

→ CountAndSay(n) is the way you "say" CountAndSay(n-1)

Case(1) = 1

Case(4) = 1 2 1 1

Case(2) = 1 1

Case(5) = 1 1 1 2 2 1

Case(3) = 2 1

s = 3 3 2 2 2 5 1 1 1 1  
j i ans = 2 3 3 2 1 5 4 1

→ Code :- ps String countAndSay(int n){

if(n==1) return "1";

String s = countAndSay(n-1) + "@";

// Now we have to modify s.

String ans = "";

int i = 0, j = 0;

while(j < s.length()){

if(s.charAt(i) == s.charAt(j)) j++;

else {

int len = j - i;

ans += len;

ans += s.charAt(i);

i = j;

}

}

return ans;

psvm {

int n = 4;

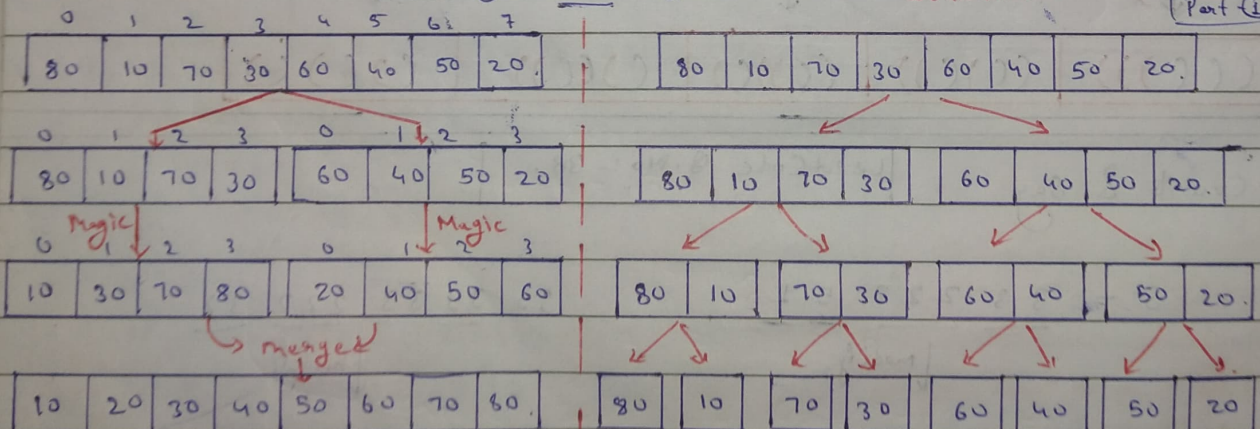
cout << countAndSay(n);

}

## # Merge Sort Algorithm

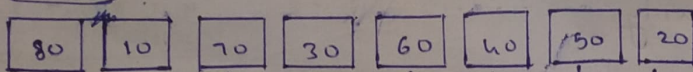
Work flow :-

Part (1)





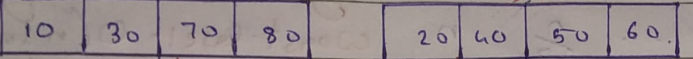
Part (2)



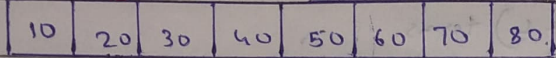
merge 2 Sorted Array.



merge 2 Sorted Array.



merge 2 Sorted Array.



## # Divide & Conquer

⇒ Code :- `psv mergesort(int[] arr){`

`int n = arr.length;`

`if (n == 1) return; // Base Case.`

`// Create two arrays of n/2 size each...`

`int[] a = new int[n/2];`

`int[] b = new int[n-n/2];`

`// Copy Pasting...`

`for (int i = 0; i < n/2; i++){`

`a[i] = arr[i];`

`}`

`for (int i = 0; i < n-n/2; i++){`

`b[i] = arr[i + n/2];`

`}`

`// Magic...`

`mergesort(a);`

`mergesort(b);`

`// Merge these 'a' & 'b'...`

`merge(a, b, arr);`

`// Delete a & b...`

`a = null; b = null;`

`}`

`psv print(int[] arr){`

`for (int ele : arr){`

`cout << ele << " ";`

`}`

`cout << endl;`

`}`

M T W T F S S

Page No.:

YOUVA

Date:

Copy pasting.

$$T(n) = \left[ n + \left( \frac{n}{2} + \frac{n}{2} \right) + \left( \frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4} \right) + \left( \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} \right) + n \right]$$

