Lecture :- Two pointers.

Agenda

- 2 pointers ✓
- 4 interview problems.

Class starts at 7:07 AM

Pointer

- C/C++ pointer ✗
- variable storing idx of array
  - 2 pointers = 2 variables
  - Extended 3 variables
  - 4 "
  - ⋮

Qu! Given arr[n] distinct sorted elements, check if there exists a pair (i, j) such that arr[i] + arr[j] = K and i != j.

Ex. arr[] = [0: 3, 1: 7, 2: 8, 3: 11, 4: 14, 5: 19, 6: 20], K = 25

arr[3] + arr[4] = 25, return true.

Ideas:

1.) Go to all pairs, calculate the sum and cmp with K

```
for(i=0; i<n; i++){
    for(j=i+1; j<n; j++){
        if( arr[i] + arr[j] == K){
            return true;
        }
    }
}
```

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

2.) Optimisation :- Hashmap

TC: O(n)
SC: O(n)

3.> Sorted array = Binary search

arr[] = [3 (0) | 7 (1) | 8 (2) | 11 (3) | 14 (4) | 19 (5) | 20 (6)], k = 25

i

Idea:

a	b	search from
3	25-3=22	from [1 ---- n-1]
7	25-7=18	[2 ---- 6]
8	25-8=17	[3 ---- 6]
11	25-8=14	[4 ---- 6]

return true;

```
boolean pair(int[] arr, int k) {
    n = arr.length;
    for(i=0; i<n; i++) {                    — O(n)
        int a = arr[i];
        int b = k-a;
        // Apply b-search from [i+1 ---- n-1]
        s = i+1, e = n-1;
        while(s<=e) {
            mid = s + ((e-s)/2);
            if(arr[mid] == b) {
                return true;
            }
            if(arr[mid] < b) {
                s = mid+1;
            } else {
                e = mid-1;
            }
        }
    }
    return false;
}
```

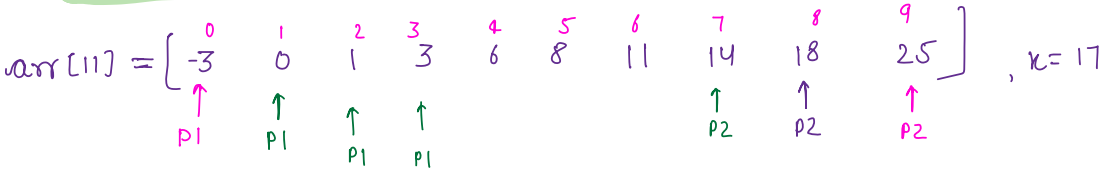
O(logn)

TC: O(nlogn)

SC: O(1)

4>

Two pointer?



P1	P2	arr[p1] + arr[p2]
0	9	-3 + 25 = 22 > 17
0	8	-3 + 18 = 15 < 17
1	8	0 + 18 = 18 > 17
1	7	0 + 14 = 14 < 17
2	7	1 + 14 = 15 < 17
3	7	3 + 14 = 17 == 17 (return true);

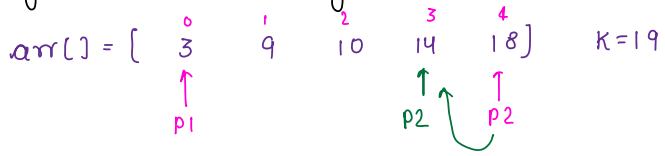
Code:

```
boolean checksum(int[] arr, int k) {
    n = arr.length;
    p1 = 0;
    p2 = n-1;
    while( p1<p2 ) {
        if(arr[p1] + arr[p2] == k) {
            return true;
        }
        if( arr[p1] +arr[p2] < k) {
            // incr p1
            p1++;
        } else {
            // decr p2
            p2--;
        }
    }
    return false;
}
```

TC: O(n)

SC: O(1)

Q Why does this logic works?



P1	P2	arr[P1] + arr[P2]
0	4 [fix]	3 + 18 = 21 > 19
1	----	9 + 18 = 27 > 19
2	---	10 + 18 = 28 > 19
3	----	14 + 18 = 32 > 19

Obs: can I get 18 as one of the element?

