Page No.
Date / /

Lecture - 16

Two pointer

* Remove Duplicate from Array :

1	1	2	2	5	5	10	10	10	20
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Put	X Already	put	X Already present	put	X	put	X	X	put.

Ans = { 1 2 5 10 20 }

Brute Force :

```
int Note = 0;
int arr[N];
int index = 0;

for(int i = 0; i < N; i++) {
    if(A[i] != Note) {
        arr[index] = A[i];
        Note = A[i];
        index++;
    }
}

for(int i=0; i<index; i++){
    cout << Arr[i] << " ";
}
```

T.C : O(N)

S.C : O(N)

Teacher's Signature.....

Page No.:
Date: / /

Optimal Approach :

0	1	2	3	4	5	6	7	8	9
1	1	2	2	5	5	10	10	10	20

int Note = 0

↳ sabse pahle 0 initialise karenge.

index = 0

↳ for writing New element and increase the index.

1	1	2	2	5	5	10	10	10	20

↑

unique because Note ≠ pahle se zero hai

then,

A[index] = A[i];

Note ≠ 1 daal denge.

Note = 1;

index ++;

1	1	2	2	5	5	10	10	10	20

↑ ↑ → unique

Not unique

A[index] = A[i];

Note = A[i];

index++;

Note = 2

Teacher's Signature.....

1 1 2 2 5 5 10 10 10 20

Not unique unique.

Note = 5

A[index] = A[i];

Note = A[i];

index++;

1 1 2 2 5 5 10 10 10 20

Not unique. unique

Note = 10

A[index] = A[i];

Note = A[i];

index++;

1 1 2 2 5 5 10 10 10 20

unique.

Note = 20

A[index] = A[i];

Note = A[i];

index++;

Code:

```
int note = 0;
int index = 0;
for (int i=0; i<n; i++) {
    if (note != a[i]) {
        a[index] = a[i];
        note = a[i];
        index++;
    }
}
return index;
}
```

Teacher's Signature.....

Page No.:
Date: / /

* Subarray With given Sum :-

2 4 5 10 11 13 17

sum = 19

If subarray present
↳ return true

otherwise
↳ return false.

Brute force Approach :

check one by one

2 4 5 10 11 13 17

→ 2 != 19
→ 2+4 = 6 != 19
→ 2+4+5 = 19 != 19
→ 2 + 4 + 5 + 10 != 19
→ 2 + 4 + 5 + 10 + 11 != 19
→ 2 + 4 + 5 + 10 + 11 + 13 != 19
→ 2 + 4 + 5 + 10 + 11 + 13 + 17 != 19
→ 4 != 19
→ 4 + 5 != 19
→ 4 + 5 + 10 = 19 (return true)

Teacher's Signature.....

Page No
Date / /

Brute force Approach : $O(N^2)$

```
for (int i = 0; i<n; i++) {
        int sum = 0;
        for (int j = i; j<n; j++) {
            sum += arr[j];
            if (sum == givensum) {
                return true;
            }
        }
}
return false;
```

* Two pointer :

2 4 11 17 25 29

Find 2 number whose sum is equal to 40.

(1) Brute Force Approach :

Take all possible operation :

2 + 4 = 6 X
2 + 11 = 13 X
2 + 17 = 19 X
2 + 25 = 27 X
2 + 29 = 31 X

4 + 11 = 15 X
4 + 17 = 21 X
4 + 25 = 29 X
4 + 29 = 33 X

11 + 17 = 28 X
11 + 25 = 36 X
11 + 29 = 40 ✓ return true.

Teacher's Signature.....

Page No.:
Date: / /

code:

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

$O(N^2)$

→ 2 4 11 17 25 29

Can we Solve this by Binary search?

We have to find two number Whose sum is 40.

$x + y = 40$

find x and y

x = start

y = end

2	4	11	17	29	29
↑					
x					

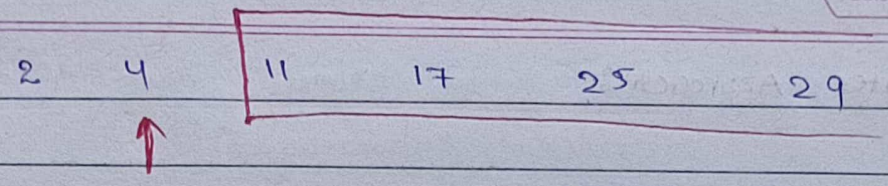
search 38 in this by Binary search

$2 + y = 40$

$y = 38$

Not found

Teacher's Signature.....

