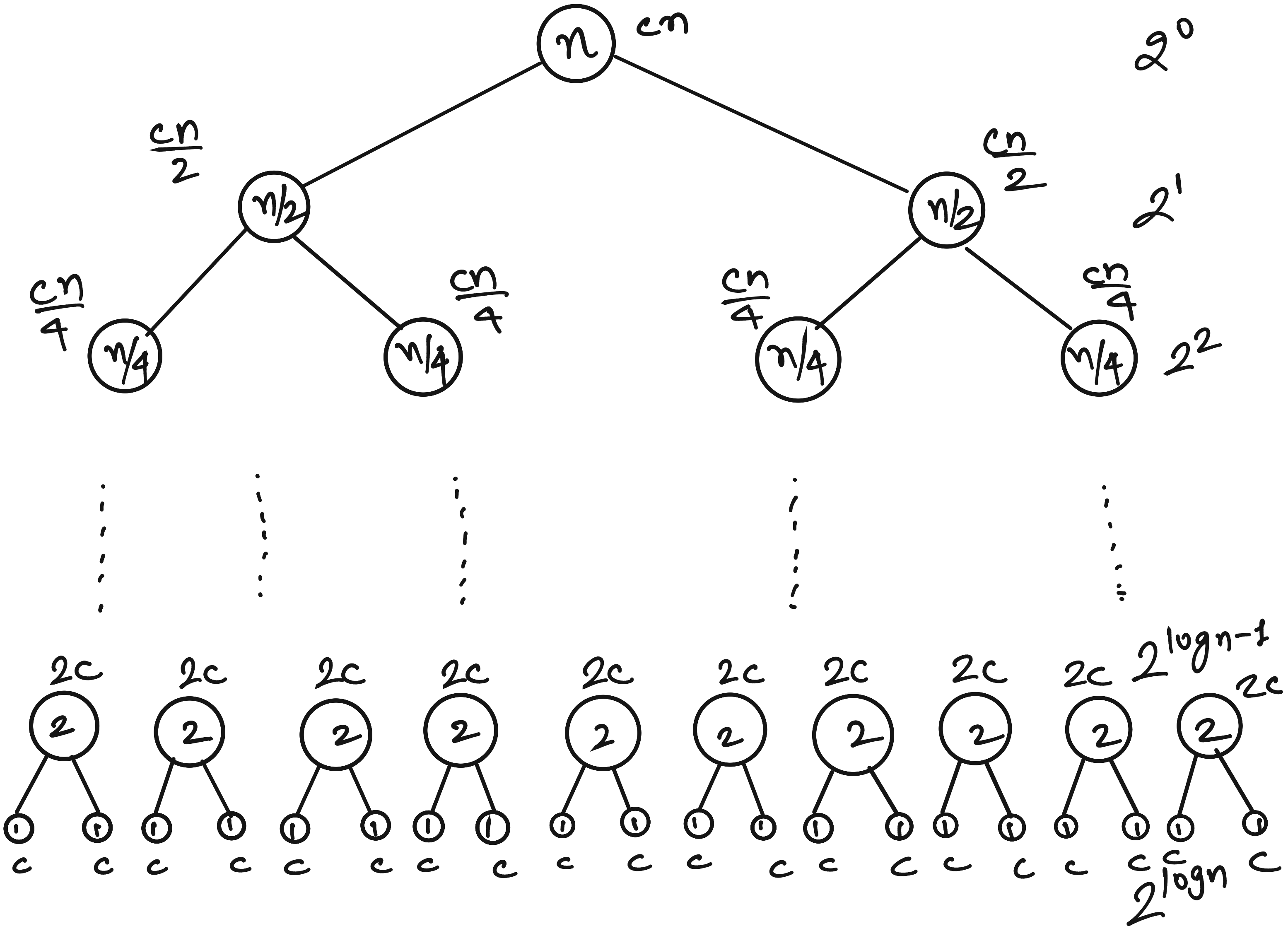
TIME TAKEN FOR THE SECOND LAST LAYER

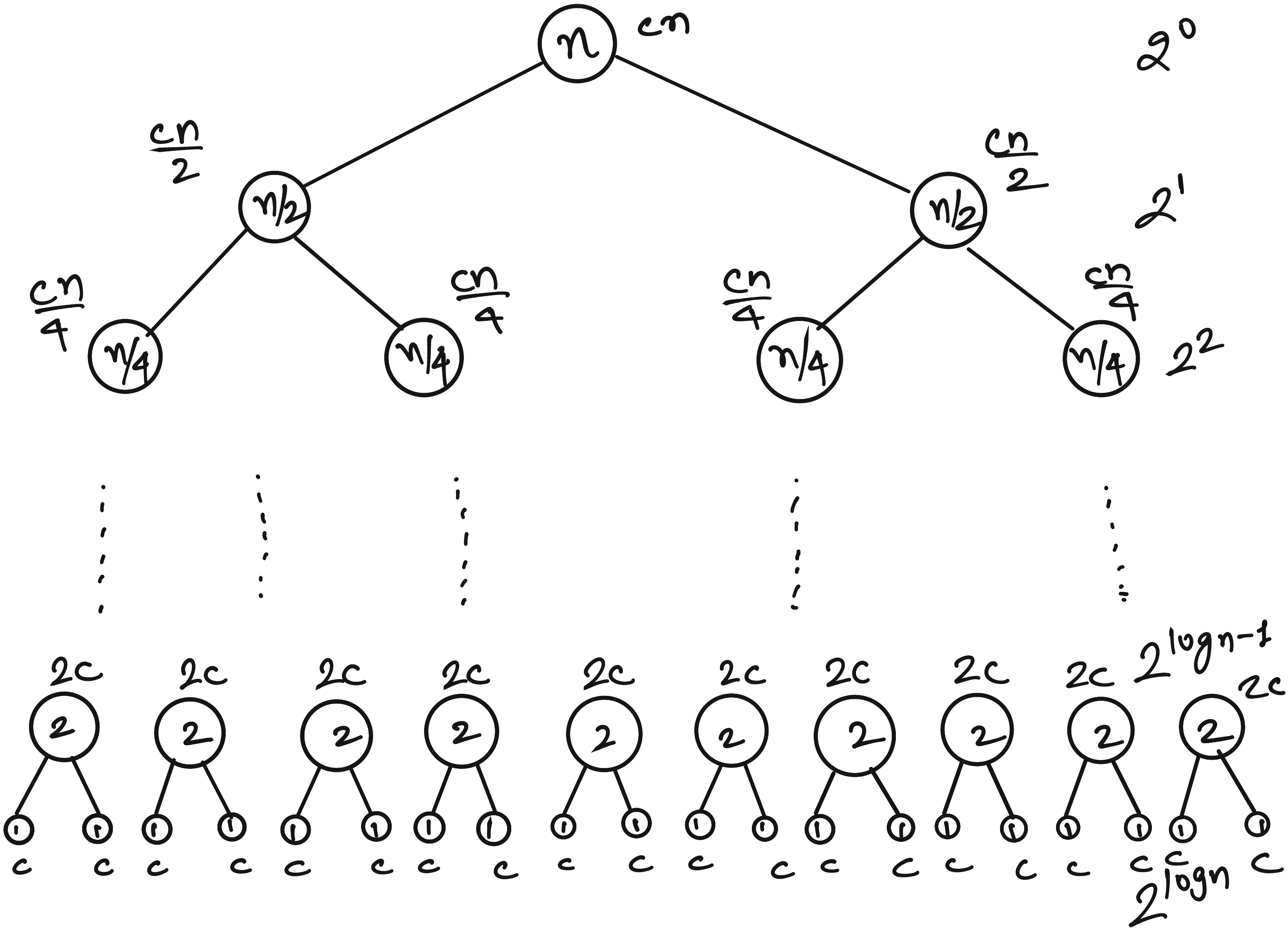
=



TIME TAKEN FOR THE SECOND LAST LAYER

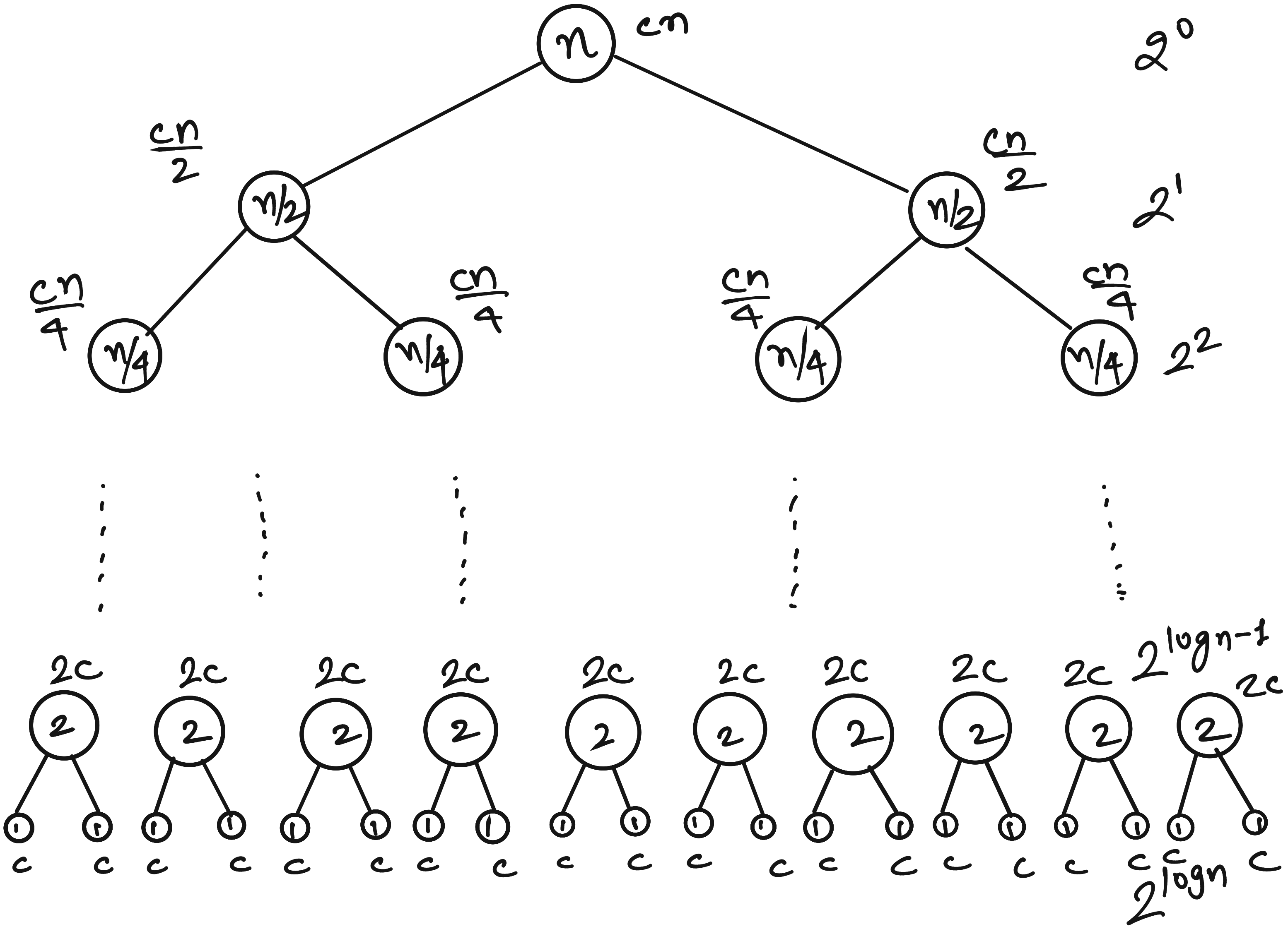
$= 2c$


n
cn
2^0
n/2
$\frac{cn}{2}$
n/2
$\frac{cn}{2}$
2^1
n/4
$\frac{cn}{4}$
n/4
$\frac{cn}{4}$
n/4
$\frac{cn}{4}$
n/4
$\frac{cn}{4}$
2^2
2c
2
$2^{\log n - 1}$
1
c
$2^{\log n}$



TOTAL RUNTIME

$$= c \cdot 2^{\log n} + 2c \cdot 2^{\log n - 1} + 4c \cdot 2^{\log n - 2} + \dots + \frac{cn}{4} \cdot 2^2 + \frac{cn}{2} \cdot 2^1 + cn$$



TOTAL RUNTIME

$$= c \cdot 2^{\log n} + 2c \cdot 2^{\log n - 1} + 4c \cdot 2^{\log n - 2} + \dots + \frac{cn}{4} \cdot 2^2 + \frac{cn}{2} \cdot 2^1 + cn$$

$$= cn + cn + \dots + cn + cn$$

$$= cn \log n$$

$$= O(n \log n).$$

ONE MORE WAY TO FIND THE RUNNING TIME

LET T(N) BE THE TIME TO SORT N NUMBERS USING MERGESORT

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N NUMBERS USING MERGESORT

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

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N NUMBERS USING MERGESORT

$$T(n) = \underbrace{T\left(\frac{n}{2}\right)}_{\text{SORT LEFT}} + \underbrace{T\left(\frac{n}{2}\right)}_{\text{SORT RIGHT}} + \underbrace{cn}_{\text{MERGE}}$$

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N NUMBERS USING MERGESORT

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

$$T(1) = c$$

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N
NUMBERS USING MERGESORT

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

$$T(1) = c$$

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

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N NUMBERS USING MERGESORT

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

$$T(1) = c$$

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

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

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N NUMBERS USING MERGESORT

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

$$T(1) = c$$

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

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

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

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N NUMBERS USING MERGESORT

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

$$T(1) = c$$

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

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

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

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

ONE MORE WAY TO FIND THE RUNNING TIME

