Rule → SC = O(1)

Q→ Given a **sorted integer array** & an integer K. Find any pair (i, j) s.t $\boxed{A[i] + A[j] = K \;\&\; i \;!= j}$

```
      0   1   2   3   4
A = [ -5  -2   1   8  10 ]
```

K = 6 A[1] + A[3] Ans = true

```
      0   1   2   3   4   5   6   7   8   9
A = [ -3   0   1   3   6   8  11  14  18  25 ]
```

K = 12 Ans = true

Bruteforce → ∀ i, j check $\boxed{A[i] + A[j] = K \;\&\; i \;!= j}$

TC = O(N²)

SC = O(1)

Sol 2 → $\boxed{A[i] + A[j] = K \;\&\; i \;!= j}$ → $x + y = y + x$

$A[i] + A[j] = A[j] + A[i]$ } $\boxed{i < j}$ OR $i > j$

$A[i] + A[j] = K$ → $\boxed{A[j] = K - A[i]}$

∀ i, search if (K − A[i]) is present from (i+1) to (N−1).

use binary search TC = O(N Log(N)) SC = O(1)

Sol 3 →

```
      i   i   i                   j   j   j   j
      0   1   2   3   4   5   6   7   8   9
A = [ -3   0   1   3   6   8  11  14  18  25 ]
```

K = 12

A[i] + A[j] = K

-3 + 25 > 12

-3 + 18 > 12

-3 + 14 < 12

0 + 14 > 12

0 + 11 < 12

1 + 11 = 12 ⇒ Ans = true

A[9] + smallest element > K

⇒ A[9] + any element > K

```
i = 0        j = N-1
while (i < j) {
    if (A[i] + A[j] == K)   return true
    if (A[i] + A[j] < K)    i++
    else                    j--
}
return false
```

TC = O(N) SC = O(1)

Q → Given a sorted integer array with unique elements, find the count of pair (i, j) s.t A[i] + A[j] = K & i != j

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

K = 10

Ans = 2

```
i = 0        j = N-1        cnt = 0
while (i < j) {
    if (A[i] + A[j] == K) {
        cnt++
        i++        j--
```

```
    } else if (A[i] + A[j] < K)   i++
      else                        j--
}

return cnt                TC = O(N)      SC = O(1)
```

If array has duplicates ?

```
          0  1  2  3  4  5  6
A = [ 1  2  2  2  5  8  8 ]        K = 10
      i  i  i  i  i  j  j          Ans = 6
                  j

ans = 0          cnt1 = 0 + 3
i = 0   j = N-1  cnt2 = 0 + 2
while (i < j) {
    x = A[i] // 2   y = A[j] // 8
    if (x == y) {
        if (x+y == K) {
            cnt = j - i + 1
            ans += cnt * (cnt-1) / 2
        }
        break
    }
    if (x + y == K) {
        cnt1 = 0    cnt2 = 0
        while (A[i] == x) { cnt1++   i++ }
        while (A[j] == y) { cnt2++   j-- }
        ans += cnt1 * cnt2   // 3*2 = 6
    } else if (x + y < K) {
        while (A[i] == x) { i++ }
    } else {
```

```
        while (A[j] == y) { j-- }
    }
}
```

TC = O(N) SC = O(1)

H.W → Form a combo of 2 food items s.t (sum of price <= 99). I/p → price ∀ food item.

Q → Given a sorted integer array & an integer K. Find any pair (i, j) s.t $\boxed{A[j] - A[i] = K \quad \& \quad j > i}$

0	1	2	3	4	5
-5	-2	1	8	10	12

$\boxed{K > 0}$

K = 12

A[4] - A[1] = 10 - (-2) = 12 Ans = true

Bruteforce → TC = O(N²) SC = O(1)

Binary Search → A[j] - A[i] = K

⇒ A[j] = K + A[i]

∀ i, search K + A[i] from (i+1) to (N-1).

TC = O(N log(N)) SC = O(1)

0	1	2	3	4	5
1	(2)	4	5	6	(12)

K = 10 Ans = true

Sol 3 →

0	1	2	3	4	5
-5	-2	1	8	10	12
i	j	~~j~~	~~j~~	~~j~~	~~j~~
	i				

K = 12

A[j] - A[i] = K

-2 - (-5) = 3 < 12 → j++

1 - (-5) = 6 < 12

8 - (-5) = 13 > 12

8 - (-2) = 10 < 12

10 - (-2) = 12 = K Ans = true

A[j] - smallest < K
⇒ A[j] - any < K

```
i = 0   j = 1
while (j < N) {
    if (A[j] - A[i] == K) return true
    if (A[j] - A[i] < K)  j++
    else  i++
}
return false
```

TC = O(N) SC = O(1)

H.W → Find count of pairs A[j] - A[i] = K, j > i

Q → Given an integer array with +ve elements & an integer K, check if there exist a subarray with sum K.

