Today's content

- → Pair Sum
- → Pair Difference
- → Count of subarrays with sum = K.
- → Triplet Sum
- → Maximum water accumulated.

Q1) Given arr[N] with sorted distinct elements, Count all the pairs (i, j) such that arr[i] + arr[j] = K and i != j.

Eg → arr → [-3 0 1 3 6 8 11 14 18 25], K = 17.

idea-1. → Consider all the pairs, TC → $O(N^2)$, S.C → $O(1)$

N log N   N

idea-2. $a + b = K \Rightarrow \{b = K - a\}$

Fix one element, then apply B.S for 2nd element. (K−a)

T.C → $O(N \log N)$, S.C → $O(1)$

idea-3.: using Hashset.

→ insert all the elements in the hs.

→ fix one element and check if second element (K−a) is present in hashset.

T.C → $O(N)$, S.C → $O(N)$

arr → [-3 0 1 3 6 8 11 14 18 25] , sum = 17.

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

↑ i ↑ j

① $i = 0, \; j = 1$

② $i = N-1, \; j = N-2$

③ $i = \frac{N}{2}, \; j = \frac{N}{2} + 1$.

× Ambiguious case.

④. $i = 0, \; j = N-1$

$-3 + 25$	$>$	17	(decrease j)
$-3 + 18$	$<$	17	(increase i)
$0 + 18$	$>$	17	(decrease j)
$0 + 14$	$<$	17	(increase i)
$1 + 14$	$<$	17	i++
$3 + 14$	$==$	17	count++, i++, j--
$6 + 11$	$==$	17	count++, i++, j--

pseudo-code

```
i = 0,  j = N-1
while ( i < j ){
    sum = arr[i] + arr[j]
    if (sum == k){
        count ++
        i++
        j--
    } else if (sum < k){
        i++
    }
    else {
        j--
    }
}
```

T.C → O(N)
S.C → O(1)

[# unsorted → sort + 2-pointer
↳ Hashmap.]

Q1) Given arr[N] with sorted distinct elements, count of all the pairs (i, j) such that arr[j] - arr[i] == K.
and i != j.

Eg → arr → [-3 0 1 3 6 8 11 14 18 25], K = 5.

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

K = -5.

K = |K|

↑ i ↑ j

① i = 0, j = N-1.

arr[j] - arr[i] > K

25 - (-3) = 28 > K.

larger - smaller > K. ⇒ decrease j or increase i

⇓

Ambiguity

② i = 0, j = 1 arr[j] - arr[i]

0 - (-3) = 3 (< K) ⇒ increase j [discarding '0' as larger element]

1 - (-3) = 4 (< K) ⇒ increase j

3 - (-3) = 6 (> K) ⇒ increase i.

3 - 0 = 3 (< K) ⇒ increase j

6 - 0 = 6 (> K) ⇒ increase i

6 - 1 = 5 (== K) ⇒ count++, i++, j++

8 - 3 = 5 (= K) ⇒ count++, i++, j++

11 - 6 = 5 (= K) ⇒ count++, i++, j++

pseudo-code.

```
i = 0 , j = 1 , count = 0 , 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++ }
        }
}
```

T.C → O(N)
S.C → O(1)

Q.1 Given an array of +ve integers, find count of subarrays with sum = K.

arr → [3 2 5 1 8 6 2 10], K = 15

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

[ans = 1].

idea-1. Consider all the subarrays & check their sum.

T.C → $O(N^3)$ $\xrightarrow[\text{Carry forward}]{\text{pSum or}}$ $O(N^2)$

Idea-2. → pSum[] →

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

Sum of subarray from i → j ⇒ pSum[j] - pSum[i-1] = K.

⇓

Sum of subarray from i to j = K.

to handle (i=0) edge case.

add '0' in the starting psum[]

{psum[j] == K.} count ++

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

[Code → #todo.]

Q) Given arr[N] with sorted distinct elements. Find triplet (i, j, K) such that arr[i] + arr[j] + arr[K] = sum. (i != j != K)

arr → [-8 -4 -3 -1 2 3 5 7 9] , sum = 14.