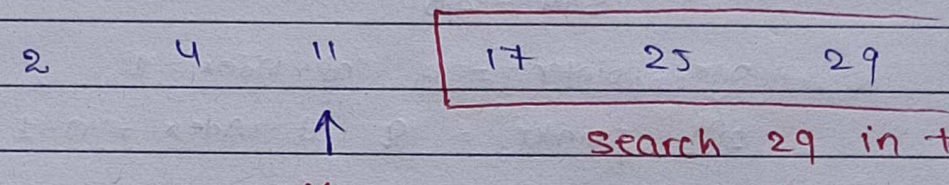


Search 36 in this by binary search

$4 + y = 40$

$y = 36$

Not found.



search 29 in this by binary search

$11 + y = 40$

$y = 29$ found.

return true.

Time Complexity : $O(N \log N)$

```
for (int i=0; i<n-1; i++) {
        find = sum - A[i];
        start = i+1;
        end = n-1;
        { logic of
          binary search; }
}
```

→ 2 pointer Approach :

2	4	11	17	25	29
↑					↑
p					q

Sum = 40

one pointer = p at index = 1

2nd pointer = q at index = n-1

check sum

2 + 29 < 40

Move right → (p)

2	4	11	17	25	29
	↑				↑
	p				q

4 + 29 = 33 < 40

Move right →

2	4	11	17	25	29
		↑			↑
		p			q

11 + 29 = 40 == 40

||

sum

return true;

Teacher's Signature.....

→ Kab tak chalega
↳ (p < q)

→ Jab value big chaye → Move right (p)

→ Jab value small chaye → Move left (q)

Code:

```
left = 0, right = n-1;
int sum = 0;
while (left < right) {
    if ( arr[left] + arr[right] == tsum) {
        return true;
    }
    else if ( arr[left] + arr[right] > sum){
        right--;
    }
    else {
        left++;
    }
}
return false;
}
```

Teacher's Signature.....

Page No.:
Date: / /

* Array : 3 7 13 41 49 51

sorted

Find two number :

x & y → given

x * y = product

product = 91

→ Brute Force Approach :

3 x 7 = 21 x 7 x 13 = 91 ✓ (True)

3 x 13 = 39 x

3 x 41 = 123 x

3 x 49 = 147 x

3 x 51 = 153 x

→ Two pointer Approach :

3 7 13 41 49 51

↑ ← ↑

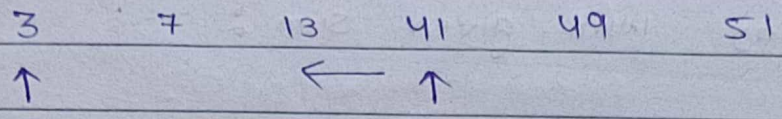
3 x 51 = 153 > 99

3 7 13 41 49 51

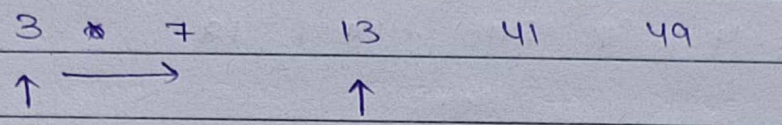
↑ ← ↑

3 x 49 = 147 > 99

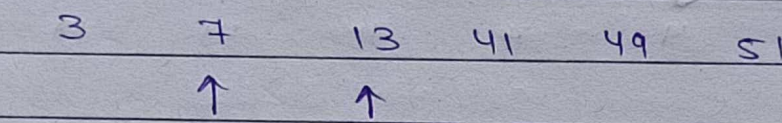
Teacher's Signature.....



$3 \times 41 = 123$



$3 \times 13 = 39 < 91$



$7 \times 13 = 91$ (True)

```
left = 0, right = n-1;
    int sum = 0;
While (left < right) {
    if ( arr[left] * arr[right] == sumgiven){
                return true;
      }
    else if ( arr[left] * arr[right] > sumgiven){
                right--;
      }
    else {
        left++;
      }
}
```

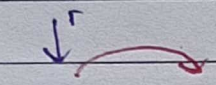
Page No.:
Date: / /

* Subarray With given Sum :

Two pointer Approach :

2 4 7 11 14 18

Sum = 25



→ 2 4 7 11 14 18

↑ l

Sum = 25

Count_sum = arr[0]

