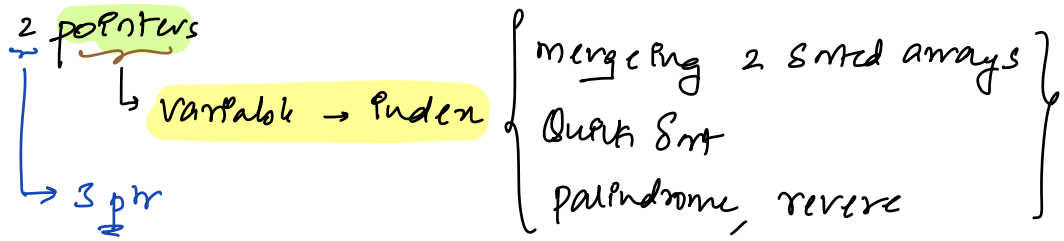
Today's Content:



→ Where to initialize

→ How to update

→ When to stop

18) Given N distinct sorted elements, Check if there exists a pair (i, j), ar[i] + ar[j] = k → return T/F (i ≠ j)

2 indices

Note:

Ex1: ar[5] = { 3 7 8 12 19 } k = 15

0	1	2	3	4
3	7	8	12	19

Ex2: ar[5] = { 2 5 8 11 15 } k = 16

0	1	2	3	4
2	5	8	11	15

a = 2, b = 14 ✗

a = 5, b = 11 ✓

Ex3:

Solutions

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

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

P_1: 0 1 1 1 — P_2: 7 8 9

a = -3

1) Check all pairs

TC: O(N²) SC: O(1)

2) Using HashSet/Map

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

3) for(i = 0; i < N; i++) {
   a = ar[i]
   b = k - a;
   Apply BS get b
}

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

P_1	P_2	sum = ar[P_1] + ar[P_2]	
0	9	22 > 17	decrease sum → P_2--
0	8	15 < 17	increase sum 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

```
bool check(ar[], N, k){
    P1 = 0,  P2 = N-1
    while( P1 < P2 ){
        if (ar[P1] + ar[P2] == k){
            return True
        }
        else if (ar[P1] + ar[P2] > k){
            P2--
        }
        else{
            P1++
        }
    }
    return false
}
```

TL: O(N) SC: O(1)

Eg:

k = 19

ar[5] = 3 9 10 14 18

P1 = 3, P2 = 18 → 21 > 19, P2--

P1 = 3, P2 = 14 → 17 < 19, P1++

3 — 18 → 21

3 — 14 → 17

9, 10, 14 > 21

10, 9 < 17

28) Given N distinct sorted elements. Check if there exists a pair (i, j), such that $ar[j] - ar[i] = k$ && $k > 0$ && $(i \neq j)$

	0	1	2	3	4	5	6	7	8	9
ar[N] =	-3	0	1	3	6	8	11	14	18	25

k = 5

N = 10

Initialize pointers

P_1	P_2	Diff
0	n-1	$ar[P_2] - ar[P_1]$
0	9	: 28 > 5 decr:

P_2--

P_1++ : diff decreases

We cannot make a decision

* Increment

P_1	P_2	Diff
0	n/2	$ar[P_2] - ar[P_1]$
0	5	11 decr

Increment

{ P_2--
P_1++ dec

P_1	P_2	Diff	k = 5
n/2	n/2+1	$ar[P_2] - ar[P_1]$	
5	6	3 < 5	Inc

{ P_2++ } Inc diffare
P_1-- }

*Increment

P_1	P_2
N-2	N-1

if $(ar[P_2] - ar[P_1] < k)$

{ increase diff
P_1--

else if $(ar[P_2] - ar[P_1] > k)$ {

{ decrease diff
P_2--

	0	1	2	3	4	5	6	7	8	9
ar[N] =	-3	0	1	3	6	8	11	14	18	25

P_1 → 2, P_2 → 4   k = 5

P_1	P_2	(ar[P_2] - ar[P_1])
0	1	: 3 < 5 : Inc → P_2++
0	2	: 4 < 5 : Inc → P_2++
0	3	: 6 > 5 : dec → P_1++
1	3	: 3 < 5 : Inc → P_2++
1	4	: 6 < 5 : dec → P_1++
2	4	: 5 == 5 : return True

Pseudocode → Pseudo Code : k > 0

k = abs(k)

