Two-sum

Saturday, August 7, 2021 9:47 PM

-1	6	2	-4	3	7	9	11	5
0	1	2	3	4	5	6	7	8

$\boxed{target = 11}$

$(2, 9) \leftarrow$

$\boxed{x + y = 11}$

1.) **Brute force**: Run a for loop from i=0 and go to each elements one by one

i.e, $x = arr[i]$, $y = arr[j]$ {where j=i+1}

$\Rightarrow$ two for loops of n >

$\Rightarrow O(n^2)$ $O(1)$ $arr[i] + arr[j] = 11$

TC SC

2.) Unordered map/set : We can store the elements of the array in a map and corresponding to each element, we store its indices.

e.g.

-1	0
6	1
2	2
⋮	⋮

we will run a for loop from i=0 which will represent x

Now, $y = 11 - x$.

So, we will find if y is present in the map or not.

Return the indices

$\Rightarrow O(n + n) = O(n) + O(1) = O(n)$ $O(n)$

traversing

to insert elements in the map

searching in map

TC SC

3.) Sorting : a) We will pair the elements with their indices before sorting them.

pair<int, int>   (-1, 0), (6, 1), ...

b) Form a vector of this pair

c) Sort the vector according to the first value

s → . . .   . . . ← e

-4   -1   2   3   5   6   7   9   11

3   0   2   4   8   1   5   6   7

d) Two pointers, s = 0, e = n-1

e) We add s & e and check with 11

if < 11, then s++

if > 11, then e--

f) when s+e = 11, return indices.

⇒ O(n log n)   O(n)

TC   SC

For unsorted array, best approach is (2) and for sorted array (3)

Best time to buy and sell stock

Monday, August 16, 2021 11:21 AM

array of prices → max. profit

$$\begin{array}{cccccccl} [7, & 1, & 5, & 3, & 6, & 4] & \leftarrow & \text{Prices} \\ 1 & 2 & 3 & 4 & 5 & 6 & \leftarrow & \text{days} \end{array}$$

i) buy at i and check at [i+1, ... n]

$O(n^2)$

ii) find minimum, and profit

- iterate over all the elements while checking.

$O(n) \leftarrow \boxed{\text{Best way}}$