Todays Content

1. 2 Pointers Intro
2. a+b=k
3. count pairs a+b=k
4. a-b=k
5. Max water between any 2 buildings

2 Things: Chat/ Unmute

optim1: If things are clear; Just send reaction
optim2: If Doubt raise hand:

a. Make you contributor &
you can unmute & Make you viewer

2 Pointer

1. C/C++ pointers *
2. int variable = index:

2 pointers = 2 variable index:
3 pointers = 3 variable index:

Q1. Given ar[N] distinct sorted elements, check if there exists a pair(i, j) such that ar[i] + ar[j] = k && i != j

Ex:

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ ar[7] = \{ & 3 & 7 & 8 & 11 & 14 & 19 & 20 \} \end{array}$$

k = 25 : ar[3] + ar[4] = Yes True.

Ideas

a) For all pairs check if their sum == k.

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

b) Optimise:

1) HashMap/HashSet : TC: $O(N)$ SC: $O(N)$

2) Binary Search : ? TC: O( ) SC: O( )

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ ar[7] = \{ & \underset{✓}{3} & \underset{✓}{7} & \underset{✓}{8} & \underset{✓}{11} & [14 & 19 & 20\}] \end{array}$$

k =

a + b = 25 Note: Fix a ; Search for b = ;

3 b = 22: search for b: from [1 6] ✗

7 b = 18: search for b: from [2 6] ✗

8 b = 17: search for b: from [3 6] ✗

11 b = 14: search for b: from [4 6] ✓ : Return True

Idea: Fix every ar[i] as a a

Search for b from [i+1.. N-1] using Binary Search

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

Worst: logN

Ideas: 2 Pointer : { 2 variable = index value }

ar[11] = { -3 0 1 3 6 8 11 14 18 25 } $\boxed{k = 17}$

	0	1	2	3	4	5	6	7	8	9
ar	-3	0	1	3	6	8	11	14	18	25
	~~P_1~~	~~P_1~~	~~P_1~~	P_1				P_2	~~P_2~~	~~P_2~~

P_1	P_2	$ar[P_1] + ar[P_2]$	Sum	update
0	9	-3 + 25 = 22 > 17 :	↓	P_2--
0	8	-3 + 18 = 15 < 17 :	↑	P_1++
1	8	0 + 18 = 18 > 17 :	↓	P_2--
1	7	0 + 14 = 14 < 17 :	↑	P_1++
2	7	1 + 14 = 15 < 17 :	↑	P_1++
3	7	3 + 14 = 17 == 17 ; return True;		

obs: for each iterate we will skip 1 element. In Total N elements. Total iterating = N

```
bool checksum(int ar[ ], int k){
    int N = ar.length;
    int P1 = 0, P2 = N-1;
    while( P1 < P2){
        if( ar(P1) + ar(P2) == sum){
            return True;
        }
        if( ar(P1) + ar(P2) < sum){
            P1++;
        }
        else{
            P2--;
        }
    }
}   return false;
```

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

Why logic works?

sorted

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \\ ar[5] = \{ & \cancel{3} & 9 & 10 & 14 & \cancel{18} \ \} \quad k = 19 \end{array}$$

$P_1 \rightarrow P_1$ $P_2 \leftarrow P_2$

$ar[P_1]$		$ar[P_2]$	
3	+	18	= 21 > 19 : dec P_2--; ?

$\left.\begin{matrix} 9 \\ 10 \\ 14 \end{matrix}\right\} + 18 > 21$ Note: 18 cannot be part of ans.

3 + 14 = 17 < 21 : Inc P_1++; ?

