Merge Procedure And Merge Sort

Merge (A, B)

```
1.  nA = A.length
2.  nB = B.length
3.  let L[nA + nB] be a new array
4.  A[nA+1] = ∞
5.  B[nB+1] = ∞
6.  i = 1
7.  J = 1
8.  for  K=1 to nA+nB
9.        if  A[i] ≤ B[J]
10.            L[K] = A[i]
11.            i = i+1
12.       else
13.            L[K] = B[J]
14.            J = J+1
15.  return L
```

• Dry Running:-

A = [1, 4]

B = [3, 5]

1. $n_A = 2$

2. $n_B = 2$

3. L = [ ]

4. A = [1, 4, ∞]

5. B = [3, 5, ∞]

6. i = 1

7. j = 1

8. # 1st Iteration Of For-Loop:-

8. K = 1

9. if 1 ≤ 3

10. L[1] = 1 → L = [1]

11. i = 2

12.

13.

14.

2nd Iteration Of For Loop:-

8. K = 2

9. if 4 ≤ 3

10.

11.

12. else

13. L[2] = 3 → L = [1, 3]

14. J = 2

3RD ITERATION OF FOR-LOOP:-

8. K = 3

9. If 4 < 5
10. L[3] = 4 → L = [1, 3, 4]
11. i = 3
12. ______
13. ______
14. ______

4TH ITERATION OF FOR-LOOP:-

8. K = 4

9. If ∞ < 5
10. ______
11. ______
12. else
13. L[4] = 5 → L = [1, 3, 4, 5]
14. J = 3

5TH ITERATION OF FOR-LOOP:-

8. K = 5 ⇒ Loop terminates

______ x ______ x ______ x ______

15. Return [1, 3, 4, 5]

Scanned with CamScanner

Line no of code	Time/instruction	frequency
1	c	1
2	c	1
3	c	1
4	c	1
5	c	1
6	c	1
7	c	1
8	c	$\frac{(n_A+n_B)}{n}+1$
9	c	$\frac{n_A+n_B}{n}$
10	c	n_A
11	c	n_A
12	c	
13	c	n_B
14	c	n_B
15	c	1

Date:

$T(n) = c[7 + n + 1 + n + 2(n_A + n_B) + 1]$

$T(n) = c[9 + n + n + 2n]$

$T(n) = c[4n + 9]$

$T(n) = 4cn + 9c$

$T(n) = 4K_1 n + K_2$

K_1 and K_2 are constants

=> Linear growth

• DISCUSSION:-

=> Merge procedure grows linearly

=> Time complexity of merge procedure is $O(n)$

Date:

PHYTHON CODE

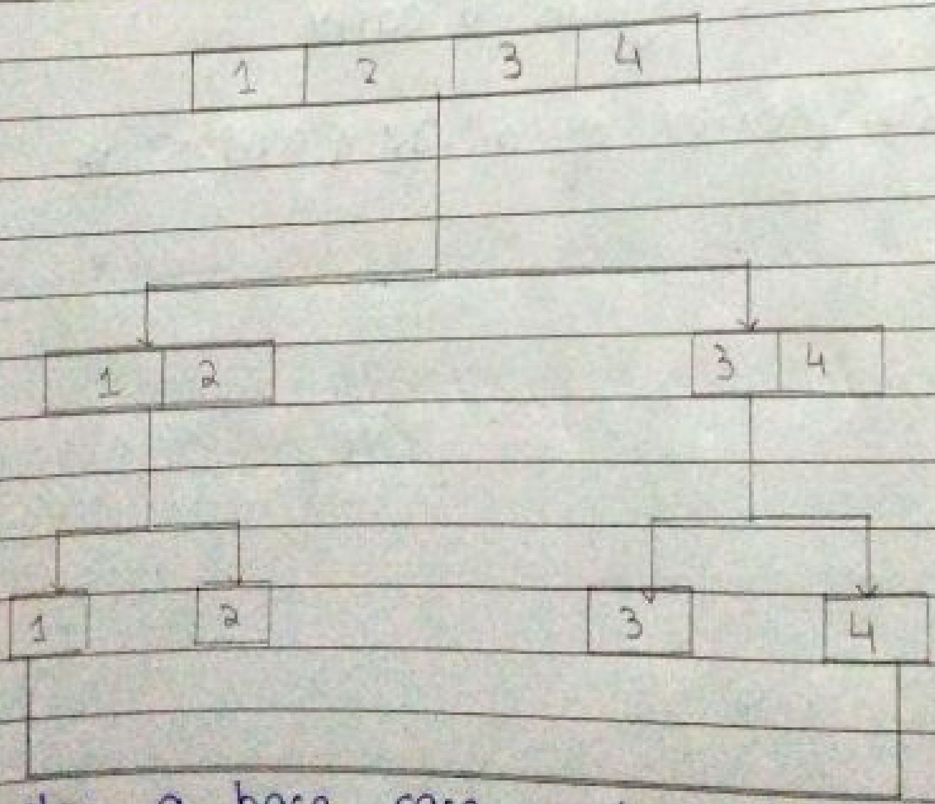
```
def mergeSort(A):
    n=len(A)
    S= list( )
    if n==1:
        S=A

    else:
        a= (n//2);
        S1= merge Sort (A[0:a]);
        S2= merge Sort (A [a:n]);
        S= merge (S1,S2)
    return S
```

• FOR BEST CASE :-

A: [1,2,3,4]

• **DRY RUNNING :-** (DIVIDE)



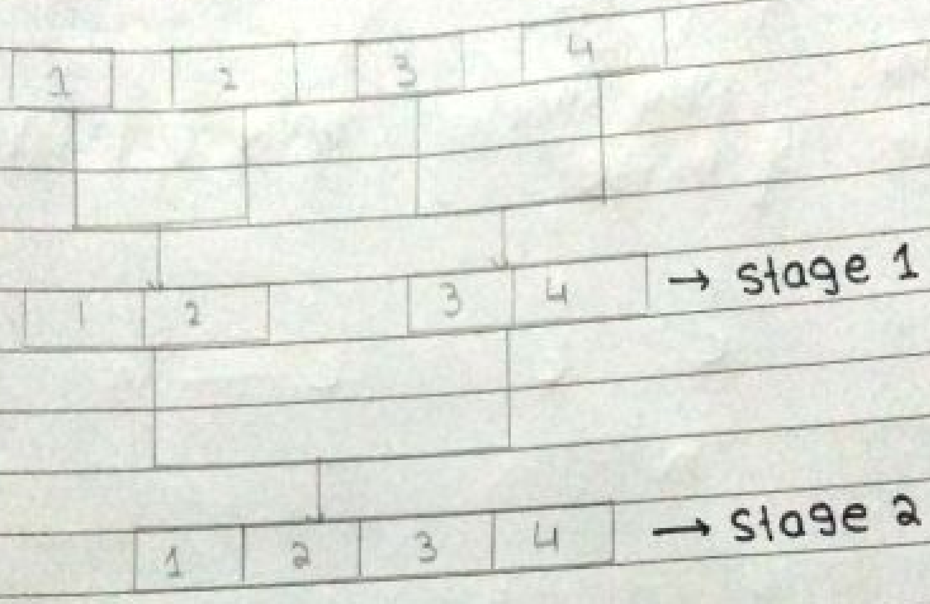
• for a base case we have 'c' second.

