# Two Pointer

## # Today's Content

- → Pairs with given Sum - 2
- → Pairs with given difference
- → Subarray with given Sum
- → Container with most water

**Quote :-** The more you sweat in peace, the less you bleed on war

**Question :-** Given an integer sorted array A & an integer K, find any pair (i, j) such that

$A[i] + A[j] = K$ , $i \neq j$ SC → O(1)

**Ex :-** $A = [-5, -2, 1, 8, 10, 12, 15]$ (indices 0, 1, 2, 3, 4, 5, 6)

$K = 11$

**Quiz :-** Check if there exists a pair with sum K

$A[] = \{-3, 0, 1, 3, 6, 8, 11, 14, 18, 25\}$ | $K = 12$

↳ yes

↳ (1, 11)

Brute force

↳ check all pairs using nested loop.

| | |
|---|---|
| TC | $O(N^2)$ |
| SC | $O(1)$ |

Binary Search (BS)

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

$$\Rightarrow A[j] = K - A[i]$$

⇒ For each $i$, apply BS over remaining part $[i+1, N]$

| | |
|---|---|
| TC | $O(N \log N)$ |
| SC | $O(1)$ |

↳ OPTIMIZATION [ Two pointers ]

Ex:- $A = [\underset{0}{-5}, \underset{1}{-2}, \underset{2}{1}, \underset{3}{8}, \underset{4}{10}, \underset{5}{12}, \underset{6}{15}]$ | $k = 11$

⇒ whenever we decide to solve question using 2 pointer. Then we need ans these question.

① From where the pointer will start?

② How the pointer will move?

Ex:- $A = [\overset{0}{-5}, \overset{1}{-2}, \overset{2}{1}, \overset{3}{8}, \overset{4}{10}, \overset{5}{12}, \overset{6}{15}]$ | $k = 11$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i \quad\; j$

Q From where I should start my pointers?

**Case I** $\quad i = 0 \;,\; j = 1$

Now, $\quad A[i] + A[j]$

$\Rightarrow -5 \quad -2$

$\Rightarrow -7 \quad < 11$

↳ which one to Move

Can't decide { $\Rightarrow j{+}{+} \rightarrow$ Sum will increase
but $i{+}{+} \rightarrow$ Sum will increase

**Case II** $\quad i = N-2 \;,\; j = N-1$

$\Rightarrow 12 \quad +15$

$\Rightarrow 27 \quad > 11$

↳ For $i{-}{-}$ & $j{-}{-}$ the Sum is decreasing so we can't decide

**Case III** :- $\quad i = 0 \;,\; j = N-1$

Ex:- $A = [\overset{0}{-5}, \overset{1}{-2}, \overset{2}{1}, \overset{3}{8}, \overset{4}{10}, \overset{5}{12}, \overset{6}{15}]$ | $K = 11$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i \quad\; j$

$\Rightarrow -5 + 15$

$\Rightarrow 10 < 11$

## DRY RUN

| A[i] | A[j] | Sum | |
|---|---|---|---|
| -5 | 15 | 10 < 11 | This convey-5 will largest is smaller so can't be paired with any other to form 11. Hence i++ |
| -2 | 15 | 13 > 11 | Because if 'i' increased than sum will increase so j-- Hence |
| -2 | 12 | 10 < 11 | |
| 1 | 12 | 13 > 11 | |
| 1 | 10 | 11 == 11 | |

Conclusion :- (1) when need to increase sum, do i++;

(2) when need to decrease sum, then do j--;

## PSEUDO CODE

```
i = 0, j = N-1

while ( i < j ){
    if ( A[i] + A[j] == k ){
        return True;
    } else if ( A[i] + A[j] > k ){
        j--;
    } else {
        i++;
    }
}

return False;
```

TC → O(N)

SC → O(1)

**Question:-** Find all pairs in a sorted array whose sum is K.

**Ex:-** [1 2 3 4 5 6 8]

K = 10

① **Unique Nos.** [Same as prev question].

```
count = 0 , i = 0, j = N-1
if ( A[i] + A[j] > K) {
        j--;
}
else if ( A[i] + A[j] < K) {
        i++
}
else {
        count++;
        i++;
        j--;
}
```

