Two Pointer →

Q) Given sorted array (distinct elements), count all pairs i, j such that $arr[i] + arr[j] = K$ $(i \ne j)$.

Eg:

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

, $K = 17$

count = ~~1~~ 2

A1 → Consider all the pairs. T.C → $O(n^2)$

→ $O(n \log n)$ → B.S.

→ $O(n)$

A2. Fix one element, apply Binary search for $(K - a)$

$a + b = K$

$b = K - a$

$a = -3$, $a + b = 17$

$b = 17 - (-3)$

$b = 20$

T.C → $O(n \log n)$

A3. : $arr[i] + arr[j] = K$

$i = 0$ $j = 1$ × Ambiguity	$i = 0$ $j = n-1$ ✓

$a[0] + a[9] = 22 \; (> 17)$ j--;

$-3 + 18 = 15 \; (< 17)$ i++;

$0 + 18 = 18 \; (> 17)$ j--;

$0 + 14 = 14 \; (< 17)$ i++

pseudo-code

```
i = 0,   j = n-1 ;
while( i < j ){
      sum = a[i] + a[j] ;
      if ( sum == K){
            count++ ;
            i++ , j-- ;
      }
      else if (sum < K){
            i++ ;
      }
      else {
            j-- ;
      }
}
```

$T.C \rightarrow O(n)$
$S.C \rightarrow O(1)$

unsorted → sort + 2-pointer   nlogn
↳ Maps

Q.1 Given sorted array (distinct elements), count all the pairs (i, j) such that arr[j] - arr[i] = K. (i != j)

arr →

-3	~~0~~	~~1~~	3	6	8	11	14	18	25	28
0	1	2	3	4	5	6	7	8	9	10

, K = 5

K = -5

$\boxed{K = |K|}$

i i i j j

greater - smaller = K

① $i = 0$, $j = n-1$

arr[j] - arr[i] = K

28 - (-3) = 31 (>K)

decrease j or increase i

② $i = 0$, $j = 1$

count

1
2
3

0 - (-3) = 3 (<K) ⟹ increase j

discarding '0' as greater element

1 - (-3) = 4 (<K) ⟹ increase j

3 - (-3) = 6 (>K) ⟹ increase i

3 - (0) = 3 (<K) ⟹ j++

6 - (0) = 6 (>K) ⟹ i++

6 - 1 = 5 (=K) ⟹ i++, j++

8 - 3 = 5 (=K) ⟹ i++, j++

11 - 6 = 5 (=K) ⟹ i++, j++

14 - 8 = 6 (>K) ⟹ i++

14 - 11 = 3 (<K) ⟹ j++

18 - 11 = 7 (>K) ⟹ i++

25 - 18 = 7 (>K) ⟹ (i++)

pseudo-code

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

$\underbrace{i == j}$

T.C → O(n)
S.C → O(1)

Q1 Given an array of +ve integers, find count of subarray with sum = K.

arr →

3	2	5	1	8	6	2	10
0	1	2	3	4	5	6	7

, K = 15

ans = 1.

B.F. Consider all the subarrays. ⇒ $\frac{n(n+1)}{2}$ T.C → $O(n^2)$

A2 : prefixSum →

pSum →

0	3	5	10	11	19	25	27	37
	0	1	2	3	4	5	6	7

$\boxed{\text{subarray sum } i \to j \Rightarrow pSum[j] - pSum[i-1]}$

$\forall\, i, j \quad pSum[j] - pSum[i-1] = K$

→ We can't consider subarrays starting from index 0. (approach)

add '0' in front of pSum.

pSum[i]
↓
sum of elements from 0 to i

$\forall i$ if(pSum[i] == K)
count++;

$\boxed{\begin{array}{l} T.C \to O(n) \\ S.C \to O(1) \end{array}}$

Q:) Given a sorted array, find triplets (i, j, k) (distinct)

(3 sum)

$arr[i] + arr[j] + arr[k] = sum$

arr →

-8	-4	-3	-1	2	3	5	7	9
0	1	2	3	4	5	6	7	8

sum = 14

B.F. → Consider all the triplets → 3 loops → $O(n^3)$

A2 :

$\underbrace{a}_{\text{let's fix this}} + b + c = sum.$

$b + c = sum - a$

⇓

T.C → $O(n)$

Approach → fix every element one by one. then apply 2-pointer approach to find all the pairs having target = sum - a[i].