↳ No [discard it]

Qu2 Given arr[n] sorted elements, check if there exists a pair (i,j) such that arr[j] − arr[i] == K and i != j and ~~K>=0~~ K can be anything.

Ex: arr[] = [-3 (0) 0 (1) 1 (2) 3 (3) 6 (4) 8 (5) 11 (6) 14 (7) 21 (8) 25 (9)], K = 5

arr[5] − arr[3] = 5 return true;

arr[4] − arr[2] = 5

Case1:

	P1	P2	arr[P2] − arr[P1]	
	0	9	25 − (−3) = 28 > 5	[Wrong]
Move p2 →	0	8	21 − (−3) = 24 > 5	Diff is dec^r [Ambiguity]
Move p1 →	1	9	25 − 0 = 25 > 5	

arr[] = [-3 (0) 0 (1) 1 (2) 3 (3) 6 (4) 8 (5) 11 (6) 14 (7) 21 (8) 25 (9)], K = 5

Case2:

	P1 ($\frac{n}{2}+1$)	P2 ($\frac{n}{2}+2$)	arr[P2] − arr[P1]	
	5	6	11 − 8 = 3 < 5	[Wrong]
move p2 →	5	7	14 − 8 = 6 > 5	Diff is inc^r [Ambiguity]
move p1 →	4	6	11 − 3 = 8 > 5	

arr[] = [-3 (0) | 0 (1) | 1 (2) | 3 (3) | 6 (4) | 8 (5) | 11 (6) | 14 (7) | 21 (8) | 25 (9)] , k = 5

P1 → 0, P2 → 1

Case3:

[correct]

P1	P2	arr[P2] − arr[P1]	
0	1	0 − (−3) = 3 < 5	[P2++]
0	2	1 − (−3) = 4 < 5	[P2++]
0	3	3 − (−3) = 6 > 5	P1++
1	3	3 − 0 = 3 < 5	P2++
1	4	6 − 0 = 6 > 5	P1++
2	4	6 − 1 = 5 == 5	return true;

arr[] = [-3 (0) | 0 (1) | 1 (2) | 3 (3) | 6 (4) | 8 (5) | 11 (6) | 14 (7) | 21 (8) | 25 (9)] , k = 5

P1, P2 → 7 ; P1 → 8 ; P2 → 9

Case4:

[correct]

P1	P2	arr[P2] − arr[P1]	
8	9	25 − 21 = 4 < 5	P1--
7	9	25 − 14 = 11 > 5	P2--
7	8	21 − 14 = 7 > 5	P2--
7	7		P1-- — return false [wrong]
6	7	14 − 11 = 3 < 5	P1--
5	7	14 − 8 = 6 > 5	P2--

⋮

so on....

Code:

```
boolean diff ( int[] arr, int k) {
        n = arr.length;
        if (k < 0) { k = k * -1 };
      // case3:  p1 = 0, p2 = 1.
        p1 = 0, p2 = 1
        while (   p2 < n.          ) {
              if ( arr[p2] — arr[p1] == k) {
                    return true;
              }
              if ( arr[p2] - arr[p1] > k) {
                    p1 ++;
                    if ( p1 == p2) {
                       p2 ++;
                    }
              } eloc {
                    p2 ++;
              }
        }
    return faloc;
}
```

Condition

① arr[3] = [4 (0) 10 (1) 13 (2)] , K = 8

Case 3:

p1	p2	arr[p2] − arr[p1]	
0	1	10 − 4 = 6 < 8	p2++
0	2	13 − 4 = 9 > 8	p1++
1	2	13 − 10 = 3 < 8	p2++
1	3	⟶ out of bound	[stop]

② arr[] = [4 (0) 10 (1) 13 (2) 23 (3)] , K = 0 [p1 == p2]

Try out yourself

p1 ↑ 4 (0)

p2 | p1 ↑ 10 (1)

case 3:

p1	p2	arr[p2] − arr[p1]	
0	1	10 − 4 = 6 > 0	p1++
1	1	if (p1 == p2)	p2++
1	2	13 − 10 = 3 > 0	p1++
2	2	if (p1 == p2)	p2++
2	3	23 − 13 = 10 > 0	p1++
3	3	if (p1 == p2)	p2++
3	4	⟶ stop	

* If $K < 0$?

