Date 03/11/2023 Page

Day-47

Hard Problems on Arrays

* **How to store 2 no. in 1 position:**

602 N = 1 to 9

into a store no. & its occurance

=) We have to store no. & its occurance

=) For storing 2 things or no. (value) at once, we will take one greater no. than the upper range.

=) Now, do (let no. is 2 & occurance is 6)
Then 2 + 6 * 100 = 602.
(value)

=) So, in this way 2 no. store in one no.

=) Number = 602 % 100 = 2

Occurance = 602 / 100 = 6

* **Find Missing & Repeating:**

=)

arr = | 4 | 3 | 2 | 1 | 2 | 7 | 6 |

=) Here, in a range of no. given in the array. There is a missing no. & in the place of missing no., one no. is repeating.

=) N = 7

That means, the range is 1 to 7.

=) Here, 2 is repeating & 5 is missing.

Date _____ Page _____

=| In Brute Force Approach, iterate the whole array & find the missing & repeating no.

=| $O(n^2)$ → Time Complexity.

=| Now for more optimized approach, first sort the array then find the repeating element then missing element by using BS. (O(nlogn) → T.C.

=| Now, for more optimized sol., create an empty array of N size then iterate the array & count the occurance of each no. & store in new array.

=| After this, calculate the missing & repeated no.. (O(N) → T.C, O(N) → S.C)

=| Here, I have solved ~~this~~ the above approach with extra space.

= So, we can reduce it by reducing the no. by 1 & storing it in their respective index with count.

Code:

```
vector<int> count(N,0);
for(i=0; i<N; i++)
    count[arr[i]-1]++;
~~missi~~
for (i=0; i<n; i++){
```

Date _____ Page _____

```
        if (count[i] == 0){
            cout << i+1;
            break;
        }
    }
}
for(i=0; i<N; i++){
    if (count[i] == 2){
        cout << i+1;
        break;
    }
}
```

T.C → O(N)

S.C → O(N)

Best & Optimized Approach

⇒ We store two no. at one place — one no. is itself no. of that index & other is occurance of that number.

0	1	2	3	4	5	6
4	3	2	1	2	7	6

First

0	1	2	3	4	5	6
3	2	1	0	1	6	7

3 % 7 = 3 ← 2 → 2 occurance of 1

Then add 7 to the no. ~~3 + 7 = 10~~

value of 3 index 2 + 7 * 2 = 16

0 + 7 = 7

0	1	2	3	4	5	6
10	16	8	7	1	13	12

Date

Page

Code

```
for(i=0; i<N; i++)
    arr[i]--;
for(i=0; i<N; i++){
    arr[arr[i]%N] += N;
}
for(i=0; i<N; i++){
    if(arr[i]/N==0){
        cout << i+1;
        break;
    }
}
for(i=0; i<N; i++){
    if(arr[i]%N == 2){
        cout << i+1;
        break;}
}
```

* Find the occurrence of Number:

3	2	5	3	1	2	3	7

⇒ We have to print the occurance of every no. in range from 1 to N.

⇒ We use the same approach that we use in last question.

Date _____ Page _____

* **Majority Element:**

0	1	2	3	4	5	6	7	8	9	10	
3	3	2	3	1	3	2	2	1	3	3	N=11

=) We have to return that no. - ~~thats' is~~ occurrence is $> N/2$.

=) Here, we will cancel the no. with other no. because if a no. is $> N/2$ than other no. than if we cancel it with other no. than the last remaining no. is our ~~desried~~ desired o/p.

~~3~~ ~~2~~ +
~~3~~ ~~2~~ T
~~3~~ ~~2~~
~~3~~
~~3~~
(3) → o/p

=) This algo. is called Moore Voting algo.

=) Sometimes, there is no gaurantee that the no. we get is more than $N/2$.

=) So, in that case, we have to verify it's count value.

2	2	2	3	3	3	4

~~2~~ ~~3~~ (4) → But it comes only 1 time
~~2~~ ~~3~~ So, it's not our desired o/p.
~~2~~ ~~3~~

Date

Page

Code

```
int candidate, count = 0;
for(i=0; i<n i++){
    if(count == 0)
        count = 1;
        candidate = arr[i];
    }
    else{
        if(candidate == arr[i])
            count++;
        else
            count--;
    }
}
count = 0;
for(i=0; i<N; i++){
    if(arr[i] == candidate)
        count++;
}
if(count > N/2)
    return ~~count~~ candidate;
else
    return -1;
```