LET $T(n)$ BE THE TIME TO SORT N NUMBERS USING MERGESORT

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

$$T(1) = c$$

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

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

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

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

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

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

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

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

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

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

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + \left(cn + cn + \dots (\text{k TIMES}) \dots + cn\right)$$

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

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

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

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

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

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + \left(cn + cn + \ldots (k \text{ Times}) \ldots + cn\right)$$

$$= 2^{\log n} T(1) + \left(cn + cn + \ldots (\log n \text{ Times}) + cn\right)$$

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

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

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

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

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

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + (cn + cn + \ldots (k \text{ Times}) \ldots + cn)$$

$$= 2^{\log n} T(1) + (cn + cn + \ldots (\log n \text{ Times}) + cn)$$

$$= (cn+1)\log n$$

$$= O(n \log n)$$

WORST CASE INPUT FOR MERGE-SORT

WORST CASE INPUT FOR MERGE-SORT

ANY INPUT

REASON: MERGING A & B TAKES
$O(\text{size}(A) + \text{size}(B))$ REGARDLESS OF INPUT

WORST CASE INPUT FOR MERGE-SORT

ANY INPUT

<u>REASON</u>: MERGING A & B TAKES
$O(\text{size}(A) + \text{size}(B))$ REGARDLESS OF INPUT

BEST CASE RUNNING TIME OF MERGESORT

WORST CASE INPUT FOR MERGE-SORT

ANY INPUT

REASON: MERGING A & B TAKES
$O(\text{size}(A) + \text{size}(B))$ REGARDLESS OF INPUT

BEST CASE RUNNING TIME OF MERGESORT

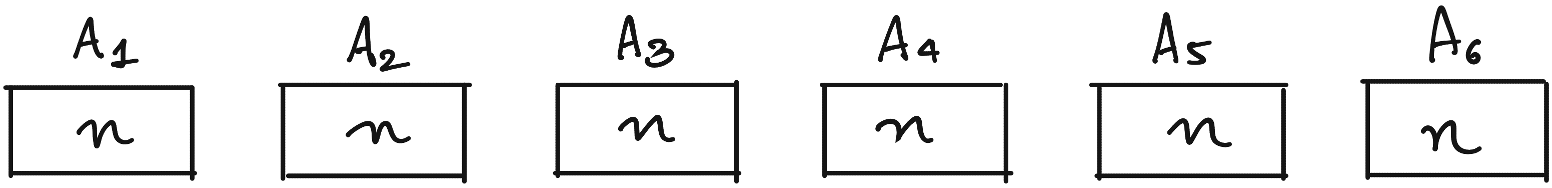
ANY INPUT

REASON: MERGING A & B TAKES
$O(\text{size}(A) + \text{size}(B))$ REGARDLESS OF INPUT

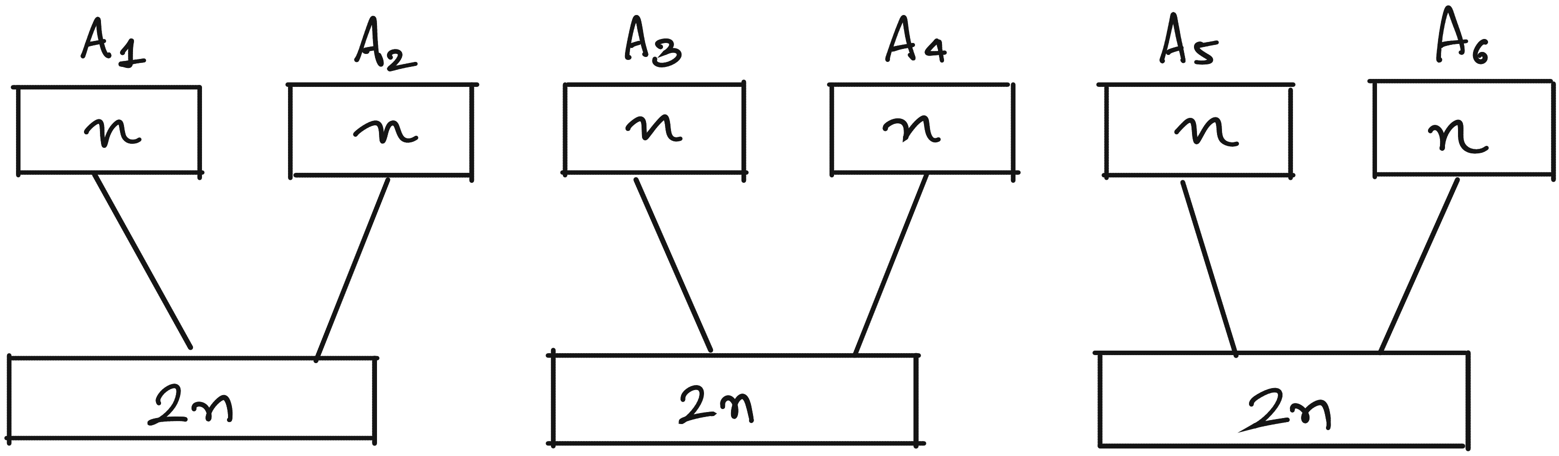
MERGESORT TAKES $O(n \log n)$ ON ANY INPUT.

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.

