SINGLE NUMBER - II

☆ In this problem, we are given an array of integers in which each element appears thrice except one. Our job is to return the element that doesnt.

Bruteforce solution is to obviously use a hashmap to store how many times each element occurs and then iterate through it to check which element occurs only once.

We can optimize this using bit manipulation. If we lay down each number in binary, and check the bits at each index, if the bit is set | unset a certain times which is not a multiple of 3 it was set | unset in our required number

Pseudocode :

```
singleNumber (arr, N)    {
    ans = 0
    for (i = 0 → 31)    {
        c = 0
        for (j = 0 → N-1)    {
            if (arr[j] && (1 << i))  {
                c += 1
            }
        }
        if (c % 3 == 1) {
            ans = ans || (1 << i)
        }
    }
    return ans
}
```

We can also solve this question using a trick where we sort the array, start from index 1 and step value 3. If at any step element is not equal to its previous, thats our answer

Optimal Solution involves concept of buckets, where we have 3 buckets, ones twos and threes. We traverse through the array and at each element we check if it is in ones, if yes, we add it to twos, and add it to threes if it is in twos. If it is not in twos though, we add it to ones. All of this is done using bitwise operators.

Pseudocode :

```
singleNumber (arr, N) {
    ones = 0
    twos = 0
    for (i = 0 → N-1) {
        ones  = (ones ^ arr[i])  & (~twos)
        twos  = (twos ^ arr[i])  & (~ones)
    }
    return  ones
}
```