$P_1 = 0$, $P_2 = 1$

no need

```
while( P1 < N && P2 < N )
    if( ar[P2] - ar[P1] == k ){
        return True
    }
    else if( ar[P2] - ar[P1] > k ){
        P1++
        if( P1 == P2 ){ P2++ }
    }
    else{
        P2++
    }
}
return false
```

TC: O(N)

SC: O(1)

Eg: 1 4 6   k = 2

P_1 → 0, P_2 → 1

P_1	P_2	ar[P_2] - ar[P_1]
0	1	3 decrease, P_1++
1	1	P_2++
1	2	2 (return True)

10 3 → 10 - 3 = 7 ✓

→ 3 - 10 = -7 ✓

ar[j] - ar[i] = 15 ✓ | 8

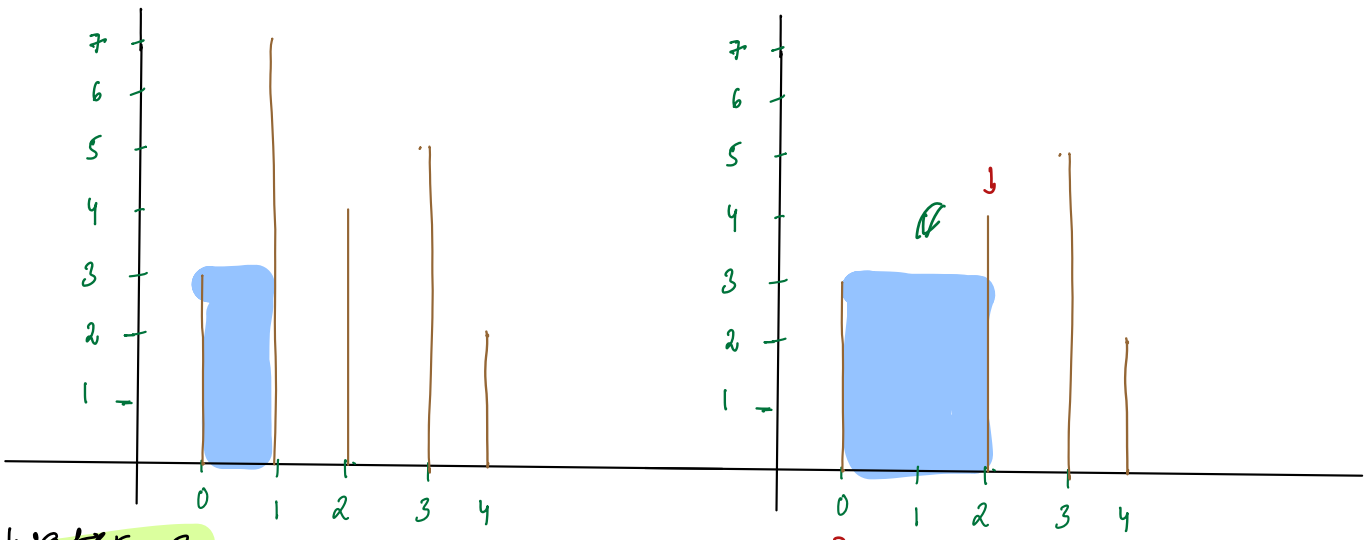
ar[i] - ar[j] = -15 ✓ | -8

39) Given N array elements,
where each arr[i] represents height of each wall

Pick any 2 walls such that max water is accumulated between them.

