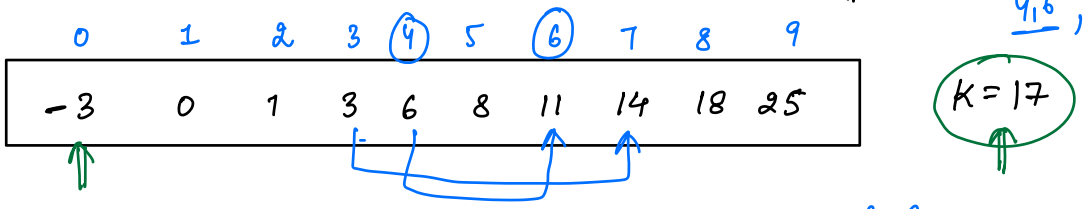
Two Pointers → indices

Q1 Given sorted arrays (distinct elements), count all pairs (i, j) such that $arr[j] + arr[i] = K$ $(i \neq j)$



B.F: $O(n^2)$ → consider every valid pair

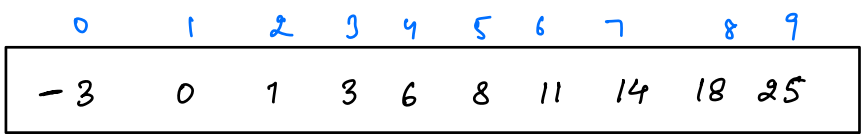
$a + b = k$ $K = 17$

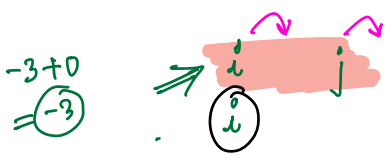
$a = -3$

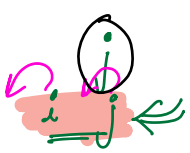
$b = 17 - (-3)$

$b = 20$ → Binary search

T.C: $O(N \log N)$





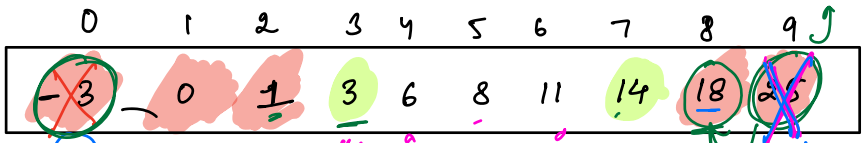


18+25 = 43 > 17

$arr[i] + arr[j] = k$ $K = 17$

increase

decrease



25+(-3)

