2 Pointers

3, 4 → C/C++ → Significance → Index values / Position of array

1a) Given a sorted array, which contains only distinct elements, check if there exists (i,j) such that $arr[i] + arr[j] = k$ && $(i \neq j)$. $k = 19$

Ex: arr[] :

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

a = -3, b = 20

Solutions

1) check all pairs $[i, j]$ $i = 0 - N-1$, $j = i+1 - N-1$, $arr[i] + arr[j] == k$

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

2) BS w. two BS?

$i = 0 - N-1$

$b = k - arr[i]$

// Apply BS from $(i+1, N-1)$ & search for b

TC: $O(N(\log N))$ SC: $O(1)$

Same as Ex [2 +]

P_1	P_2	sum
0	9	23 > 19 // Reduce sum P_2--
0	8	15 < 19 // Increase sum P_1++
1	8	18 > 19 // Reduce P_2--
1	7	14 < 19 P_1++
2	7	15 < 19 P_1++
3	7	19 = 19 [return True]

Why the above logic works?

0, 9 → 22 > 19 // we discard index 9

-3, 25

0, 8 → 15 < 19 // we discard index 0

-3, 18

Note: In above logic at every step we are neglecting couple of pairs without checking them

Pseudocode:

```
bool compare(int* arr[], int N, int k){
    int p1 = 0, p2 = N-1, int cnt = 0
    while( p1 < p2 ){
        ① int sum = arr[p1] + arr[p2];
        ② if( sum > k ){ p2--
        ③ else if( sum < k ){ p1++
        ④ else {
            [cnt++;
            p1++; p2--;
        }
    }
    return cnt;
}
```

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

If elements repeating, calculate how many different (i,j) are there

arr[]: 1 4 4 5 5 5 5 6 6 11 k = 10

→ 4 pairs, → 6 pairs } 10 pairs

arr[i] + arr[j] == 10

Number of 4's = 2 } Total cut = 4 = $2*2$

Number of 6's = 4 } Total cut = $\binom{4}{2} = 6$

Modified else

```
else {
    if( arr[i] != arr[j] ){
        // Iterate & count how many arr[i] & arr[j]
        int c1 = 0, int j = p1;
        while( arr[j] == arr[i] ){ c1++; p1++; }
        j = p2, int c2 = 0
        while( arr[j] == arr[p2] ){ c2++; p2--; }
        cnt += c1 * c2;
    }
    else {
        // arr[p1] = arr[p2] all elements p1 to p2 are same, b/c array is sorted
        int c = (p2 - p1 + 1)
        cnt += ((c)(c-1))/2
        break;
    }
}
```

5 ⟷ 10, 10 ⟷ 10

2a) Given a sorted array, which contains distinct elements, check if there exists a pair (i,j) such that $(i \neq j)$ & $arr[i] - arr[j] == k$. $k = 5$

Ex: arr[] :

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

Solutions

1) All pairs with single iteration

TC: $O(N^2)$ SC: $O(N)$

2)

a = arr[i]

b = a + k

// Apply BS

TC: $O(N \log N)$, SC: $O(1)$

Initialization of pointers:

1) 0, N-1 *

P_1	P_2
0	9 → 28 > 5 * decrease diff

P_1++ diff decrease, P_2-- diff decrease } we cannot choose

2) N/2, N/2+1 *

P_1	P_2
4	5 → 8 - 6 = 2 < 5 Increase

P_1-- Inc decrease, P_2++ Inc decrease } we cannot choose

3) 0, N/2 *

P_1	P_2
0	4 → 9 > 5 decrease

P_1++ diff decrease, P_2-- diff decrease } we cannot choose

4) 0, 1 ✓

P_1	P_2
0	1 → 3 < 5 Increase (P_2++)
0	2 → 4 < 5 Increase (P_2++)
0	3 → 6 > 5 Decrease (P_1++)
1	3 → 3 < 5 Increase (P_2++)
1	4 → 5 == 5 Matching

