class 2 Arrays

**Q1** Missing number

[3,0,1]  0 - - - n

$1^{st}$ approach : [0, 1, 3]

0+1 = 1

1+1 = 2

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

$2^{nd}$ approach : Hashset

[3, 0, 1]

| |
|---|
| 1 |
| 0 |
| 3 |

[0,1,2,3]

Size = n

No. of elements = n+1

0 - - - n+1

[9, 6, 4, 2, 3, 5, 7, 0, 1]

| |
|---|
| 1 |
| 0 |
| 7 |
| 5 |
| 3 |
| 2 |
| 4 |
| 6 |
| 9 |

no = 10

0 - - - - 10

8

**Algorithm :**

1) Maintain a hashset of elements
2) No. of elements = n+1

Check if no. from 0 - - - n+1 are present in hashset/not of not, return the no.

TC : $O(n)$

SC : $O(n)$

**Q2** Merge the intervals

[[1, 3], [2, 6], [8, 10], [15, 18]]

**Ans :** ~~[1,3], [2,6]~~ = [1,6], [8,10], [15,18]

[1,4], [4,5], [8, 11], [10, 13]

**Ans** ~~[1,4] [4,5]~~ = [1, 5], ~~[8,11]~~, [8, 13]

= [1, 5], [8, 13]

## Algorithm :

1) Sort the intervals based on starting element (start time),
2) Insert 1st element into the answer
3) continue inserting each interval

   we'll check if starting of new interval $\leq$ end of last inserted interval

   If it is true, update end time of last inserted interval to be max of end time of both intervals.

   If it is false, simply insert the interval
4) Return the ans.

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

## Q3 Merge sorted array

[1, 2, 3, 0, 0, 0]    [2, 5, 6]

m = 3    n = 3

↓

[1, 2, 2, 3, 5, 6]

1, 2, ~~2~~, ~~3~~, ~~5~~, ~~6~~    2, 5, 6    = [1, 2, 2, 3, 5, 6]

~~ptr 2~~ ~~ptr 2~~ ptr 2

new_array [1 | 2 | 3]

~~ptr 1~~ ~~ptr 1~~ ~~ptr 1~~ ptr 1

## Algorithm :

1) Initialise a new array (→ nums1copy) containing the first m elements of num 1.
2) Initialise p1 to beginning of nums1 copy
3) Initialise p2 to beginning of nums 2
4) If nums1 copy [p1] exists & is less than/= nums 2 [p2]

   write nums1copy [p1] in num1 & increment p1

   else

   write nums 2 [p2] in num 1 & increment p2.

TC : $O(m+n)$

SC : $O(m)$

Q4: calculate majority element

3 4 3 7 3 3

Brute force:

$n-1+n-2+\ldots+1 = O(n^2)$

Approach 2

3 3 3 3 4 7 $O(n \log n)$ ← sorting

Approach 3

Hash Map

| Number | Frequency |
|---|---|
| 3 | ~~1~~ ~~2~~ ~~3~~ 4 |
| 4 | 1 |
| 7 | 1 |

$> 6/2$

$> 3$

= 3 is the ans

TC: $O(n)$

SC: $O(n)$

Algorithm: 1) we can use a hashmap to store elements & their count/frequency

2) Return the element with frequency $> n/2$

Q5: Duplicate Number

[1, 3, 4, 2, 2]

Approach: 1 Brute force: $n-1+n-2+\ldots 1 = O(n^2)$

2: Sorting

1, 2, 2, 3, 4 $O(n \log n) + O(n) = O(n \log n)$

3:

| |
|---|
| 2 |
| 4 |
| 3 |
| 1 |

2

Algorithm:

1) Try to store elements in the hashset

2) If an element is already present, return that element

Q6 Bit manipulation

{2, 10} {a, b}

→ 2, 4, 5, 6, 8, 10 [b-a]+1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

$\{2, 4, 5, 6, 8, 10\}$

num % 2 == 0

num % 5 == 0

**Algo:**

1) creating an array of size $b-a+1$
2) For each element we are checking if it is divisible by 2 or 5
   if " = mark it as 1
   else mark it as 0
3) Return all numbers marked as 1

