6/9/21

Design Techniques
Divide and Conquer

→ Algorithm design technique is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

→ BRUTE FORCE is a straightforward approach to solving a problem.

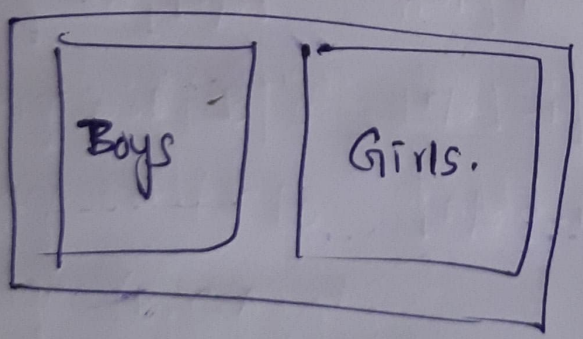
→ EXHAUSTIVE SEARCH is a bruteforce approach to combinatorical problems.

Design Techniques:-

→ Divide & Conquer
→ Backtracking
→ Dynammic Programming
→ Branch and Bound.
→ Greedy approach

Divide and Conquer

class



Aim-
Find no. of students in class.

Brute force approach -

1 person counts the no of students

Divide & conquer -

1 person counts boys & 1 person counts girls parallely and at last merge.

→ If problem instance is small enough, solve problem directly, return solution.

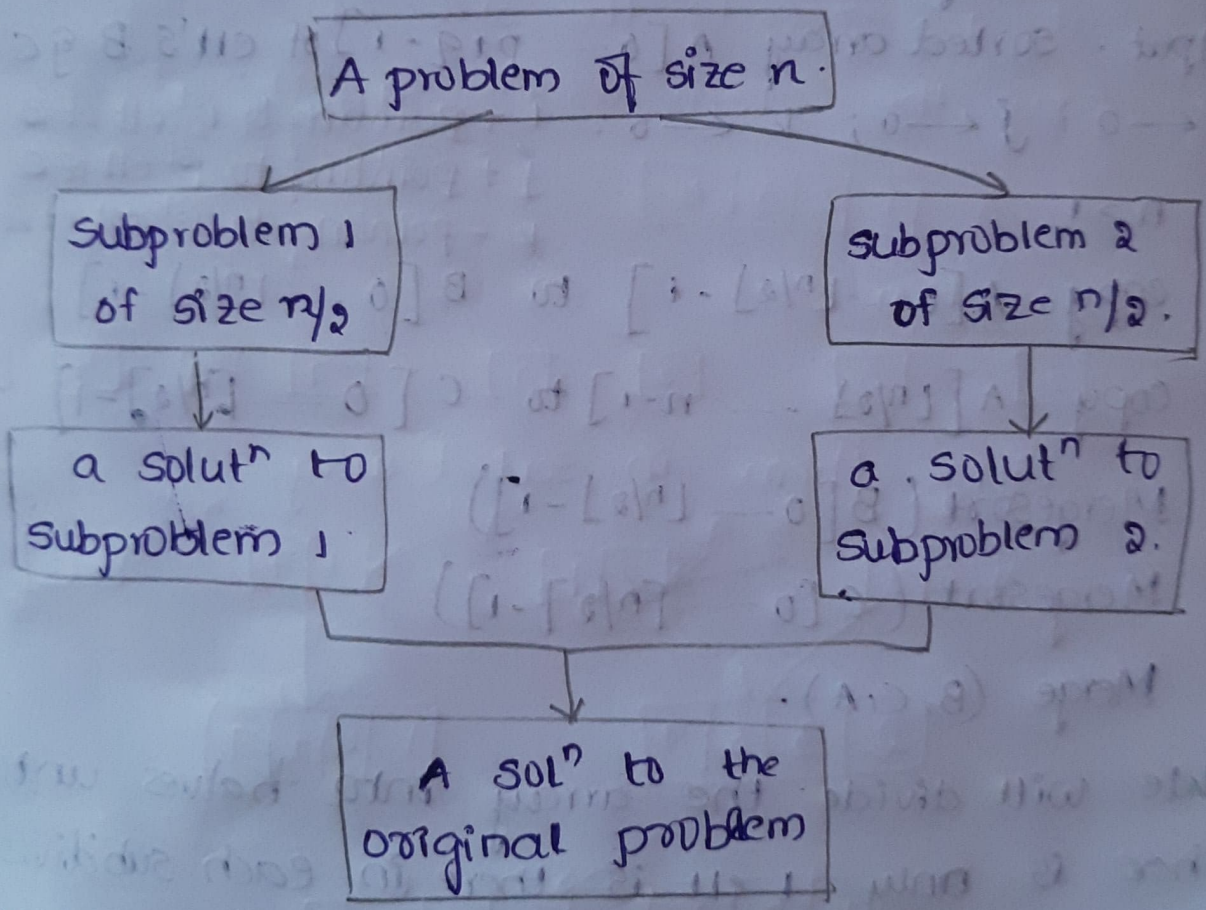
→ Otherwise, divide a problem instance into smaller instances of the same problem.