17

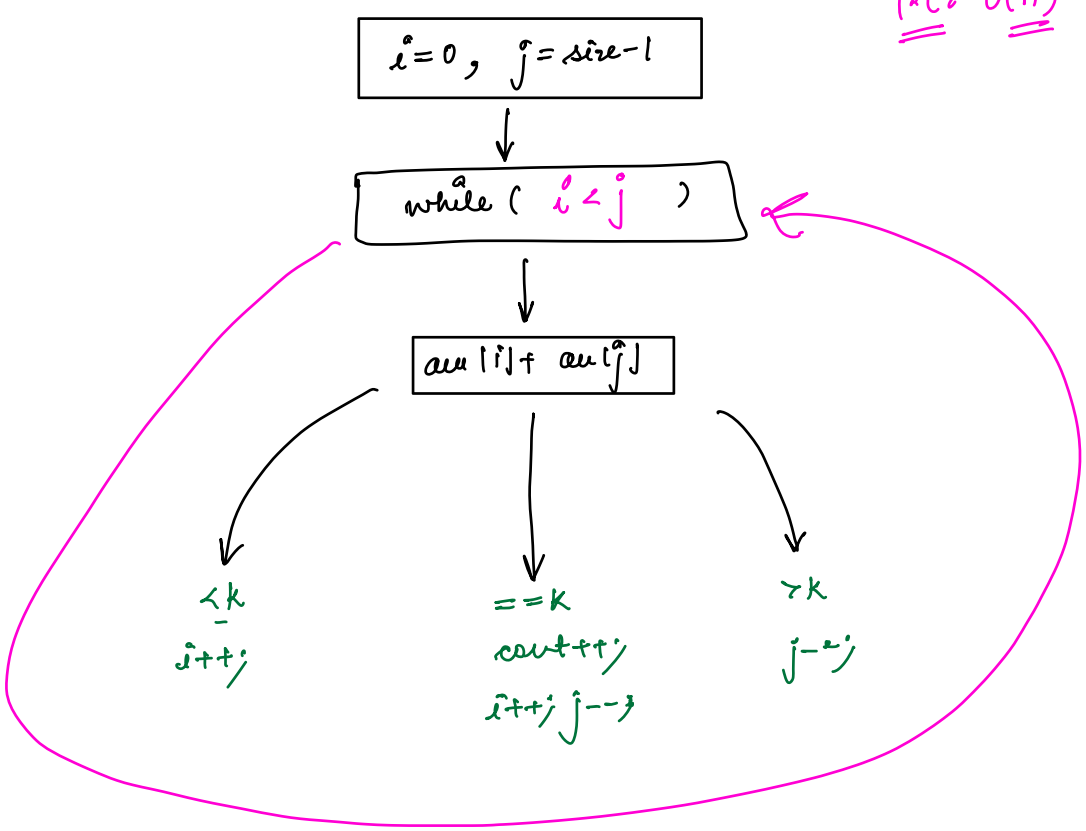
-3 + max

i	j		
0	9	22 > 17	j--
0	8	18+(-3) = 15 < 17	i++
1	8	18 > 17	j--
1	7	14 < 17	i++
2	7	15 < 17	i++
3	7	3+14 == 17	count++; i++; j--;
4	6	6+11 == 17	count++; i++; j--;
5	5		

k=17

n^2

T.C: O(n)



When you should move your pointer? ⇒ only if we are sure that it is not going to be used in the ans

Q2 Given sorted arrays (distinct elements), count all pairs (i, j) such that $arr[j] - arr[i] = K$ $(i \neq j)$

$K = abs(K)$

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

$K = -5$

$K = 5$

count = 3

$O(n^2)$

$O(n \log n)$

greater - smaller

$i = 0$ $j = 1$

$i = 0$

smaller

$8 - 3 = 5$

$3 - 8 = -5$

$j < n-1$

greater

$arr[j] - arr[i] = k$ $K = 5$

$25 - (-3) = 28 > 5$ ⇓

$0 - (-3) = 3 < 5$ ⇑

decrease

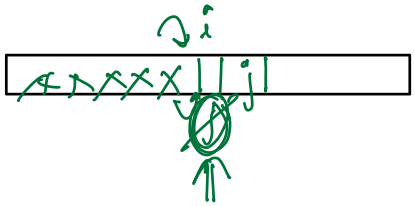
increase

5

0	1	2	3	4	5	6	7	8	9	10	11
-3	0	1	3	6	8	11	14	18	25	28	32

rejected 0 as lower value ←

i	j	diff		
0	1	3	3<5	⇑ j++;
0	2	4	4<5	⇑ j++;
0	3	6	6>5	⇓ i++;
1	3	3	3<5	⇑ j++;
1	4	6	6>5	⇓ i++
2	4	5	5==5	cout++; i++; j--
3	5	5	5==5	cout++; i++; j--



4	6	5	s==5	cout++; i++ j++
5	7	6	6 > 5	i++
6	7	3	3 < 5	j++
6	8	7	7 > 5	i++
7	8	4	4 < 5	j++
7	9	11	11 > 5	i++
8	9	7	7 > 5	i++
9	9			

i==j → j++

int i=0, j=1

while (j<=n)

arr[j] - arr[i]

<k : j++;

=k : count++; i++; j++;

>k : i++;

if(i==j) j++;

T.C: O(n) S.C: O(1)

i==j

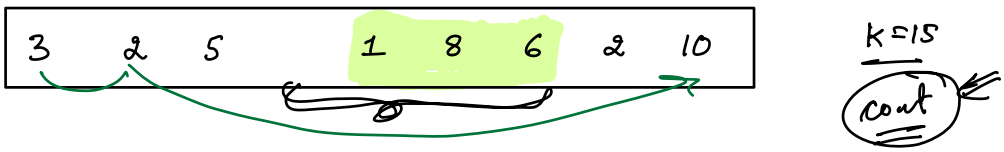
j

i = n-2
j = n-1

Q3 Given an array of +ve integers, find count of subarray with sum = k.