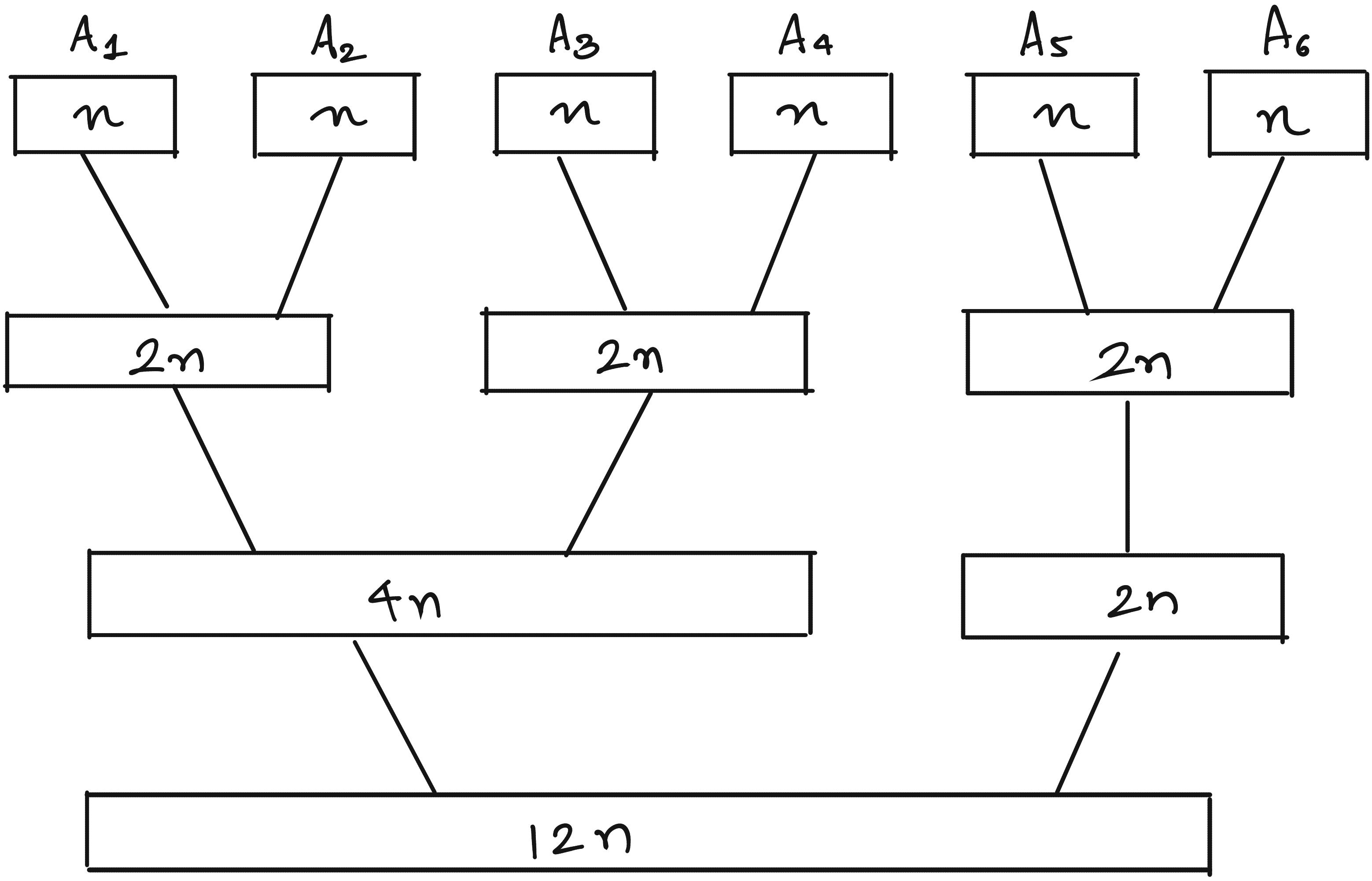
Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



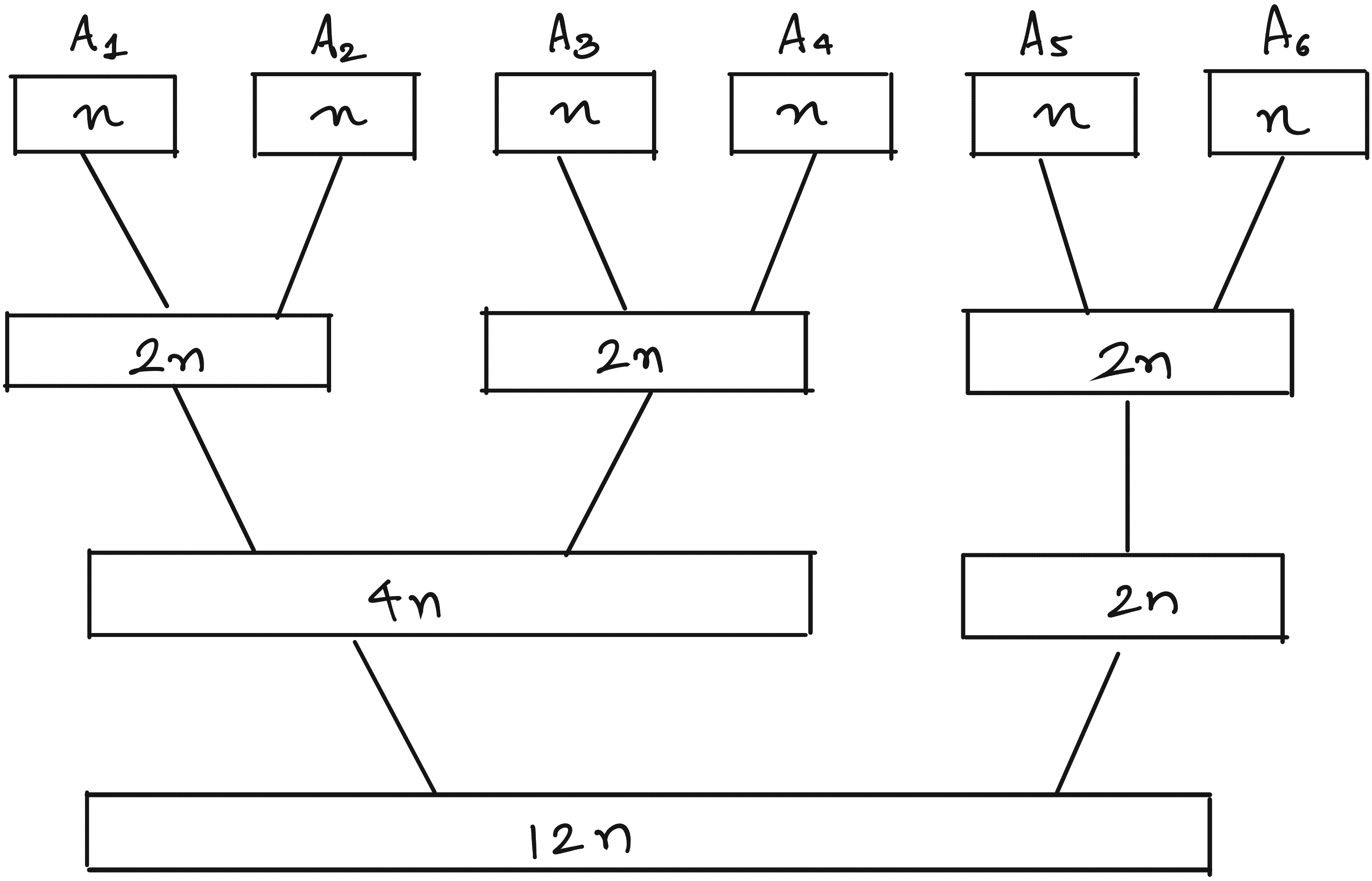
Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.

