Two Pointers

TABLE OF CONTENTS

Notes

< Question > : Given an integer sorted array A and an integer K. Find any pair[i , j] such that A[i] + A[j] = K, $\boxed{i \mathrel{!}= j}$

Example : A[ ] → [-5 -2 1 8 10 12 15], K = 11

0 1 2 3 4 5 6

A[2] + A[4] = 11 Ans = true

Quiz : [-3 0 1 3 6 8 11 14 18 25], K = 12

0 1 2 3 4 5 6 7 8 9

A [2] + A [6] = 12 Ans = true

Bruteforce → ∀ i, j check if A[i] + A[j] = K & i != j. TC = O(N²) SC = O(1)

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

∀ i, search if $\boxed{K - A[i]}$ is present in array s.t i != j. TC = O(N log(N)) SC = O(1)

Hashset

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

Two - pointer approach

[-3 0 1 3 6 8 11 14 18 25], K = 12

0 1 2 3 4 5 6 7 8 9

1. i = 0, j = 1 ✓
2. i = N-1, j = N-2

} same

3. i = 0, j = N-1

i j

[-3 0 1 3 6 8 11 14 18 25],

0 1 2 3 4 5 6 7 8 9

$A[i] + A[j] = K$

$-3 + 0 = -3 < 12$

⇒ increase the sum

⇒ increase i or j

(no defined update) ⇒ X

i j

[-3 0 1 3 6 8 11 14 18 25], K = 12

0 1 2 3 4 5 6 7 8 9

$A[i] + A[j] = K$

$-3 + 25 = 22 > 12$ ⇒ decrease j

$A[N-1] + A[0] > K$

⇒ $\underline{A[N-1]}$ + any element > K

can never be the ans.

$i \rightarrow$ $\qquad\qquad\qquad\qquad \leftarrow j$

[-3 0 1 3 6 8 11 14 18 25], K = 12

0 1 2 3 4 5 6 7 8 9

$A[i] + A[j] = K$

$-3 + 25 = 22 > 12$, $j--$

$-3 + 18 = 15 > 12$, $j--$

$-3 + 14 = 11 < 12$, $i++$

$0 + 14 = 14 > 12$, $j--$

$0 + 11 = 11 < 12$, $i++$

$1 + 11 = 12 = K$ Ans = <u>true</u>

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

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

< Question > : Find count of all the pairs in a sorted array whose sum is K. (i != j)

arr[ ] → [1 3 4 5 6 7 10], K = 10

0 1 2 3 4 5 6

Ans = 2

arr[ ] → [1 3 4 4 5 6 7 10] K = 10

0 1 2 3 4 5 6 7

Ans = 3

t = 2 + 0 + 1 = 3

cnt x = ~~0~~ + 2

cnt y = 0

x = y = 6

K = 12

arr[ ] → [1 3 4 4 5 6 6 6 7 10] K = 10

0 1 2 3 4 $\boxed{5 \ 6 \ 7}$ 8 9

$^3C_2 = \frac{3 * 2}{2} = 3$

A[i] + A[j] = K

1 + 10 = 11 > 10 , j--

1 + 7 = 8 < 10 , i++

3 + 7 = 10 = K → cnt3 & cnt7

#pairs = cnt3 * cnt7

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

```
    // i++  OR  j--  OR  both → distinct element
    } if ( A[i] + A[j] > K )   j--
      else   i++
  }
  return cnt                    TC = O(N)    SC = O(1)
```

```
i = 0    j = N-1    ans = 0
while ( i < j ) {
    if ( A[i] + A[j] == K ) {
        x = A[i]      cntx = 0
      → while ( i < j  &&  A[i] == x ) { i++    cntx++ }
        y = A[j]    cnty = 0
      → while ( i < j  &&  A[j] == y { j--  cnty++ }
        if (x != y)    ans = cntx * cnty
        else  {   t = cntx + cnty + 1
                 //    5   5   5   5          K = 10
                       i   i   i   j i
                       cntx = 0 1 2 3    cnty = 0

            ans += t*(t-1)/2      // tC2
        }
    } if ( A[i] + A[j] > K )   j--
      else i++                  TC = O(N)     SC = O(1)
} return ans
```