contiguous

Total = $\frac{n*(n+1)}{2}$



Brute: → find all possible subarrays ≡ $O(n^2)$

prefix sum

l → r

pf[r] − pf[l−1] = k

cnt pairs i, j

How to consider subay starting from 0?

```
if ( pf[i] == k)
    cnt++;
```

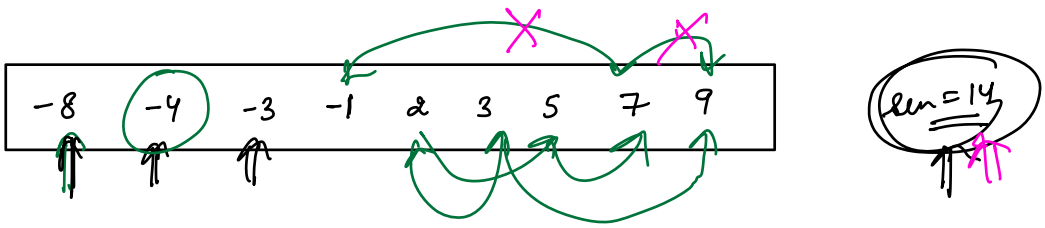
where cnt the pf sum

T.C: O(n)

S.C: O(n) – pf sum

Q4 given a sorted array, find triplets (i, j, k) (distinct)

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



B.F → consider all possible triplets → O(n3)

a + b + c = sum

a = -8

b + c = sum + 8 → two pointers

b + c = 22

a = -4

b + c = sum - (-4)

= 18

for(i = 0 → n)

a = arr[i];

apply two pointer

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

i+1 → n-1

4sum

a + b + c + d = sum

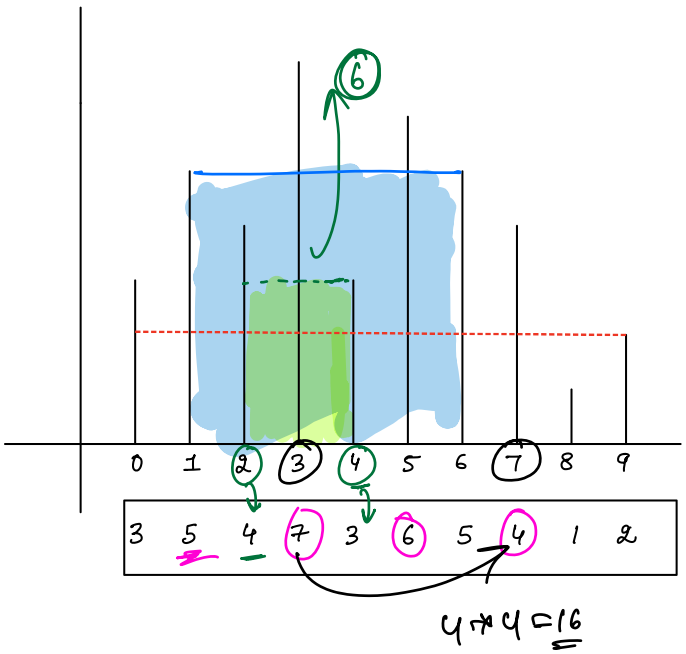
Ksum

T.C: O(n^2)

$$\begin{cases} a+b-c=k \\ a-b+c=k \end{cases}$$

Qs given N array elements → height of walls

Pick any 2 walls such that max water accumulated b/w walls.