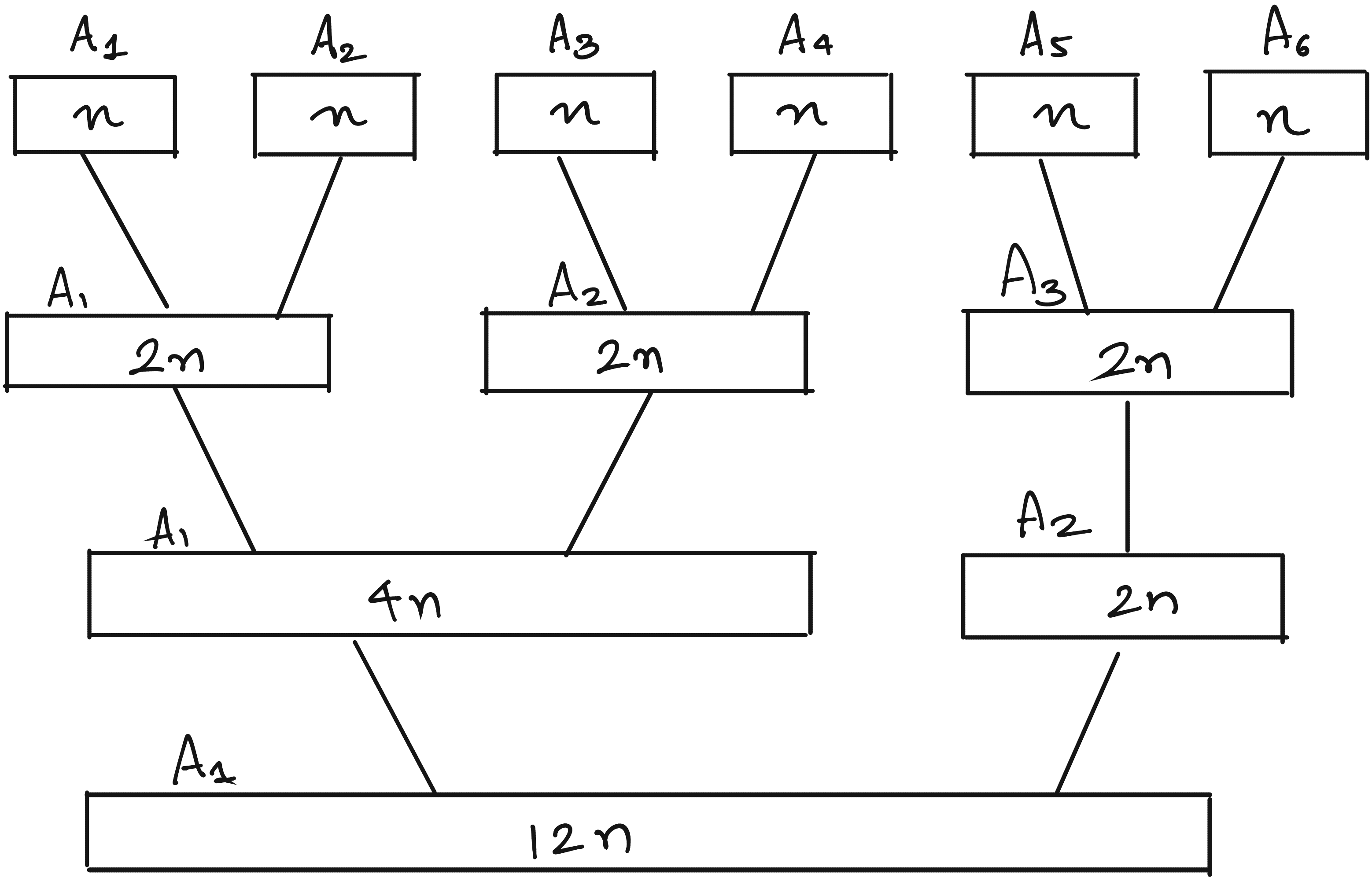


Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



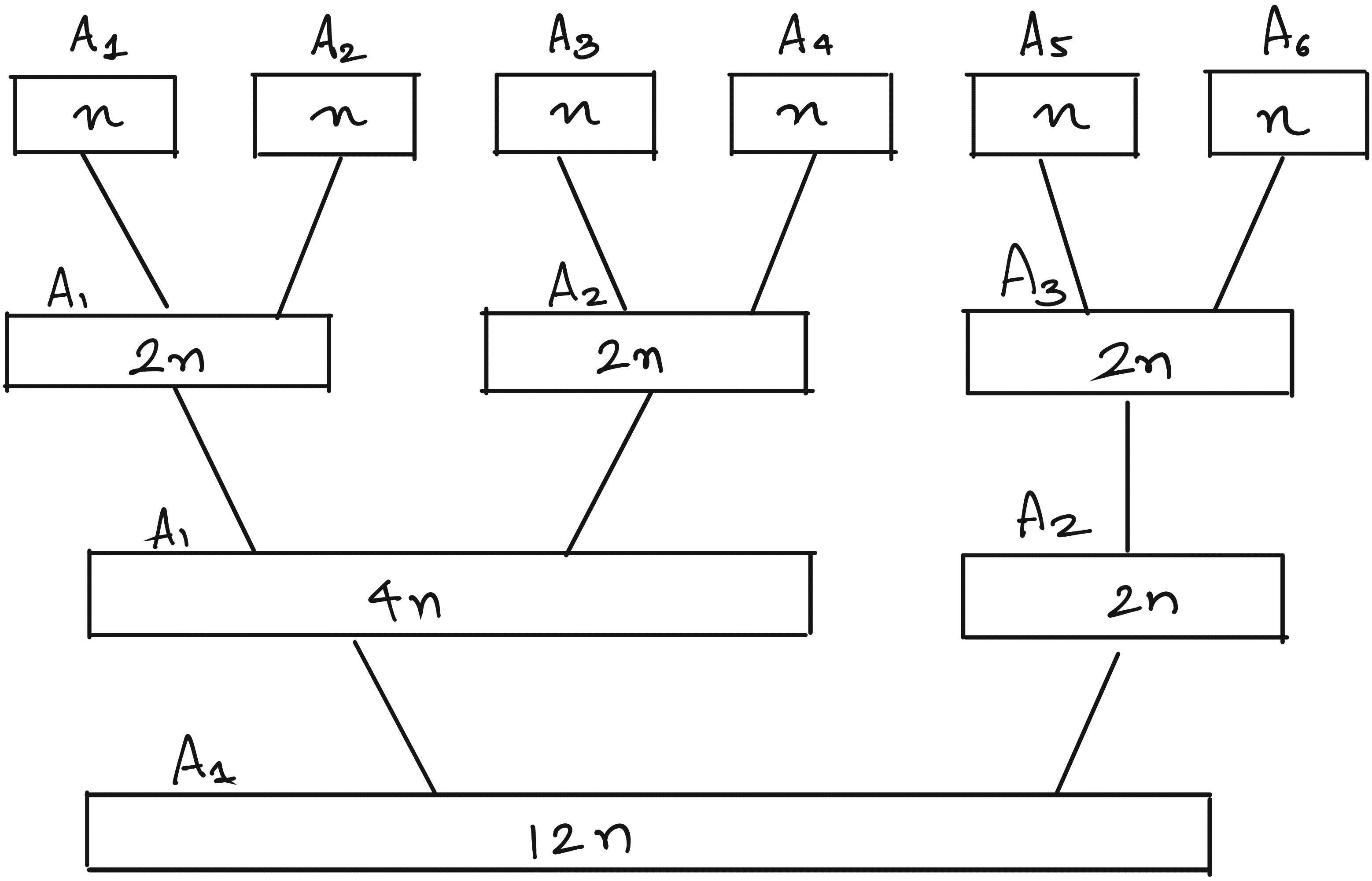

```
WHILE (K > 1)
{   i ← 1;
    WHILE ( i ≤ ⌊K/2⌋)
        A_i ← MERGE( A_{2i-1}, A_{2i});
        i ← i+1;
    K ← ⌊K/2⌋
}
```

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



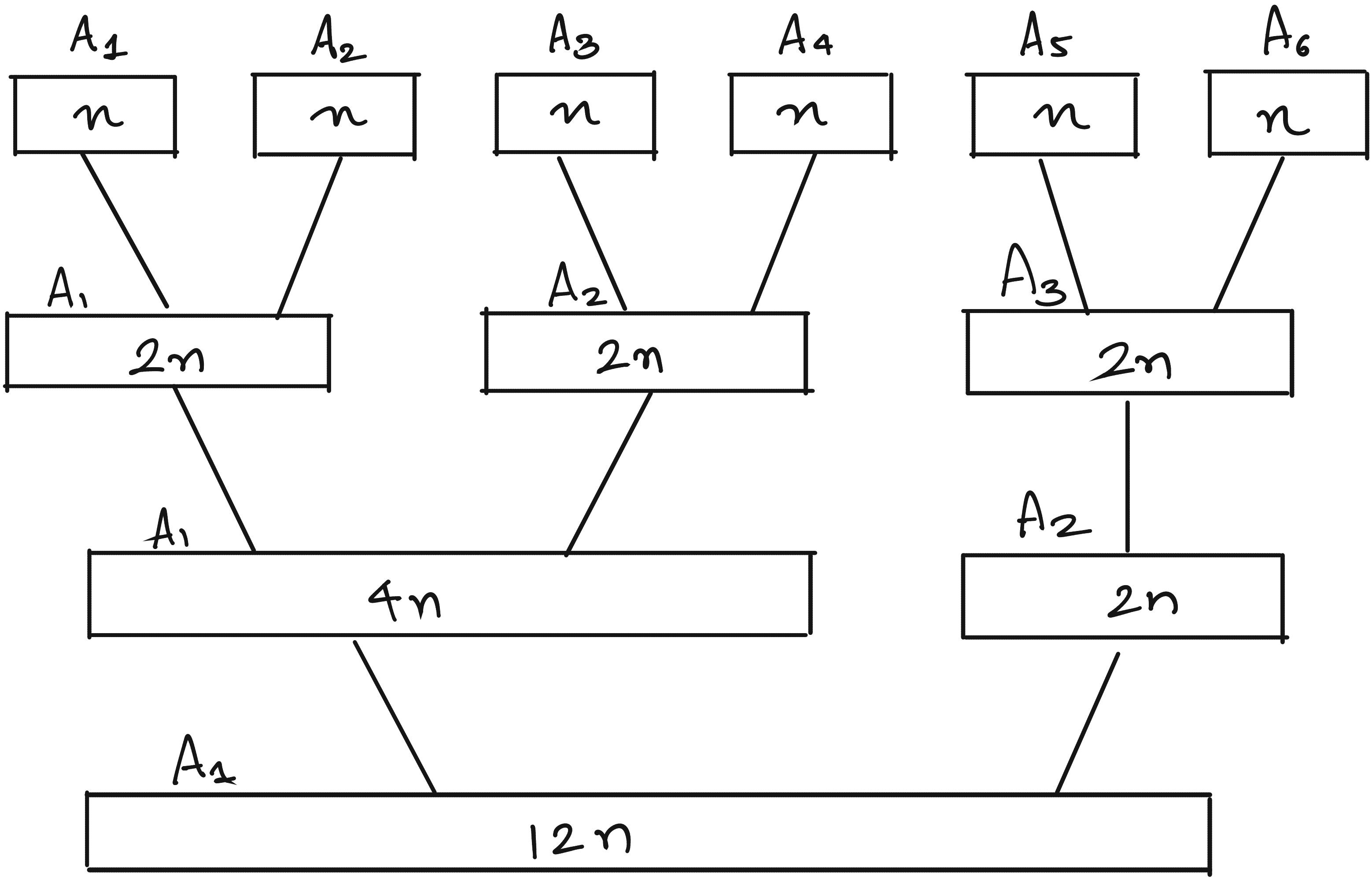
WHILE ($K > 1$)
{ $i \leftarrow 1$;
    WHILE ($i \leq \lfloor \frac{K}{2} \rfloor$)
        $A_i \leftarrow$ MERGE(A_{2i-1}, A_{2i})
        $i \leftarrow i+1$;
    $K \leftarrow \lfloor \frac{K}{2} \rfloor$
}

Q: Given k sorted array of size n, merge them into a single sorted array.