$arr[3] = [\ \overset{0}{4} \quad \overset{1}{10} \quad \overset{2}{13}\ ]$, $K = -3$

Pair: $K = -3$; $arr[1] - arr[2]$

$K = 3$; $arr[2] - arr[1]$

Obs: if $arr[i] - arr[j] == K \iff arr[j] - arr[i] == -K$

Pair with K exists, -K will also exist.

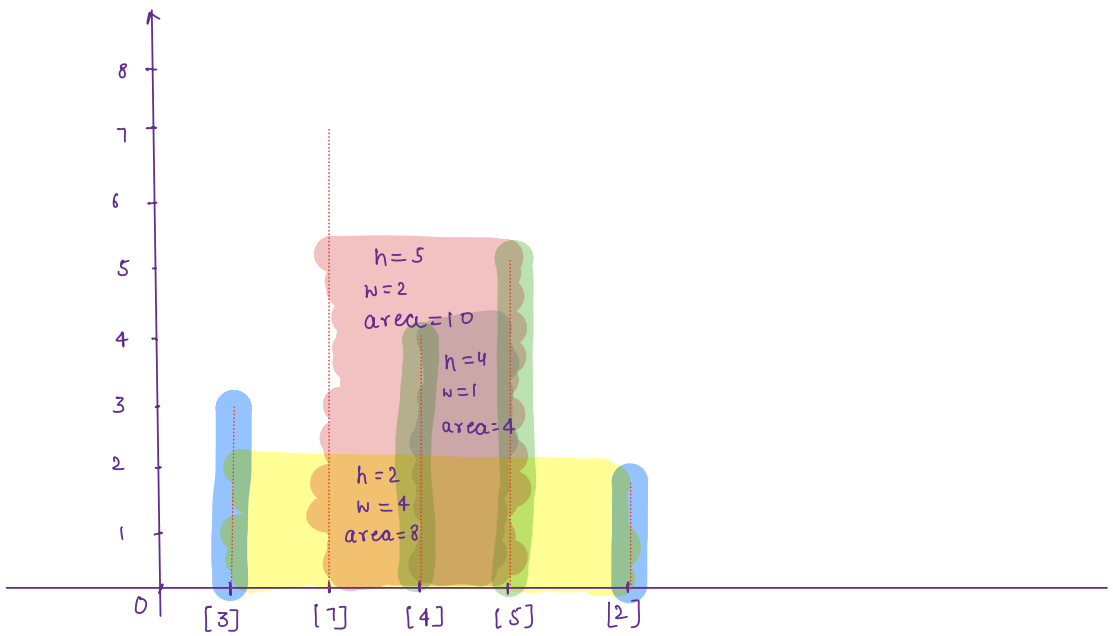
Break :- 8:38 AM

Qu3: Given arr[n] elements, arr[i] represents height of each wall.

find **max water accumulated** b/w any 2 walls ?

Note: Between any two walls, width == 1.

Example: arr[5] = [3 (0) 7 (1) 4 (2) 5 (3) 2 (4)], ans = 10.



Idea1: Go to all pairs. —

calculate area and compare with max area encountered

TC: $O(n^2)$

SC: $O(1)$

Idea2: calculate area b/w max and s.max wall. [wrong]

