Today's agenda

- Reverse an array
- Reverse a given part of array.
- Rotate array by k.
- greater than itself.
- Two Sum.

Q) Reverse array

In Given array of length N, Reverse the whole array.

ex: arr[5]: {10 20 30 40 50}

0 1 2 3 4

50 40 30 20 10

arr[8]: {10 20 30 40 50 60 70 80}

0 1 2 3 4 5 6 7

80 70 60 50 40 30 20 10

arr[8]: {10 20 30 40 50 60 70 80}

0 1 2 3 4 5 6 7

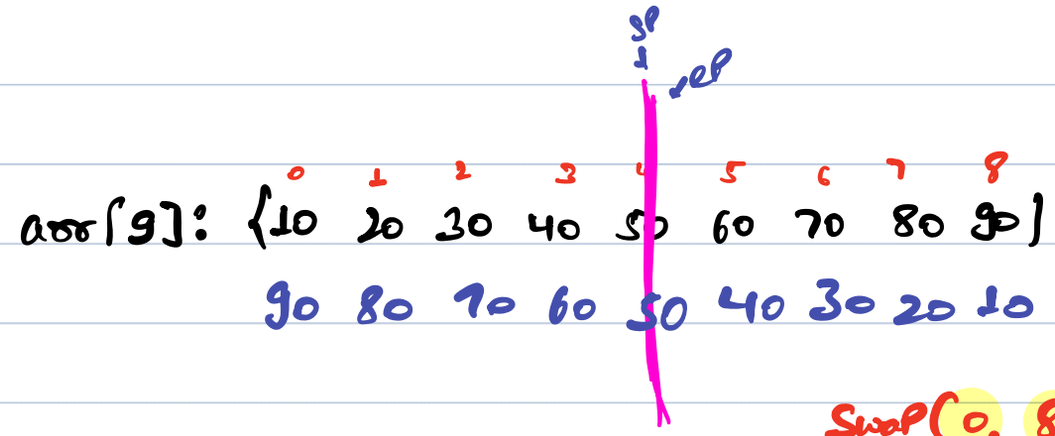
80 70 20 50 40 20 20 10

Swap (0, 7)

Swap (1, 6)

Swap (2, 5)

Swap (3, 4)




```
int main() {
    // input - arr?
    reverse(arr);
    for(int i=0; i<arr.length; i++){ S.O.P(arr[i]); }
}
```

```
Public static void reverse(int[] arr){

    int sp = 0;
    int ep = arr.length - 1;

    while(sp < ep){
        int temp = arr[sp];
        arr[sp] = arr[ep];
        arr[ep] = temp;
        sp++;
        ep--;
    }
}
```

arr.length

TC: O(n)

sp: 1 2 3 4
ep: 7 6 5 4

```
while ( 0 ){
    int temp = arr[sp];
    arr[sp] = arr[ep];
    arr[ep] = temp;
    sp++;
    ep--;
}
```

arr[8]: {10 20 30 40 50 60 70 80}
(0 1 2 3 4 5 6 7)
80 70 60 50 40 30 20 10

sp ep
Swap (0,7)
Swap (1,6)
Swap (2,5)
Swap (3,4)

Q) Reverse a Part of array

↳ Given N array element and [s,e], reverse the array from [s,e] index.

[3,7]

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

arr[10] = { -3 4 2 8 3 9 6 2 8 10 }

{ -3 4 2 [2 6 9 3 8] 8 10 }

swap(3,7)
↓ till 2nd
↓
swap(4,6)

```
Public static void reverse (int[] arr, int s, int e){
    int sp = s;
    int ep = e;
    while (sp < ep){
        int temp = arr[sp];
        arr[sp] = arr[ep];
        arr[ep] = temp;
        sp++;
        ep--;
    }
}
```

Big O notation → worst case scenario

T.C: O(N)

S.C: O(1) (O(X) → ignore input)

(Space Complexity)

Q) Rotate the array → {google, meta, amazon}

↳ Given N elements, Rotate array from last to first by K times.

K=3 arr[7]: { 3 -2 1 4 6 9 8 } (0 1 2 3 4 5 6)

↓ last pos.

{ 8 3 -2 1 4 6 9 }

↓ 2nd last.

{ 9 8 3 -2 1 4 6 }

↓ 3rd last.