$3 + \left\{\begin{matrix} 9 \\ 10 \end{matrix}\right\} < 21$ Note: 3 cannot be part of ans

If arr[] is not sorted will logic work? Nope ✗

Given a sorted arr[], where elements repeat count no: of pairs (i, j)

such that ar[i] + ar[j] = k. (i != j)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Ex: ar[ ] = { 3 5 5 5 6 6 8 8 8 8 10 10 11 11 11 11 17 } ans = 22

k = 16 P1 P2

Pairs:

(1 12) (2 12) . . (6 7)
(1 13) (2 13) . . (6 8)
(1 14) (2 14) . . (6 9)
(1 15) (2 15) . .

22 pairs:

Formula:

How many we can select 2 from N = $\frac{(N)(N-1)}{2}$

How many ways we can select 2 from 4 = 6

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Ex: ar[ ] = { ~~3~~ 5 5 5 6 6 8 8 8 8 10 10 11 11 11 11 ~~17~~ } ans = 22

k = 16 P1 P1 → P1 → P1 P2 ← P2 ← P2 P2

C_5 = 0 +1 +1 +1 = 3 | +1 +1 = 2 | 6 pairs | 2 = +1 +1 | 4 = +1 +1 +1 +1 C_{11} = 0

P1	P2	ar[P1] + ar[P2]	Sum	Update
0	16	3 + 17 = 20 > 16	↓	P2--
0	15	3 + 11 = 14 < 16	↑	P1++
1	15	5 + 11 = 16 = 16		
4	11	6 + 10 = 16 = 16		
6	9	8 + 8 = 16 = 16		

(1 15) Q: How many pairs with 5, 11.

Q1: How many 5's C_5 = 3 } = 12 pairs
Q2: How many 11's. C_{11} = 4

(4 11) Q: How many pairs with 6, 10.

Q1: How many 6's C_6 = 2 } = 4 pairs
Q2: How many 10's. C_{10} = 2

(6 9) Q: if ar[P1] == ar[P2] :

Q: How many ele: [P1 P2] = C = P2 - P1 + 1 = 4. } 6 pairs

& Break code & return ans;

```
long countSum(int ar[ ],int k){ TC: O(N) SC: O(1)
    int N=ar.length, P1=0, P2 = N-1;
    long ans=0;
    while(P1<P2){
        if( ar(P1) + ar(P2) > k){ // dec sum
            P2--; // 1 iteration : 1 ele is skipped
        }
        else if( ar(P1)+ar(P2) < k){ // inc sum
            P1++; // 1 iteration : 1 ele is skipped
        }
        else{ // ar(P1)+ar(P2) == k:
            if ( ar(P1) != ar(P2)){
                int t = ar(P1], cl = 0, cr=0;
                while( ar(P1] == t){ // count no:f ar[P1]
                    cl++; // count      1 iteration : 1 ele is skipped
                    P1++; // inc P1++;
                }
                t = ar(P2);
                while(ar(P2)==t){ // count no:f ar[P2]
                    cr++; //count;      1 iteration : 1 ele is skipped
                    P2--;
                }
                ans = ans + cl*cr;
            }
            else{ //ar(P1) = ar(P2)
                long c = P2-P1+1;
                ans = ans + c*[c-1]
                            -------
                               2
                break; // No more elements
            }
        }
    }
    return ans;
}
```

Q2: Given ar[N] sorted elements, check if there exists a pair (i,j) such that ar[j] - ar[i] = k && i != j && k >= 0 : If k is anything?

	0	1	2	3	4	5	6	7	8	9	
ar[10] = {	-3	0	1	3	6	8	11	14	21	25 }	k = 5

↳ ar[4] - ar[2] = 5 : Return True;

Ideas

1. Generate all pairs. TC: $O(N^2)$
2. Using HashMap: TC = $O(N)$ SC: $O(N)$
3. Using Binary Search: TC = $O(N \log_2 N)$

	0	1	2	3	4	5	
ar[ ] = {	1	2	4	5	6	12 }	k = 10

↳ ar[5] - ar[2] = 10 : Return True;

Idea 4: 2 Pointers: Initialize in such a way we can update pointers without any confusion.