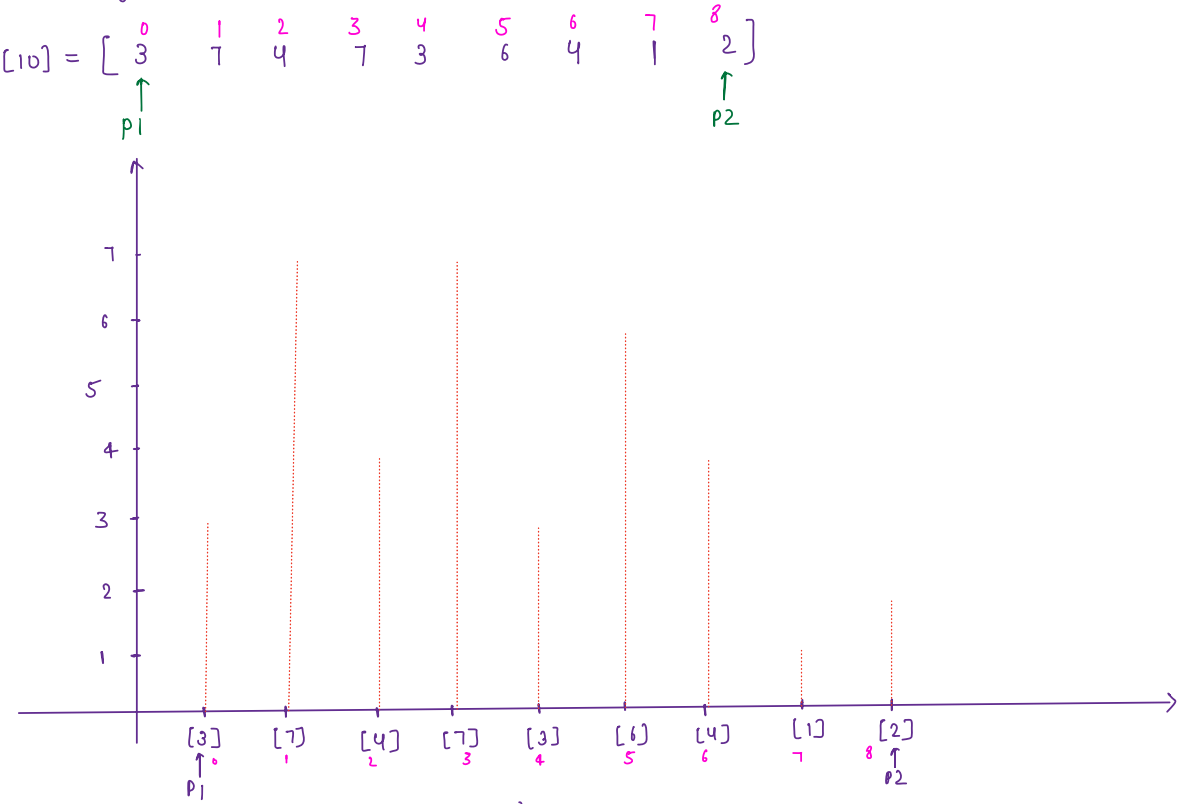
arr: [3 2 5 6 1 2 3 4 1 2]

area: 5 * 1 = 5

area = 2 * 9 = 18

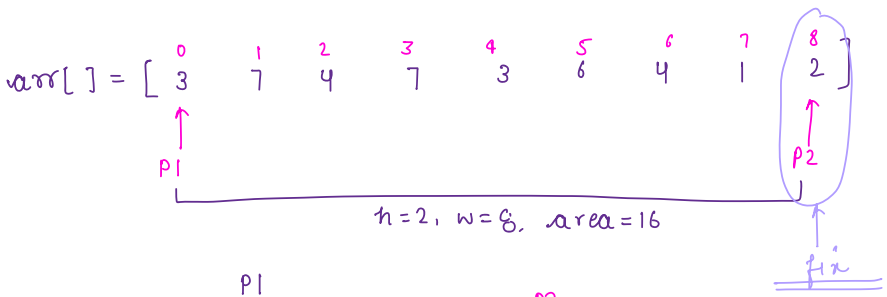
Idea3 Using 2 pointer ?

