Question: write a function that takes in an array of non-empty distinct integers and a int → targetSum

if any two numbers add up to targetSum, return them in array, if none, return empty array

Can't add number to itself, at most one pair.

e.g.

$[3, 5, -4, 8, 11, 1, -1, 6]$, $t = 10$

↑

find x and y, such that $x + y = t$

Brute force approach? → not a good approach

two for loops → $O(n^2)$

#1: use hash table

seenNums = { } → initialize empty hash table

$x + y = 10$

$y = 10 - x$ → currentNum ← in traversing

traverse the array, check if $10 - x$, which is number need to make the two sum pair, is in hash map, if yes return y and x, if not add it to hash map.

traverse the array (for loop):

X = CurrentNum = array[i]

if 10-array[i] is in seen_hash:

return [10-array[i], array[i]]

else:

seen_hash[10-array[i]] = true

return []

Example run

[3, 5, -4, 8, 11, 1, -1, 6], t = 10

↑

→ CurrentNum = 3

does y = 10-3 = 7 exist in array? no

add to hash table

→ CurrentNum = 5

does y = 10-5 = 5 exist in array? no

add to hash

→ CurrentNum = -4

y = 10-(-4) = 10+4 = 14? no

add

→ Curr = 8

y = 10-8 = 2? no

→ Curr = 11

y = 10-11 = -1? no

has_seen: {
3: true
5: true
-4: true
8: true
11: true
1: true
}

→ Cum = 1

y = 10 - 1 = 9 ? no

→ CurrentNum = -1

y = 10 - (-1) = 11 ? YES break out of loop

return [11, -1]

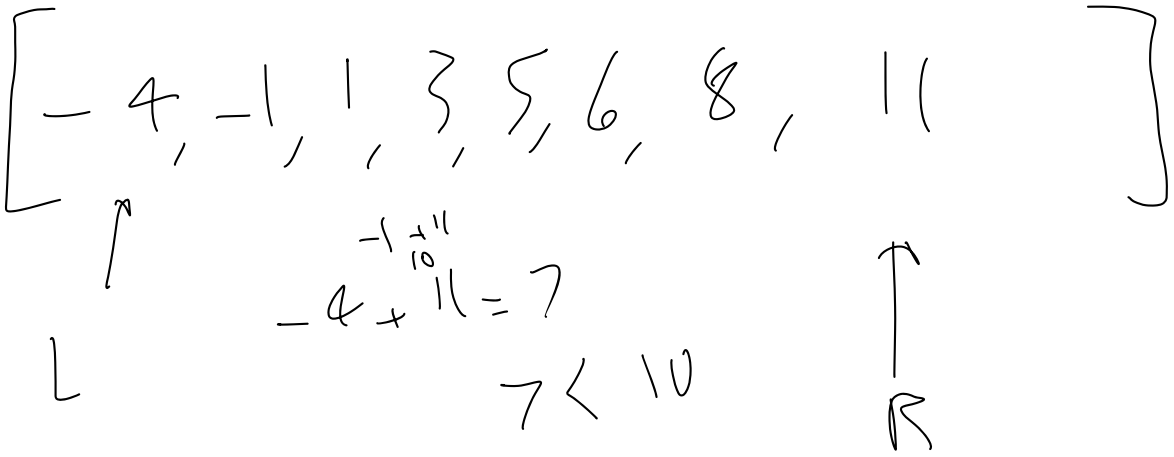
time Complexity → O(N) → array traverse

Space Complexity → O(N) → hash map

2nd Solution

- Sorting the array → $O(N \log(N))$

target = 10



L + R = -4 + 11 = 7 7 == 10? no

if > < 10 :

L += 1 → move to the right

if > > 0 :

R -= 1 → move to the left

→ -1 + 11 = 10 ✓

Time Complexity → $O(N \log(N))$

Space Complexity → $O(1)$ → if sorting can be done in place.