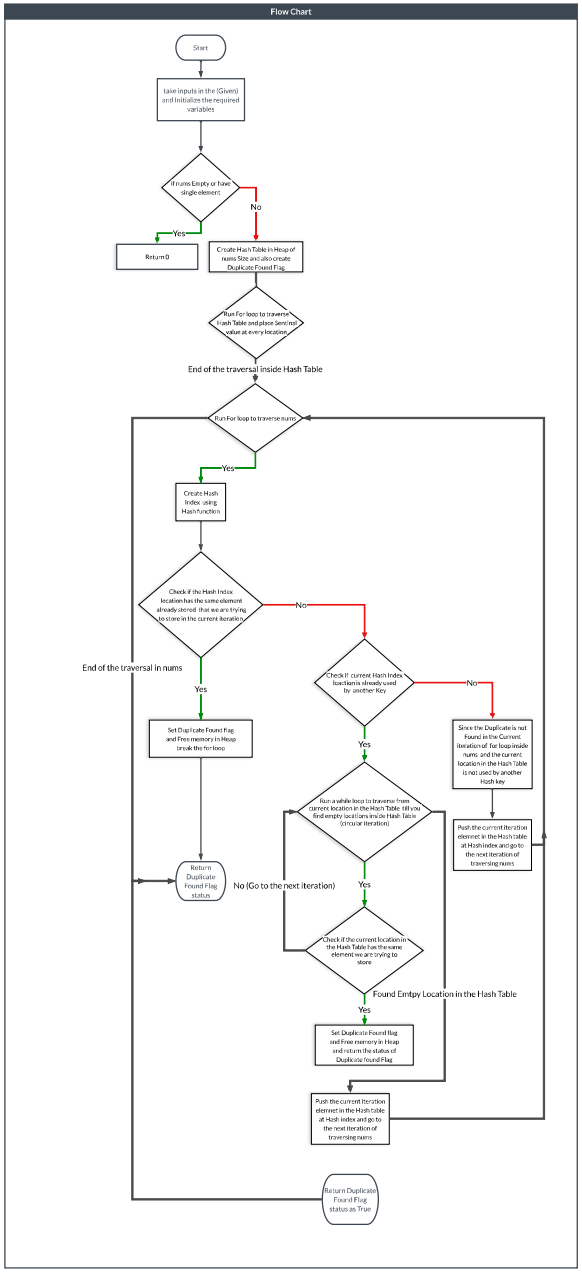
Contains Duplicate

https://leetcode.com/problems/contains-duplicate/

Flow Chart



Simplified Problem Statement: Given an integer array nums, return true if any value appears **at least twice** in the array, and return false if every element is distinct.

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

Key Points: Same as problem statement

My algorithm: Using Hash Table

1) If Array size is 0 or 1 return 0 No duplicates in Empty or single element array.

2) We create a Flag which indicates if Duplicate is Found or not and also an array in the heap if the input array contains more than one element. Also the size of the array in the heap is same as input array, we call it our Hash Table and the pointer to the first element hashPtr

3) We traverse our newly created Hash Table which is empty acessing each location and putting a sentinal value (INT_MIN) inside it. Sentinals are used to indicate whether the location is used by element from input array or not.

4) We traverse our input array and pass each element as key to the Hash Function which gives us the Hash Index inside the Hash Table and by adding it with numsSize and taking modulo with the same again we avoid getting a negative Hash Index inside Hash Table.

int HashIndex = (((nums[i] % numsSize) + numsSize)% numsSize);

4a) Once we go the Hash index if the current location in the table has the same element already stored that we are trying to store we set the **DuplicateFound** flag , free the heap memory, stop traversing and return the status of **DuplicateFound** flag.

4b) if 4a) is false then we check if the current location in hast table is filled by other element or not. If the current location in the hash table has a value other than a sentinal value it means this location is used by other element from the input array and hence storing our current input array element here will result in **collision.**
To avoid this we run a while loop from our current position in the Hash Table if the location is non empty **(other than sentinal value)** we keep checking next location in the Hash Table till the current location is non empty.

While this traversing we also check if the current non empty location has the same element that we are trying to store if yes then we set the **DuplicateFound** flag, free the heap memory and return the status of **DuplicateFound** flag. Otherwise if we found the empty location in the Hash Table while loop stops and then we store our current input array element there.

4c) if 4a) and 4b) is false then it indicates that the current location in the Hash Table is empty and we simply store our current input array element in that location inside the Hash Table.

5) Once done traversing the input array we return the status of **DuplicateFound** flag.