Todays Content.

a) 2 Pointers

b) 4 Interview Problems based on them

Pointer:

a) C/C++ *

b) Variable storing index of array

c) 2 pointer = 2 variables

↳ Extend 3 pointer

4 pointers

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

Eg:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| ar[7] = { | 3 | 7 | 8 | 11 | 14 | 19 | 20 } |

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

Ideas

a) Check all pairs sum & compare == k. TC: O(N^2) SC: O(1)

b) Optimise

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

2) Using Binary Search?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|---|
| ar[7] = { | 3 | 7 | 8 | 11 | 14 | 19 | 20 } | k = 25 |

a + b = 25 Note: Fix a, calculate b & search 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] ✓ Target found

```
boolean pair(int ar[], int k){ TC: O(NlogN) SC: O(1)
    int N = ar.length;
    for(int i=0; i<N; i++){
        int b = k-ar[i];
        Search b in ar[] from [i+1 N-1]
        int l = i+1, h = N-1;
        while(l<=h){ // search b in range
            TODO?
}}}
```

Ideas: 2 Pointer

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ar[11] = { | -3 | 0 | 1 | 3 | 6 | 8 | 11 | 14 | 18 | 25 } | k = 17 |
| | * | * | * | $P_1$ | | | | $P_2$ | * | * | |

| $P_1$ | $P_2$ | ar[$P_1$] + ar[$P_2$] | |
|---|---|---|---|
| 0 | 9 | 22 > 17 | dec $P_2$-- |
| 0 | 8 | 15 < 17 | inc $P_1$++ |
| 1 | 8 | 18 > 17 | dec $P_2$-- |
| 1 | 7 | 14 < 17 | inc $P_1$++ |
| 2 | 7 | 15 < 17 | inc $P_1$++ |
| 3 | 7 | 17 == 17 return true | |

Note: After each iteration we neglect 1 element.

At max we have N iterations.

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

TC: After each iteration we neglect 1 element, hence TC: O(N) SC: O(1)

Why logic works?

| | 0 | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|---|
| ar[5] = { | ~~X~~ | 9 | 10 | 14 | ~~18~~ } | k = 19 |
| | | $P_1$ | | $P_2$ | | |

ar[$P_1$] ar[$P_2$]

3 + 18 = 21 > 19 : $P_2$--?

9, 10, 14 + 18 > 19 Note: Can 18 be 1 one of the element : No

Discard element

3 + 14 = 17 < 19 : P1++?

3 + {9, 10} < 19 Note: Can 3 be 1 one of the element : No

Discard element

If arr[] is not sorted will logic work?

* Wont work.

Q2: Given ar[N] sorted elements, check if there exists a pair (i, j)

such that ar[j] - ar[i] = k && i != j && k can be 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[5] - ar[3] = 5 True |
| | | $P_1$ | | $P_2$ | | | | | | | |

Case-3: $P_1 = 0$ $P_2 = 1$ ✓

| $P_1$ | $P_2$ | : 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 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: $P_2$-- |
| 7 | 8 | : so on | |

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

| $P_1$ | $P_2$ | : ar[$P_2$] - ar[$P_1$] |
|---|---|---|
| 0 | 9 | : 25 - (-3) = 28 > 5 |

Dec diff:

If $P_1$++ : Diff dec } We cannot
If $P_2$-- : Diff dec } decide *

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

| $P_1$ | $P_2$ | : ar[$P_2$] - ar[$P_1$] |
|---|---|---|
| 5 | 6 | : 11 - 8 = 3 < 5 |

Inc diff:

If $P_1$-- : Diff inc } We cannot
If $P_2$++ : Diff inc } decide *

```
bool diff(int ar[], int k){
    // Case-3 Code:
    if(k<0){ k=k*-1; } // It works all
    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){
            P1++ //dec diff
            if(P1==P2){ P2++; }
        }
        else{ // inc diff
            P2++;
        }
    }
    return false;
}
```

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

If k<0:

ar[3] = {4, 10, 13} k = -3

Pair k=-3 : ar[1]-ar[2] = -3

Pair k=3 : In above eqn mul -1

(ar[1]-ar[2])*-1 = -3*-1

ar[2]-ar[1] = 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

Edge Case

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| ar[3] = | 4 | 10 | 13 | k=8 |
| | | P1 | | P2 |