→ Recursively solve the smaller instances.

→ Combine the solutions to the smaller instances to get solution for the original instance, return solution.

→ Every subdivision is independent of the other subdivision & are solved parallely.

→



MASTER'S METHOD -

The master method applies to recurrences of the form:

$$T(n) = a \cdot T(n/b) + f(n)$$

where $a \geq 1$, $b > 1$ and f is asymptotically

positive,

If $f(n) \in O(n^d)$ where $d \geq 0$

$T(n) = O(n^d)$ if $a < b^d$

$= O(n^d \log_b n)$ if $a = b^d$

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

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]

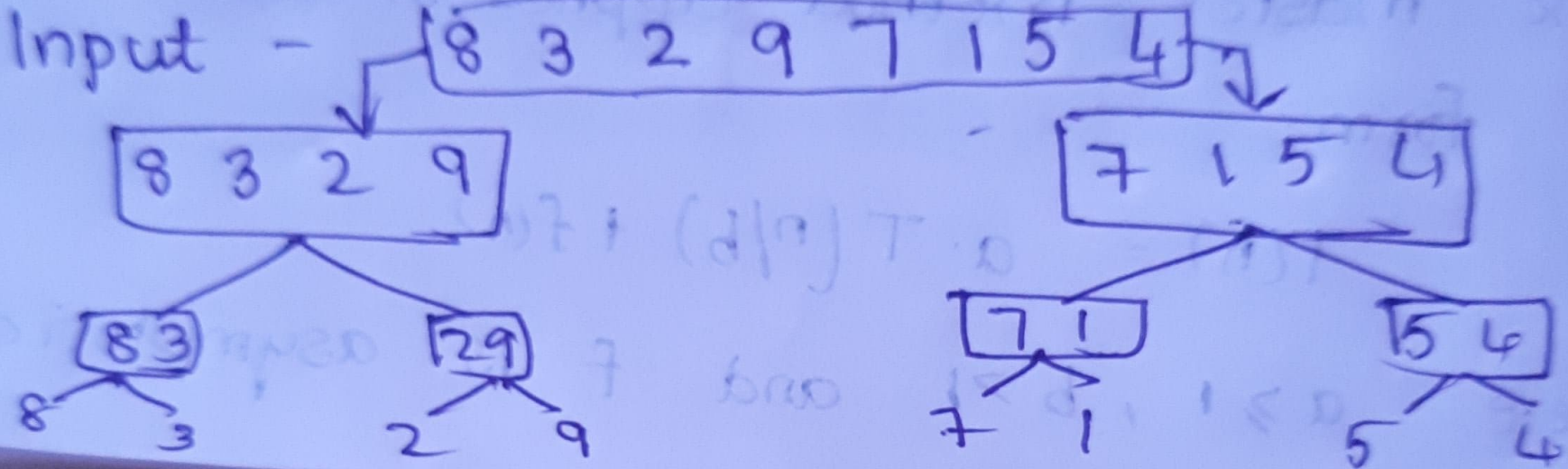
Mergesort (B[0 -- ⌊n/2⌋ - 1])

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.

eg :-



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

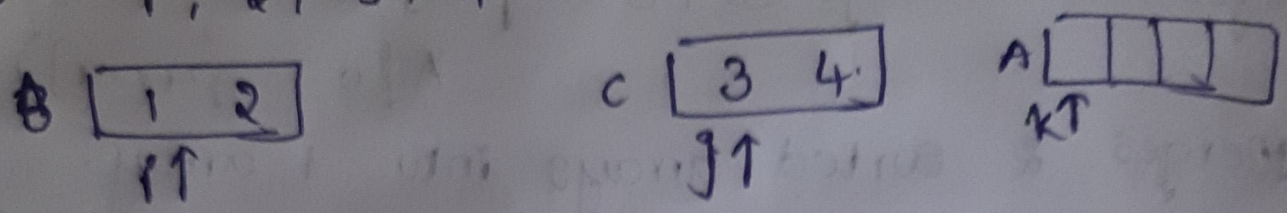
Merges 2 sorted arrays into 1 array.