{ 6 9 8 3 -2 1 4 }

expected T.C: O(N)
S.C: O(1)

K=3 arr[7]: { 3 -2 1 4 6 9 8 } (0 1 2 3 4 5 6)

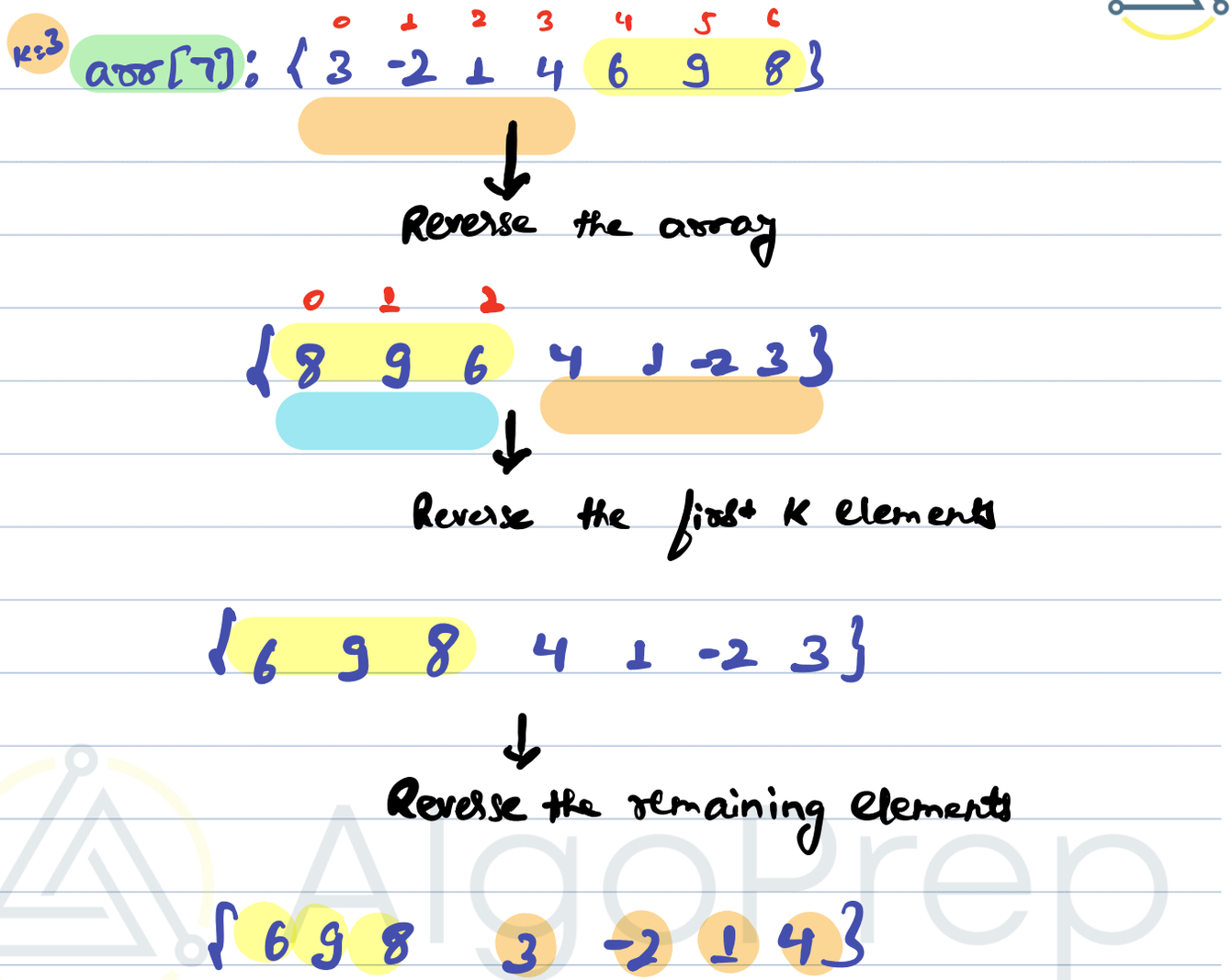
↓

{ 6 9 8 3 -2 1 4 }

K=4 arr[7]: { 3 -2 1 4 6 9 8 } (0 1 2 3 4 5 6)

↓

4 6 9 8 3 -2 1


k=3
arr[7]: { 3 -2 1 4 6 9 8 }
0 1 2 3 4 5 6
Reverse the array
{ 8 9 6 4 1 -2 3 }
0 1 2
Reverse the first K elements
{ 6 9 8 4 1 -2 3 }
Reverse the remaining elements
{ 6 9 8 3 -2 1 4 }

//Pseudo code

```
int main() {
    input → arr[N], K
    K = K%N
    //Step 1: Reverse the whole array.
    reverse(arr, 0, N-1);
    //Step 2 : Reverse the first K elements.
    reverse(arr, 0, K-1);
    //Step 3: Reverse the remaining elements
    reverse(arr, K, N-1);
}
```

T.C: O(N)