Q: What is the height of this tree?

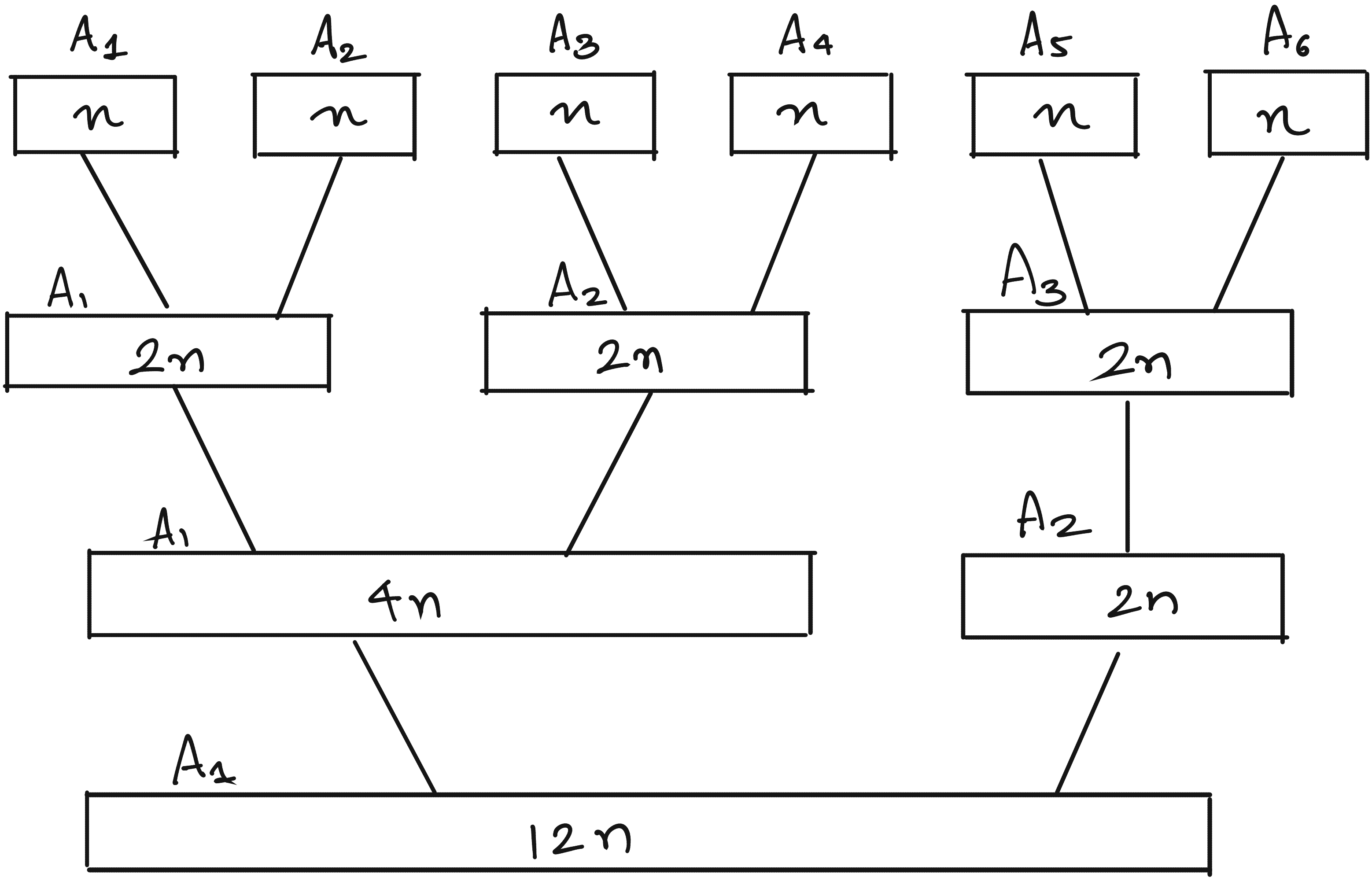
Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



Q: WHAT IS THE HEIGHT OF THIS TREE?

LAYER 0 — K ARRAY

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



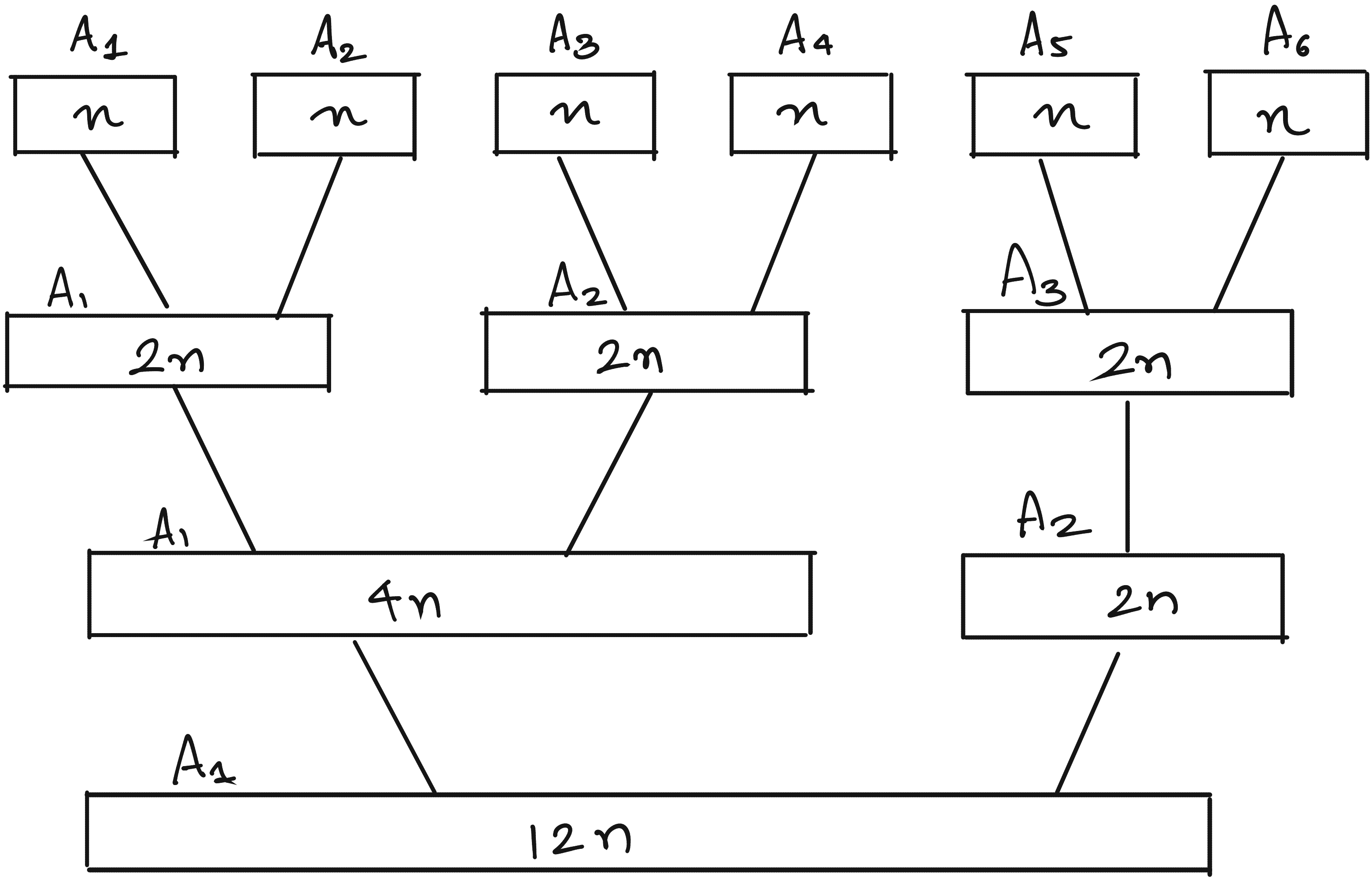
Q: WHAT IS THE HEIGHT OF THIS TREE?

LAYER 0 — K ARRAY

LAYER 1 — $\frac{K}{2}$

⋮

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



Q: WHAT IS THE HEIGHT OF THIS TREE?