	0	1	2	3	4	5	6	
A = [	1	3	15	10	20	3	23	] K = 33

Ans = true

Bruteforce → Y subarrays, calculate sum & check if subarray sum = K.

N*(N+1)/2

TC = O(N² * N) → c.f. O(N³) SC = O(1)

```
        0  1  2  3  4
A = [ 1  2  5  4  3 ]    K = 9
```

Ans = true

subarray sum → prefix sum

$P[i] = P[i-1] + A[i]$

$sum(i \_\_ j) = P[j] - P[i-1]$

HW → Try solving via prefix sum.

Sol →

```
          i  i  i  i
          0  1  2  3  4  5  6
A = [ 1  3  15 10 20 3  23 ]    K = 33
      i  i ------------> i  i
```

subarray → (i __ j)

∵ ∀i, A[i] > 0

⇒ sum ∝ size

subarray sum		K
1	<	33
4	<	33
19	<	33
29	<	33
49	>	33
48	>	33
45	>	33
30	<	33
33	=	K

Ans = true

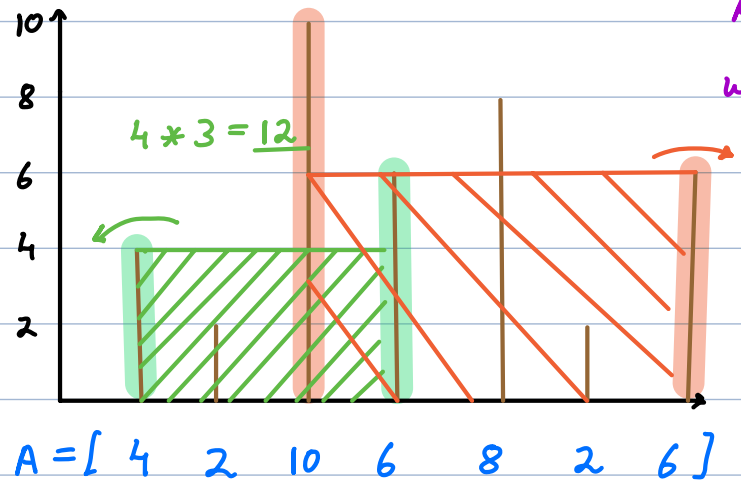
```
i = 0   j = 0
sum = A[0]
while (j < N) {
    if (sum == K)
        return true
    if (sum < K) {
        j++   sum += A[j]
    } else {
        sum -= A[i]   i++
    }
} return false
```

TC = O(N) SC = O(1)

A → Given an integer array,

A[i] → Height of i^{th} wall.

Find 2 walls that can form a container to store max water.



A = [4 2 10 6 8 2 6]

Area of water width = 1 b/w consecutive walls

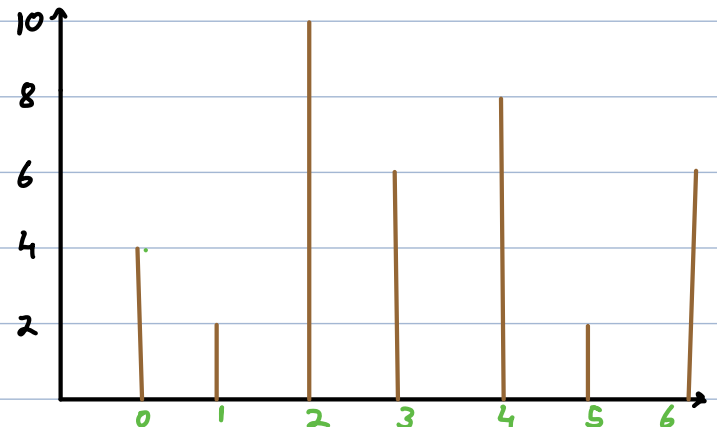
6 * 4 = 24 (Ans)

$$\text{Area} = \min(A[i], A[j]) * (j - i)$$

Bruteforce → ∀ i, j

calculate area & take max.

TC = O(N^2) SC = O(1)



A = [4 2 10 6 8 2 6]

i i i j j j j

ans = 24

Area = H * W

4 * 6 = 24

2 * 5 = 10

6 * 4 = 24

2 * 3 = 6

8 * 2 = 16

6 * 1 = 6

small height * max width → x

h * any width <= x

```
i = 0          j = N-1          ans = 0
while (i < j) {
    area = min(A[i], A[j]) * (j - i)
    ans = max(ans, area)
    if (A[i] < A[j])  i++
    else if (A[j] < A[i])  j--
    else { i++  j-- }
} return ans
```

TC = O(N) SC = O(1)