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

idea-1. Consider all the triplets. T.C → $O(N^3)$

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

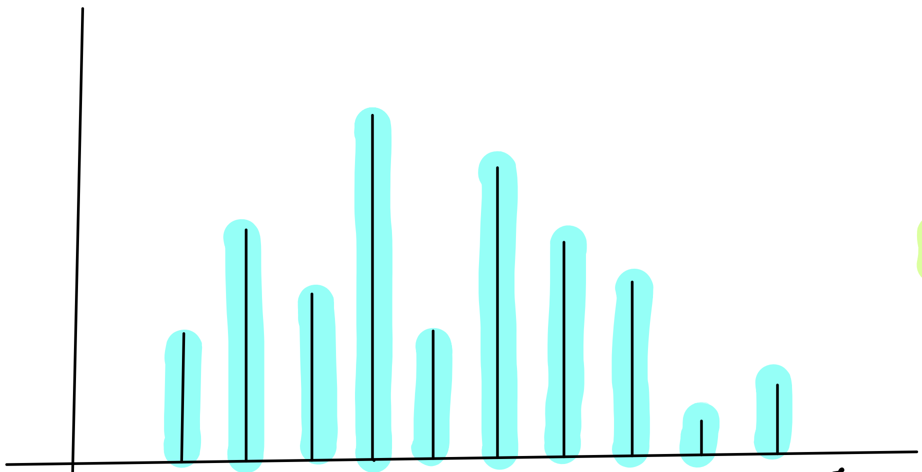
$\underbrace{b + c = \text{sum} - a}_{T.C \to O(N)}$

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

$$\boxed{\begin{matrix} T.C \to O(N^2) \\ S.C \to O(1) \end{matrix}}$$

```
count = 0;
for( i = 0; i < N; i++) {
        target = sum - arr[i]
    count += pairsum ( arr, target, i);
}
count;
```

Q) Given arr[N] where every element represents height of walls. You need to pick any two walls such that water accumulated is maximum.



$(1,7) \rightarrow \underline{24}$

$(1,6) \rightarrow \underline{25}$.

arr→ [3 5 4 7 3 6 5 4 1 2]

0 1 2 3 4 5 6 7 8 9

i↑ ↑j

idea-1. Consider all the pairs.

Water accumulated for pair $(i,j) = \text{Min}(arr[i], arr[j]) * (j-i)$

$[T.C \rightarrow O(N^2)\ ,\ S.C \rightarrow O(1)]$

→ Walls should be as far as possible.

⇓

$i = 0\ ,\ j = N-1.$

9 * 2 = 18.

8 * 1 = 8

7 * 3 = 21

6 * 4 = 24

5 * 5 = 25

4 * 4 = 16

5 * 3 = 15

6 * 2 = 12.

3 * 1 = 3.

pseudo-code.

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

while ( i < j ){
        water = min(arr[i], arr[j]) * (j - i);
        ans = max( ans , water);
        if (arr[i] < arr[j]){
                i++
        } else {
                j-- ;
        }
}
return ans;
```

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

- Pair Sum / Pair Difference / pairs.
- Binary arr
- Triplets

think of applying 2-pointer approach.

$$|A[i]-A[j]| + |B[i]-B[j]| + |C[i]-C[j]| + |D[i]-D[j]| + |i-j|$$

$$|A[i]-A[j]| + |i-j| + |B[i]-B[j]|$$

$\Downarrow$

$A[i]-A[j]+i-j+B[i]-B[j] \Rightarrow A[i]+B[i]+i-(A[j]+B[j]+j)$

$A[i]-A[j]+i-j+B[j]-B[i] \Rightarrow A[i]-B[i]+i-(A[j]-B[j]+j)$

$A[i]-A[j]+j-i+B[i]-B[j] \Rightarrow A[i]+B[i]-i-(A[j]+B[j]-j)$

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

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

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

$A[j]-A[i]+j-i+B[i]-B[j] \Rightarrow A[j]-B[j]+j-(A[i]-B[i]+i)$

$A[j]-A[i]+j-i+B[j]-B[i] \Rightarrow A[j]+B[j]+j-(A[i]+B[i]+i)$

$$\left\{\begin{matrix} + & + & + \\ + & + & - \\ + & - & + \\ + & - & - \end{matrix}\right\}$$

$$\begin{array}{ccccc} a[i] & b[i] & c[i] & d[i] & i \end{array}$$

$$\left\{\begin{matrix} + & + & + & + & + \\ + & + & + & + & - \\ + & + & + & - & + \\ + & + & - & + & + \\ + & - & + & + & + \\ + & + & + & - & - \\ + & + & - & - & + \\ + & - & - & + & + \end{matrix}\right\}$$

2d array.

$$\begin{array}{c} 0 \to \\ 1 \to \\ 2 \to \\ 3 \to \\ 4 \to \\ 5 \to \\ 6 \to \\ 7 \to \end{array} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 \end{bmatrix}$$

$$\underbrace{a[i] + b[i] + c[i] + d[i] + i}$$