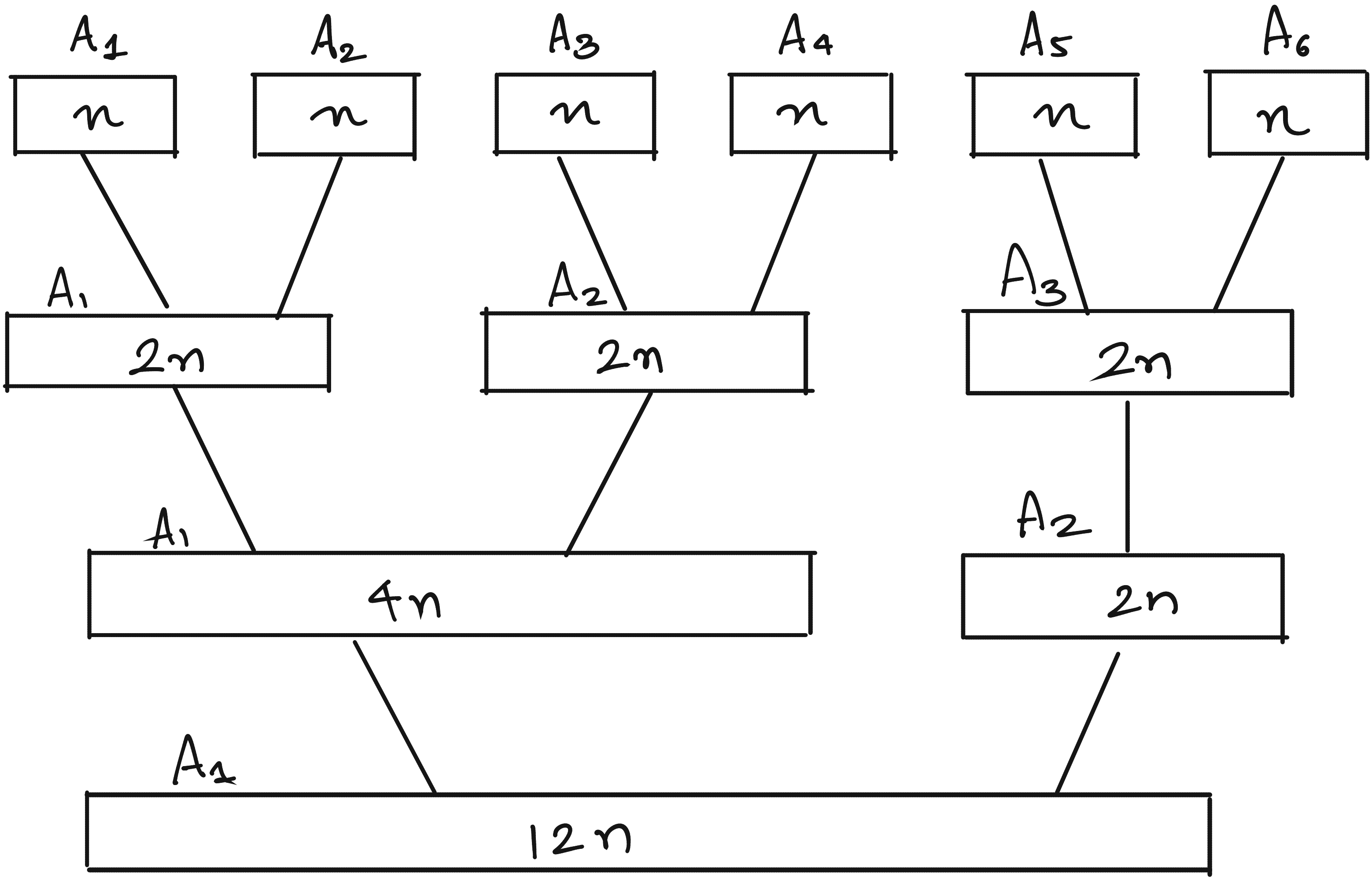
LAYER 0 — K ARRAY

LAYER 1 — $\frac{K}{2}$

⋮

LAYER l — $\frac{K}{2^l} = 1$

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



Q: WHAT IS THE HEIGHT OF THIS TREE?

