2 pointers

↓

variables iterating over indices of the array.

Quicksort

Reverse

Palindrome

merged 2 sorted

Expedia | Morgan Stanley | Goldman | JP morgan | Uber

Que 1. Given N distinct sorted ele, check if there exists a pair (i, j) A[i] + A[j] = K [i != j]

Ex1 : A[] = { 3 (7) (8) 12 19 } (0 1 2 3 4) K=15 True

Ex2 : A[] = { 2, 5, 8, 11, 15 } K=16

Ex3 : A[] = { -3, 0, 1, 3, 6, 8, 11, 14, 18, 25 } K=17

Brute

(1) Check all pairs TC: $O(N^2)$ SC: $O(1)$

(2) Use HashMap / HashSet TC: $O(N)$ SC: $O(N)$

(3) { 2, 5, 8, 11, 15 } K=16

16-2 = 14 | 16-5 = 11 | 16-8 = 8 | . . .

1 → $\log_2 N$

N → $N \log_2 N$

TC: $O(N \log_2 N)$

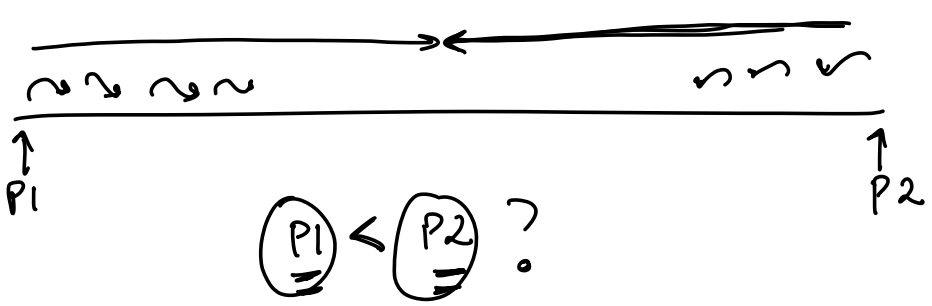
SC: $O(1)$

K = 17

(4)

	0	1	2	3	4	5	6	7	8	9
A[] =	-3	0	1	3	6	8	11	14	18	25
	~~P1~~	~~P1~~	~~P1~~	P1				P2	~~P2~~	~~P2~~

P1	P2	sum	
0	9	22 > 17 Decrease the sum	P2 --
0	8	15 < 17 increase	P1 ++
1	8	18 > 17 Decrease	P2 --
1	7	14 < 17 increase	P1 ++
2	7	15 < 17 Increase	P1 ++
3	7	17 = 17	Return true.



Code

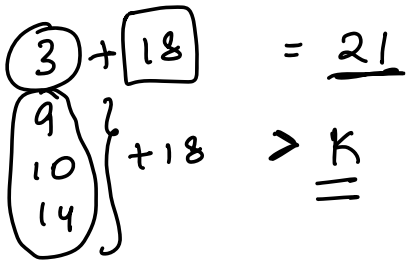
```
bool check ( A[], N, K) {
    P1 = 0      P2 = N-1
    while( P1 < P2 ) {
        if ( A[P1] + A[P2] == K) {
            return true
        }
        else if ( A[P1] + A[P2] < K){
            P1++
        }
        else P2--
    }
    return false
}
```

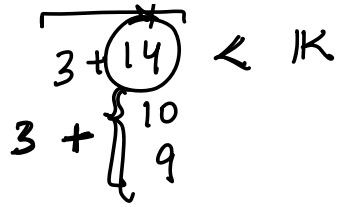
TC : O(N)

SC : O(1)

	0	1	2	3	4	
arr[] = {	3	9	10	14	18 }	K = 19
	↑			↑	↑	
	P1			P2	~~P2~~	

P1	P2		
0	4	21 > 19	P2--
0	3	17 < 19	


3 + 18 = 21
9
10
14
+18
> K


3 + 14 < K
3 +
10
9

Amazon | Visa

Q2. Given N distinct sorted elements, check if there exists a pair (i, j) s.t $A[j] - A[i] = K$ and $K > 0$ $[i \neq j]$

$a - b = K$

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

K = 5

True.

	0	1	2	3	4	5	6	7	8	9
A[] = {	-3,	0,	1,	3,	6,	8,	11,	14,	18,	25 }
						P1	P2			

K=5

	P1	P2	A[P2] - A[P1]		
Op 1	0 0	N-1 9	28 > 5 Decrease the difference	P2-- P1++	Unsure!
Op 2	0 0	N/2 5	11 > 5 Decrease the difference	P1++ P2--	Unsure!
Op 3	N/2 5	N/2+1 6	3 < 5 Increase the difference	P1-- P2++	unsure!

