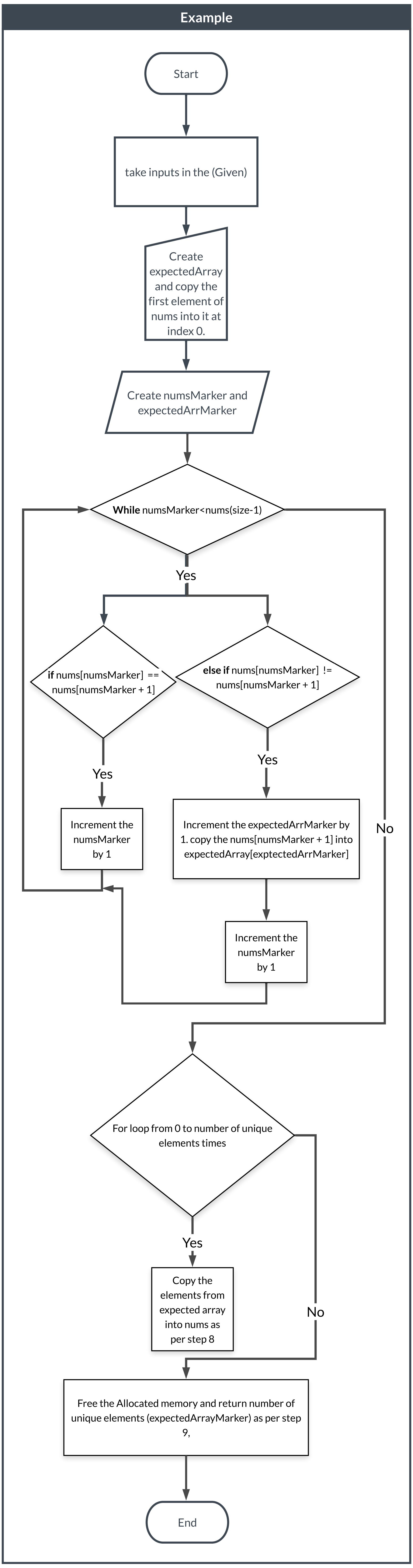
Remove Duplicates from Sorted Array

Link: https://leetcode.com/problems/remove-duplicates-from-sorted-array/?envType=problem-list-v2&envId=array


Example
Start
take inputs in the (Given)
Create expectedArray and copy the first element of nums into it at index 0.
Create numsMarker and expectedArrMarker
While numsMarker<nums(size-1)
Yes
if nums[numsMarker] == nums[numsMarker + 1]
else if nums[numsMarker] != nums[numsMarker + 1]
Yes
Yes
Increment the numsMarker by 1
Increment the expectedArrMarker by 1. copy the nums[numsMarker + 1] into expectedArray[exptectedArrMarker]
Increment the numsMarker by 1
No
For loop from 0 to number of unique elements times
Yes
Copy the elements from expected array into nums as per step 8
No
Free the Allocated memory and return number of unique elements (expectedArrayMarker) as per step 9,
End

Simplified Problem Statement: Rearrange the given Sorted Acending array such that each unique element occur only once. Keep the relative order of elements and size of input array same and also return the number of unique elements in this Array.

Given : Pointer to the first element of the Array (nums) and Size of this Array nums.

KeyPoints: 1) This array is sorted in non-decreasing order.
2) The **relative order** of the elements should be kept the **same**.
3)return *the number of unique elements in* nums.
4)The remaining elements of nums are not important as well as the size of nums.

My algorithm

Step 1:

Initialization:
1)First create another array of same size let's call it expectedArray in heap.

2)Take the first element of the input array and copy it in the expectedArray because it's a sorted array and the final array should have the element in the same order as input. This element is lowest in it's array and will always be first in the expected array as well.

3) Create a marker which is pointing to the first index of input array (numsMarker).

4)Create another marker which is pointing to the first index of the expected array (expectedArrMarker.)

Checking for unique elements in the given array:

5) Keep checking till the input array marker is at the second last element of the array (2nd last because to avoid index out of range in input array).

6) if the current index and next index have same elements increment the input array marker.

7) if the current index and next index have different elements then increment the expected array marker and copy the next index element of the input array into the expected array at it's new current index. After this increment the input array marker.

Step 2:

Returning the number of unique elements and rearranging the given array to match the expected array

8) Run a loop from 0 to expectedArrayMarker(number of unique elements) times. inside this loop copy elements of expectedArray[Current Loop counter] into nums[Current Loop counter]. This makes sures the relative order is inside the input array is matched to the expectedArray. Also during step we found our number of unique elements.

9) free the memeory in heap since we no longer require the temporary array and return the number of unique elements which will expectedArrayMarker + 1. Since this marker starts from 0. Also Create a seperate variable for final return value to avoid confusion.