< Question > : Given an integer sorted array A and an integer K. Find any pair[i , j] such that A[i] - A[j] = K, i != j

arr[ ] → [-5 -2 1 8 10 12 15] K = 11

0 1 2 3 4 5 6

Ans = true

Bruteforce → TC = O(N²) SC = O(1)

Binary Search → TC = O(N log (N)) SC = O(1)

Hash Set → TC = O(N) SC = O(N)

i					j
0	1	2	3	4	5

A → [1 2 4 5 6 12] K = 10

↑A[j] - A[i]↓ = K

12 - 1 = 11 > 10 ⇒ decrease ⇒ j-- OR i++

~~i~~	i ~~j~~	~~j~~	~~j~~	~~j~~	j
0	1	2	3	4	5

A → [1 2 4 5 6 12] K = 10

A[j] - A[i] = K

2 - 1 = 1 < 10 ⇒ increase ⇒ j++

4 - 1 = 3 < 10 ⇒ j++

5 - 1 = 4 < 10 ⇒ j++

$6 - 1 = 5 < 10 \Rightarrow j{+}{+}$

$12 - 1 = 11 > 10 \Rightarrow i{+}{+}$

$12 - 2 = 10 = K$ Ans = true

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

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

H.W → count of pairs s.t $A[j] - A[i] = K$, $i \,!= j$.

< Question > : Given an arr[N] of positive integers and an integer K. Check if there exists a subarray with sum = K.

$\underline{K > 0}$

arr[ ] → [1 2 5 4 3], K = 9

Ans = $\underline{\text{true}}$

$\underline{\text{Bruteforce}}$ → ∀ subarray, check if sum = K.

TC = $\underline{O(N^3)}$ ⟶ $\underline{O(N^2)}$ (carry forward)

SC = $\underline{O(1)}$

$$\begin{array}{cccccccccc} & \cancel{i} & \cancel{i} & \cancel{i} & i & & & & & \\ & \cancel{j} & \cancel{j} & \cancel{j} & \cancel{j} & \cancel{j} & j & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ A = [& 1 & 3 & 15 & \underline{10} & \underline{20} & \underline{3} & 23 & 33 & 43\ ] \end{array} \quad K = \underline{33}$$

$\boxed{i <= j}$

sum i — j = K

$\boxed{\begin{array}{l} A[i] \text{ positive} \Rightarrow x + A[i] > x \\ \qquad\qquad\qquad\quad x - A[i] < x \end{array}}$

1 < 33, increase ⇒ j++

4 < 33 ⇒ j++

19 < 33 ⇒ j++

29 < 33 ⇒ j++

49 > 33 ⇒ i++

48 > 33 ⇒ i++

45 > 33 ⇒ i++

30 < 33 ⇒ j++

33 = K ⇒ Ans = $\underline{\text{true}}$

sum for empty array = 0

i = 0 j = 0 sum = A[0]

(sum > K) ⇒ sum -= A[0] // 0

i++ // 1

i = 0 j = 0 sum = A[0]