	P1	P2
OP4	0	1

A[] = { -3, 0, 1, 3, 6, 8, 11, 14, 18, 25 }

0 1 2 3 4 5 6 7 8 9

P1 P2

K = 5

P1	P2	A[P2] - A[P1]			
0	1	3	3 < 5	↑	P2++
0	2	4	4 < 5	↑	P2++
0	3	6	6 > 5	↓	P1++
1	3	3	3 < 5	↑	P2++
1	4	6	6 > 5	↓	P1++
2	4	5	5 = 5		Return True.

0 1 2

1 4 6

P1 P2

K = 2

P1++

P2++

P1	P2	
0	1	3 > 2

```
bool diff(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)

What if K is ve -

{ 3, 8 } K = 5 .

8 - 3

3 - 8 = -5

K = abs(K)

Conditions to apply 2 pointers

** IMP

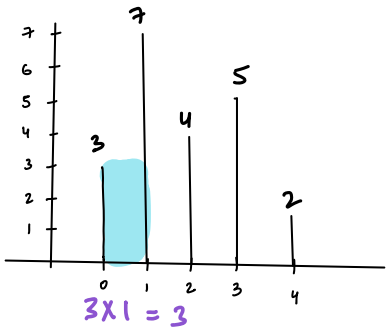
1) Where to initialize pointers.
2) How to update your pointers.
3) When to stop.

Que 3. Given an array of size N where A[i] represents height of each wall. Pick any 2 walls s.t max water it accummulated b/w them.

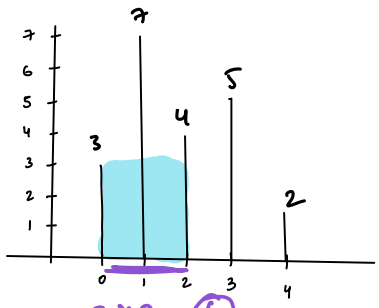
Ex: {3, 7, 4, 5, 2}

Arcesium
Swiggy
Intuit
Samsung

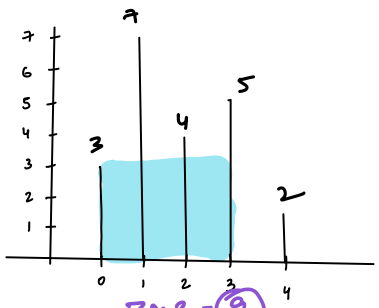
$\{3, 7, 4, 5, 2\}$



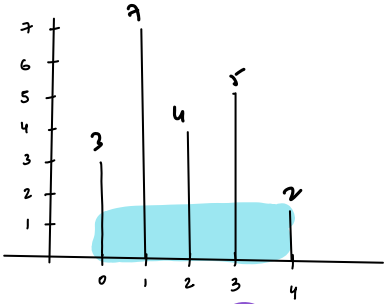
$3 \times 1 = 3$



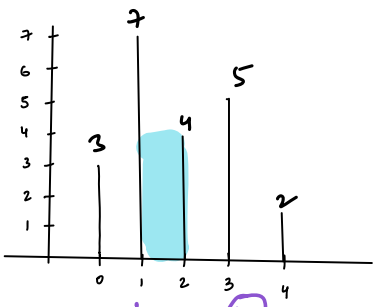
$3 \times 2 = 6$



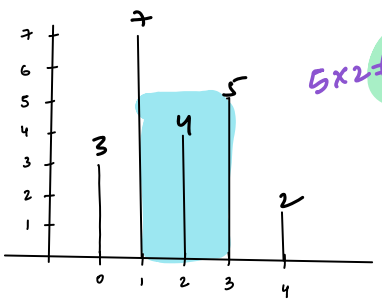
$3 \times 3 = 9$



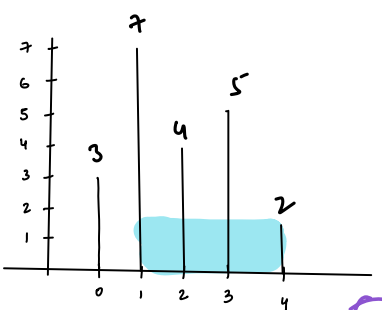
$2 \times 4 = 8$



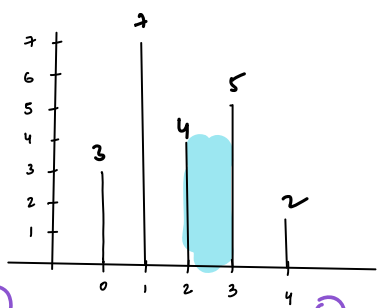
$4 \times 1 = 4$



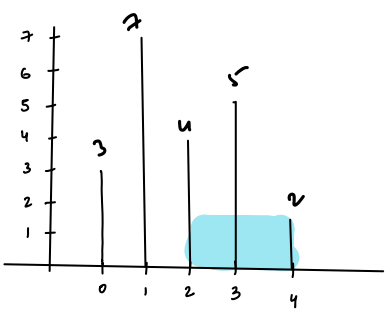
$5 \times 2 = 10$



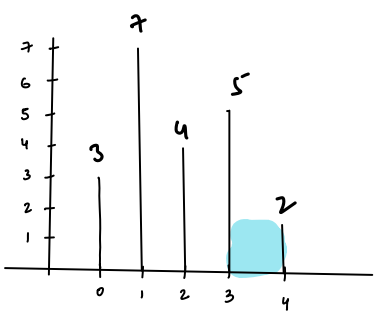
$2 \times 3 = 6$



$4 \times 1 = 4$



$2 \times 2 = 4$



$2 \times 1 = 2$

Brute Force