**Scenario 2 [Follow up]:-** Duplicate are there.

K = 13

**Ex:-** [2 , 3, 3, 10, 10, 10, 15]

Create freq Map/array.

```
  i     j
[ 2, 3, 10, 15 ]
  ↓  ↓  ↓  ↓
  1  2  3  1
```

~~17~~ ~~15~~ 13

**Approach:-** Create a distinct array & store freq of every element, whenever find a pair such that sum = K, in distinct array multiply frequency

**Follow up :-** In Constant space

Ex :- [2, 4, 4, 4, 5, 5, 7, 10, 10, 10, 15]
(indices 0 1 2 3 4 5 6 7 8 9 10)

i, i, i, j, j, j

17 12

K = 14

## PSEUDO CODE

```
i = 0 , j = N-1
ans = 0
while ( i < j ) {
    sum = A[i] + A[j];
    if ( sum < K ){
        i++
    } else if ( sum > K ) {
        j--;
    } else {
        if ( A[i] == A[j] ) {
            cnt = j - i + 1;
            ans += cnt * (cnt-1) / 2
            break;
        }
        i2 = i , cnt1 = 0
        while ( A[i2] == A[i] ){
            i2++;
            cnt1++;
```

Edge Case

```
i = i2;
j2 = j , cnt2 = 0

while( A[j2] == A[j] ){
    j2--;
    cnt2++;
}

j = j2;
ans += cnt1 * cnt2;
}
}
```

Edge Case:- K = 14

[2, 4, 4, 4, 7 7 7 7 7, 10, 10, 15]

i i → i  j ← j

j - i + 1

Q How many pair we can form now?

↳ $\frac{cnt * (cnt-1)}{2}$

TC → O(N)
SC → O(1)

**Question** :- Given a sorted integer array A & an integer K, find any pair (i, j) such that

SC → O(1)  $A[j] - A[i] = K$, $i != j$ & $K > 0$

Ex1  [-5, -2, 1, 8, 10, 12, 15]  | K = 11
(indices: 0, 1, 2, 3, 4, 5, 6; i = 0)

**Brute force**

↳ check all pairs using Nested loops.

TC → O(N²)

SC → O(1)

**Binary Search (BS)**

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

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

↳ For j apply BS on right side of i [i+1, N]

**Quiz 2** :- Given an array A is [5, 4, 2, 12, 1, 6] & K = 10. Find any pair (i, j) such that

→ $A[j] - A[i] = K$

→ $i != j$

Note :- 0-based indexing

↳ [2, 3]

## IDEA :- 2 Pointers

$A = [\underset{0}{-5}, \underset{1}{-2}, \underset{2}{1}, \underset{3}{8}, \underset{4}{10}, \underset{5}{12}, \underset{6}{15}]$ | $k = 11$

$i$ $\quad$ $j$

**Case I** $\quad i = 0 \quad j = N-1$

$\Rightarrow A[j] - A[i]$

$\Rightarrow 15 - (-5)$

$\Rightarrow 20 > 11$

★ Need to Decrease the Sum but $i{+}{+}$ will increase $A[i]$ & $j{-}{-}$ will decrease $A[j]$. In both scenarios the Sum is getting decreased. So can't take call.

**Case II** $\quad i = N-2 \quad , \quad j = N-1$

$\Rightarrow 15 - 12$

$\Rightarrow 3 < 11$

$\Rightarrow i = N-3 \quad , \quad j = N-1$

$\Rightarrow 15 - 10$

$\Rightarrow 5$

| $i{-}{-}$ | $j{-}{-}$ |
|---|---|
| $\Rightarrow 15 - 8$ | $\Rightarrow 12 - 10$ |
| $\Rightarrow 7$ | $\Rightarrow 2$ |

WORKS

A = [ 0: -5, 1: -2, 2: 1, 3: 8, 4: 10, 5: 12, 6: 15 ] | k = 11

Q why it worked?

largest element - A[i] < k

⇒ any element - A[i] < k

A[i] = 12

If we subtract from largest element then also its < 11. So it can't be subtracted from anything else. So Do i--.