0-9 = 9*2=18

maximum

18

0-7 = 3*7=21

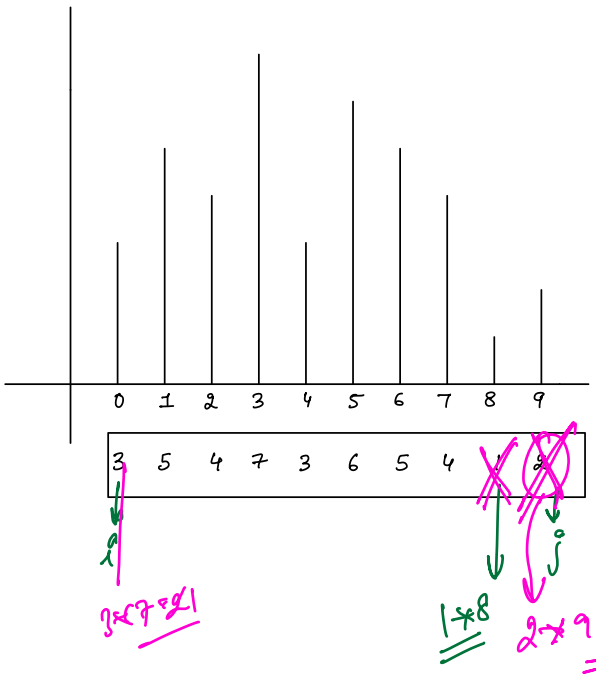
2-7 = 4*5=20

1-6 = 5*5=25

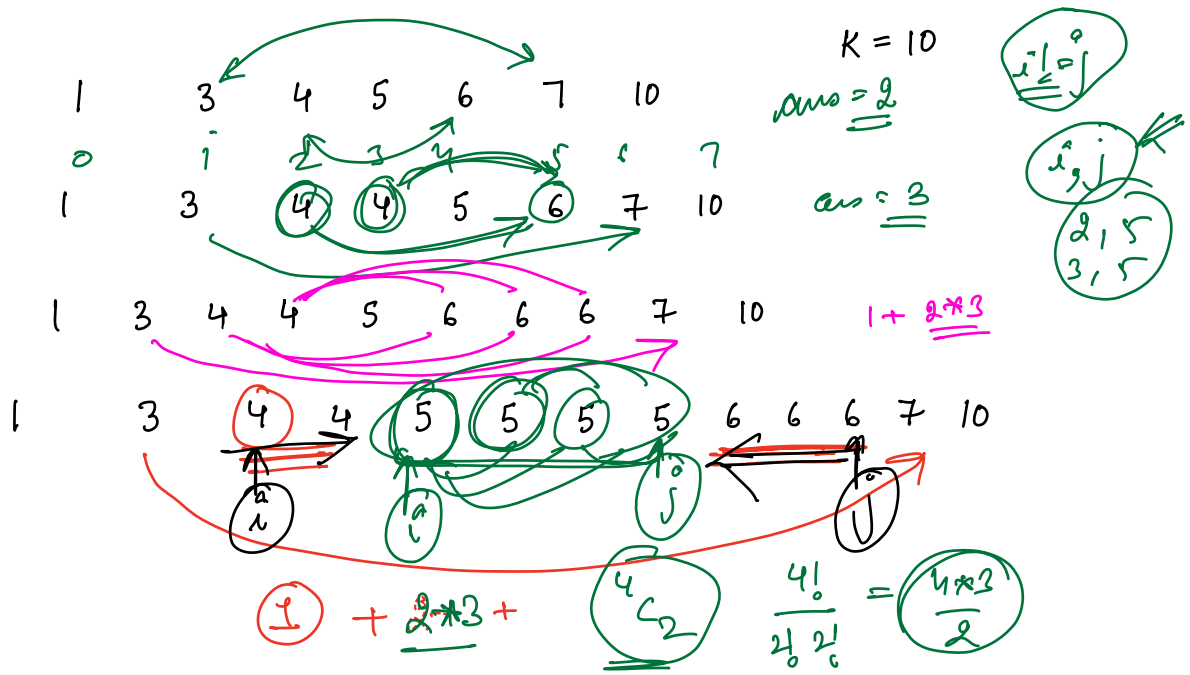
3-5 = 2*6=12

i-j ⇒ (j-i) * min(arr[i], arr[j])

Brute: consider all pairs → $O(n^2)$




```
i=0, j=n-1
ans = max(ans, water)
if( arr[i] < arr[j] )
    i++;
else
    j--;
```



$$nC_2 = \frac{n*(n-1)}{2}$$

```
if (arr[i]+arr[j]==k)
{
    if (arr[i] == arr[j])
    {
        cnt = j - i + 1;
        ans += cnt * (cnt-1) / 2;
    }
    else
    {
        cnt = 0;  int x = arr[i];
        while (i<j && x==arr[i])
        {
            cnt++;
            i++;
        }
        cnt1;  x = arr[j];  while (i<j && x==arr[j])
        {
            cnt1++;
            j--;
        }
        ans += cnt * cnt1
    }
}
```