```
for (i = 0; i < N; i++) {
    for (j = i+1; j < N; j++) {
        height = min(A[i], A[j])
        width = j - i
        water = height x width
        ans = max(ans, water)
    }
}
return ans
```

TC: $O(N^2)$
SL: $O(1)$

Break 15 min.

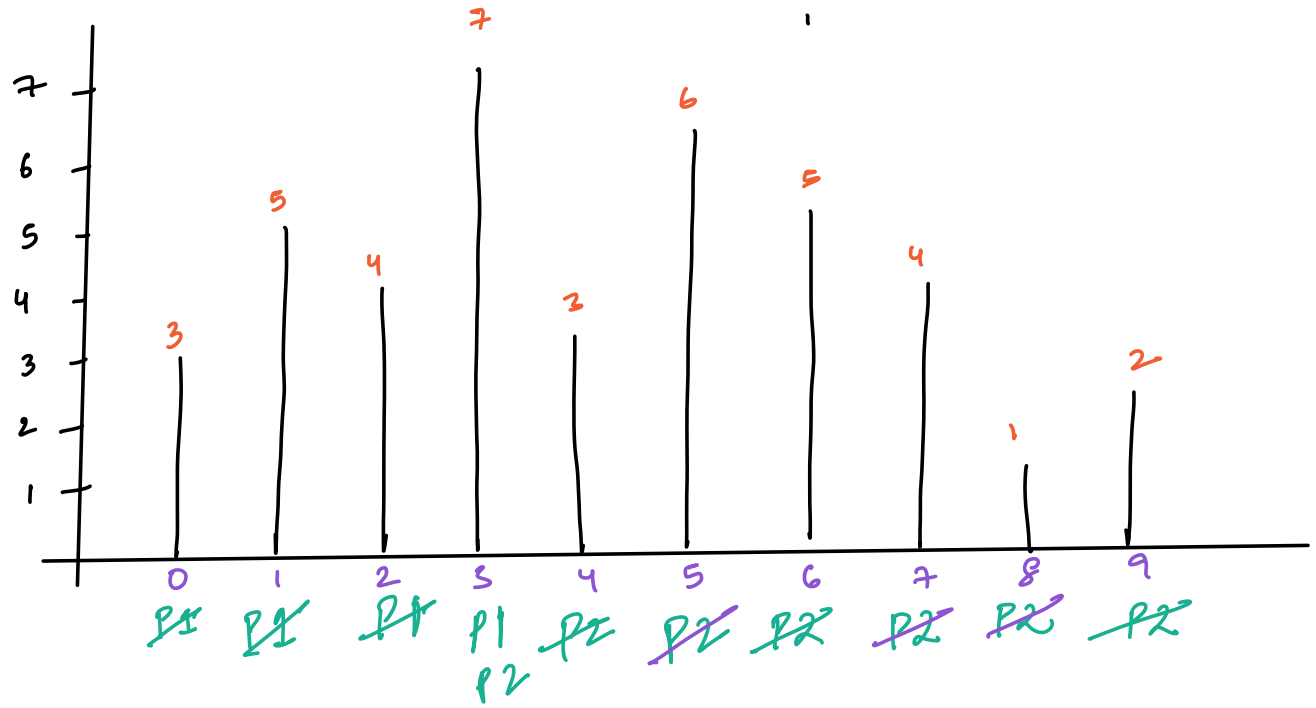
10 min → noting.

5 min.

15 min

10:42

A[] = { 3, 5, 4, 7, 3, 6, 5, 4, 1, 2 }

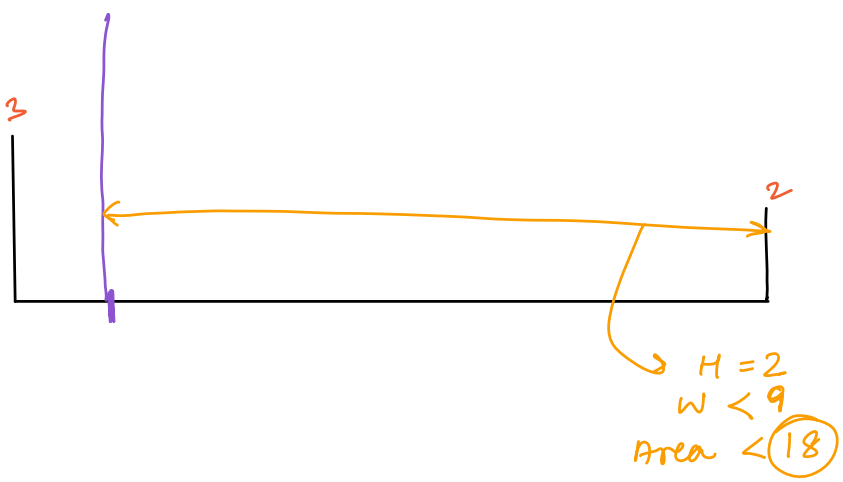


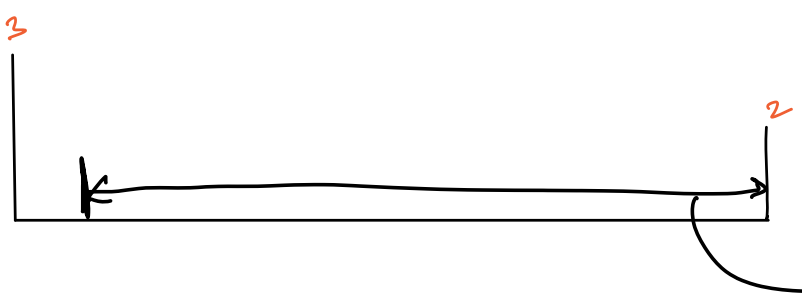
P1	P2	H = min(A[P1], A[P2])	W (P2 − P1)	Area = H * W
0	9	2	9	18
0	8	1	8	8
0	7	3	7	21
1	7	4	6	24
1	6	5	5	25
2	6	4	4	16
3	6	5	3	15
3	5	6	2	12

any

3 4 3 1 3

3 3 STOP





Code

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

↑ Promise ?

MICROSOFT

Que. Given 3 sorted arrays A[], B[], C[] of size N find i, j, K s.t

max(A[i], B[j], C[K]) - min(A[i], B[j], C[K]) is minimized

	0	1	2	3	4	5
A = {	3 (P1)	14	16	20	29	40 }
B = {	-6 (P2)	23	24	30	35	50 }
C = {	-15 (P3)	15	26	31	39	42 }

P1	P2	P3	A[P1]	B[P2]	C[P3]	max	min	ans
0	0	0	3	-6	-15	3	-15	18
0	1	1	3	23	15	23	3	20
1	2	1	14	24	15	24	14	10
0	5	5	3	50	42	50	3	47
4	3	3	29	30	31	31	29	2

Brute Force