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

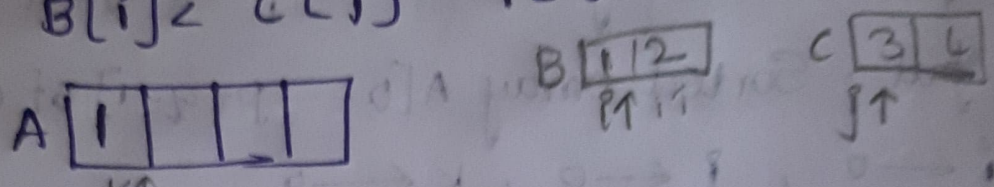
Output - Sorted array A[0 — p+q-1] of elt's of B & C.

```
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.

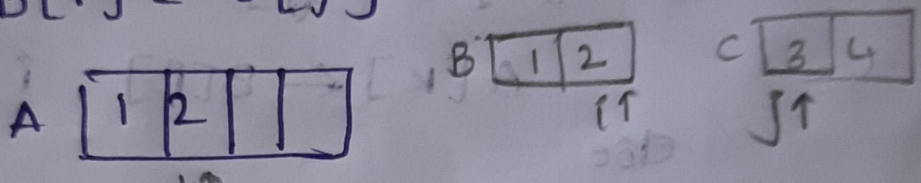


① st step - B[i] < C[j] i < 3.



i ++ & k ++.

② nd step - B[i] < C[j] 2 < 3



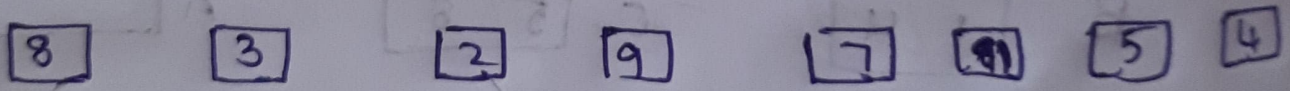
i ++ & k ++

But 'i' exceeds the size of array.

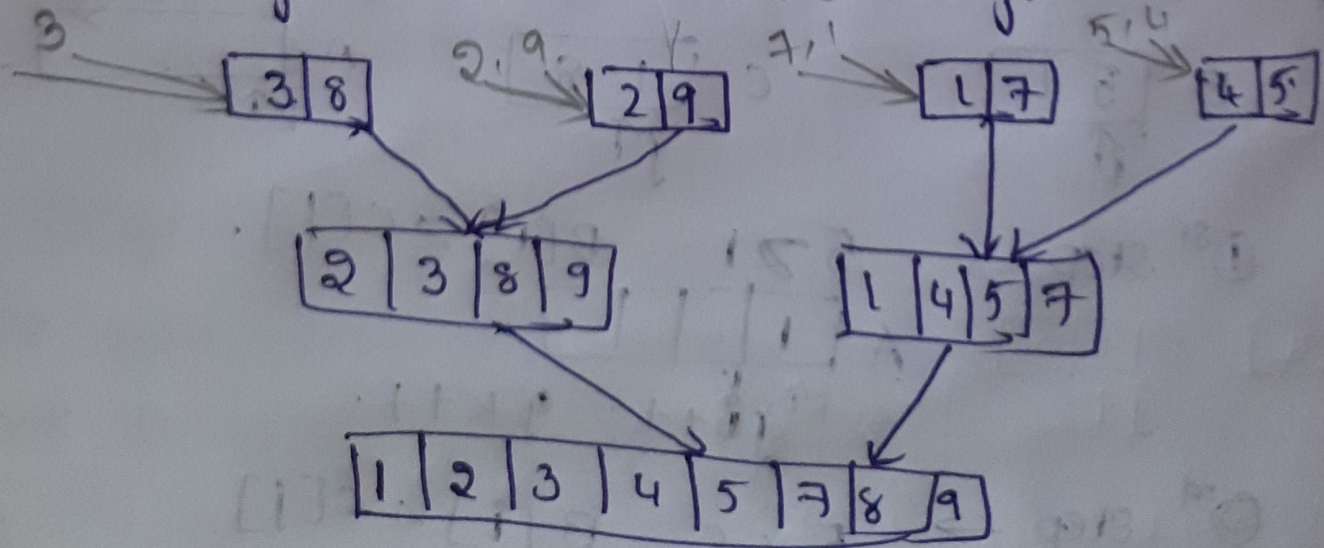
So, copy the elements in C array to A.

A 1 2 3 4.

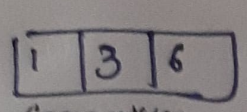
Main q :- Continuation -

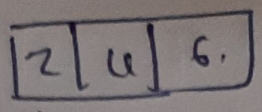


Merge them in sorted way.



eg: (2)





Comparision		Comparision count	
1 & 2	-	1	A [1]
2 & 3	-	1	A [1 2]
3 & 4	-	1	A [2 2 3]
4 & 6	-	1	A [1 2 3 4]
6 & 5	-	1	A [1 2 3 4 5]

A [1 2 3 4 5 6]

Worst case: 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

$$C(n) = 2C(n/2) + C\_merge(n) \quad \text{for } n > 1$$

$$C(1) = 0.$$

In the worst case -

$$C\_merge(n) = n-1 \quad \text{(refer eq - (2))}$$

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

$$C\_worst(1) = 0.$$

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,

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