Q6. Given 3 sorted arrays, A, B, C

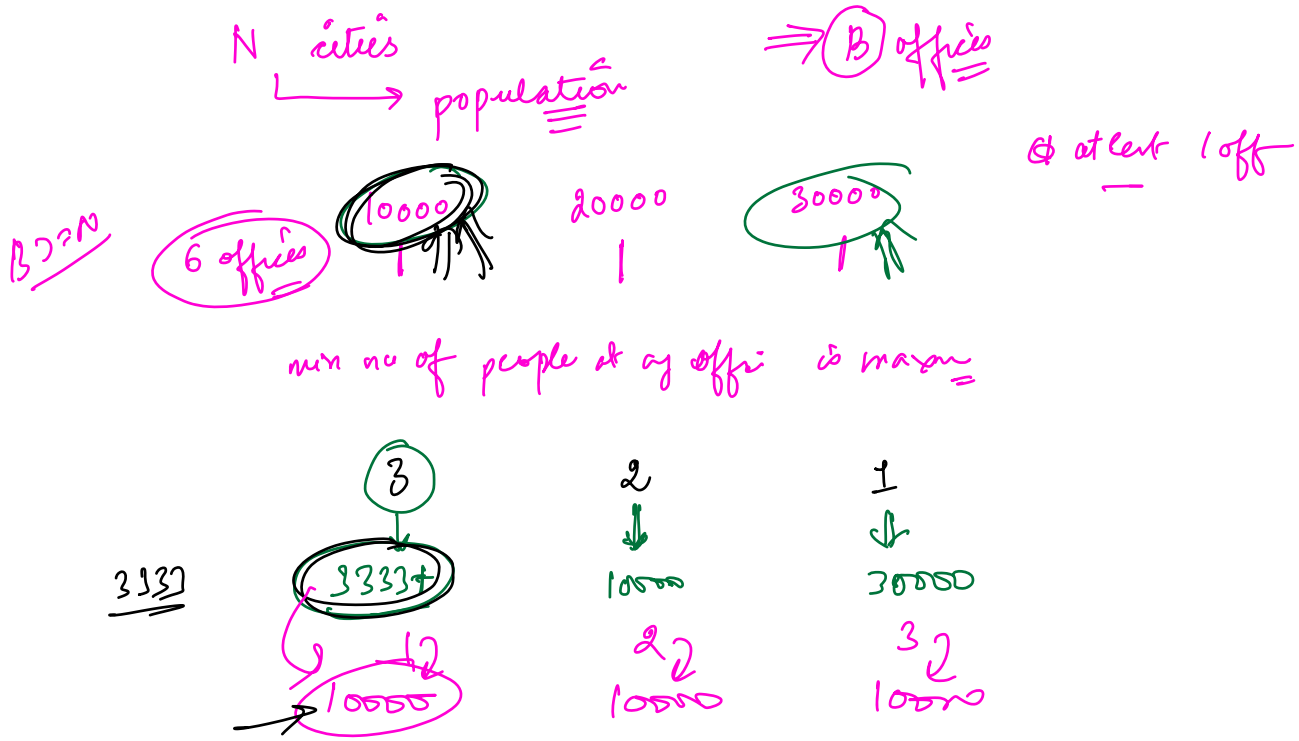
We need choose a triplet (one element from each) such that

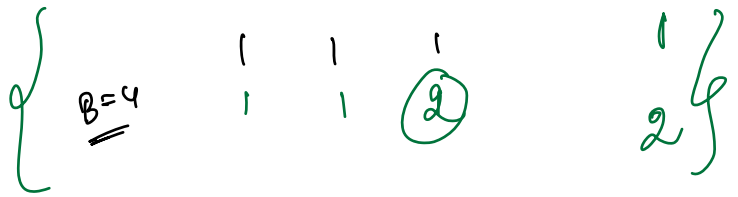
$$\max(A[i], B[j], C[k]) - \min(A[i], B[j], C[k])$$

is minimised. Find the minimum difference.

A:	1	4	5	8	10
B:	6	9	15		
C:	2	3	6	8	

Food packets destribution




B=4
2
2