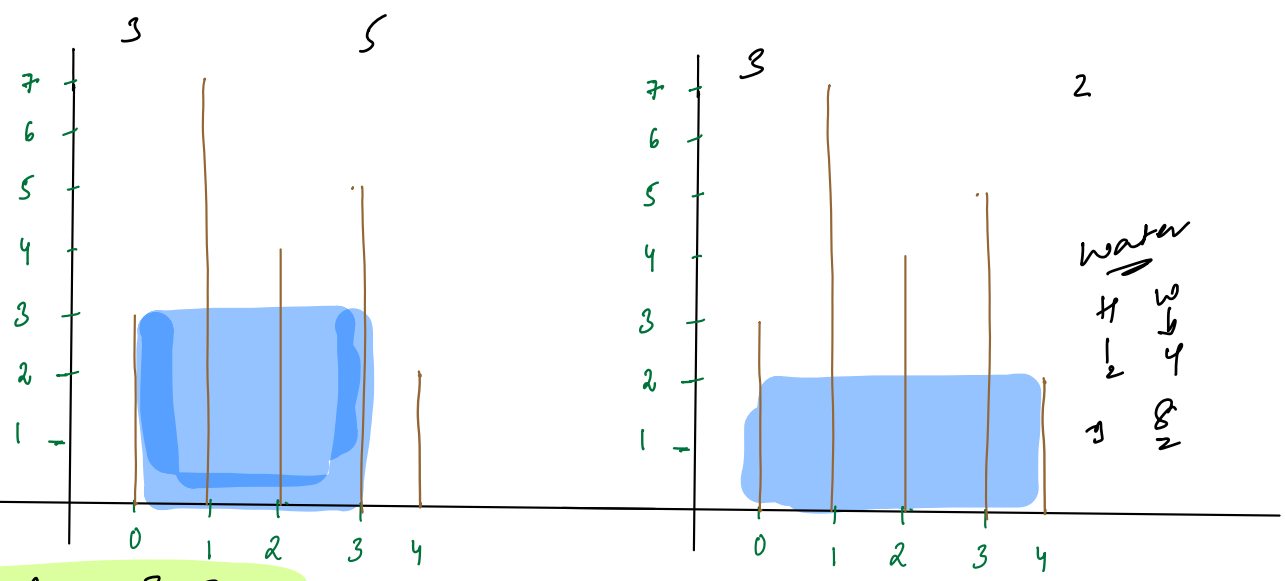
→ 10:40 break

Eg1: {3, 7, 4, 5, 2}



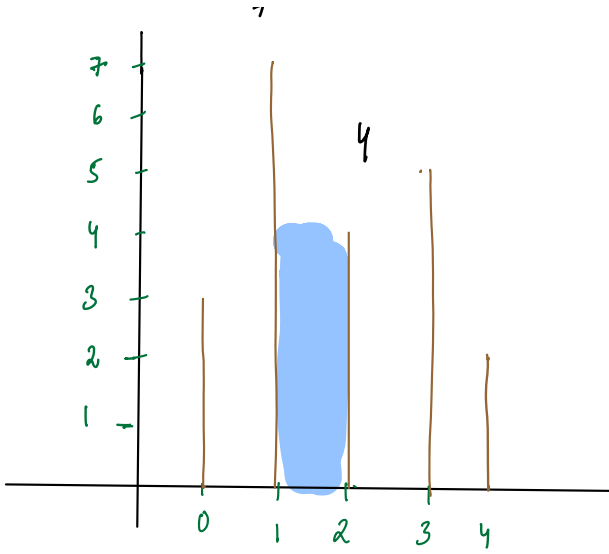
water = 3

→ 3 * 2 = 6 water

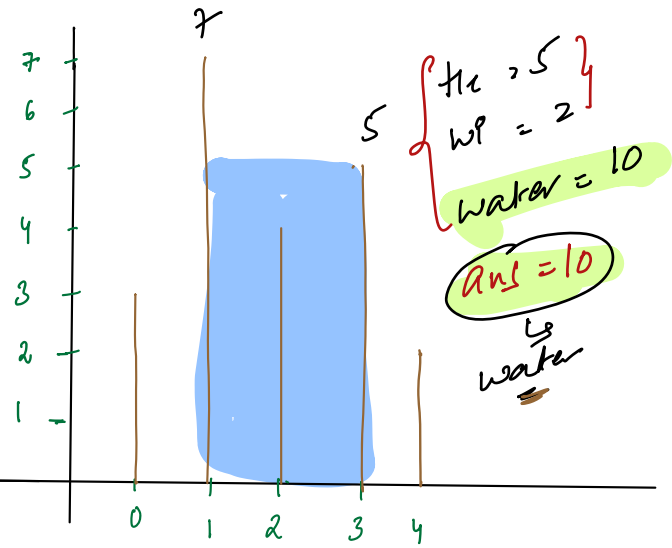


water = 3 * 3 = 9