Scanned with CamScanner

- For 'n' number of elements we will call base case for n-times

- Time for all base cases: nc

Merge:-



• COUNT STAGES OF MERGE:-

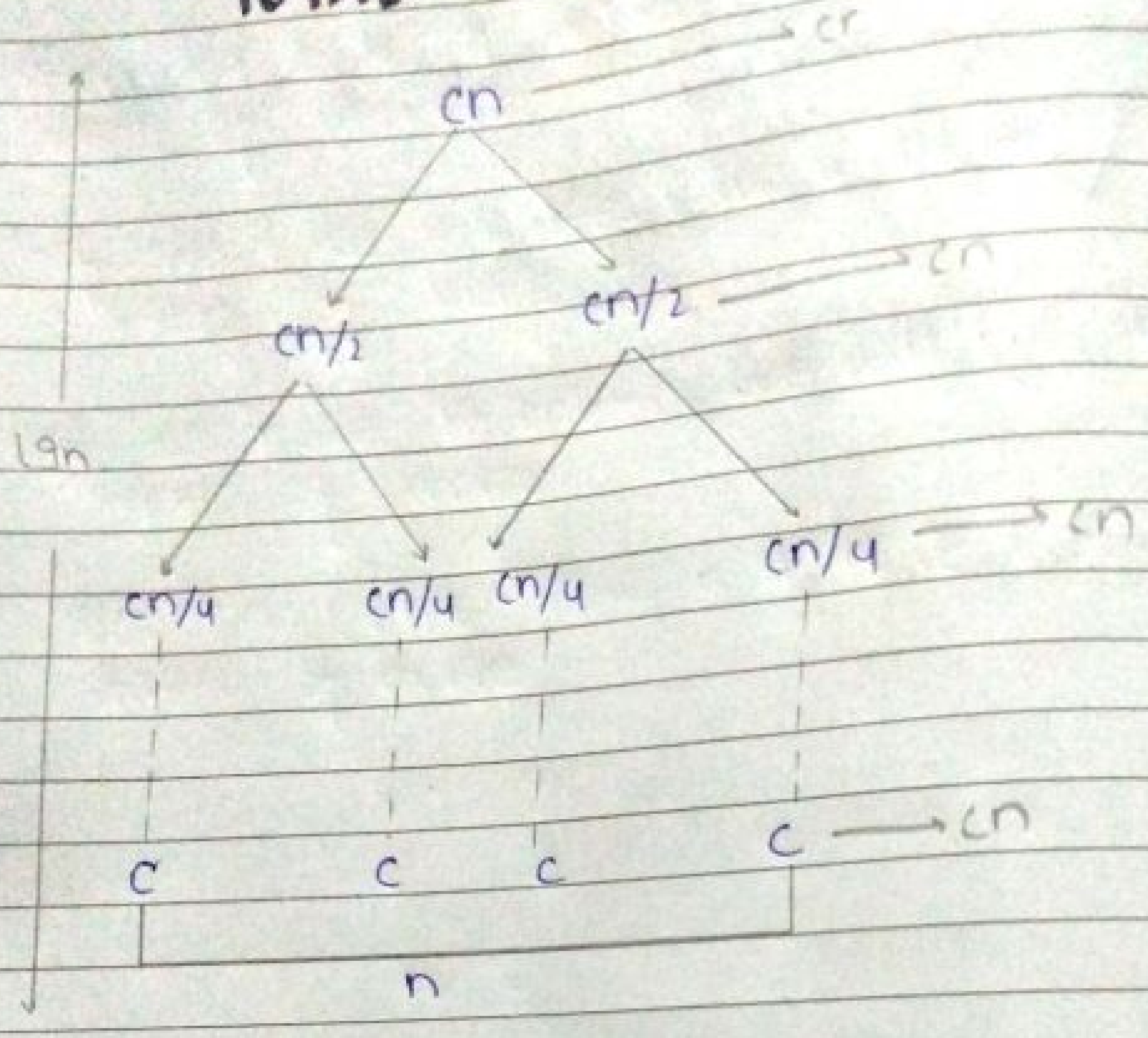
- When we have 4 entries, we got two stages of merge

