[3, 5, -2, -6, 10, -1] → -8

min sub array sum = -8.

[-3, -5, 2, 6, -10, 1] → 8

[3, 5, -2, -6, 10, -1]

3 5 -2 $\boxed{-8}$ 2 -1

min_sum[i] = min (arr[i], min_sum[i-1] + arr[i])

TC: O(n).
AS: O(n) → O(1).

[-2, -3, 1, -100]
[2, 3, -1, 100]
6.100

[-2, -3, 1, -100]

(1, 2)

-2 -5 -7 -104

(j+1 → i-1)

[1, 2, 4, 5, 9], tar=8
↑
False.

[1, 2, 4, 3, 5, 9], tar=8.
True.

i: 0 → n-1
j: i+1 → n-1
if $a_i + a_j$ == tar
return true.

TC: O(n²)
AS: O(1)

i: 0 → n-1
Binary Search for
(tar - a_i) in [i+1 -- n-1]

TC: O(n log n)
AS: O(1)

→ i

[1, 3, 5, 9, 13, 15], tar = 12

Hashing Based: TC: O(n)
AS: O(n).

visited
1 → true
3 → true.
5

Two-Pointer

[1, 3, 5, 9, 13, 15], tar = 12

TC: O(n)
AS: O(1).

Triplet with target

i: 0 → n-1
j: i+1 → n-1
k: j+1 → n-1
...

TC: O(n³)
AS: O(1)

i: 0 → n-1
Hashmap-based app. to find pair with sum: tar - a_i in arr[i+1 --- n-1].

TC: O(n²)
AS: O(n).

if the array was sorted:
i: 0 → n-1
Two-pointer app. to find pair with sum: tar - a_i

TC: O(n log n + n²) = O(n²).
AS: O(1)

	BF	Two-ptr
2-sum:	O(n²)	O(n)
3-sum:	O(n³)	O(n²)
4-sum:	O(n⁴)	O(n³)
k-sum:	O(nᵏ)	O(nᵏ⁻¹)

i: _
j: _

(TP.)

tar - a_i - a_j

(arr, k, t).
a_i
→ (arr, k-1, t - a_i)
a_j
→ (arr, k-2, t - a_i - a_j)

[1, 1, 1, 2, 3, 3, 8, 9, 9, 10]

Hashmap Based: TC: O(n)
AS: O(n).

[1, 2, 3, 8, 9, 10]

[1, 1, 1, 2, 3, 3, 8, 9, 9, 10]

n = 100

i: 0 → n-1
j: (i) → n-1

i=0 : j = 1 → 99
i=1 : j = 2 → 98
i=2 : j = 3 → 97

i: 0 → n-1
j: 0 → n-1

[1, 1, 1, 2, 3, 3, 8, 9, 9, 10]
[1, 2, 3, 8, 9, 10, x, x, x, x]

return = 6