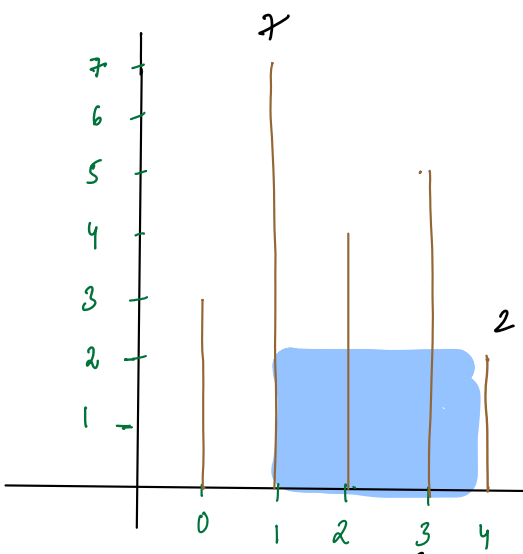
water



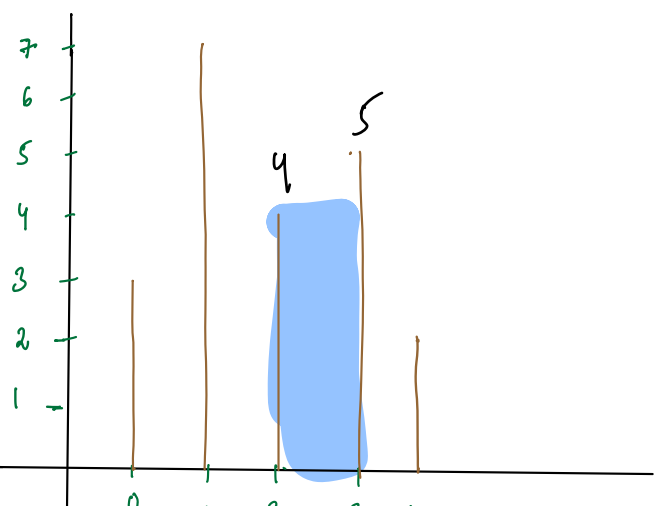
water = 4



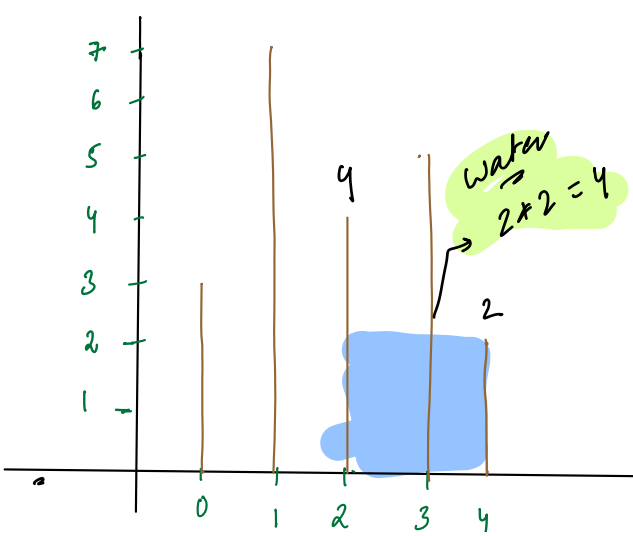
$tlx > 5$
$wp = 2$
water = 10
Ans = 10
is water