LAYER 0 — K ARRAY

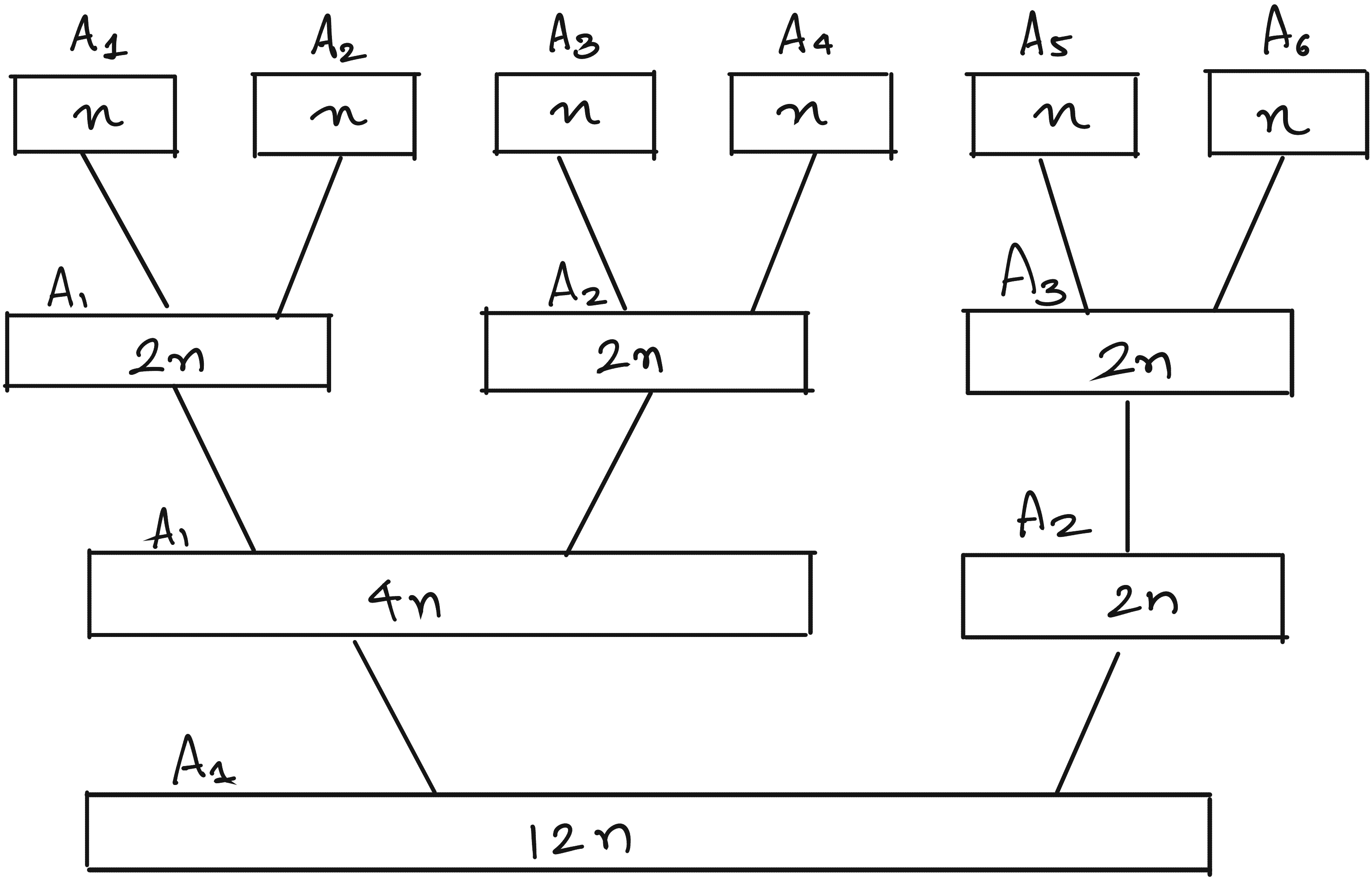
LAYER 1 — $\frac{K}{2}$

⋮

LAYER l — $\frac{K}{2^l} = 1$

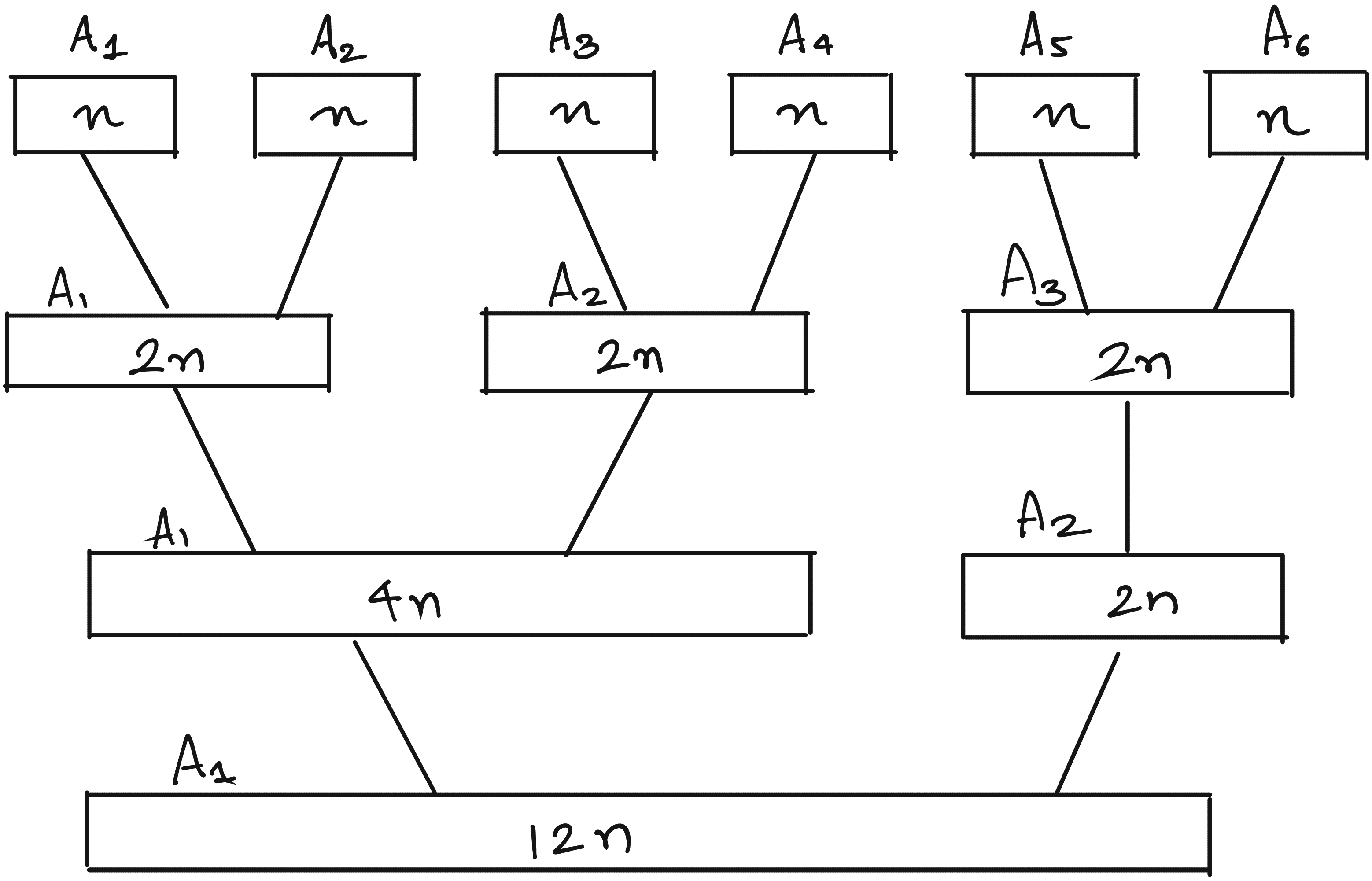
$\Rightarrow l = \log K$

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



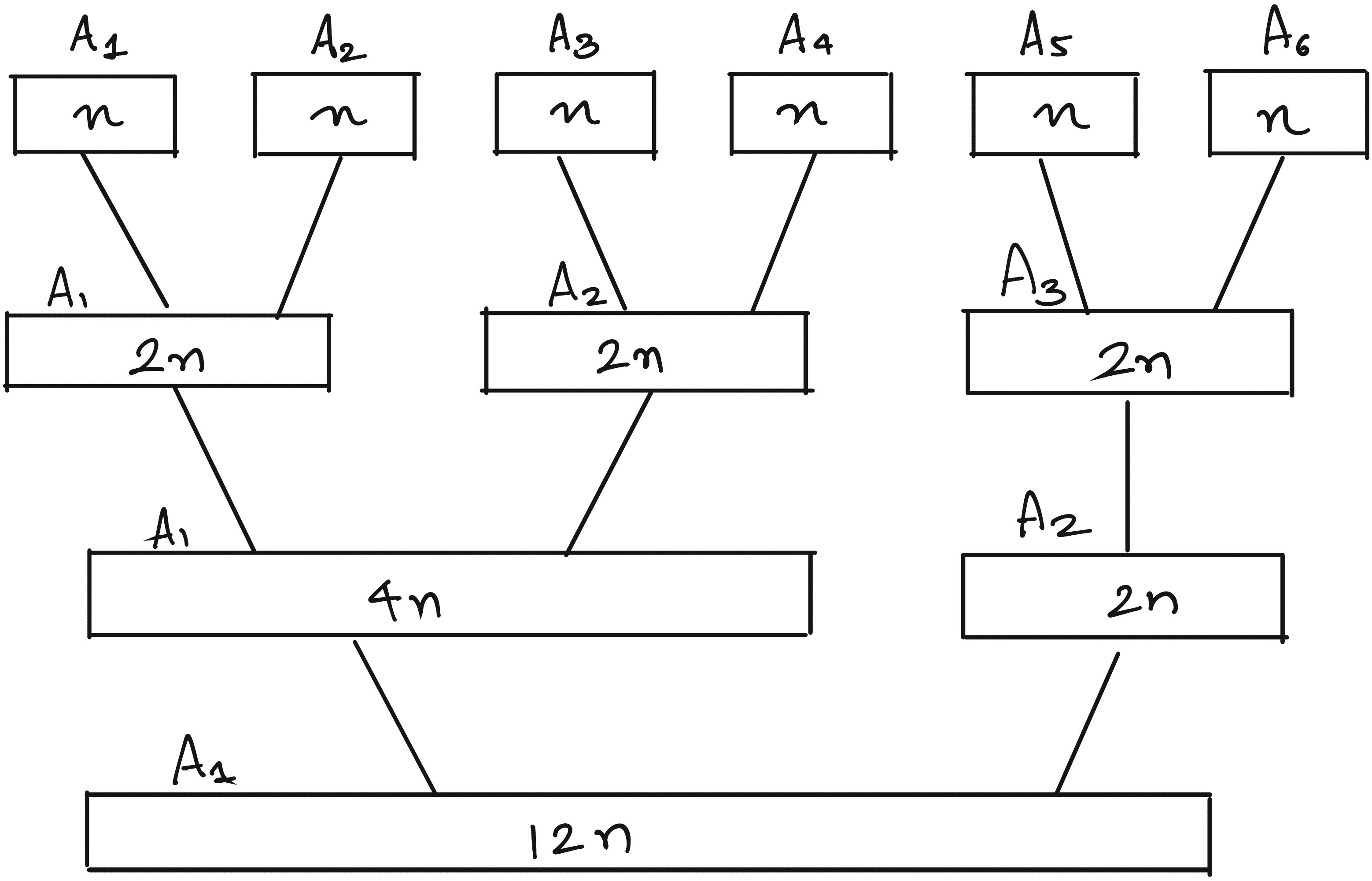
TIME REQUIRED AT LAYER 1

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



TIME REQUIRED AT LAYER 2 $\leq 2n \cdot \frac{K}{2} = nK$

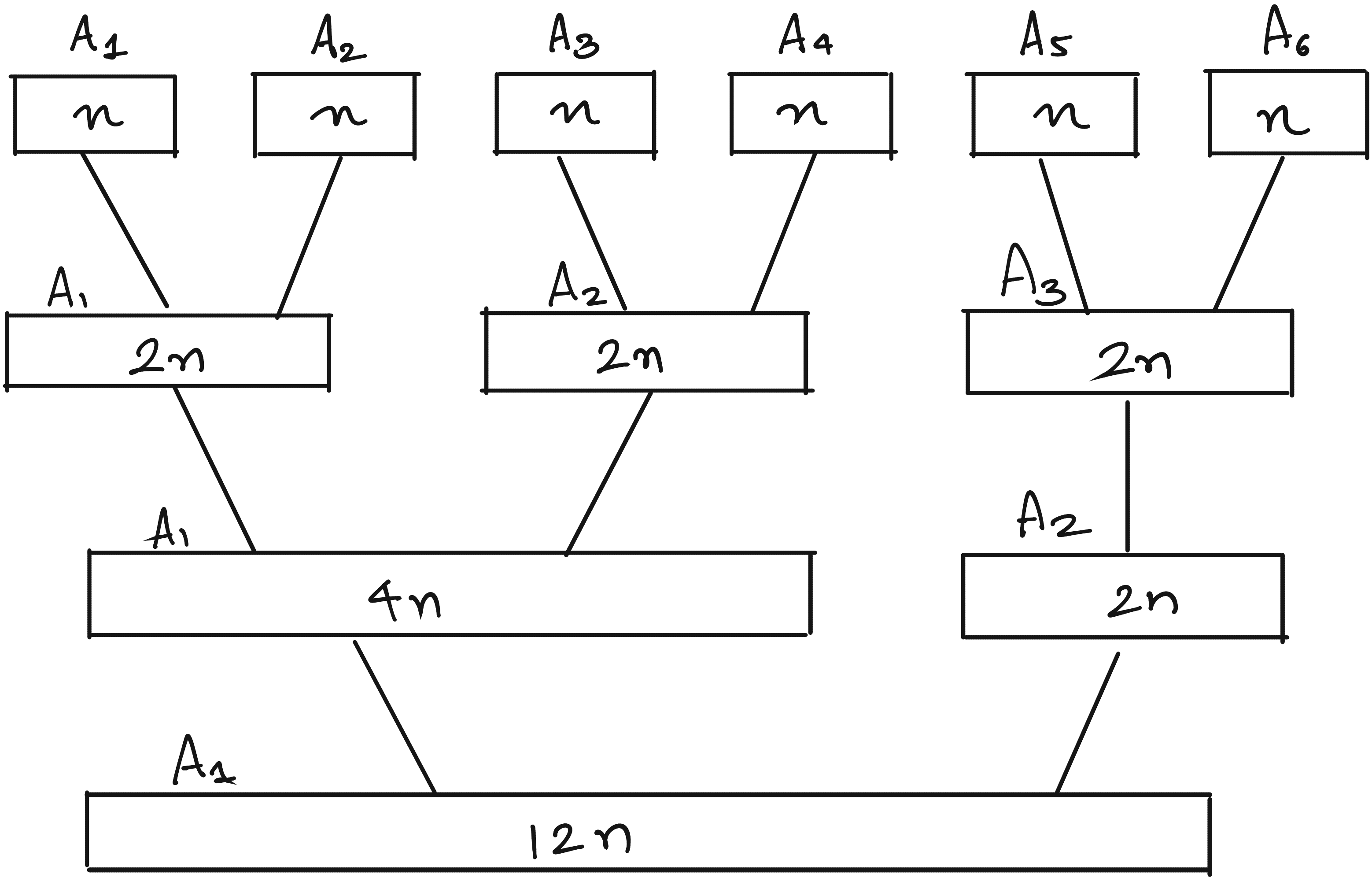
Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



TIME REQUIRED AT LAYER 1 $\leq 2n \cdot \frac{K}{2} = nK$

TIME REQUIRED AT LAYER 2

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.