arr[10] = [3 (0) 7 (1) 4 (2) 7 (3) 3 (4) 6 (5) 4 (6) 1 (7) 2 (8)]



p1	p2	h = min (arr[p1], arr[p2])	w = p2 - p1	area	move	final ans
0	8	h = min (3, 2) = 2	8 - 0 = 8	2 * 8 = 16	p2--	16
0	7	h = min (3, 1) = 1	7 - 0 = 7	1 * 7 = 7	p2--	16
0	6	h = min (3, 4) = 3	6 - 0 = 6	3 * 6 = 18	p1 ++	18
1	6	h = min (7, 4) = 4	6 - 1 = 5	4 * 5 = 20	p2 --	20
1	5	h = min (7, 6) = 6	5 - 1 = 4	6 * 4 = 24	p2 --	24
1	4	h = min (7, 3) = 3	4 - 1 = 3	3 * 3 = 9	p2 --	24
1	3	h = min (7, 7) = 7	3 - 1 = 2	7 * 2 = 14	p1 ++	24
2	3	h = min (4, 7) = 4	3 - 2 = 1	4 * 1 = 4	p1 ++	24
3	3	→ stop				

* Which pointer to move any why?
  - Pointer with min height.



P1	P2		
0	8	⟶	area = 16
1	8	⟶	area = 2 * 7 = 14
2	8	⟶	area = 2 * 6 = 12
3	8	⟶	area = 2 * 5 = 10
4	8	⟶	area = 2 * 4 = 8
5	8	⟶	area = 2 * 3 = 6
6	8	⟶	area = 2 * 2 = 4
7	8	⟶	area = 1 * 1 = 1.

area ↓

Discard P2.

Code:

```
int maxwaterAccumulated(int[] arr){
    ans = 0;
    n = arr.length;
    p1 = 0, p2 = n-1;
    while( p1 < p2){
        if( arr[p1] <= arr[p2]){
            ans = max(ans, arr[p1] * (p2-p1));
            p1++;
        } else{
            ans = max(ans, arr[p2] * (p2-p1));
            p2--;
        }
    }
    return ans;
}
```

TC: O(n)
SC: O(1)

Qu4: Given 3 sorted arrays A[], B[], C[] of size n.

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

Example:

A[4] = [3 (0) 14 (1) 16 (2) 23 (3)] ⇒ ans = 3

B[4] = [-6 (0) 23 (1) 24 (2) 30 (3)]

C[4] = [-15 (0) 15 (1) 26 (2) 31 (3)]

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

Ideal: consider all triplets — find ans and compare with min ans encountered

TC: $O(n^3)$

SC: $O(1)$

Idea2:

A[4] = [3 (0) , 14 (1) , 16 (2) , 23 (3)] ↑ P1

B[4] = [-6 (0) , 23 (1) , 24 (2) , 30 (3)] P2

C[4] = [-15 (0) , 15 (1) , 26 (2) , 31 (3)] P3

P1	P2	P3	max (A[P1], B[P2], C[P3])	min (A[P1], B[P2], C[P3])	ans
0	0	0	max(3, -6, -15) = 3	min(3, -6, -15) = -15 Increase P3; P3++	3-(-15) = 18
0	0	1	max(3, -6, 15) = 15	min(3, -6, 15) = -6 Inc' P2; P2++	15-(-6) = 21
0	1	1	max(3, 23, 15) = 23	min(3, 23, 15) = 3 P1 ++	23-3 = 20
1	1	1	max(14, 23, 15) = 23	min(14, 23, 15) = 14 P1++	23-14 = 9
2	1	1	max(16, 23, 15) = 23	min = 15 ; P3++	23-15 = 8
2	1	2	max(16, 23, 26) = 26	min = 16 P1++	26-16 = 10
⋮	⋮	⋮	⋮	⋮	⋮

TC: O(n)

SC: O(1)

Code: Try out yourself.

*

A = [3 (P1) 14 16 23] (0 1 2 3)

B = [-6 (P2) 23 24 30]

C = [-15 (P3) 15 26 31]

p1	p2	p3	max	min	ans
0	0	0 (fix)	3	-15	3-(-15) = 18
1	1		23	-15	23-(-15) = 38
1	2		24	-15	24-(-15) = 39
2	2		24	-15	24-(-15) = 39

for p3, I can nver get any min ans. so discard it p3++.

Thankyou ☺