P_1++ Decrease

P_2++ Increase

```
bool diff(int arr[], int N, int k){
    int p1 = 0, p2 = 1; k = abs(k)
    while( p2 < N ){
        int val = (arr[p2] - arr[p1])
        if( val > k ){
            p1++; if( p1 == p2 ){ p2++; }
        }
        else if( val < k ){
            p2++;
        }
        else {
            return True;
        }
    }
    return False;
}
```

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

Ex: 2 6 9 k = 3

P_1	P_2
0	1 → 4 > 3 P_1++
1	1 → Both same (P_2++)
1	2 → 3 == 3 [return True]

If k means P_2 will always be greater than P_1

arr[P_2] - arr[P_1] > 0 always

If k < 0, will above code work [NO]

If this covers a pair with diff = 10

Can I say there will also exist a pair with diff = -10

Based on above points, can I always make k = abs(k)

Steps to follow with 2 Pointers

1) Initialize the pointers, in such a way a decision can be made at every step.
2) Updation of pointers
3) When to stop

3a) Given an array of +ve Integers check if there exists a subarray with sum = K

Ex: arr[] :

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

(K = 15), (K = 19)

pf[] : 3 5 10 10 11 19 25 27 37

If I subtract pf[6] - pf[2] = sum of subarray of all elements [3 - 6]

Hint: Difference between any 2 pf[] elements is a subarray sum.

Solution:

1) Calculate pf[] & check if there exists any 2 elements with diff k.

TC: $O(N+N)$ SC: $O(1)$ Using same array

Note: With above logic we will get subarray sums starting with 0 correct.

// To solve above thing 2 ways

1) Add '0' at start of pf[]
2) Iterate on pf[] & check if any of the element is k.

4a) Given an distinct sorted array elements, arr[] represents length. Find number of rectangles whose area is less than k. k = 85

Note: Pair (i,j) = (j,i)

Explanation

arr[2, 3, 5] k = 10

(2,3) : 6 < 10

(2,5) : 10 < 10

(3,5) : 9 < 10

(3,3) : 15 > 10

(5,5) : 25 > 10

cnt = 3

Ex: 2 3 4 5 8 9 12 14 15 16 19 20

P_1 → 2, P_2 → 20, 285 > 85, < 80

P_1	P_2	Area	cnt
0	11	40 < 85	+ P_2 - P_1 + 1 + 11 } P_1++
1	11	60 < 85	+ 10 P_1++
2	11	80 < 85	+ 9 P_1++
3	11	140 ≥ 85	P_2--
3	10	133 ≥ 85	P_2--
3	9	122 ≥ 85	P_2--
3	8	105 ≥ 85	P_2--
3	7	98 ≥ 85	P_2--
3	6	84 < 85	+ 4 P_1++
4	6	96 ≥ 85	P_2--
4	5	80 < 85	+ 2 P_1++
5	5	81 < 85	+ 1 P_1++

6 ⟷ 5

We stop

```
int countRect(int* arr[], int N, int k){
    int p1 = 0, p2 = N-1, cnt = 0
    while( p1 < p2 ){
        int area = arr[p1] * arr[p2];
        if( area < k ){
            cnt += (p2 - p1 + 1)
            p1++
        }
        else {
            p2--
        }
    }
    return cnt;
}
```

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

Note: If (i, j) + (j, i)

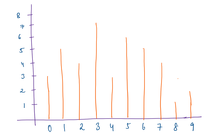
Simply return 2*cnt *

arr[] = [2 3 5] k = 11 → 2*5 < 11

cnt += (p2 - p1 + 1) * 2 - 1

= (2 - 0 + 1) * 2 - 1 = 5

(2, 2)
(2, 3) (3, 2)
(2, 5) (5, 2)

5a) Given N array elements, where each arr[i] represents height of each building, calculate Area of water accumulated on a rainy day

Ex: arr[] : 3 5 4 7 3 6 5 4 1 2