$\boxed{\begin{array}{l} T.C \rightarrow O(n^2) \\ S.C \rightarrow O(1) \end{array}}$

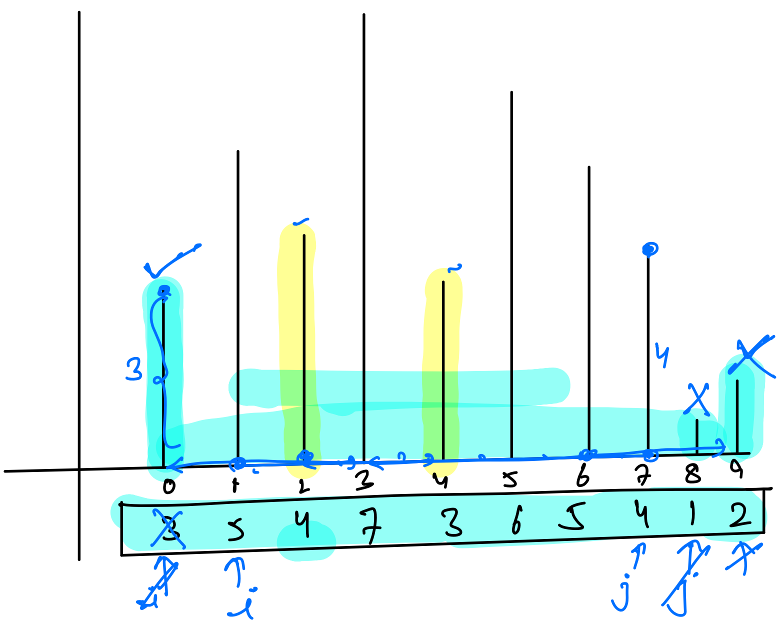
```
for( i = 0 ; i < n-2 ; i++) {
        target = sum - a[i];
        count += 1st question(a, target, i);
}
```

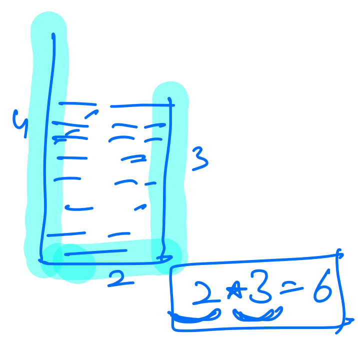
4 sum $a+b+c+d = \text{sum} \Rightarrow$

Q) Given N-array elements
↳ height of walls.

Pick any two walls such that max water accumulated b/w walls.

ans →





$(1,6) \rightarrow 25$

$(2,7) \rightarrow 20$

$(1,7) \rightarrow 24$

① B.F → consider all pair → $O(n^2)$

→ walls should be as far as possible

⇓

$i=0, \; j=n-1$

9 * 2 = 18.

8 * 1 = 8

3 * 7 = 21

⋮

pseudo-code

```
i = 0,  j = n-1,  ans = 0;
while ( i < j ) {
    ans = Max( ans, (j - i) * min(a[i], a[j]) );
    if ( a[i] < a[j] ) {
        i++;        // we have considered the
                    best possible ans for arr[i]
    }
    else {
        j--;        //    "
    }
}
return ans;
```

Q) Given 3 sorted arrays → A, B, C.

We need to choose a triplet (one element from each) such that

$$\boxed{\max(A[i], B[j], C[k]) - \min(A[i], B[j], C[k])}$$

is minimised. Find the minimum difference.

A : 1 4 5 8 10

B : 6 9 15

C : 2 3 6 8

Idea → 3 pointers