## Dry Run

Ex A = [ 0: -5, 1: -2, 2: 1, 3: 8, 4: 10, 5: 12, 6: 15 ] | k = 11

i = 0, j = 6

| A[i] | A[j] | Diff | |
|---|---|---|---|
| -5 | -2 | -2 + 5 = 3 < 11 | → j++ |
| -5 | 1 | 1 - (-5) = 6 < 11 | → j++ |
| -5 | 8 | 8 + 5 = 13 > 11 | → i++ |
| -2 | 8 | 8 + 2 = 10 < 11 | → j++ |
| -2 | 10 | 10 + 2 = 12 > 11 | → i++ |
| 1 | 10 | 10 - 1 = 9 < 11 | → j++ |
| 1 | 12 | 12 - 1 = 11 | → Required |

## PSEUDO CODE

```
i = 0, j = 1
while ( j < n) {
    diff = A[j] - A[i];
    if ( diff == K){
        return (i, j);
    } else if ( diff < K) {
        j++
    } else {
        i++
    }
}
```

↳ In worst case i can become equals to j but since K > 0 is required. So in very next iteration j will increase

TC -> O(N)

SC -> O(1)

10:50

Quiz3 :- If the given array is [1, 2, 5, 4, 3] & K=9, Does there exist a subarray with Sum = K ?

↳ yes : [5, 4]

Question :- Find if in the Given Array there exist a subarray summing to K.

Ex:- A = [ 1 (0) 3 (1) 15 (2) 10 (3) 20 (4) 3 (5) 23 (6) ]

K = 33 → yes

K = 43 → No

Brute force

↳ check all subarray Sum.

↳ Carry Forward

TC → O(N²)

SC → O(1)

IDEA 2.

Ex:- A = [ 1 (0) 3 (1) 15 (2) 10 (3) 20 (4) 3 (5) 23 (6) ]

Q → Best Way to get subarray Sum?

↳ Prefix Sum array.

PS = [ 1 4 14 29 44 52 75 ]

Subarray Sum ⇒ PS[j] , i = 0

↓

[i, j] Pf[j] − Pf[i−1] i != 0

| i = 0 | i != 0 |
|---|---|
| check Pf[j] | ⇒ Pf[j] − Pf[i−1] |
| ↓ | ↳ Sum[i − j] |
| check all subarray starting from 0. | Brute question |
| | ↳ Diff = K. |

Twist :- SC → O(1), without changing I/P array.

IDEA :- SLIDING WINDOW

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| A = [ | 1 | 3 | 15 | 10 | 20 | 3 | 23 | 33 | 43 ] |
| | ✗ | ✗ | ✗ | i | ✗ | ! | | | |
| | | | | k = 33 | | j | | | |

sum = ~~4~~ ~~14~~ ~~29~~ ~~44~~ ~~48~~ ~~45~~ ~~30~~ 33

Conclusion :- ① To increase sum we need to increase size of window

i.e. j++

② To decrease sum we need to decrease size of window.

i.e. i++

## PSEUDO CODE

```
i = 0 , j = 0
Sum = A[0]

while ( j < n ) {
    if ( Sum == K ){
        return True;
    } else if (Sum < K){
        j++;
        if ( j == n) {break};
        Sum += A[j];
    } else {
        Sum -= A[i];
        i++;
    }
}
```

