**APPROACH 1: brute force**

1. Store nums in a set
2. Iterate from 0 to n
3. If i not in set:
   1. Return i

Time -> O(n)

Spac -> O(n)

**APPROACH 2: concept of sum**

Find sum of n numbers. Subtract with sum of array, gives missing number

1. S initialized to size of array(sum of n numbers will be sum(0 to n+1) )
2. For index, number in array:
   1. Add index subtract number
3. Return remaining

TIme -> O(n)

Space -> O(1)

**CODE:**

s = len(nums)

for i in range(len(nums)):

s += i - nums[i]

return s

**APPROACH 3: BITWISE SUM**

1. When a is xored with itself it returns 0, therefore if we xor every element of the array with the index, elements that have occurred, will become 0 eventually xor holds value that didn’t have its pair to be xored with to return 0
2. Return remaining

CODE:

xor = len(nums)

for i in range(len(nums)):

xor ^= i ^ nums[i]

return xor

**Approach 4: Binary Search**

Used when the input array is already sorted.

nums = sorted(nums)

left = 0

right = len(nums)-1

while left <= right:

mid = left + (right-left)//2

if mid == nums[mid]:

# move right

left = mid+1

elif mid < nums[mid]:

# move left

right = mid-1

return left