Today's content

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

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

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

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

ans = 2

Idea-1. Consider all the pairs. T.C → O(N^2), S.C → O(1)

Idea-2 a + b = K ⇒ b = K - a.

Fix one element, then apply B.S to find the corresponding second element.

T.C → O(N log N), S.C → O(1)

Idea-3. Use hashmap / hashset / dictionary

→ insert all elements in hashset.

→ fix one element, then search for the corresponding second element in hashmap / hashset / dictionary.

[T.C → O(N) S.C → O(N)]

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

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

i, j

① i = 0, j = 1 X

② i = N-1, j = N-2 X

③ i = 0, j = N-1

arr[i] + arr[j] = K.

-3 + 25 > 17 ⇒ j--

-3 + 18 < 17 ⇒ i++

0 + 18 > 17 ⇒ j--

0 + 14 < 17 ⇒ i++

1 + 14 < 17 ⇒ i++

3 + 14 = 17 ⇒ count++, i++, j--

6 + 11 = 17 ⇒ count++, i++, j--

pseudo-code

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

while( i < j){

    sum = arr[i] + arr[j] ;

    if (sum == K) {
    {           count++ , i++, j--      // replace this code for duplicates
    }
    else if ( sum < K){
    {           i++;
    }
    else{
    {           j-- ;
    }
}

    return count;
```

T.C → O(N)
S.C → O(1)

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

K → |K|

arr[] = [-3, 0, 1, 3, 6, 8, 11, 14, 18, 25, 30], K=5

indices: 0 1 2 3 4 5 6 7 8 9 10

[ans=4]

i, j

① i=0, j=N-1 ✗

arr[j] - arr[i] > K

increment i or decrement j ⇒ Ambiguity

② i=0, j=1 ✓

arr[j] - arr[i]

0 - (-3) = 3 < 5 ⇒ j++ [discarding '0' as larger element]

1 - (-3) = 4 < 5 ⇒ j++ [discarding '1' as larger element]

3 - (-3) = 6 > 5 ⇒ i++ [discarding '-3' as smaller element]

3 - 0 = 3 < 5 ⇒ j++ [discarding '3' as larger element]

6 - 0 = 6 > 5 ⇒ i++ [discarding '0' as smaller element]

6 - 1 = 5 = 5 ⇒ i++, j++

8 - 3 = 5 = 5 ⇒ i++, j++

11 - 6 = 5 = 5 ⇒ i++, j++

code →

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

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++ }
        }
}

return count;
```

T.C → O(N)
S.C → O(1)

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

arr[] → [3 2 5 1 8 6 2 10], K = 15, ans = 1.
(indices: 0 1 2 3 4 5 6 7)

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

T.C → O(N^3), S.C → O(1)

idea-2.

pSum[] → [3 5 10 11 19 25 27 37]
(indices: 0 1 2 3 4 5 6 7)

subarray sum from i to j ⇒ pSum[j] − pSum[i-1] = K.

a − b = K.
(previous question)

∴ Question reduced to →
Find no. of pairs (i, j) in pSum[] such that difference of the pair is equal to K.

edge-case → if K itself is present in pSum[], then count++.

T.C → O(N)
S.C → O(1)

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.
0 1 2 3 4 5 6 7 8

Idea-1 → Consider all the triplets.

```
for( i=0; i<N; i++){
    for( j = i+1; j<N; j++){
        for( K= j+1; K<N; K++){
            if(arr[i] + arr[j] +arr[K] == sum){
                count ++
            }
        }
    }
}
```

[T.C → O(N^3)
S.C → O(1)]

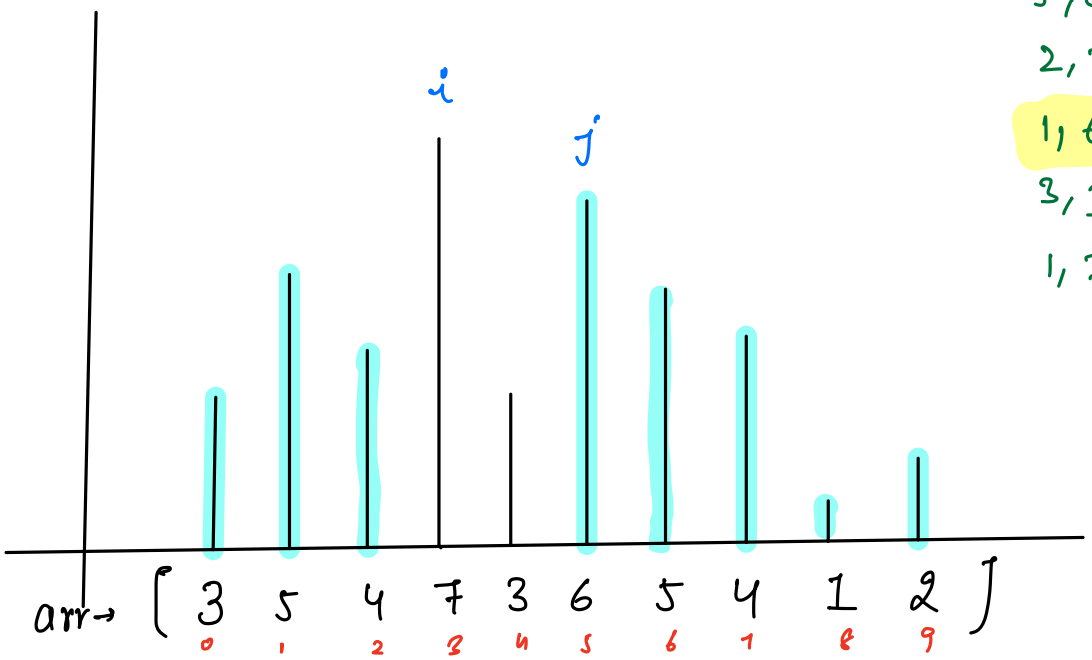
a + b + c = K ⇒ b + c = K - a ⇒ pair sum equals to K-a.
fix