```
for( i = 0 ; i<N ; i++){
    for(j = 0 ; j<N ; j++){
        for(K=0 ; K<N ; K++){
            .
            .
            .
```

TC : O(N3)
SC : O(1)

	0	1	2	3	4	5
A = {	3	14	16	20	29	40 }
	P1 ↑					
B = {	-6	23	24	30	35	50 }
	P2 ↑					
C = {	-15	15	26	31	39	42 }
	P3 ↑					

A	B	C	max	min	diff
3	-6	15	15	-6	21

A[]	B[]	C[]	max	min
3	-6	-15	3	-15
14	23			
16	24			
20	30			
29	:			
40	1			

18

3 - (-15) = 18

-(-15) > 18

P1

	0	1	2	3	4	5
A = {	3	14	16	20	29	40 }

P2

B = {	-6	23	24	30	35	50 }

P3

C = {	-15	15	26	31	39	42 }

P1	P2	P3	A[P1]	B[P2]	C[P3]	max	min	diff
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 = P2 = P3 = 0
while (P1<N && P2<N && P3<N) {
    TODO
}
```

TC: N+N+N

3N

O(N)

SC: O(1)

Doubts =

a b c N=6

p1

0	1	2	3	4	5
1	2	3	4	5	90
pr 7	8	9	10	11	90
p3 13	14	15	16	17	90

13 1

13 2

	0	1	2	3	4	5
A =	3	14	16	20	29	50 p1
B =	-6	23	24	30	35	50 p2
C =	-15	15	26	31	39	49 p3