Arrays

Problem link:

<https://leetcode.com/problems/majority-element/description/>

Intuition:

nums = [2,2,1,1,1,2,2]

**Output:** 2

Here

2 2 1 1 1 2 2

Assume the first element is the majority

Make a | whenever count becomes 0

2

Count = 1

Again 2

Count 2

1

Count = 2 – 1 = 1

1

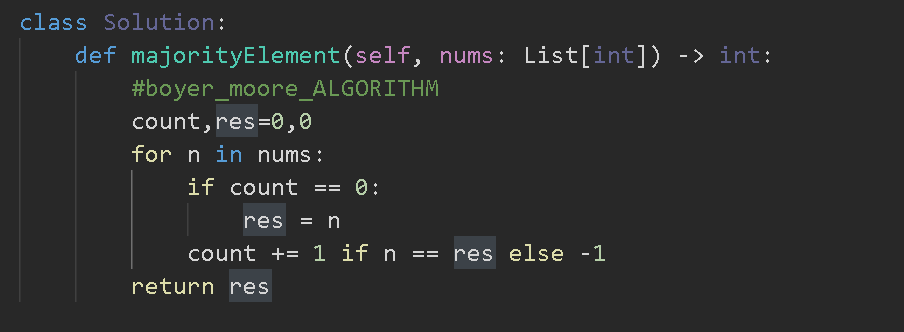
Count = 1 – 1 =0

Since count has become 0 , 2 cannot be the majority element anymore as it count is decreased by two 1s

And now the third 1 (index 4) becomes the new majority element

Likewise the algorithm goes on

Eventually the count of 1 also gets cancelled by two new 2s



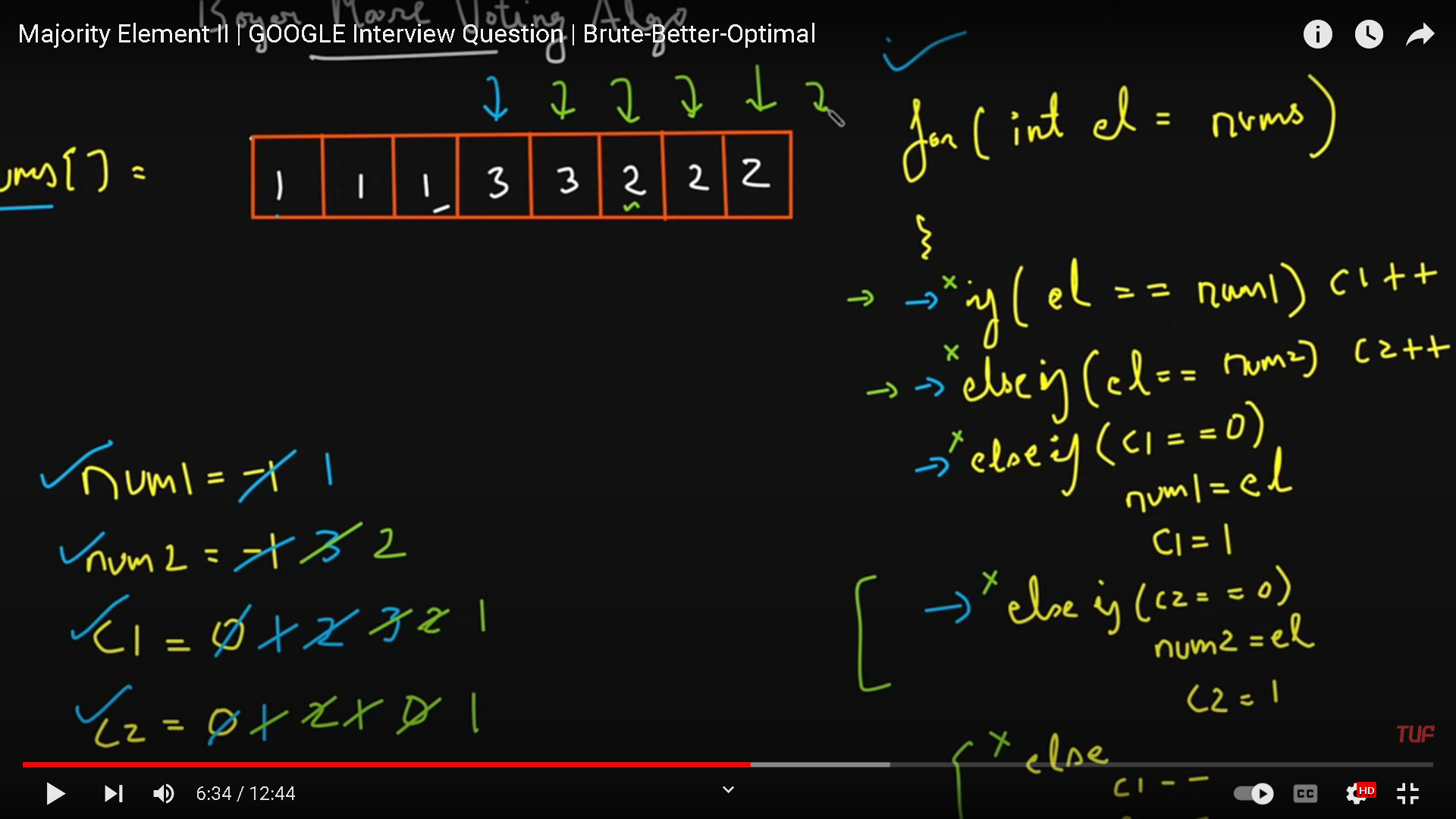
Problem link:

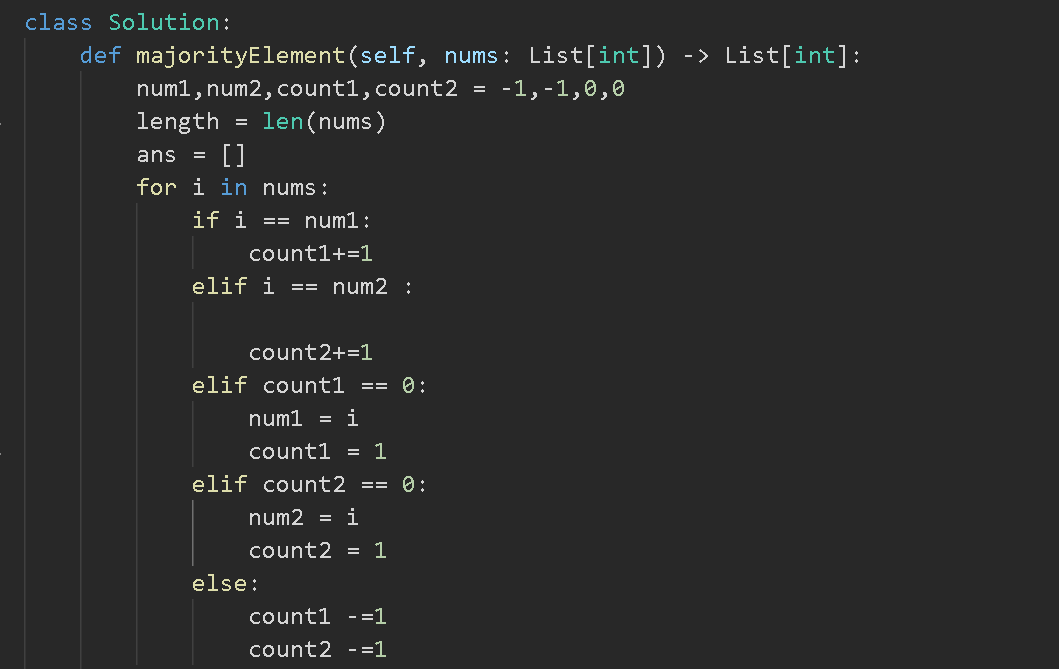
<https://leetcode.com/problems/majority-element-ii/description/>

There cannot be more than 2 majority elements for n/3

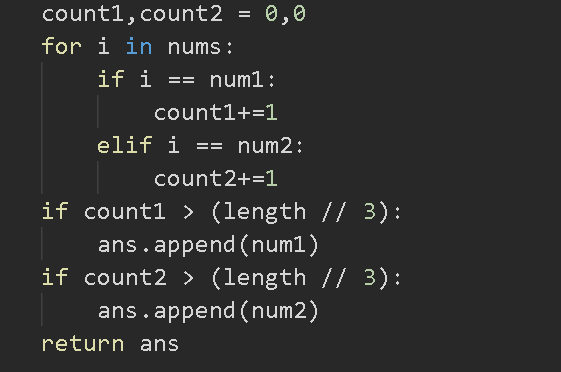
Because if n = 9

Atleast one element must occur 4 times to satisfy the condition of majority element