Edge Case :- when i passes j

K = 15

Exit :-

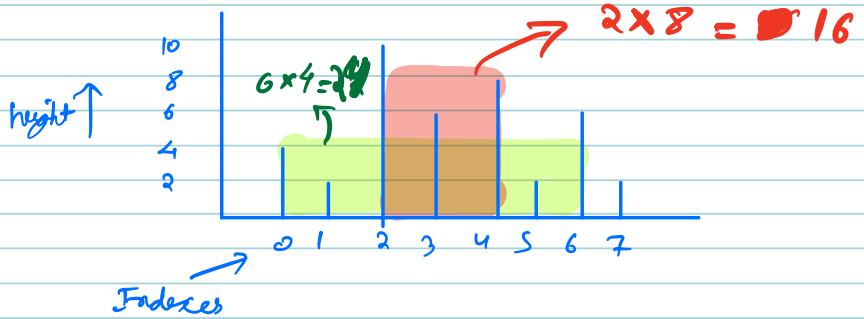
```
if ( i > j  &&  i <= n-1) {
    j = i  or  j++;
    Sum = A[j];
}
```

* one more edge case might be there. → H.W.

TC → O(N)
SC → O(1)

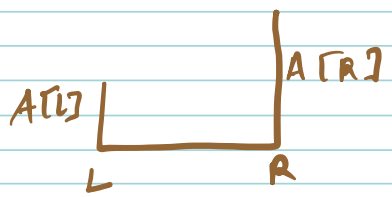
**Question:-** Given an integer array A where array elements represents the height of the wall. Find any two walls that can form a container to store the maximum amount of water.

**Ex:-** [4, 2, 10, 6, 8, 2, 6, 2]




**Ans = 24**

**Quiz 4** what is the water trapped b/w 2 walls at index L & R.
Array A gives the heights of buildings.
Choose the correct answer.



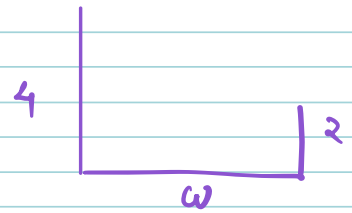

↳ min (A[L], A[R]) * (R-L)

**Q** where to start 2 pointers ?

↳ Start at max Distance apart

(i) L = 0 & (j) R = N-1

Q How to move the pointer ?



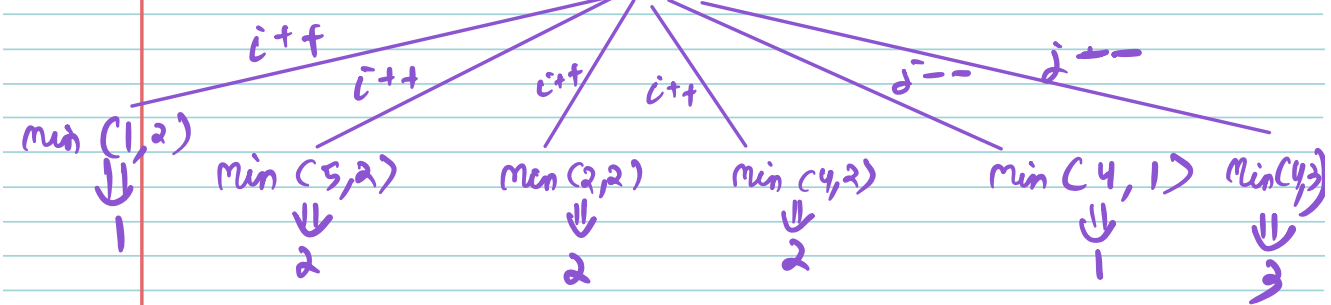

Ans = w * 2

=> Ans => width * height

i++ & j--, it will decrease

Need to work here

=> min (4, 2) = 2



j → move left
i → move Right

## DRY RUN

A = [4 2 10 6 8 2 6 2]
(0 1 2 3 4 5 6 7)

i, j

| A[i] | A[j] | water stored | |
|---|---|---|---|
| 4 | 2 | 14 | → j-- |
| 4 | 6 | 24 | → i++ |
| 2 | 6 | 10 | → i++ |
| 10 | 6 | 24 | → j-- |
| 10 | 2 | 6 | → j-- |
| 10 | 8 | 16 | → j-- |
| 10 | 6 | 6 | → j-- |
| 10 | 10 | → | Stop |

## PSEUDO CODE

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

TC - O(N)
SC - O(1)

return area;