$TC = O(b-a)$

**Q7** Make array a palindrome

abcba = abcba  121

NAMAN = NAMAN

$\{15, 4, 15\} \equiv \{15, 4, 15\}$ } - Palindrome array

0

1, 4, 5, 9, 1  1, 9, 5, 4, 1

~~p1~~ p1  p2 ~~p2~~

1, 17, 35

p1  p2

1 9 9 1

18 35

1

[53] , [53]

2

Reverse of array

**Algorithm:**

1) Let $f(i,j)$ be the minimum no. of operations to make subarray $[i \cdots j]$ a palindrome

we start from $i=0$ & $j=n-1$

2) If $arr[i] == arr[j]$, we don't need to do any operation,

else $arr[i] > arr[j]$, we'll merge $j-1$ & $j$

$$arr[j-1] = arr[j-1] + arr[j]$$

$$ans = 1 + f(i, j-1)$$

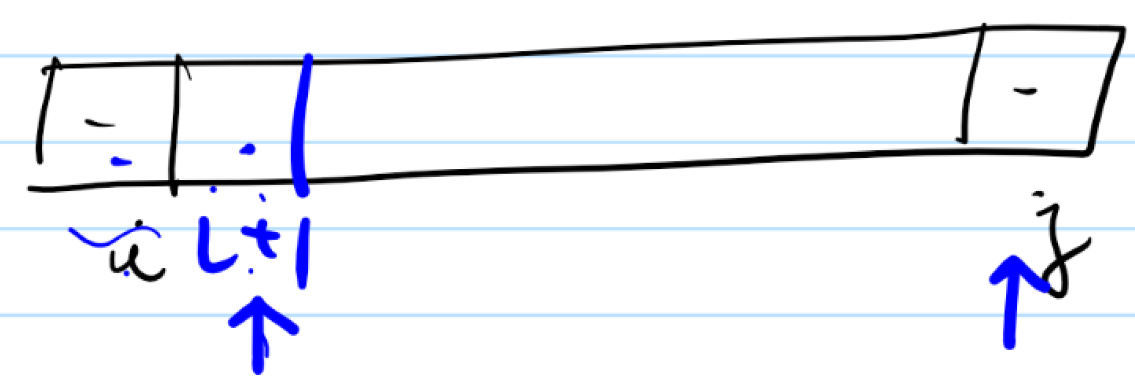
$arr[i] < arr[j]$,

$$arr[i+1] = arr[i] + arr[i+1]$$

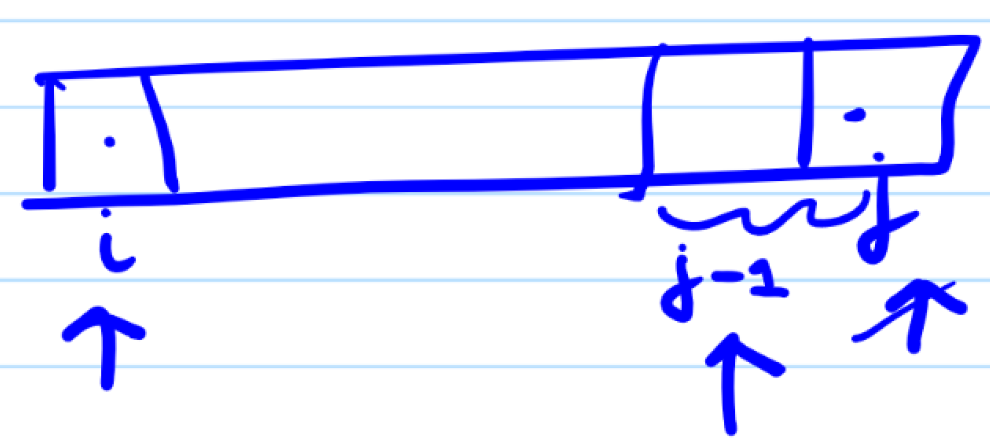
$$ans = 1 + f(i+1, j)$$

3) Return $f(0, n-1)$

$i < j$




$arr[i] > arr[j]$




$\{15, 4, 15\}$ $= O(n)$ ← TC

$O(1)$ ← SC