arr[] = [1, 2, 4, 5] N = 5

1 to N

⇒ 1 to 5 here n = 3 is missing

Brute force

→ take every number and check if it's present in the arr[] or not (Linear Search)

```
for (i=1; i<=N i++)
{ flag = 0
  for(j=0; j<n-1; j++)
  {
    if (arr[j] == i)
        flag = 1;
        break;
  }
if (flag == 0)
  return i;
}
```

T.C ⇒ O(N×N)

Better Approach (Hashing) arr[] = {1, 2, 4, 5} N = 5

Declare hash array of size = 6

hash

0	1	2	3	4	5
0	1	1	0	1	1

N = 6

→ Everything will be '0' first then we iterate over the array and

mark '1' if its there.

→ Then again we iterate the hash array from 1 to N and figure out which is not marked.

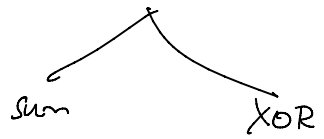
Pseudo Code

```
hash [n+1] = {0}
for(i=0; i<n; i++)
{
    hash [arr[i]] = 1;
}
for(i=1 → N)
{
    if (hash[i] == 0)
        return i
}
```

T.C ⇒ O(N) + O(N)

S.C ⇒ O(N) (Hash array)

Optimised Solution (two method)



arr[] = [1, 2, 4, 5] N = 5

Sum

Sum of N natural nos

$$Sum = \frac{N(N+1)}{2}$$

$$Sum = \frac{5 \times 6^{3}}{2}$$

Sum = 15

1 2 3 4 5

XOR

$a \wedge a = 0$ $2 \wedge 2 = 0$, $2 \wedge 2 \wedge 5 \wedge 5 = 0$

$0 \wedge 0 = 0$

$2 \wedge 2 \wedge 2 \wedge 2 \wedge 2$

$0 \wedge 0 \wedge 2$

$0 \wedge 2$

{1, 2, 4, 5}

S2 = 1 3 7 12

now ⇒ Sum − S2

= 15 − 12 = (3) output

Sum = $\frac{n(n+1)}{2}$ S2 = 0

for(i=0 → n) { S2 t= arr[i] }

return Sum − S2.

T.C ⇒ O(N) S.C ⇒ O(1)

0 ^ 2

= 2

0 ^ a = a

when n = 5

1 2 3 4 5

arr[] = 1 2 4 5

XOR1 = 1 ^ 2 ^ 3 ^ 4 ^ 5

XOR2 = 1 ^ 2 ^ 4 ^ 5

XOR1 ^ XOR2

(1 ^ 1) ^ (2 ^ 2) ^ (3) ^ (4 ^ 4) ^ (5 ^ 5)

0 ^ 0 ^ 3 ^ 0 ^ 0

0 ^ 3 = 3 ⇒ output.

XOR1 = 0

for (i=1 → N; i++)

XOR1 = XOR1 ^ i;

will not run this extra loop so for that

XOR2 = 0

for(i=0; i<N-1; i++)

XOR2 = XOR2 ^ arr[i]

XOR1 ^ XOR2 ⇒ ans.

```
XOR1 = 0
XOR2 = 0
for ( i = 0; i < N-1; i++)
{
    XOR2 = XOR2 ^ arr[i]
    XOR1 = XOR1 ^ (i+1)
}
    XOR1 = XOR1 ^ N
return XOR1 ^ XOR2
```

T ⊂⇒ O(N)

S ⊂⇒ O(1)

XOR is better than Sum

8. Max Consecutive Ones

485. Max Consecutive Ones

Hint

Easy 4.1K 423

Companies

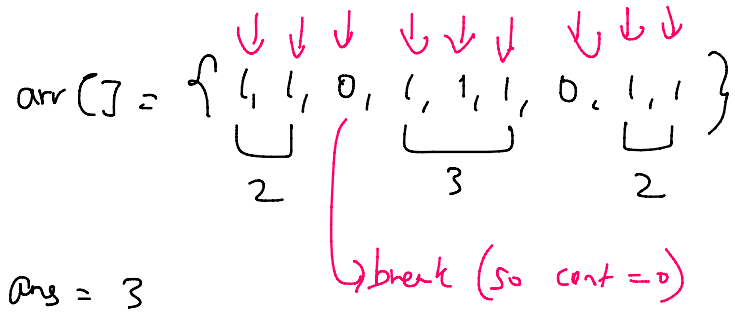
Given a binary array `nums`, return *the maximum number of consecutive* `1`*'s in the array.*

