2 pointers
- C++ ✗
- indices iterating over the Array.

Q.1 Amazon Given a sorted Array of size N of distinct elements. Check if there exists a pair (i,j) s.t $A[i] + A[j] = K$, $i != j$.

⇒ 2 Sum.

A: { 3, 7, 8, 12, 14 } K = 15

→ True

Approach # 1 :-

TC : $O(N^2)$

SC : $O(1)$

Approach # 2 :-

TC : $O(N)$

SC : $O(N)$

Approach # 3 :-

A: { 3, 7, 8, 12, 14 } K = 15

Search (K - A[i]) in Array.

⇒

TC : $O(N \log N)$

SC : $O(1)$

Approach # 4 :-

	0	1	2	3	4	5	6	7	8	9	
A :	-3	0	1	3	6	8	11	14	18	25	K = 17
	i									j	

i	j	A[i]	A[j]	Sum
0	9	-3	25	22 > 17
0	8	-3	18	15 < 17
1	8	0	18	18 > 17
1	7	0	14	14 < 17
2	7	1	14	15 < 17
3	7	3	14	17 == 17
				return true.

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

TC : O(N)
SC : O(1)

Q.2 Amazon/Visa Given a sorted Array of size N of distinct elements. Check if there exists a pair (i, j) s.t $A[i] - A[j] = K$, $K > 0$

0	1	2	3	4	5	6	7	8	9
-3	0	1	3	6	8	11	14	18	25

P1 → 1, P2 → 6

K = 5 ⇒ True.

25 - 0 = 25

	P1	P2	A[P2] - A[P1]	
1)	0	N-1	✗ 28 > 5 (decrease the diff)	P1++ P2--
2)	0	N/2	✗ 11 > 5	
3)	N/2	N/2 + 1	3 < 5	
4)	0	1	3 < 5	
	0	2	4 < 5	
	0	3	6 > 5	
	1	3	3 < 5	
	1	4	6 > 5	
	2	4	5 == 5	return true.

A[] : { 1 (0), 4 (1), 6 (2) } K = 2

P1 = 1, P2 = 2

P1	P2	A[P2] - A[P1]
0	1	3 > 2 ⇒ P1++
1	1	if (P1 == P2) ⇒ P2++
1	2	2 == 2 ⇒ return true.

```
bool check ( A[], N, K ) {
    P1 = 0
    P2 = 1
    while ( P2 < N ) {
        if ( A[P2] - A[P1] == K )
            return true;
        else if ( A[P2] - A[P1] > K ){
            P1++
            if(P1 == P2)  P2++
        }
        else {
            P2++
        }
    }
    return false;
}
```

TC : O(N)

SC : O(1)

A : { 3, 7, 8, 12, 14 } K = 16.

P1 → 3, P2 → 14

P1	P2	diff (A[P2] - A[P1])
0	1	4 < 16 P2++
0	2	5 < 16 P2++
0	3	9 < 16 P2++
0	4	16 == 16 ⇒ return true.

What if K < 0.

{ 3, 8 }

P2 → 3, P1 → 8

Abs(K)

* Important points about 2 pointers approach

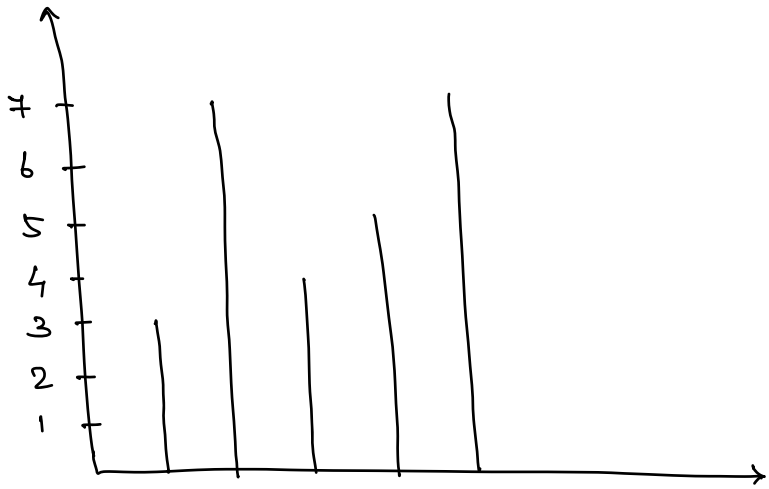
1) Where to intiliaze the pointers.
2) How to update your pointers.
3) When to stop ?