Case 3 P1=0 P2=1

| P1 | P2 | : ar[P2]-ar[P1] |
|---|---|---|
| 0 | 1 | : 10-4 = 6 < 8 : Inc diff: P2++ |
| 0 | 2 | : 13-4 = 9 > 8 : Dec diff: P1++ |
| 1 | 2 | : 13-10 = 3 < 8 : Inc diff: P2++ |
| 1 | 3 → P2 out of bounds Stop | |

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| ar[3] = | 4 | 10 | 13 | 13 k=0 |
| | | | P1 | P2 |

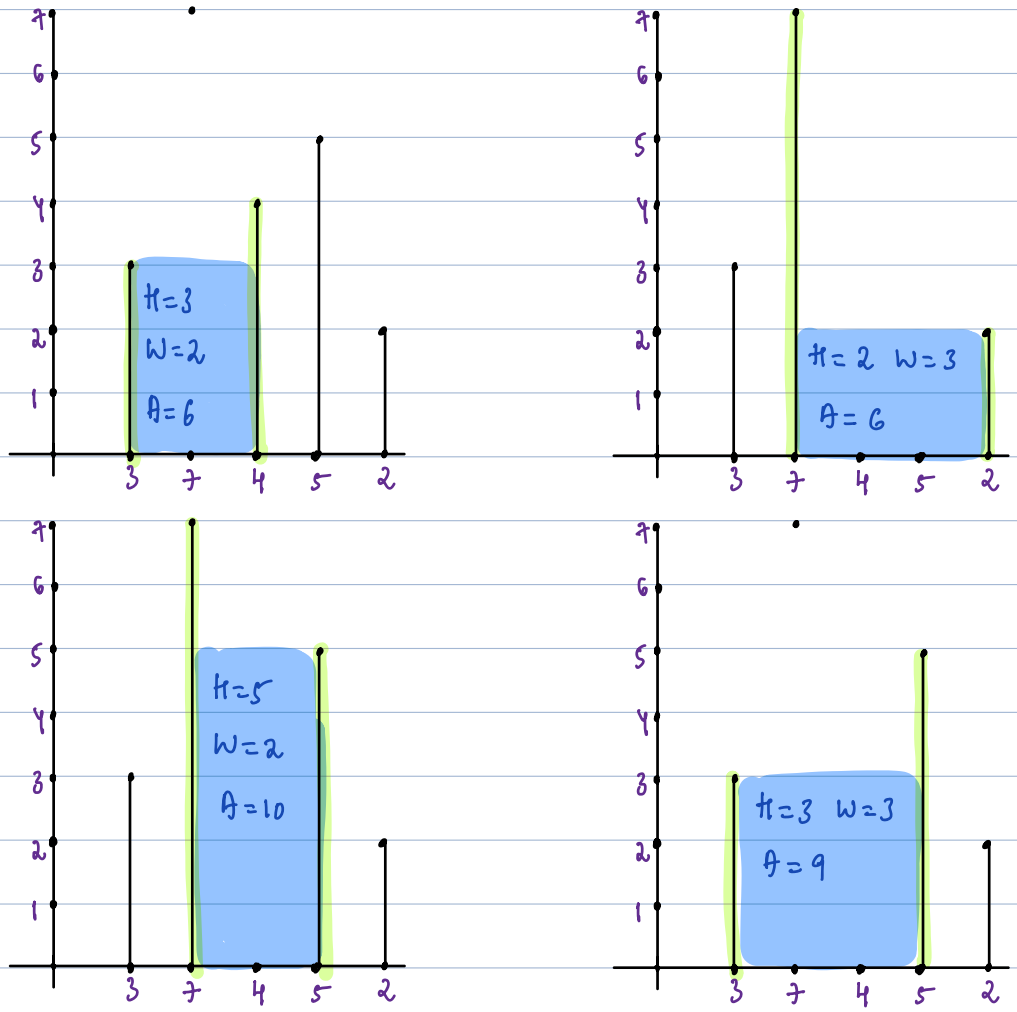
Case 3 P1=0 P2=1

| P1 | P2 | : ar[P2]-ar[P1] |
|---|---|---|
| 0 | 1 | : 10-4 = 6 > 0 : Dec diff: P1++ |
| 1 | 1 | : If P1==P2 : P2++ |
| 1 | 2 | : 13-10 = 3 > 0 : Dec diff: P1++ |
| 2 | 2 | : If P1==P2 : P2++ |
| 2 | 3 | : 13-13 == 0 : return true |

30) Given ar[N] ele, ar[i] represents height of each wall.
Find Max water accumulated between any 2 walls?