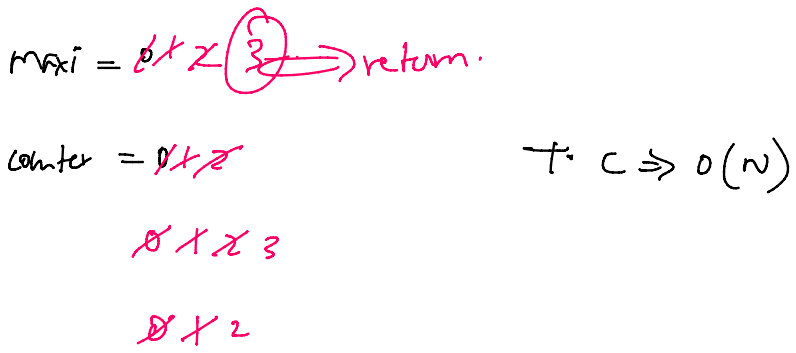
Example 1:

```
Input: nums = [1,1,0,1,1,1]
Output: 3
Explanation: The first two digits or the last three digits are
consecutive 1s. The maximum number of consecutive 1s is 3.
```

Example 2:

```
Input: nums = [1,0,1,1,0,1]
Output: 2
```





```cpp
class Solution {
public:
    int findMaxConsecutiveOnes(vector<int>& nums) {
        int maxi = 0;
        int cnt = 0;
        for(int i=0; i<nums.size();i++){
            //if the current element of the input vector is equal to 1.
            if(nums[i] == 1){
               // If the current element is 1, these two lines increment the "cnt" variable by 1 and update the value of "maxi" to be the maximum of its current value and the current value of "cnt".
                cnt++;
                maxi = max(maxi, cnt);
            } else {
                cnt = 0;
            }
        }
        return maxi;
    }
};
```

9. Single Number (Find the number that appears once, and other numbers twice.)

arr [] = {1, 1, 2, 3, 3, 4, 4} → ans

Brute force

⟷ Pick every number and do Linear search and check how many times it appear.

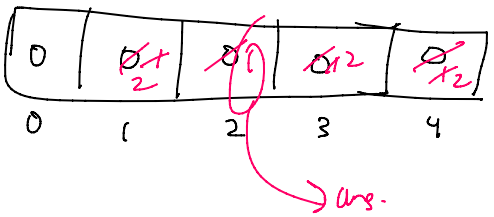
```
for(i=0 → n)
{  num = arr[i]
   cnt=0
   for(j=0 → n)
   {  if(arr[j] == num)
         cnt++
   }
   if(cnt==1) return num;
}
```

T.C ⇒ $O(n^2)$

2nd Approach (Hashing)

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

→ maxElement = 4



So you declare arr of size = 5

What size of hash array you define?

→ hash[maxelement + 1]

```
maxi = arr[0]
for(i=0; i<n; i++)
    maxi = max(maxi, arr[i]) // this will max element   } O(n)
hash[maxi] = {0}   // initialise to {0}
// Iterate in the arr
for(i=0 ⟶ n)
    hash[arr[i]]++   } O(n) // for marking
for(i=0 ⟶ n)
if(hash[arr[i]] == 1) return arr   ] ⇒ O(n)
```

T.C ⇒ O(3N) S.C ⇒ O(max Element)

3rd Approach

Given a **non-empty** array of integers `nums`, every element appears *twice* except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

```
Input: nums = [2,2,1]
Output: 1
```

1 2 1 2 4

XOR

1 ^ 1 = 0

$2 \wedge 2 = 0$

$x \wedge x = 0$

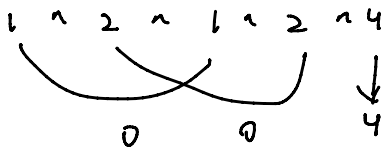
$0 \wedge 1 = 1$

$1 \wedge 0 = 0$

$0 \wedge x = x$

1) Traverse the array

2) Do XOR



$0 \wedge 0 \wedge 4 = 4$

T.C $\Rightarrow O(N)$ S.C $\Rightarrow O(1)$

C++ Auto

```cpp
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int ans = 0;
        for(int i = 0;i<nums.size();i++){
            // performs a bitwise XOR operation on the current value of "ans" and the current element of the input vector,
            ans ^= nums[i];
        }
        return ans;
    }
};
```