	0	1	2	3	4	5	6	7	8	9	
ar[10] = {	-3	0	1	3	6	8	11	14	21	25 }	k = 5
								P_1	P_2		

Case-2: $P_1 = 0$ $P_2 = 1$ ✓

P_1	P_2	: ar[P_2] - ar[P_1]
0	1	: 0 - (-3) = 3 < 5 : Inc P_2++
0	2	: 1 - (-3) = 4 < 5 : Inc P_2++
0	3	: 3 - (-3) = 6 > 5 : Dec P_1++;
1	3	: 3 - 0 = 3 < 5 : Inc P_2++
1	4	: 6 - 0 = 6 > 5 : Dec P_1++;
2	4	: 6 - 1 = 5 = 5 : Return True;

Case-4: $P_1 = N-2$ $P_2 = N-1$ ✓

P_1	P_2	: ar[P_2] - ar[P_1]
8	9	: 25 - 21 = 4 < 5 : Inc P_1--
7	9	: 25 - 14 = 11 > 5 : Dec P_2--

Case-1: $P_1 = 0$ $P_2 = 9$ *

P_1	P_2	: ar[P_2] - ar[P_1]
0	9	: 25 - (-3) = 28 > 5 ↓ Dec diff:

Dec Diff: Ambiguity / Confusion / We cannot
P_1++; 25 - 0 = 25 decide which pointer
P_2--; 21 - (-3) = 24 to update *
Wrong initialization.

Case-3: $P_1 = N/2$ $P_2 = N/2 + 1$ *

P_1	P_2	: ar[P_2] - ar[P_1]
4	5	: 8 - 6 = 2 < 5 : ↑ Inc Diff

Inc Diff:

P_1--; or P_2++ : Both will Inc Diff.

/ Confusion We cannot decide which pointer update Wrong initialization

```
bool diff(int ar[], int k){
    k = Math.abs(k);
    int N = ar.length;
    int P1 = 0, P2 = 1;
    While( P2<N ){
        if( ar(P2) - ar(P1) == k){
            return True;
        }
        if( ar(P2) - ar(P1) < k){
            //Inc Diff
            P2++;
        }
        else{ // Dec Diff
            P1++;
            if(P1==P2){P2++;}
        }
    }
    return false;
}
```

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

Edge Case

$$ar[3] = \begin{array}{cccc} 0 & 1 & 2 & \boxed{3} \\ \{4 & 10 & 13\} & \downarrow \\ & & & P_2 \end{array} \quad k = 10$$

Case 3 $P_1 = 0$ $P_2 = 1$

P_1	P_2	: ar[P_2] - ar[P_1]
0	1	: 10 - 4 = 6 < 10 : Inc Diff : P_2++
0	2	: 13 - 4 = 9 < 10 : Inc Diff : P_2++
0	3	: Stop.

$$ar[3] = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ \{4 & 10 & 13 & 13\} \\ & & P_1 & P_2 \end{array} \quad k = 0$$

Case 3 $P_1 = 0$ $P_2 = 1$

P_1	P_2	: ar[P_2] - ar[P_1] Diff
0	1	: 10 - 4 = 6 > 0 : Dec Diff : P_1++
1	1	: if(P_1==P_2){P_2++;}
1	2	: 13 - 10 = 3 > 0 : Dec Diff : P_1++
2	2	: if(P_1==P_2){P_2++;}
3	2	: 13 - 13 = 0 = 0 : Return True;

If k<0: Make it +ve & search for pair

$$ar[3] = \begin{array}{ccc} 0 & 1 & 2 \\ \{4 & 10 & 13\} \end{array} \quad k = 3$$

Pair k=3 : ar[2] - ar[1] = 3 ✓

Pair k=-3 : ar[1] - ar[2] = -3 ✓

obs:

if ar[i] - ar[j] = k ⟺ ar[j] - ar[i] = -k

Con: If pair with diff k exists ⟺ pair with -k also exists