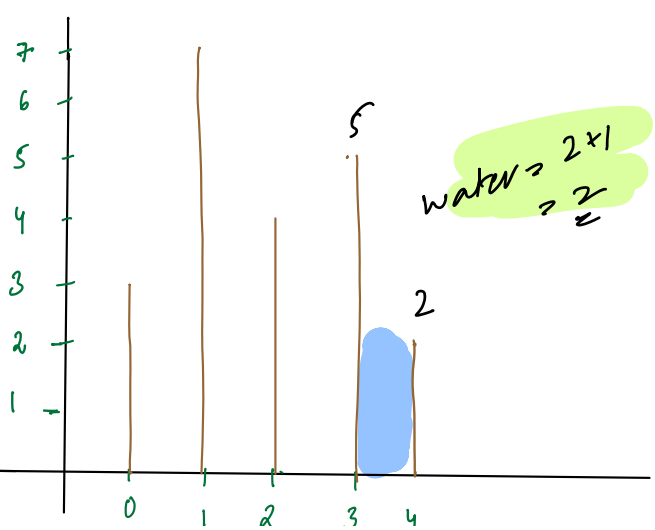
water = 2 * 3 = 6



water = 4 * 1 = 4

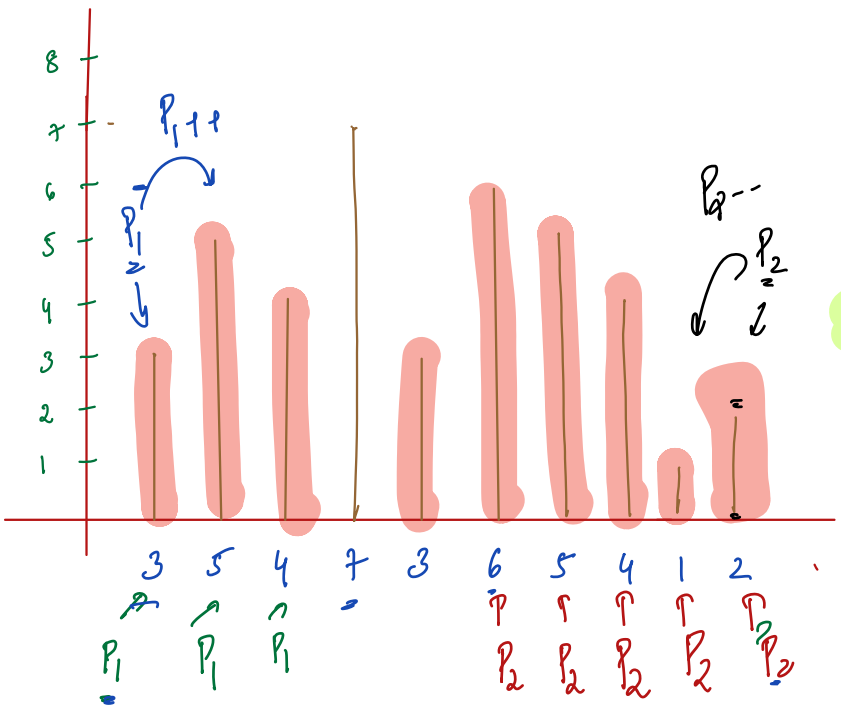


water = 2 * 2 = 4



water = 2 * 1 = 2

	0	1	2	3	4	5	6	7	8	9
ar[10] = {	3	5	4	7	3	6	5	4	1	2 }



→ obs:

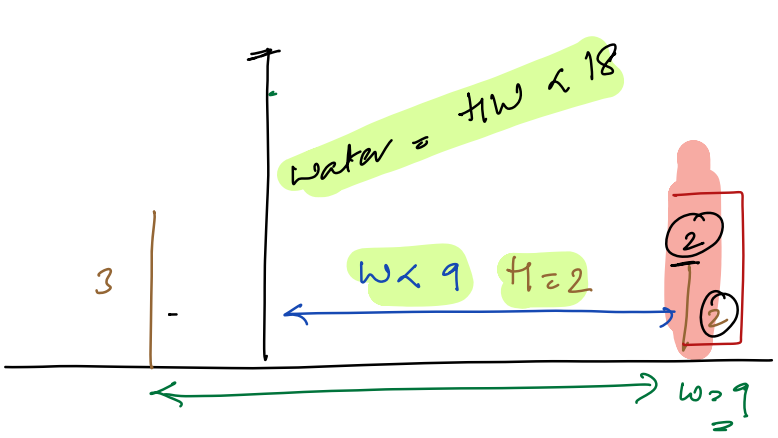
1) We cannot sort order changes

Note

2) We are moving the pointer → minimal bcoz we cannot get better ans

ans = 0

P_1	P_2	$H = (\min(ar[P_1], ar[P_2]))$	$W : (P_2 - P_1)$	water: (H*W)		
0	9	2	9	18	→	18
0	8	1	8	8	→	18
0	7	3	7	21	→	21
1	7	4	6	24	→	24
1	6	5	5	25	→	25
1	5	5	4	20	→	25
2	5	4	3	12	→	25
3	5	6	2	12	→	25
3	4	3	1	3	→	25
3	3	Break				



→ skip min height wall because we cannot get a better for that.

Pseudo Code:

1) Check all pairs → TC: O(N²) SC: O(1)

```
i = 0; i < N; i++
    j = i+1; j < N; j++
        H = min(ar[i], ar[j])
        W = j - i
        water = H * W
        ans = max(ans, water)
return ans
```

Pseudocode

TODO ?

