Explanation done in meeting -

Arrays -

```
Int a[n];                       // array declaration of size n
vector<int> v;                  // vector declaration of size 0
vector<int> v(n);               // vector declaration of size n
vector<int> v(n,0);             // vector declaration of size n and initialize all elements
with 0
```

Repeating and missing number in array in range 1-n

5 1 4 2 3 4

```
// try to map element to index
i=0
while(i<n){
        if (a[i]==i+1 && a[i]!=a[a[i]-1])
                swap(a[i],a[a[i]-1]);
        Else
                I++;
}
```

3 1 4 2 5 4
4 1 3 2 5 4
2 1 3 4 5 4
1 2 3 4 5 4
Repeating - 4
Missing - 6

Ternary search -

Binary search -
0 - n l,r
M = (l+r)/2
R = m-1
L = m+1
O(log 2 n)

Ternary search -
0 - n l,r
M1 = l+ (r-l)/3
M2 = r- (r-l)/3

L - m1, m1+1 - m2, m2+1 - r
Time complexity - O(log 3 n)

Maximum size array

10^ 8 - boolean - bitset
10^7 - global
10^ 6 - function(main also)

Count sort -
0 n 0 - 6
1 3 4 5 6 2 3 4 5 3 2 4 5

A[7] = {0, 1, 2, 3, 3, 3, 1};

Merge overlapping intervals
Que - [1,3], [1,4], [2,3],[4,5]
Ans - [1,4],[4,5]

```
vector<pair<int,int>> v;
for(int i=0;i<n;i++)
{
        //Take input
}

sort(v.begin(),v.end());
// complete rest code
```

Largest consecutive subsequence
2 3 6 1 4 5 8 9 10

1 2 3 4 5 6 8 9 10

Try to find the sol in best complexity -
Best - O(n)
Min
Max
Repeating
Max-min+1 = length

Explanation (meeting -3)
Kadane's algorithm
Array[n] = 3 0 -2 3 -4 2 1 -2 4
Find the maximum subarray sum

Brute force - try all subarrays(n*n) - calculate their sum and take max
O(n*n*n)

Prefix sum = 0 -2 1 -3 3 4 2 6
Sum[l,r] = p[r]- p[l-1]
O(n*n)

Code :

```
Int csum=0,msum=0;
for(int i=0;i<n;i++)
{
        Csum += a[i];
        if(csum<0)
                Csum = 0;
        msum= max(msum,csum);
}

// if all negative or 0
if(msum==0)
{
        Int d=a[0];
        for(int i=0;i<n;i++)
        {
                d = max(d,a[i]);
        }
        msum=d;
}
cout<<msum;
```

Stock buy sell problem
One time buy and one time sell

Array = 2 4 6 3 5 3 5 1
Find max profit
profit= sell - buy

Suppose you buy on ith day and sell on jth day then i<j

2 4 6 3 5 3 5 1
Min till now (i-1)
And if we sell on ith day then take max of profit

Do its 6 variations from pepcoding videos after you study DP
(solution includes DP, that's why)
Do only this one now - https://www.youtube.com/watch?v=4YjEHmw1MX0

Links for rest variations
https://www.youtube.com/watch?v=3YILP-PdEJA

https://www.youtube.com/watch?v=HWJ9kIPpzXs
https://www.youtube.com/watch?v=wuzTpONbd-0
https://www.youtube.com/watch?v=pTQB9wbIpfU
https://www.youtube.com/watch?v=GY0O57llkKQ

Array - 0 2 4 6 3 5 1
Diff arr - 0
Diff - A[i]-a[i-1]

Prefix sum = p[i] = p[i-1]+a[i]

Sum in range[l,r] = p[r]-p[l-1]

Update [l,r] by k
L r k
2 4 5
1 6 -2

Q queries and n length
q*n not valid

Array -a 0 2 4 6 3 5 1
query
L r k
2 5 3
Final array after updation - 0 2 7 9 6 8 1

Diff arr -d 0 2 5 2 -3 2 -7
B = final elements after updation

```
while(q--)
{
        Int l,r,k;
        cin>>l>>r>>k;
        d[l]+=k;
        if(r+1<n)
                D[r+1] -=k;
}
Int s=0;
for(int i=0;i<n;i++)
{
        b[i] += s+d[i];
        S += d[i];

}
```

(code written in meeting was correct)

Matrix - 90 deg rotation clockwise
1 2 3 7 4 1
4 5 6 8 5 2
7 8 9 9 6 3
transpose
1 4 7
2 5 8
3 6 9

Print the matrix in spiral form
1 2 3 6 9 8 7 4 5

Staircase search
Matrix n*n rows and columns sorted and we have to find an element k

Linear traverse O(n*n)
Binary search on every row/column - O(n*logn)

1 2 6
3 4 8
5 7 9

K = 8

Set matrix zero
Set every row and column zero if that element is zero

1 0 3 5 0
2 4 0 6 4
1 5 1 2 5

Actual ans
0 0 0 0 0
0 0 0 0 0
1 0 0 2 0

Brute force - O(n*m*(n+m)) time, O(n*m) space
Time optimized - O(n*m), O(n+m)
Space optimized - O(n*m), O(1)

(Try to solve in above both complexities)

Median of row wise sorted matrix

Time = O(n*m*(_)
Space O(1)

Kth smallest element
Let k=3 => ans = 3
1 2 6
3 4 8
5 7 9

Sliding window

0 1 2 3 4 5 6 7 8 9 10
Size j-i+1
I =1, j=4 => i=2,j=5

K - size
N total

N-k+1 = total subarrays of size k
I j
0 k s
1 k+1 s + a[j] - a[i-1]
2 k+2
3 k+3
…
n-k n

Array -only 0 and 1
Find Max subarray size all 1
0 0 1 0 1 1 1 0 1 1 ans=3

---

k th root of number n
k=3 n=7
(Ans)^k = n

Square root of n
STL functions - sqrt(n), cbrt(n)

0 - n
k*k <= n
K should be max

5
2

We use Binary search
0 1 2 3 4 5 6 7 8

Code -

```
while(L<=r)
{
        M = (l+r)/2

if(pow(m,k) <=n)
        ans=m, l = m+1;
Else
        R = m-1;
}
```

Unique number 1 = All elements appear twice except 1 element and find that element
Unique number 2 = All elements appear twice except 2 element and find those element
Unique number 3 = All elements appear thrice except 1 element and find that element

Try these questions
Required space O(1) and time O(n) (can be O(n*(log (base 10)n)))