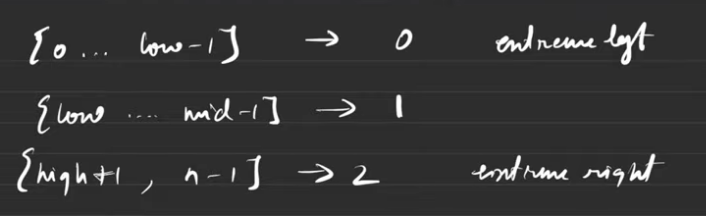


Now we have to check whether the current num occurs for more than n/3 times



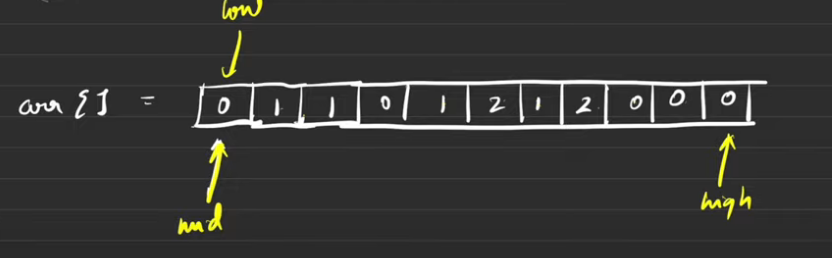
Dutch national Flag Algorithm

After result we want:

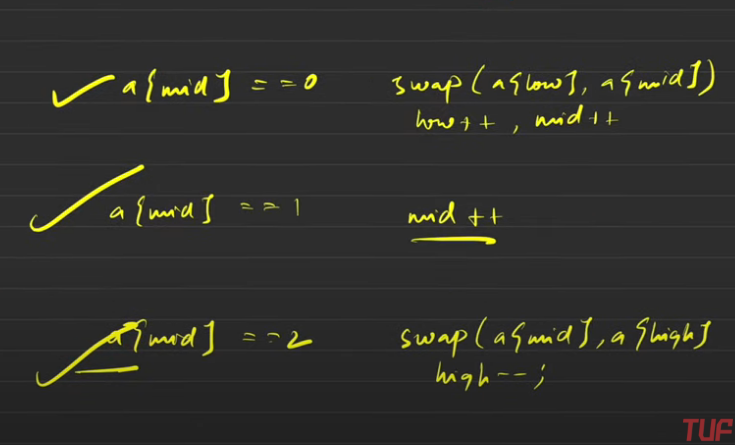


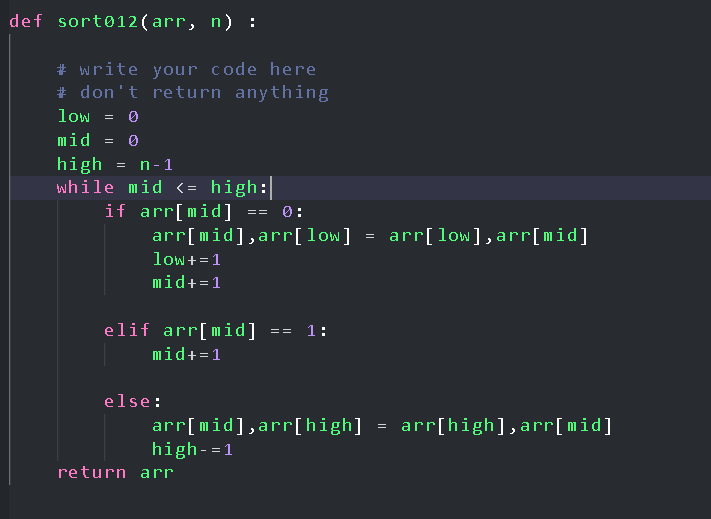
Intuition on how to swap elements:





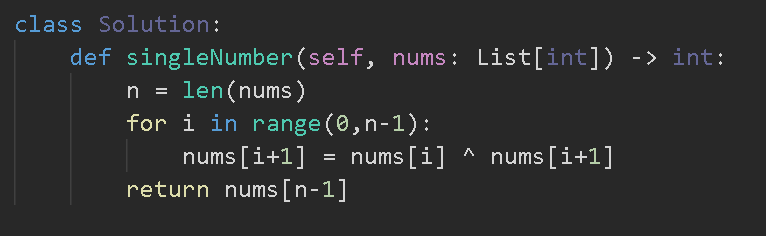
From mid to high all elements are unsorted as in the intuition diagram





Problem link:

<https://leetcode.com/problems/single-number/description/>



Two same numbers xor would cancel out each other but there no need to rearrange the elements such that all same numbers are adjacent as xor have the commutative property