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

find i, j, k such that

max(A[i], B[j], C[k]) - min(A[i], B[j], C[k]) is minimized

	0	1	2	3	4	5	
A[] = {	3	14	16	20	27	40	} ← P_1
B[] = {	-6	23	24	30	35	50	} ← P_2
C[] = {	-15	15	26	31	39	42	} ← P_3

i	j	k	A[i]	B[j]	C[k]	max	min	ans
0	0	0	3	-6	-15	3	-15	18
0	1	1	3	23	15	23	3	20
2	2	2	16	24	26	26	16	8
3	2	1	20	24	15	24	15	9
4	3	3	27	30	31	31	29	2

ans = 2

BF:

1) Check all pairs:

O(N³) SC: O(1)

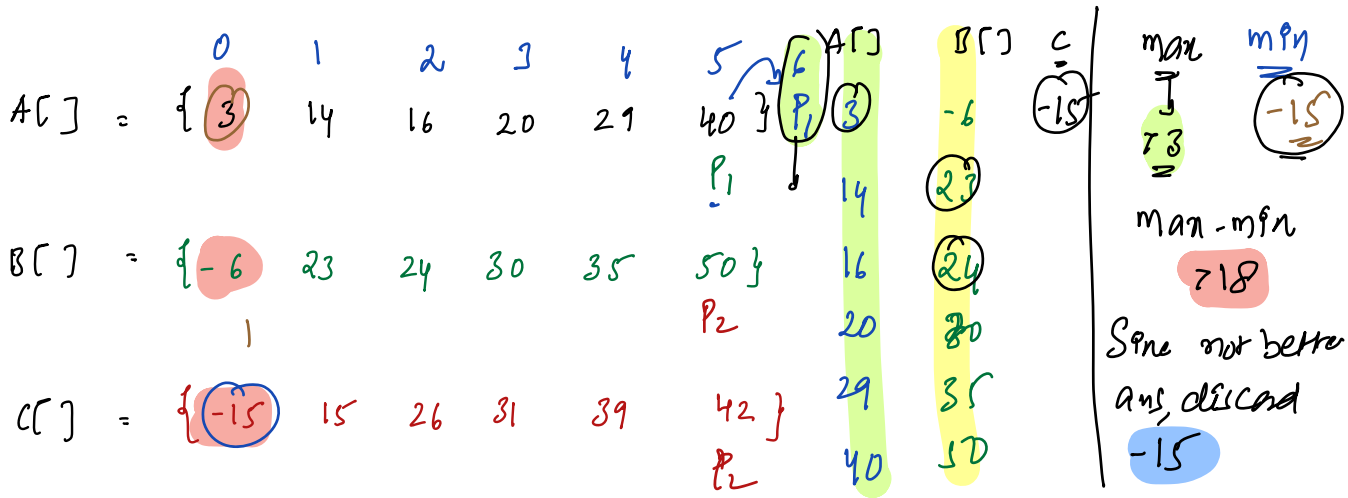
```
i = 0 ; i < N
    j = 0; j < N
        k = 0; k < N
```

3 different arrays

Thursday

→ BS

→ 2 pointers



// decrease - diff)

P_1	P_2	P_3	$A[P_1]$	$B[P_2]$	$C[P_3]$	max	min	(max - min)
0	0	0	3	-6	-15	3	-15	18
0	0	1	3	-6	15	15	-6	21
0	1	1	3	23	15	23	3	20
1	1	1	14	23	15	23	14	9
2	1	1	16	23	15	23	15	8
2	1	2	16	23	26	26	16	10
3	1	2	20	23	26	26	20	6
4	1	2	29	23	26	29	23	6
4	2	2	29	24	26	29	24	5
4	3	2	29	30	26	30	26	4
4	3	3	29	30	31	31	29	2
5	3	3	40	30	31	40	30	10
5	4	3	40	35	31	40	31	9
5	4	4	40	35	39	40	35	5
5	5	4	40	50	39	50	39	11

5 5 5 40 50 42 50 40 10

Pseudo code

$[P_1 = 0 \quad P_2 = 0 \quad P_3 = 0]$ & $[P_1 = N-1, \quad P_2 = N-1, \quad P_3 = N-1]$

while ($P_1 < N_1$ && $P_2 < N_2$ && $P_3 < N_3$)

