First Missing Positive Integer

Ex:-

arr[5] → 3 -2 1 2 7 ans = 4

arr[7] → -8 2 6 4 -7 1 3 ans = 5

arr[6] → 2 1 6 4 3 5 ans = 7

arr[5] → -4 8 3 -1 0 ans = 1

//idea-1

Sort the array & traverse upto last element

TC : $O(n \log n) + O(n) \approx O(n \log n)$

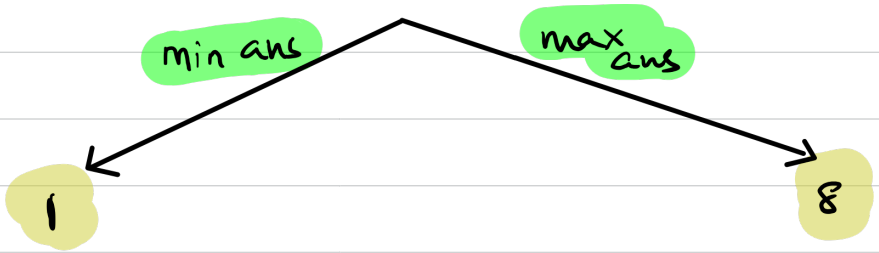
Made with Goodnotes

// idea 2

Observation - 1

if numbers are assigned properly

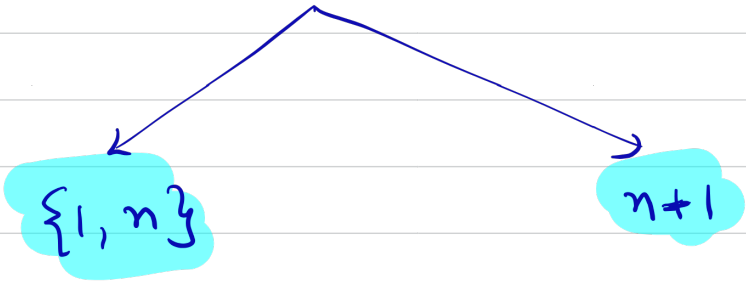
arr[7] :	1	2	3	4	5	6	7
idx =	0	1	2	3	4	5	6



So, for array size = 7 $\Longrightarrow$ Range = (1, 8)

In general Size = n $\Longrightarrow$ Range = (1, n+1)

Answer could be from (1 $\rightarrow$ n+1)



(If each corresponding index is having its (+1) value then ans = n+1)

Made with Goodnotes

	0	1	2	3	4	5	6	7
arr[8] :	1	2	3	4	5	6	7	8

Each corresponding index has its value as (index + 1)

Our main moto is to convert given array into above pattern, from that we can easily get the first missing value by O(n)

So, how to do that Mapping? ⇒ By "Swapping"

	0	1	2	3	4	5	6	7
arr[8] :	4	2	-7	6	9	1	-8	3
Assumption[]:	1	2	3	4	5	6	7	8

arr[0] = 4 ⟶ index = 3 , Swap (0, 3)

If we observe arr, arr[0] = 4 (According to our assumption Value 4 need to be index = 3)
So Swap Current idx with 3

Writing every iteration in-detail ↓

Made with Goodnotes

arr[0] = 4 ⟹ Actual Index = 3, Swap(0,3)

arr[0] = 6 ⟹ Actual Index = 5, Swap(0,5)

arr[0] = 1 ⟹ Actual Index = 0, increment i

arr[1] = 2 ⟹ Actual Index = 1, increment i

arr[2] = -7 ⟹ Irrelevant, increment i

arr[3] = 4 ⟹ Actual Index = 3, increment i

arr[4] = 9 ⟹ Irrelevant, increment i

arr[7] = 3 ⟹ Actual Index = 2, Swap(7,2)

Note:

If Swapping Values are Same ⟹ increment i

just i++

for example:

arr[5] = {4, 1, 1, 3, 4, 2}

Here if we Swap (arr[0], arr[2])

No use

Made with Goodnotes

```java
class Solution {
    public int firstMissingPositive(int[] nums) {
        int n = nums.length;

        int i = 0;
        while(i<n){
            if(nums[i]<1 || nums[i]>n || nums[i] == i+1){
                i++;
            }
            else{
                int idx = nums[i]-1;
                if(nums[i] == nums[idx])i++;
                else{
                    int temp = nums[i];
                    nums[i] = nums[idx];
                    nums[idx] = temp;
                }
            }

        }
        for(i = 0;i<n;i++){
            if(nums[i]!=i+1)return i+1;
        }
        return n+1;
    }
}
```

case irrelevant

Already Solved

if Swapping values are Same Case

Made with Goodnotes