Note: Between 2 walls width = 1

Ex: ar[5] = { 3 7 4 5 2 } ans = 10



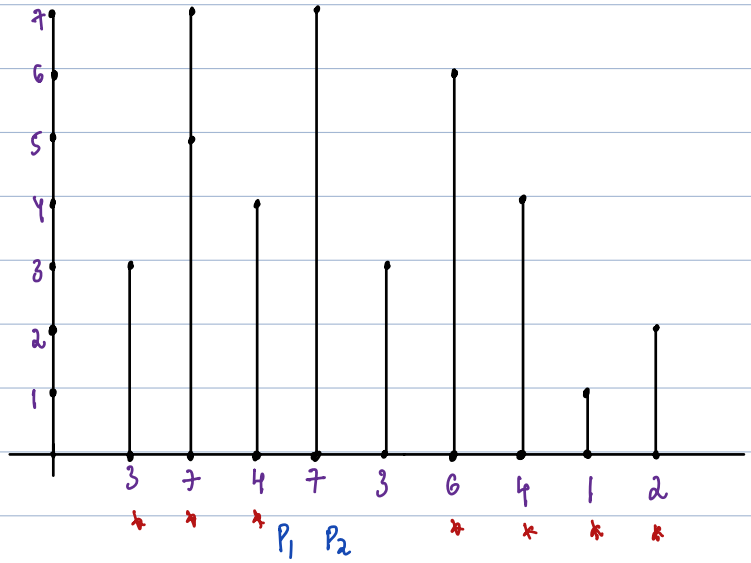

Idea1: Fr all pair walls get water, & calculate overall max:

TC: O(N²) SC: O(1)

Idea2: Calculate water between 1st Max & 2nd Max *

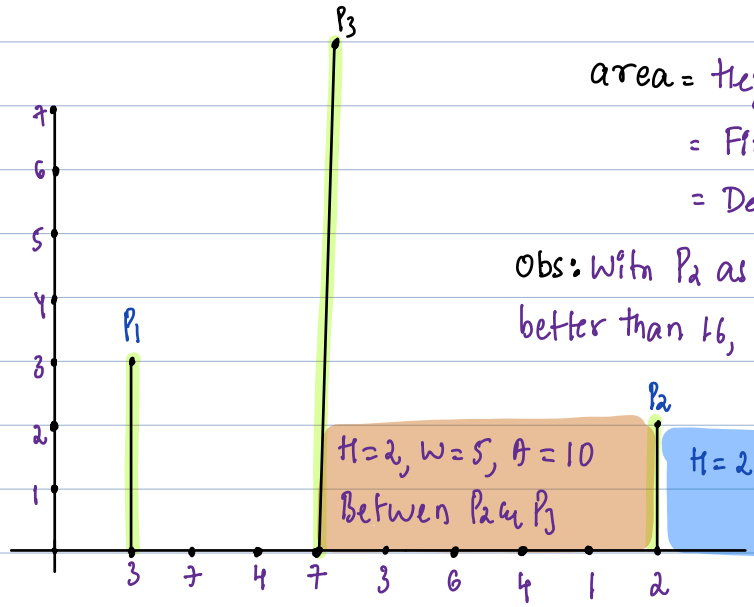
Ideal: Using 2 pointer?

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| ar[10] = { | 3 | 7 | 4 | 7 | 3 | 6 | 4 | 1 | 2 } |
| | $P_1$ | | | | | | | | $P_2$ |




| $P_1$ | $P_2$ | $H = \min(ar[P_1], ar[P_2])$ | $W = P_2 - P_1$ | Area : Move min height wall | | ans = 16 |
|---|---|---|---|---|---|---|
| 0 | 8 | $H = \min(3, 2) = 2$ | $W = 8$ | $A = 16$ | $P_2$ --; | ans = 16 |
| 0 | 7 | $H = \min(3, 1) = 1$ | $W = 7$ | $A = 7$ | $P_2$ --; | 16 |
| 0 | 6 | $H = \min(3, 4) = 3$ | $W = 6$ | $A = 18$ | $P_1$++ | 18 |
| 1 | 6 | $H = \min(7, 4) = 4$ | $W = 5$ | $A = 20$ | $P_2$ --; | 20 |
| 1 | 5 | $H = \min(7, 6) = 6$ | $W = 4$ | $A = 24$ | $P_2$ --; | 24 |
| 1 | 4 | $H = \min(7, 3) = 3$ | $W = 3$ | $A = 9$ | $P_2$ --; | 24 |
| 1 | 3 | $H = \min(7, 7) = 7$ | $W = 2$ | $A = 14$ | $P_1$++; | 24 |
| 2 | 3 | $H = \min(4, 7) = 4$ | $W = 1$ | $A = 4$ | $P_1$++ | 24 |
| 3 | 3 | Stop | | ans = max A = 24 | | |