- If we have 8 entries, we got three stages of merge

- If we have 16 entries, we will have 4 stages of merge

- For n entries we will have $\log_2(n)$ stages of merge

TOTAL TIME:-



Total time = $cn\lg n + cn = cn(1 + \lg n)$

• logarithmic growth

- **DISCUSSION:-**

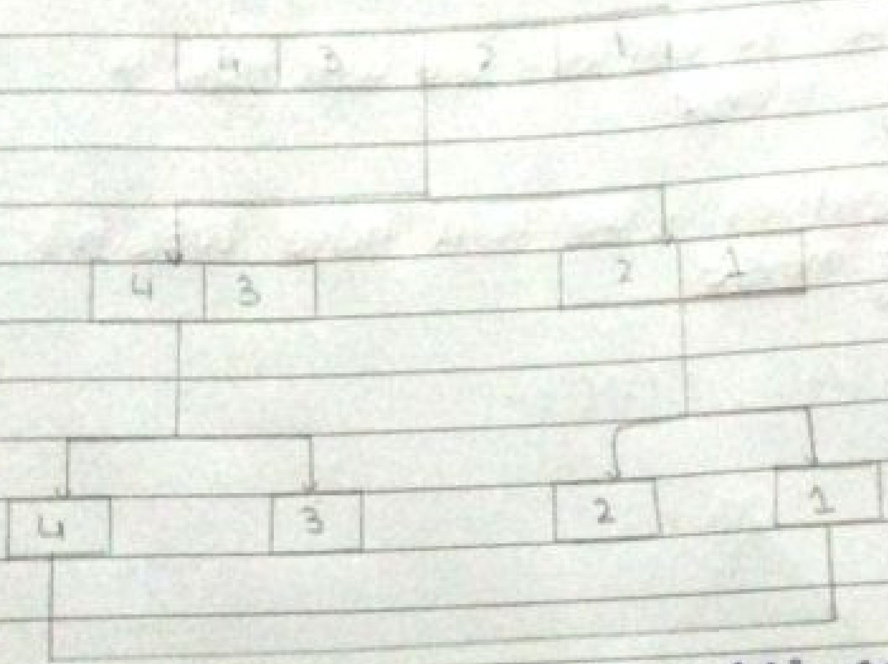
- Merge sort in best case grows logarithimic.
- Time complexity of merge sort in best case is $O(n\log n)$

Scanned with CamScanner

• For Worst case:-

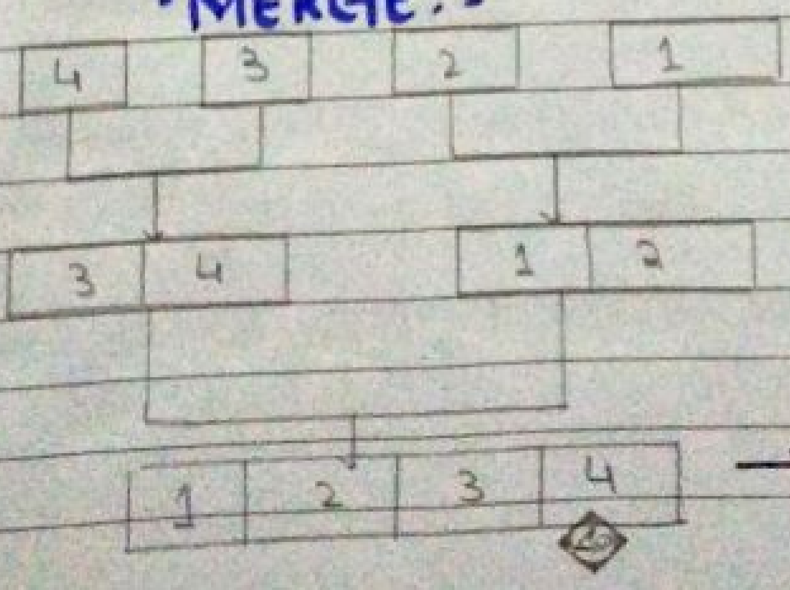
A = [4, 3, 2, 1]

DIVIDE:-



- For a base case we have 'c' second.
- For 'n' number of elements we will call base case for n times
- Time for all base cases = n×c