2 != 25 2 < 25

r

↓

→ 2 4 7 11 14 18

↑ l

Count_sum = 2 + arr[r] = 6

6 != 25 6 < 25

↓r

→ 2 4 7 11 14 18

↑ l

Count_sum = 13

13 != 25 13 < 25

↓r

→ 2 4 7 11 14 18

↑ l

Count_sum = 24

24 != 25 24 < 25

↓r

→ 2 4 7 11 14 18

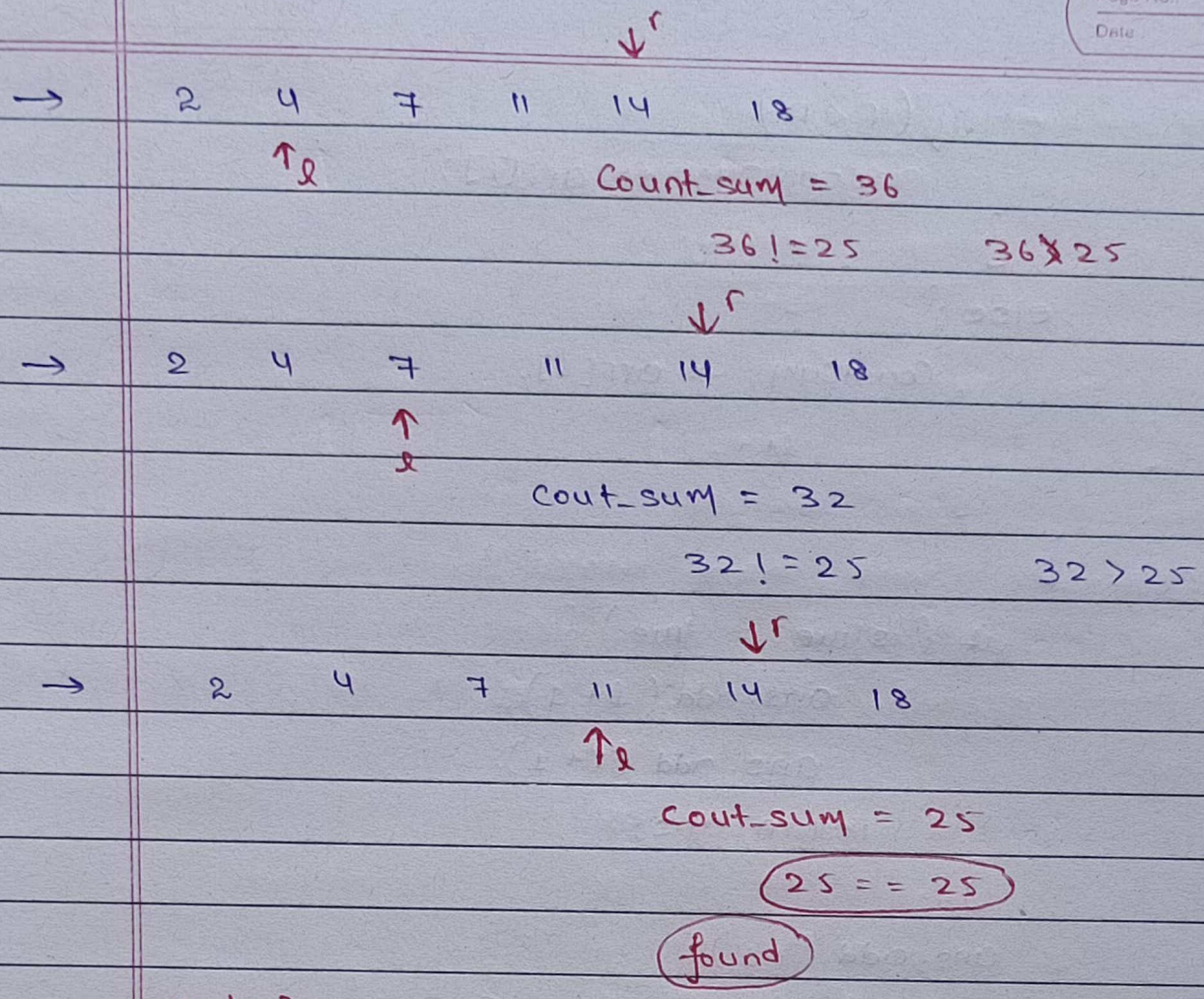
↑l

Count_sum = 38

38 != 25 38 > 25

Move l.

Teacher's Signature.....



Code :

```
Arraylist<Integer> ans = new Arraylist<>();
        if (s == 0) {
            ans.add(-1);
            return ans;
        }
        int l = 0;
        int r = 0;
        int Countsum = arr[0];
        boolean isTrue = false;
        While (r < n) {
            if (Countsum == s) {
                isTrue = true;
                break;
            } else if (Countsum < s) {
                r++;
```

```
        if (r < n)
            countSum += arr[r];
        }
    else {
        countSum -= arr[l];
            l++;
        }
    }
    if (isTrue = true) {
            ans.add (l+1);
            ans.add (r+1);
        return ans;
    }
    ans.add (-1);
      return ans;
}
```

* Union of two sorted array :

Arr1[] = { 1, 1, 2, 3 } size = n

Arr2[] = { 2, 3, 3, 4, 5 } size = m

Ans[] = { 1, 2, 3, 4, 5 }

Step 1 : Remove all the duplicates element from both the array.

