Two Sum :-

2	6	5	8	11	7
0	1	2	3	4	5

target = 14

There can be two varities :-

1) If there exist x and y where x+y=14 where x and y are available at different indices. So, return true or false.
2) Assume, there always exist such x & y where x+y=14. So, return the indices of x and y.

• Brute Force :-

Iterate over the array. And for each element i check if there exist j where arr[i] + arr[j] = target.

$T = O(n^2)$; $S = O(1)$

1st Best Approach :-

i	i	i	i		
2	6	5	8	11	7
0	1	2	3	4	5

Hash Map

(2, 0)
(6, 1)
(5, 2)

target = 14

$2 + x = 14$
$\Rightarrow x = 14 - 2 = 12$
$\Rightarrow x = 12$ (Search for 12 in Map if not found then put x in the map)

$x = 14 - 6 = 8$

$x = 14 - 5 = 9$

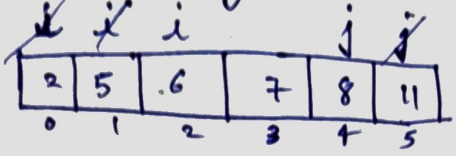
$x = 14 - 8 = 6$ $\Rightarrow$ 6 is found in the Map

Here, hasMap is used to keep track of the elements which are already visited.

$T = O(n) * O(1) = O(n)$
$S = O(n)$

2nd Best Approach (Two pointer)

⇒ Sort the array



target = 14

$2+11 = 13 < 14$ (i++)

$5+11 = 16 > 14$ (j--)

$5+8 = 13 < 14$ (i++)

$6+8 = 14 == 14$

$T = O(n \log n) + O(n) = O(n \log n)$

$S = O(1)$

In this approach we are changing the array. So, we can say true or false but we can't find the indices.

Note :—

There can be a third variety of this problem. We have to find all unique pairs and also in a pair same element can't be taken twice.

We can use 1st and 2nd best approach (Both) to find all unique pairs.

↳ We just have to make slight modification

- In 1st Best Approach, we will take a Hashset

Here, we won't stop the loop as soon as we get a valid pair. We will continue the loop and keepon add elements in hashMap.

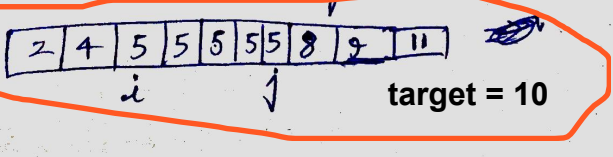
During the looping, as soon as we find a valid pair, we will ~~keep the pair in list~~ sort the elements of the pair and store it in hashset.

HashSet will itself Make sure that the pairs are unique. Because it stores only unique elements.

- In 2nd Best Approach, we will use the same logic as previous. Additionally, we have to do i++ and j-- when we find a pair and keep on adding the pairs to a list.

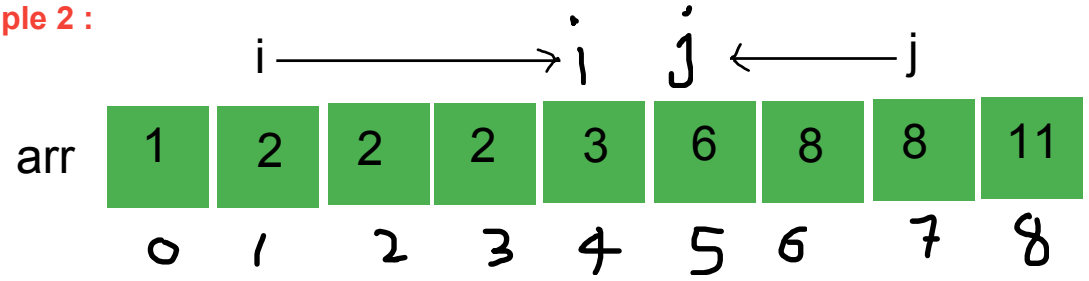
There is an edge case here, when some of the elements are repeated :—

Example 1 :



In 2nd best solution, we have one catch. If some of the elements of the array are repeated then:-

Example 2 :



target = 10

Here, (2,8) forms a pair so, i++, j - -

Now, since (arr[i-1] == arr[i]). So, do i++ till we reach a new value.

Since (arr[j] == arr[j+1]). So, do j -- till we reach a new value.

Do check code for better understanding

Note:-

1st Best Solution - Using HashMap & HashSet

2nd Best Solution - Using two pointer solution

Even though, I am saying 1st best solution and 2nd best solution. But actually, 2nd best solution (Two Pointer Solution is the most optimal algorithm)

Because HashMap and HashSet gives amortized and average time performance of O(1), not worst case. This means, we can suffer an O(n) operation from time to time.