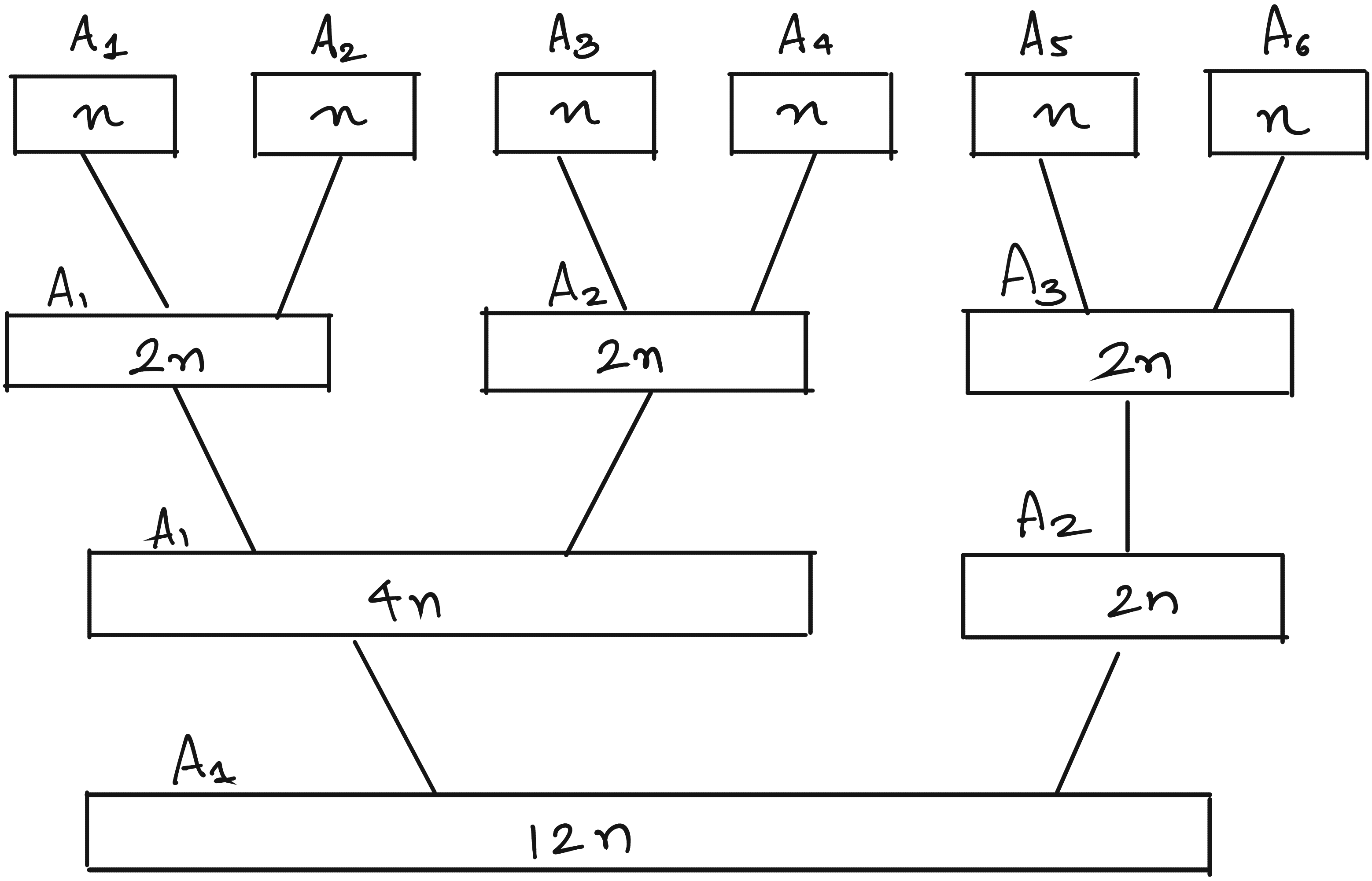


TIME REQUIRED AT LAYER 1 $\leq 2n \cdot \frac{k}{2} = nk$

TIME REQUIRED AT LAYER 2 $\leq 2^2 \cdot n \cdot \frac{k}{2^2} = nk$

$\vdots$

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



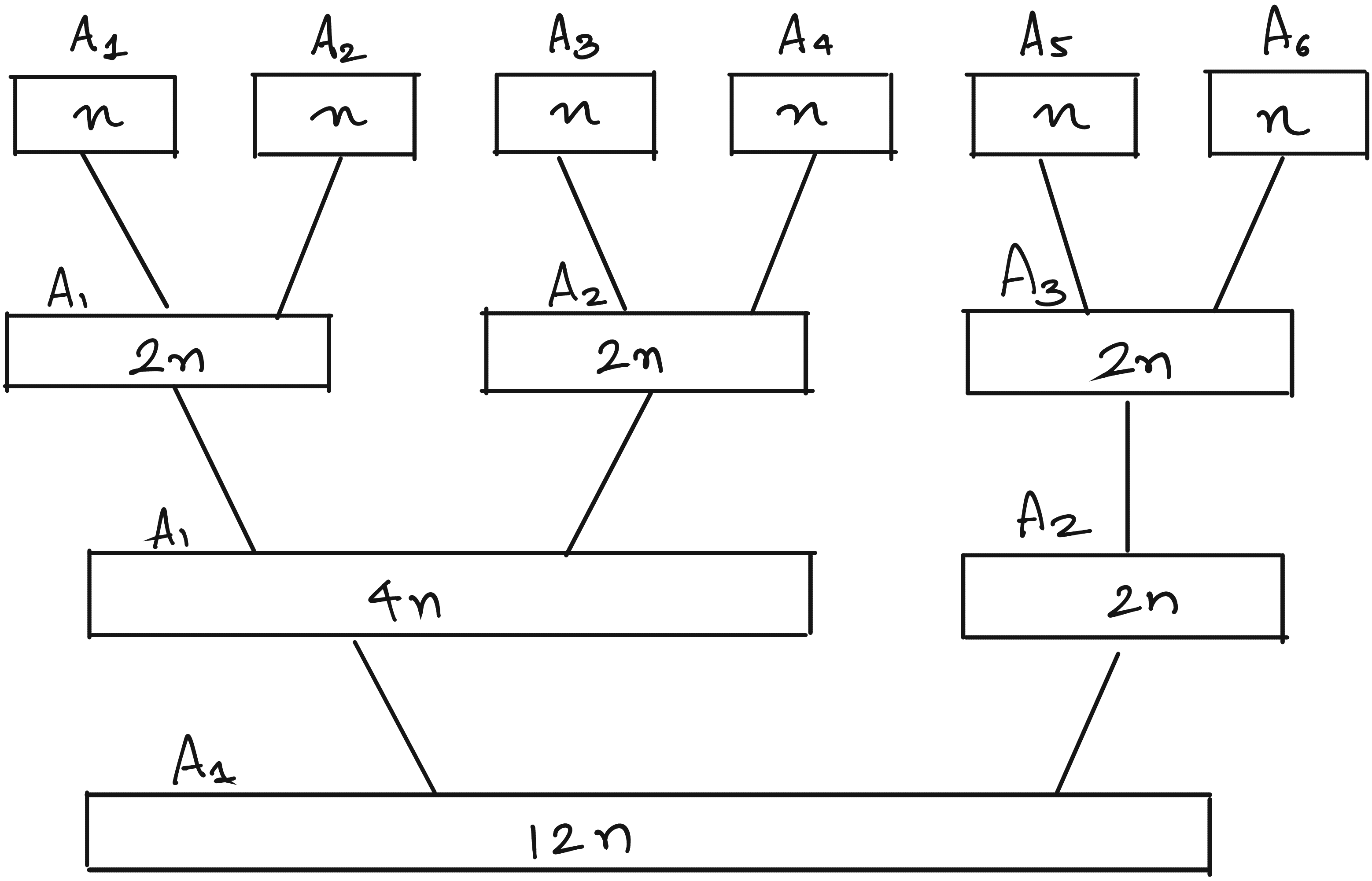
TIME REQUIRED AT LAYER 1 $\leq 2n \cdot \frac{k}{2} = nk$

TIME REQUIRED AT LAYER 2 $\leq 2^2 \cdot n \cdot \frac{k}{2^2} = nk$

$\vdots$

TIME REQUIRED AT LAYER $\log k \leq 2^{\log k} \cdot n \frac{k}{2^{\log k}} = nk$

Q: GIVEN K SORTED ARRAY OF SIZE n, MERGE THEM INTO A SINGLE SORTED ARRAY.



TIME REQUIRED AT LAYER 1 $\leq 2n \cdot \frac{k}{2} = nk$

TIME REQUIRED AT LAYER 2 $\leq 2^2 \cdot n \cdot \frac{k}{2^2} = nk$

$\vdots$

TIME REQUIRED AT LAYER $\log k \leq 2^{\log k} \cdot n \frac{k}{2^{\log k}} = nk$

TOTAL TIME $= O(nk \log k)$