• MERGE:-



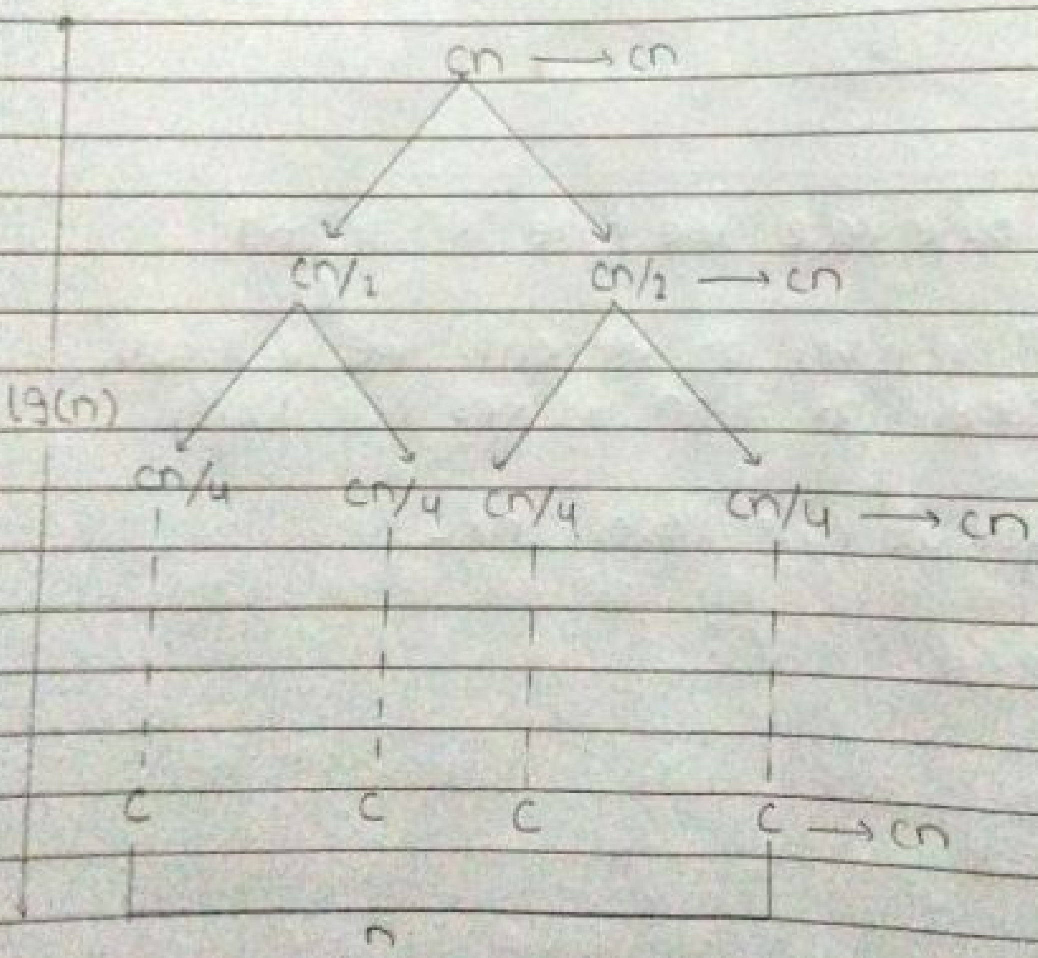
→ stage 1

→ stage 2

Date:

- When we have 4 entries, we got 2 stages of merge
- If we have 8 entries, we will have 3 stages of merge
- If we have 16 entries, we will have 4 stages of merge.
- For 'n' entries, we will have $\log_2(n)$ stages of merge

TOTAL TIME:-



Date:

Total time. $cn\lg n + cn = cn(1+\lg n)$

=> logarithimic growth

• Discussion:-

=> Mege sort in worst case also grows logarithimic

=> Time complexity of merge sort for worst case is $O(n\log n)$