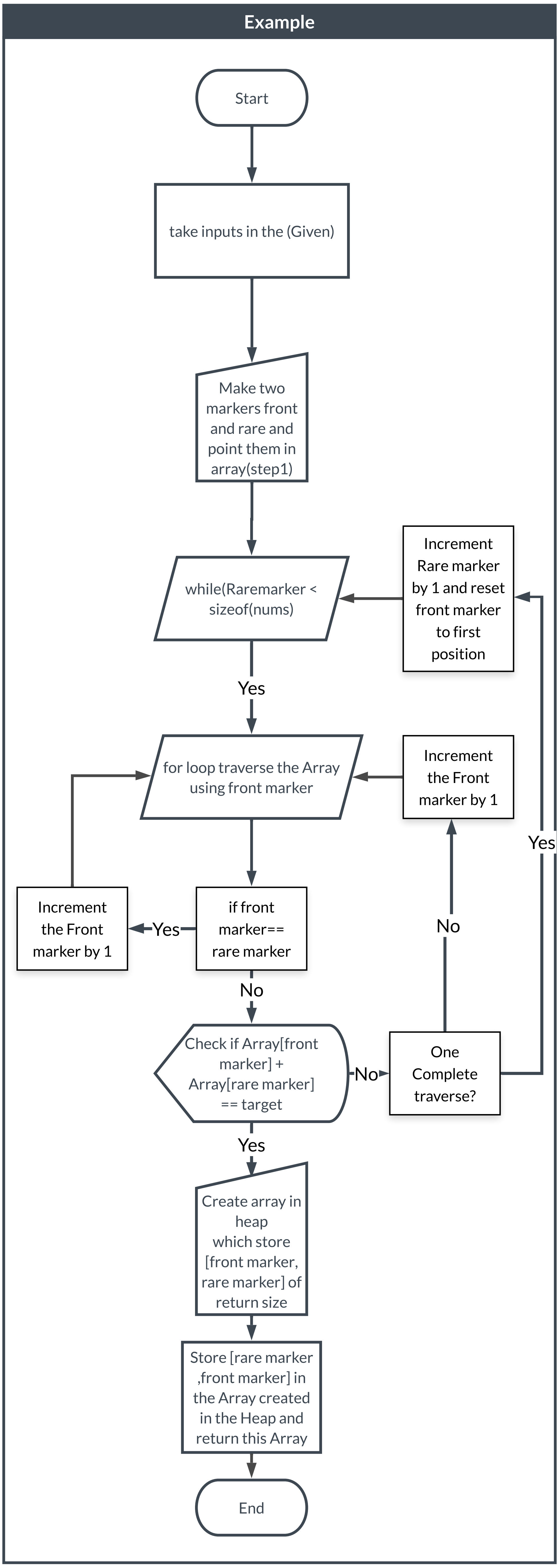
TWO SUMS

Two Sums Leet Code: https://leetcode.com/problems/two-sum/description/?envType=problem-list-v2&envId=array

Example



Start
take inputs in the (Given)
Make two markers front and rare and point them in array(step1)
while(Raremarker < sizeof(nums)
Yes
for loop traverse the Array using front marker
if front marker== rare marker
Yes
Increment the Front marker by 1
No
Check if Array[front marker] + Array[rare marker] == target
No
One Complete traverse?
No
Increment the Front marker by 1
Yes
Increment Rare marker by 1 and reset front marker to first position
Yes
Create array in heap which store [front marker, rare marker] of return size
Store [rare marker ,front marker] in the Array created in the Heap and return this Array
End

My algorithm

Simplified Problem Statement:

Given : pointer to the int Array (nums) ,integers (target) ,integer size of int array(nums) and int pointer returnsize;

if(Array[i] + Array[j] == target)

return [Array[i] and Array[j]]

KeyPoints: Each Input has One solution and not to use the same element twice for example: arr[i] + arr[i] = target is not accepted.

My model : 1) Create two Markers first points to the first element(rare marker) second also points to the first element (Front marker).

2) keep checking till the rare marker becomes greater than the size of the nums Array using while loop (outer loop)

3) During Each iteration of the while loop there should be a nested for loop where we iterate the Front marker from the first element till it's on the last element of the array by one step at a time. after each complete traversal of Front marker we reset it to 0 (first element) and increment the Rare marker by 1.

4) Inside Each iteration of the For Loop we only check for the target when Rare and Front marker are not equal to avoid repeating the same element. if they are equal the iteration will be skip automatically.

5) If the Front and the Rare marker are not equal then we simple check if Array[Front marker] + Array[Rare marker] == target. if yes then we create the Array of returnSize in heap and store [Rare marker, Front marker]. else we increment the Front marker.