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

$$T(n) = 2(1-1) \cdot n$$

`psv main {`

`int[] arr = {80, 30, 50, 20, 60, 10, 70, 40};`

`print(arr);`

`mergesort(arr);`

`print(arr);`

`}`

`psv merge(int[] a, int[] b, int[] c){`

`int i = 0, j = 0, k = 0;`

`while (i < a.length & j < b.length) {`

`if (a[i] <= b[j])`

`c[k++] = a[i++];`

`else`

`c[k++] = b[j++];`

`}`

`while (j < b.length)`

`c[k++] = b[j++];`

`while (i < a.length)`

`c[k++] = a[i++];`

`}`



# # Time And Space Complexity :-

$n$   
 $n/2$   
 $n/4$   
 $n/8$   
 $n/16$   
 $\vdots$   
 $1$

$n \quad \frac{n}{2} \quad \frac{n}{4} \quad \frac{n}{8} \quad \dots \quad 1$

or

$1 \quad 2 \quad 4 \quad \dots \quad \frac{n}{2} \quad n$

$(\log_2 n + 1)$  terms.

levels

Best Case :-  $O(n \log n)$

Avg Case :-  $O(n \log n)$

Worst Case :-  $O(n \log n)$

$$T.N.O = 2(1-1).n = 2 \cdot (\log_2 n + 1 - 1) n$$

$$= 2 n \log n$$

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

In this basic code, the amount of space used =  $(1-1).n$   
 $= n \log_2 n$

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

we can improve it by deletion of a & b after merging into arr.

Best case :-  $O(n)$

Worst Case :-  $O(n)$

Avg case :-  $O(n)$

$$\text{Extra Space Used} \therefore n + \frac{n}{2} + \frac{n}{4} + \dots = n \left( 1 + \frac{1}{2} + \frac{1}{4} + \dots \right) < 2n$$

$$S.C = O(n)$$

Maths :-

$$1 + \frac{1}{2} + \frac{1}{4} + \dots < 2$$

$$n(1 + \frac{1}{2} + \frac{1}{4} + \dots) < 2 * n$$

## # Stability of Merge Sort :-

Yes. merge sort is stable.

arr = 5 1 2 5\* 3

↓  
Sorting.

1 2 3 5 5\*

Make sure to code like this.

if(a[i] <= b[j])

c[k++] = a[i++];

if I use '<' instead of '<=' then it won't be stable.



→ Applications of Merge Sort

- ①  $O(n \log n)$  worst case time complexity.
- ② Custom Sorting.
- ③ Sorting Linked Lists
- ④ Inversion Count & Related Problems

Page No.:

YOUVA

Date:

[LeetCode 493]

# Inversions Count Problem

→ You have to count the no. of inversions.

arr = { 8, 2, 5, 3, 1, 4 }

→ (8, 2), (8, 5), (8, 3), (8, 1), (8, 4)

→ (2, 1), (3, 1)

→ (5, 3), (5, 1), (5, 4)

[ans = 10]

$i < j$

~~arr[i] > arr[j]~~

Note :- Just need to modify the merge sort code.

# Basic Approach [Brute Force]

→ Code :-

```
psv inversion(int[] a, int[] b){
```

```
    int i = 0, j = 0;
```

```
    while (i < a.length && j < b.length){
```

```
        if (a[i] > b[j]){
```

```
            count += (a.length - i);
```

```
            j++;
```

```
        }
```

```
        else i++;
```

```
    }
```

```
}
```

T.C  $\Rightarrow O(n^2)$

S.C  $\Rightarrow O(1)$

Not the Best

arr = { 8, 2, 5, 3, 1, 4 }

a = 2, 5, 8

b = 1, 3, 4

count = ~~3~~ + ~~2~~ + ~~2~~ = 7

arr[i] > arr[j]

↓  
ele from a

↓  
ele from b

\* Merge sort

→ Create two array a & b of  $n/2$  size each & copy paste.

→ Sort(a).

→ Sort(b).

(add the inversion function & call it)

→ Count inversions in a & b. [Extra]

→ Merge(a, b, arr). [also create count global variable]

M-2 → [just ~~need~~ modify the merge function in mergesort code]

→ Code :- while (i < a.length && j < b.length){

```
    if (a[i] <= b[j])
```

```
        c[k++] = a[i++];
```

```
    else { // a[i] > b[j]
```

```
        count += (a.length - i); // Extra
```

```
        c[k++] = b[j++];
```

```
    }
```

[Only thing need to modify]

T.C  $\Rightarrow$

$O(n \log n)$

No need to create extra function