**Discard:** Move pointer with min height because




area = height * width
= Fixed * decrease
= Decrease

Obs: With $P_2$ as one of wall can we get better than 16, update ans & discard $P_2$ because we cannot get a better ans

```
int water(int H[]){ TODO   TC: O(N)  SC: O(1)
    int ans = 0, N = H.length;
    int P1 = 0, P2 = N-1
}
```

Note: In each iteration we neglect 1 wall, We have N walls, hence N iterations

48) Given 3 sorted arrays A[] B[] C[] of size N

Find min value of Below expression

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

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| i A[4] = { | 3 | 14 | 16 | 23 } | ans=3 |
| j B[4] = { | -6 | 23 | 24 | 30 } | |
| k C[4] = { | -15 | 15 | 26 | 31 } | |

| i | j | k | max(A[i] B[j] C[k]) - min(A[i] B[j] C[k]) |
|---|---|---|---|
| 0 | 0 | 0 | max(3 -6 -15) - min(3 -6 -15) = 3 - (-15) = 18 |
| 1 | 1 | 1 | max(14 23 15) - min(14 23 15) = 23 - 14 = 9 |
| 3 | 3 | 3 | max(23 30 31) - min(23 30 31) = 31 - 23 = 8 |
| 3 | 1 | 2 | max(23 23 26) - min(23 23 26) = 26 - 23 = 3 |

Idea: Consider all triplets, & for all triplets calculate its value & get overall min.

TC: O(N3) SC: O(1)

Idea2:

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| A[4] = { | 3 | 14 | 16 | 23 | } 4 |
| | | | | | $P_1$ |
| B[4] = { | -6 | 23 | 24 | 30 | } |
| | | | $P_2$ | | |
| C[4] = { | -15 | 15 | 26 | 31 | } |
| | | | $P_3$ | | |

| A[] | B[] | C[] | max - min | val |
|---|---|---|---|---|
| 3 | -6 | -15 | 3 - (-15) = | 18 |
| 14 | 23 | | 23 - (-15) = | 41 |
| 16 | 24 | | 14 - (-15) = | 29 |
| 23 | 30 | | 23 - (-15) = | 38 |

max: inc - fixed = ↑

Con: With $P_3$ at -15, can we get less than 18 & discard

Updater pointer with min value ?

| $P_1$ | $P_2$ | $P_3$ | max(A[$P_1$] B[$P_2$] C[$P_3$]) - min(A[$P_1$] B[$P_2$] C[$P_3$]) : |
|---|---|---|---|
| 0 | 0 | 0 | max(3, -6, -15) - min(3, -6, -15) = 3-(-15) = 18 |
| 0 | 0 | 1 | max(3, -6, +15) - min(3, -6, 15) = 15-(-6) = 21 |
| 0 | 1 | 1 | max(3 23 15) - min(3 23 15) = 23-3 = 20 |
| 1 | 1 | 1 | max(14 23 15) - min(14 23 15) = 23-14 = 9 |
| 2 | 1 | 1 | max(16 23 15) - min(16 23 15) = 23-15 = 8 |
| 2 | 1 | 2 | max(16 23 26) - min(16 23 26) = 26-16 = 10 |
| 3 | 1 | 2 | max(23 23 26) - min(23 23 26) = 26-23 = 3 |
| 3 | 2 | 2 | max(23 24 26) - min(23 24 26) = 26-23 = 3 |
| 4 | 2 | 2 | $P_1$ == 4 // out of bounds Stop. |

min of all value = 3.

```
int MinCost(int A[], int B[], int C[]) : TODO
{
}
```

Note: Any 2 pointer

1. Initialize
2. Update pointer:
   We move away from an element, if it cannot give a better ans.

Topics:

1. TC = 3
2. Arrays = 7
3. Bits = 2
4. Hashing = 3
5. Recursion = 3
6. Sorting + = 3
7. Binary Search + 2 Pointer = 4
8. Strings = 3
9. Stacks = 4
10. LinkedList = 3
11. Tree + Trie + Greedy + BT = 8
12. Dp = 5
13. graphs = 3