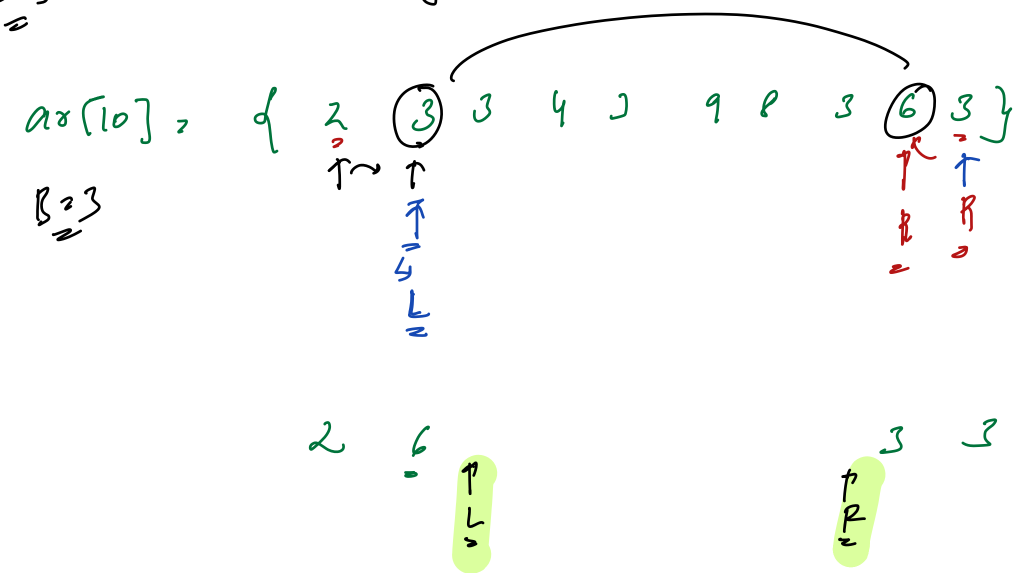
TC: $O(N * N * N)$

$\Rightarrow O(3N)$

$\Rightarrow O(N)$

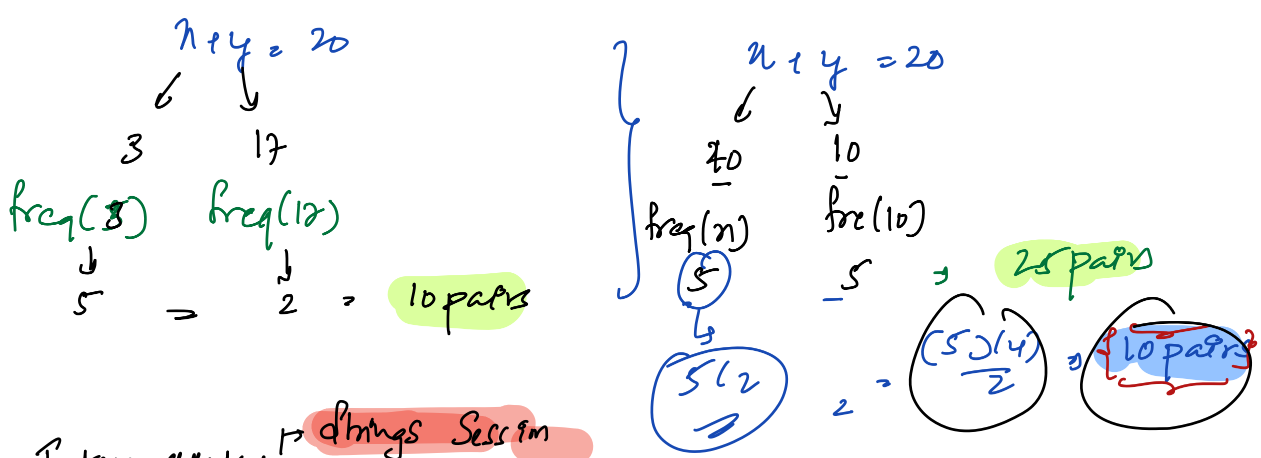
SC: $O(1)$

// 5Q) Given N array elements make all Elet = B on right



//

All pairs using map. {a+b with Hashmap}



// Intermediate: → doing Session

//