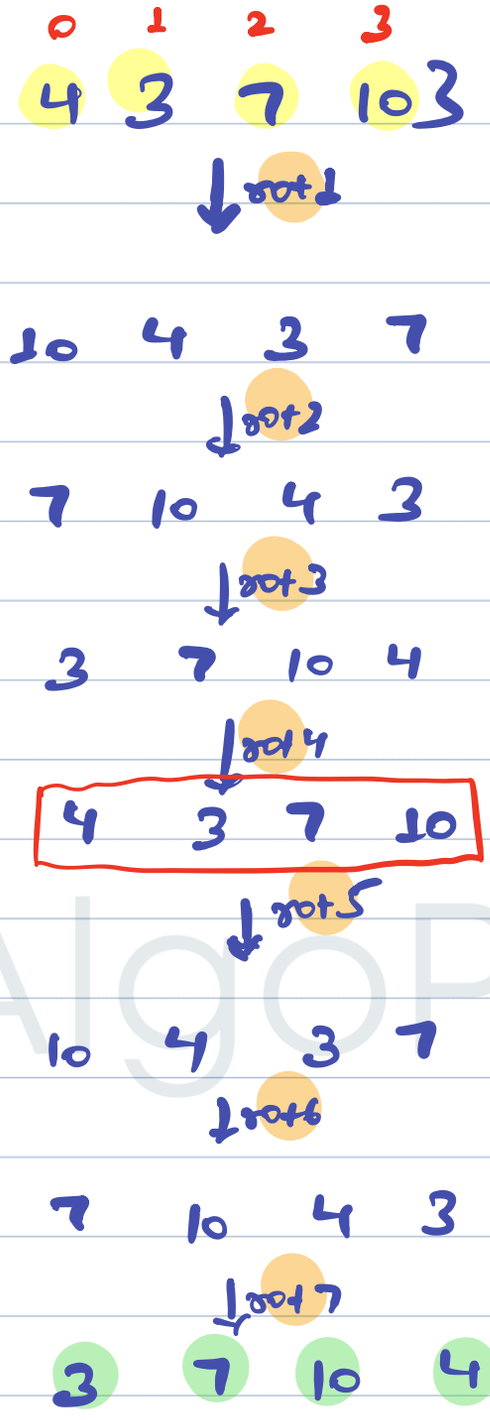
S.C: O(1)

```
Public static void reverse(int[] arr, int s, int e){

    int sp = s;
    int ep = e;

    while(sp < ep){
        int temp = arr[sp];
        arr[sp] = arr[ep];
        arr[ep] = temp;
        sp++;
        ep--;
    }

}
```

K=7

arr[4]: { 4 3 7 10 }

0 1 2 3



→ K > N → K = K - N ??

K=10

arr[4]: { 4 3 7 10 }

0 1 2 3

rot1
rot2
rot3
rot4
rot5
rot6
rot7
rot8
rot9
rot10

effective no. of rotation: K%N

N	K	effective rotation
7	24	24-7 = 17-7 = 10-7 = 3
		↳ 24%7 = 3
{_ _ _ _ _ _ _}		
7	31	31%7 = 3
7	3	3%7 = 3

Break till 10:40 Pm

Q) Given N array elements, count total no. of elements having atleast 1 element greater than itself.

ex: arr[7]: { -4 -3 7 9 3 9 4 }
(0 1 2 3 4 5 6)
↳ ans = 5

arr[8]: { 3 4 11 8 2 10 9 11 }
↳ ans: 6

arr[5]: { 7 7 7 7 7 }
↳ ans = 0

arr[6]: { 1 2 2 3 3 4 }
↳ ans: 5

obs1: max elements of the array are not valid.

obs2: non-max elements are always valid.

→ find occurrence of max element. → maxcount

↳ ans = no. of elements - maxcount.

arr[8]: { 3 4 2 8 11 10 9 11 }
0 1 2 3 4 5 6 7

//Pseudo Code

max: 11

maxcount: 2

```
int CountGreater(int arr[N]){
    int max = arr[0];
    for(int i=1; i<N; i++){
        if(arr[i] > max){max = arr[i];}
    }
    int maxCount=0;
    for(int i=0; i<N; i++){
        if(arr[i] == max){maxCount++;}
    }
    return N - maxCount;
}
```

T.C: O(2N) ≈ O(N)

S.C: O(1)

Q) Two Sum

↳ Given N array elements, check if there exists a Pair (i,j) such that arr[i] + arr[j] = K and i != j

Note: i and j are index value, K is given sum.

ex: arr[7] : { 2 -1 0 3 2 5 7 }

K=8 ↳ True

arr[4] : { 1 3 -2 6 }

K: 5 ↳ false

arr[5] : { 2 4 -3 7 10 }
0 1 2 3 4

K: 8 ↳ arr[1] + arr[2] = 8 → false

arr[6] : { 3 5 2 8 3 7 }

K=6 ↳ true

K=12

arr[5]: {3 5 2 7 5}
0 1 2 3 4

n

i j

(0,0) (1,0) (2,0) (3,0) (4,0)
(0,1) (1,1) (2,1) (3,1) (4,1)
(0,2) (1,2) (2,2) (3,2) (4,2)
(0,3) (1,3) (2,3) (3,3) (4,3)
(0,4) (1,4) (2,4) (3,4) (4,3)

i → 0 1 2 3 (n-2)

j → (1,2,3,4) (2,3,4) (3,4) (4)

```
Public static boolean TwoSum(int arr[n]){
    for(int i=0; i<=n-2 ; i++){        // i < n-1
        for (j=i+1; j<=n-1 ; j++){     // j <= n
            if(arr[i]+arr[j])==k){return true;}
        }
    }
    return false;
}
```

How
↓
T.C: ?!
S.C: O(1)