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

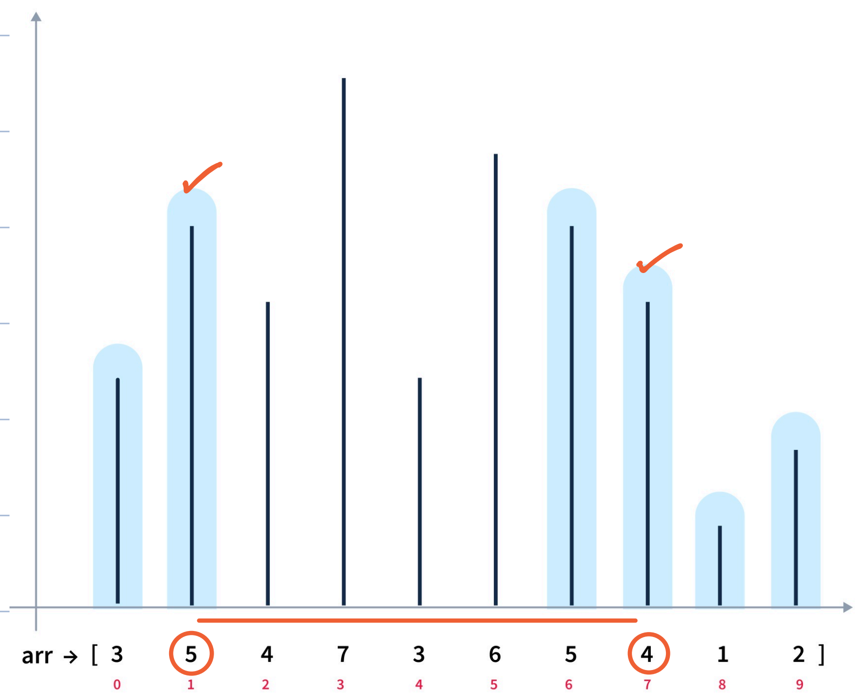
TC = O(N) SC = O(1)

Container with most water

< Question > : Given an arr[N] where every element represents height of walls. You need to pick any two walls such that water accumulated is maximum.

Area = height * width

$\boxed{min(A[i], A[j]) * (j - i)}$



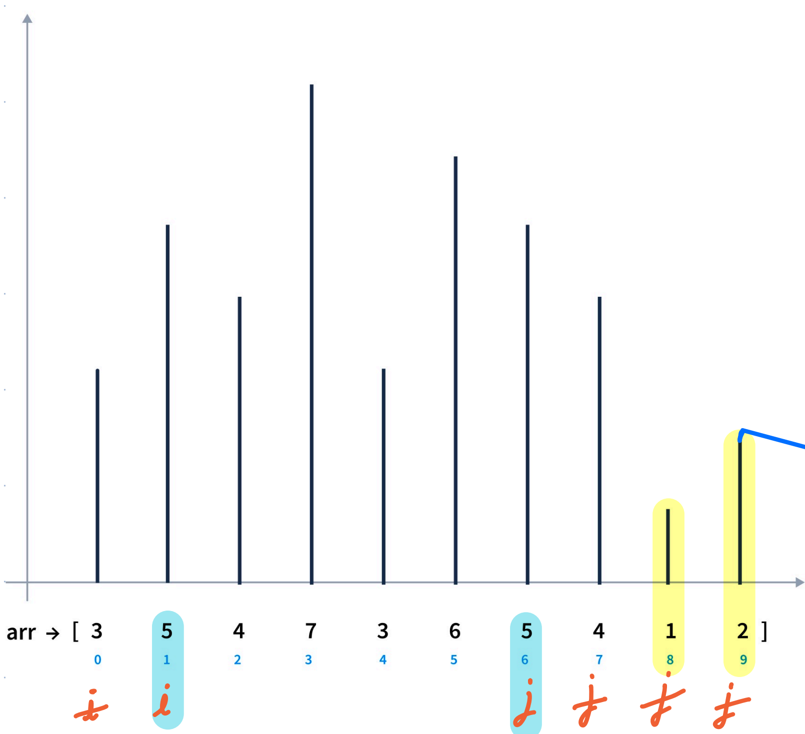
w = 6 h = 4

Area = 24

Bruteforce → ∀ i, j calculate area & take max.

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

Two Pointers



Area

$\boxed{\min(A[i], A[j]) * (j - i)}$

$\min(3, 2) * (9 - 0) = 18$

2 9

pair with any wall

max h = 2

max w = 9

$\min(\infty, 2) = 2$

i	j	Area $\boxed{\min(A[i], A[j]) * (j - i)}$	Ans
0	9	$\min(3, 2) * (9 - 0) = 18$	18
0	8	$\min(3, 1) * (8 - 0) = 8$	
0	7	$\min(3, 4) * (7 - 0) = 21$	21
1	7	$\min(5, 4) * (7 - 1) = 24$	24
1	6	$\min(5, 5) * (6 - 1) = 25$	25
2	5	...	

```
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[i] > A[j])  j--
    else { i++   j-- }
}
return ans
```

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