8) MERGE SORT ON

123 34 189 56 150 12 9 240

ANSWER

DIVIDE THE ARRAY

| 123 | 34 | 189 | 56 |
|---|---|---|---|

| 150 | 12 | 9 | 240 |
|---|---|---|---|

SORT EACH ARRAY

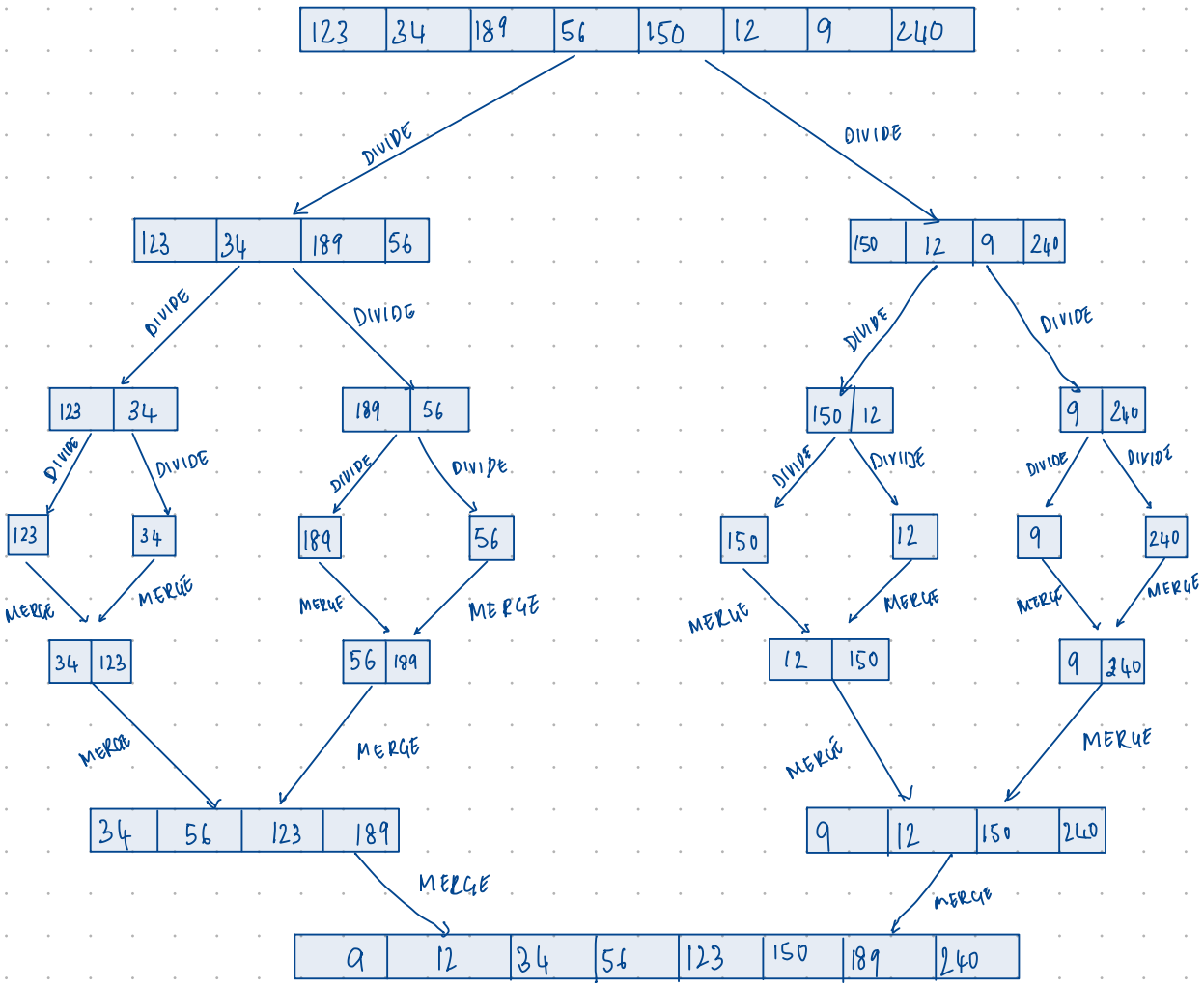
| 34 | 56 | 123 | 189 |
|---|---|---|---|

| 9 | 12 | 150 | 240 |
|---|---|---|---|

MERGE THE SUBARRAY

| 9 | 12 | 34 | 56 | 123 | 150 | 189 | 240 |
|---|---|---|---|---|---|---|---|

a) RECURSIVE TREE CALL



37)

Consider $n! = 1 \times 2 \times \dots \times n$

To recursively determine the factorial of n using divide & conquer

A Divide and Conquer algorithm consists of 3 parts:

→ DIVIDE
→ CONQUER
→ COMBINE

INPUT SIZE : n (n: represents the input number)

Consider $n! = [1 \times 2 \times \dots (n/2)] \times [(n/2+1) \times \dots \times n]$

Calculate $(n/2)!$ Calculate product of $(n/2+1)$ to $n$

## PSEUDO CODE

```
Function fact (n, start):
    if start == n :
        return n
    if start + 1 == n
        return n * start

    mid = (start + n) // 2

    left = fact ( mid, start)
    right = fact ( n, mid + 1)

    return left * right

end function
```

DIVIDE - The problem is divided at the midpoint between start & n
CONQUER - Recursively calculate factorial of numbers from start to mid and from mid + 1 to n
COMBINE - The results of the 2 recursive calls are multiplied to get factorial

CODE

```
def fact (n, start = 1):
    if start == n :
        return n
    if start + 1 == n:
        return n * start

    mid = (start + n) //2
    left = fact (mid, start)
    right = fact ( n, mid + 1)
    return left * right
```

## TIME COMPLEXITY -

if statements have a linear complexity of O(1)

We recursively calculate the mid element and divide the array into halves

→ This has a time complexity of O (log n)

We must multiply the elements (n) times

∴ The time complexity is O (n log n)

This is not exponential time complexity

- It does not violate case 1 in section 2.8:
  - ↳ We get multiple cases of size $n/2$ where $n/2$ is lesser than $n$