Arr1[] = { 1, 2, 3 }

Arr2[] = { 2, 3, 4, 5 }

Teacher's Signature.....

Page No.
Date / /

Arr1 = {1, 2, 3} Arr2[] = {2, 3, 4, 5}

1 < 2

ans[] = {1}

Arr1[] = {1, 2, 3} Arr2[] = {2, 3, 4, 5}

2 == 2

ans[] = {1, 2}

Arr1[] = {1, 2, 3} Arr2[] = {2, 3, 4, 5}

3 == 3

ans[] = {1, 2, 3}

Arr1[] = {1, 2, 3} Arr2[] = {2, 3, 4, 5}

ans = [] = {1, 2, 3, 4, 5}

Code :

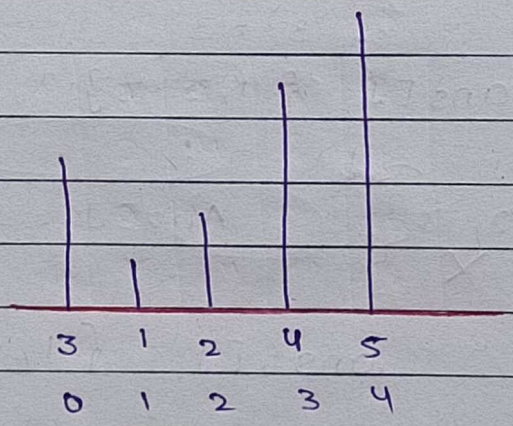
```
While (i<n && j<m) {
    if (arr1[i] == arr2[j]) {
        ans[k] = arr1[i];
        k++;
        i++;
        j++;
    }
    else if (arr1[i] > arr2[j]) {
        ans[k] = arr2[j];
        k++;
        j++;
    }
```

Teacher's Signature.....

```
        else {
            ans[k] = arr1[i];
            k++;
            i++;
        }
    }
```

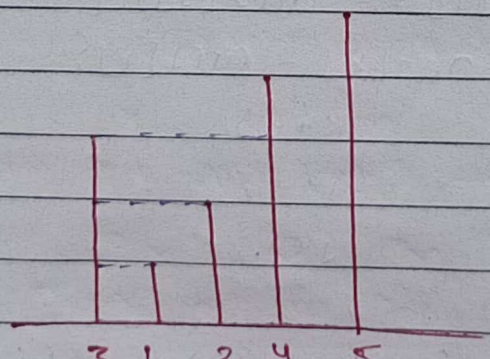
* Container With Most Water :

Container : { 3, 1, 2, 4, 5 }



Brute force :

Check for all possible and find maximum.



Max water = 0

(3,1) (1,2) (2,4) (4,5)
(1,4) (2,5)
(3,2) (1,5)
(3,4)
(3,5)

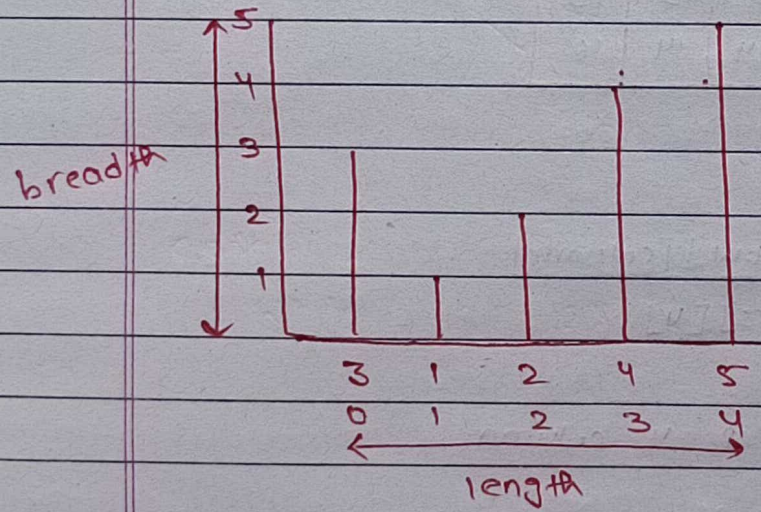
Teacher's Signature.....

```
for(int i = 0;   i < n-1;   i++) {
    for (int j= i+1;   j<n;   j++) {

        int bd  = Math.Min ( Arr[i], Arr[j]);
        int length  = j - i;
        int water  = bd * length;
        MaxWater =  Math.max (water, MaxWater);
    }
}
return 0;
}
```

$O(N^2)$

Optimise:



Answer = lenght * breadth;

Both side → one pointer

i at 0

j at 4

Find Min-m from both & calculate MaxWater.
Where we find min-m increase that pointer one step.

Teacher's Signature.....