$= \theta(n^{\log_b a})$ if $a > b$

MERGE SORT -

Algorithm - Mergesort (A[0 -- n-1])

Sorts array A[0 — n-1] by recursive mergesort.

Input - Arrays B[0 -- p-1] & C[0 -- q-1] both sorted.

Output - sorted array A[0 -- p+q-1] of elt's B & C.

i ← 0 ; j ← 0 ; k ← 0.

i - pointing to 1st elt of B
j - pointing to 1st elt of C
k - pointing to 1st elt of A

If n > 1

copy A[0 -- ⌊n/2⌋ - 1] to B[0 — ⌊n/2⌋ - 1]

copy A[⌊n/2⌋ — n-1] to C[0 — ⌈n/2⌉ - 1]

Mergesort (B[0 — ⌊n/2⌋ - 1])

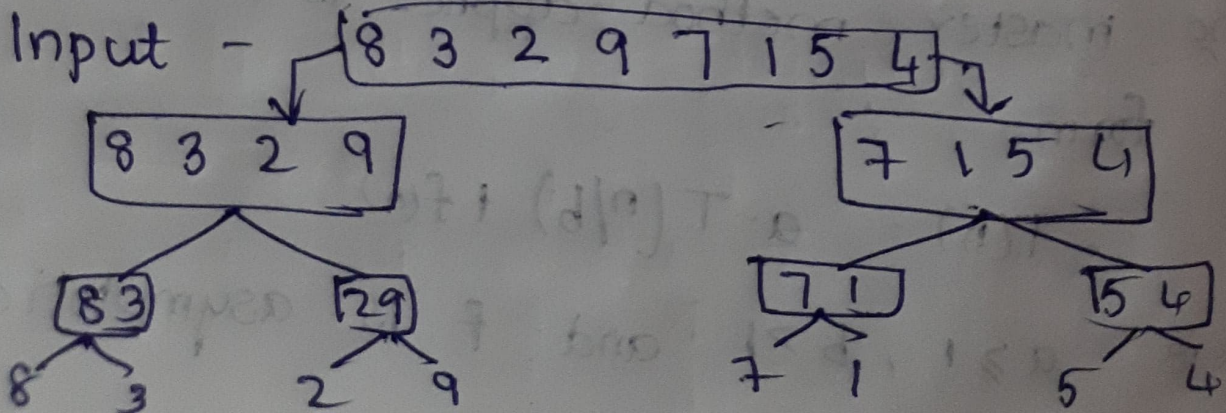
Mergesort (C[0 — ⌈n/2⌉ - 1])

Merge (B, C, A).

We will divide the array into halves until there is only 1 elt is there in each subdivision.

Main

eg :-



Algorithm - Merge (B[0 -- p-1], C[0 -- q-1], A[0 -- p+q-1])

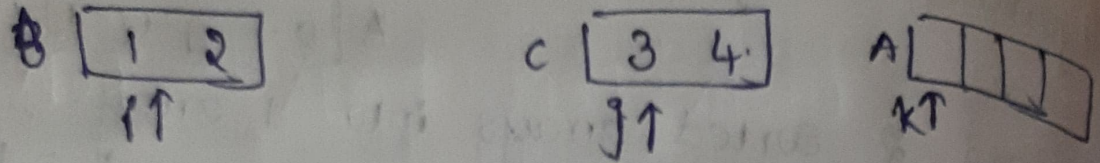
Merges 2 sorted arrays into 1 array.

Input - Arrays B[0 -- p-1] & C[0 -- q-1] both sorted

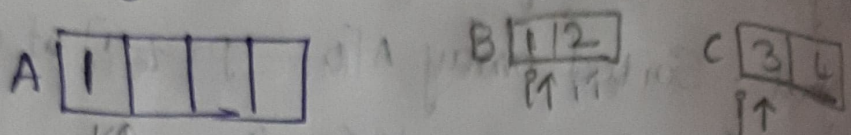
Output - Sorted array A[0 -- p+q-1] of elt's of B & C.

```
i ← 0 , j ← 0 , k ← 0
while i < p and j < q do
        if B[i] ≤ C[j]
            A[k] ← B[i] ; i ← i+1
        else
            A[k] ← C[j] ; j ← j+1
        k ← k+1
if i = p
    copy C[j -- q-1] to A[k -- p+q-1]
else
    copy B[i -- p-1] to A[k -- p+q-1]
```

① Eg :- 1, 2, 3, 4.

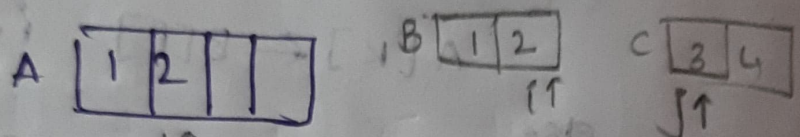


① st step - B[i] < C[j] 1 < 3.



i ++ & k ++.

② nd step - B[i] < C[j] 2 < 3



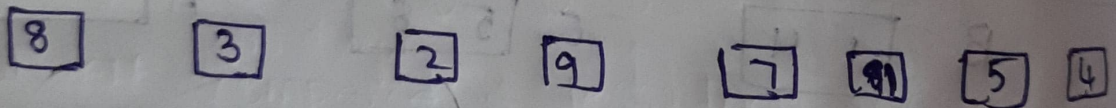
i ++ & k ++

But 'i' exceeds the size of array.