idea-2 Fix one element & apply two-pointer approach on all rhs elements.

```
count = 0
for( i = 0; i < N-2; i++){
    target = K - arr[i]
    count += pairSum( arr[], target, i);
}
return count;
```

[T.C → O(N^2)
S.C → O(1)]

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



5,6 → 5 X
2,7 → 20 X
1,6 → 25
3,5 → 12 X
1,7 → 24 X

ans → 25.

idea → Consider any two walls or
Consider all the pairs & find maximum water accumulated.
water accumulated → $(j-i) * \min(arr[i], arr[j])$

[T.C → $O(N^2)$ S.C → $O(1)$]

idea-2 Walls should be as far as possible.

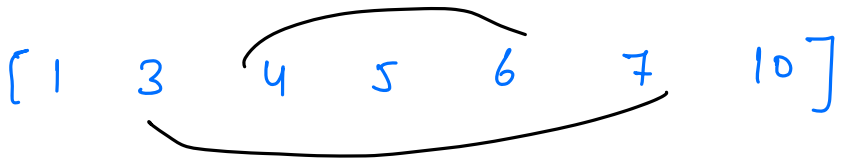
2 * 9 → 18
1 * 8 → 8
3 * 7 → 21
4 * 6 → 24
5 * 5 → 25
4 * 4 → 16
4 * 3 → 12

pseudo-code.

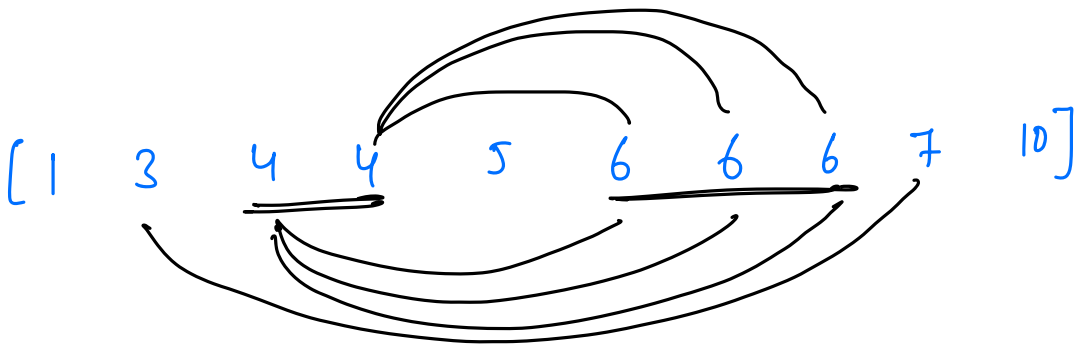
```
i = 0, j = N-1, ans = 0

while (i < j){
    water_accumulated = (j-i) * min(arr[i], arr[j])
        ans = Max(ans, water_accumulated);
        if (arr[i] < arr[j]){
            i++;
        }
        else{
            j--;
        }
}

return ans;
```

T.C → O(N)
S.C → O(1)

K = 10



ans = 2.



ans = 7.

[1 3 4 4 5 5 5 5 6 6 6 7 10]

ans = 13.

i j

count = 0 ~~1~~ ~~7~~ 13.

$^4C_2 \rightarrow \frac{4 \times 3}{2} \rightarrow 6.$

$^NC_2 \rightarrow \frac{N(N-1)}{2}$

Code -

```
if (arr[i] + arr[j] == k){
    if (arr[i] == arr[j]){
        c = j - i + 1
        count += c*(c-1)/2 ;
        break;
    }
    else{
        left = arr[i], lc = 0
        while( arr[i] == left){
            i++;
            lc++;
        }
        right = arr[j], rc = 0
        while( arr[j] == right){
            j--;
            rc++;
        }
        count += (lc*rc);
    }
}
```

{T.C → O(N)
S.C → O(1)}

Contest → 1st Sept.

{ 9:00 PM → 10:30 P.M. (contest).
10:30 P.M → 12:00 A.M [contest discussion] }

Syllabus → Maths, Recursion, Sorting

- → Prime no.
- → GCD

dry-run

Sorting questions

[Please go through lecture note bfore the contest.)