Q.3 Rain Water Trapping

Accesium Swiggy Intuit

Given an Array of size N, where A[i] represents the height of ith wall. Pick any 2 walls s.t max water can be stored b/w them.

Ex: { 3, 7, 4, 5, 7 } (indices 0 1 2 3 4)



i, j

water = Height * width

water = min(A[i], A[j]) * (j - i)

Brute force

```
for( i = 0; i < N; i++ ) {
    for( j = i+1; j < N; j++ ) {
        H = min( A[i], A[j] )
        w = j - i
        water = H * w
        ans = max( ans, water );
    }
}
```

TC : $O(N^2)$

SC : $O(1)$

A : { 3, 5, 4, 7, 3, 6, 5, 4, 1, 2 }

indices: 0 1 2 3 4 5 6 7 8 9

P1 P2

$$water = \overbrace{min(A[i], A[j])}^{H} * \underbrace{(j - i)}_{w.}$$

P1	P2	H = min(A[P1], A[P2])	W = P2 - P1	water
0	9	2	9	18.
0	8	1	8	8
0	7	3	7	21
1	7	4	6	24
1	6	5	5	25
2	6	4	4	16
3	6	5	3	15
3	5	6	2	12
3	4	3	1	3
3	3	7	0	0

* Update pointer which is having lesser height wall.
* If the heights of the wall at P1 & P2 are equal then we can update any of them.

Code :

Todo.

Q.4 Given 3 sorted Arrays A[], B[] & C[] of sizes N. Find i, j & k s.t

MS: $\max(A[i], B[j], C[k]) - \min(A[i], B[j], C[k])$ is minimized.

A: { 3 (0), 14 (1), 16 (2), 20 (3), 29 (4), 40 (5) }

B: { -6, 23, 24, 30, 35, 50 }

C: { -15, 15, 26, 31, 39, 42 }

i	j	k	A[i]	B[j]	C[k]	max	min	ans
0	0	0	3	-6	-15	3	-15	18
0	1	1	3	23	15	23	3	20
4	3	3	29	30	31	31	29	2
0	5	5	3	50	42	50	3	47
5	5	5	40	50	42	50	40	10
5	4	4	40	35	39	40	35	5
5	5	4	40	50	39	50	39	11

Brute force

TC: $O(N^3)$

SC: $O(1)$

A: { 3 (0), 14 (1), 16 (2), 20 (3), 29 (4), 40 (5) }
P1

B: { -6, 23, 24, 30, 35, 50 }
P2

C: { -15, 15, 26, 31, 39, 42 }
P3.

A[P1] = 3

B[P2] = -6

C[P3] = -15

max = 3

min = -15

ans = 18

X - Y

max(Ai, Bj, Ck) min(Ai, Bj, Ck)

* To minimize the value X-Y, we are trying the increase the value of Y.

A : { 3 , 14 , 16 , 20 , 29 , 40 } (0 1 2 3 4 5)

P1

B : { -6 , 23 , 24 , 30 , 35 , 50 }

P2

C : { -15 , 15 , 26 , 31 , 39 , 42 }

P3

P1	P2	P3	A[P1]	B[P2]	C[P3]	max	min	ans.
0	0	0	3	-6	-15	3	-15	18
0	0	1	3	-6	15	15	-6	21
0	1	1	3	23	15	23	3	20
1	1	1	14	23	15	23	14	9
2	1	1	16	23	15	23	15	8
2	1	2	16	23	26	26	16	10
3	1	2	20	23	26	26	20	6
4	1	2	29	23	26	29	23	6
4	2	2	29	24	26	29	24	5
4	3	2	29	30	26	30	26	4
4	3	3	29	30	31	31	29	2
5	3	3	40	30	31	40	30	10
5	4	3	40	35	31	40	31	9
5	4	4	40	35	39	40	35	5
5	5	4	40	50	39	50	39	11
5	5	5	40	50	42	50	40	10
6	5	5						

p1 == N

break.

ans = 2

Code (HW)

```
p1 = p2 = p3 = 0
while(p1 < N && p2 < N && p3 < N ) {
|
|
}
```

TC : O(N+N+N) ⇒ O(N)

SC : O(1)