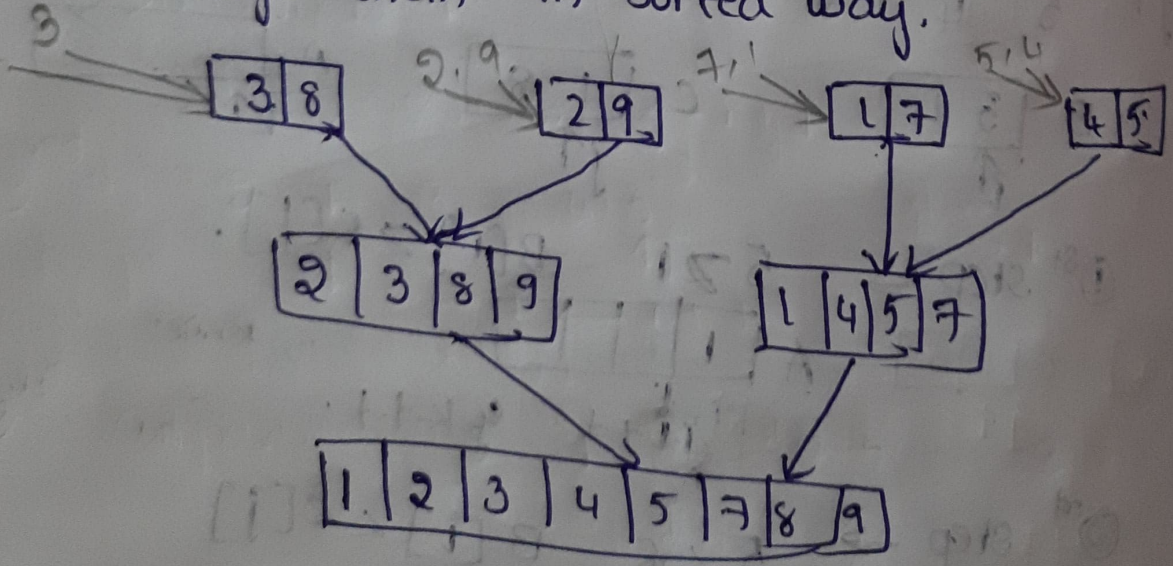
So, copy the elements in C array to A.

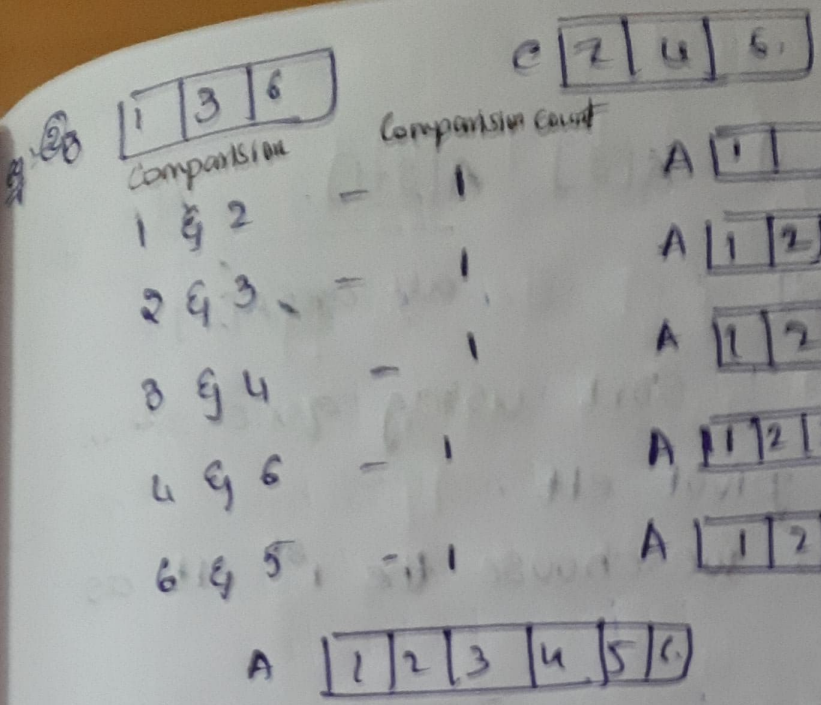
A | 1 | 2 | 3 | 4 |

Main eg :- Continuation -



Merge them in sorted way.





A array has 6 elements - 5 comparisions
A array has n elements - n-1 comparisions

Merge Sort Worst Case Analysis.

When n is a power of 2, we have exactly

$C(n) = 2C(n/2) + C\_merge(n)$ for $n > 1$

$C(1) = 0$.

In the worst case -

$C\_merge(n) = n - 1$ (refer eq - (2))

$C\_worst(n) = 2\,C\_worst(n/2) + (n-1)$ for $n > 1$

$C\_worst(1) = 0$.

Using Master's Theorem,

$C\_worst(n) = \Theta(n \log n)$ → Worst case complexity

→ $a = 2 \quad b = 2 \quad d = 1$

$C\_worst(n) = 2 \cdot C(n/2) + (n-1)$

$2 = 2^1 \quad (a = b^d)$

So,

$C\_worst(n) = \Theta(n^d \log n) = \Theta(n \log n)$

C_worst(n) = C_worst + 1

BINARY SEARCH

Binary Search (A[1 ... n], K):

```
l ← 1
r ← n
while l <= r
    m ← ⌊(l+r)/2⌋
    if K == A[m]
        return m
    else if K < A[m]
        r ← m-1
    else
        l ← m+1
return -1
```

13/9/21

Analysis

C_worst(n) = C_worst(floor(n/2)) + 1

C_worst(1) = 1

Using

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

a = 1 b = 2 . d = 0 a = b^d

Using Masters Theorem,

$$C_{worst}(n) = O(\log_2 n)$$