Today's agenda

- Reverse an array
- Reverse a given Part of array.
- Rotate array by K.
- greater than itself.
- Two Sum.

Q) Reverse array

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

```
            0  1  2  3  4
ex: arr[5]: {10 20 30 40 50}
             50 40 30 20 10
```

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

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

```
           sp   ep
           ↓    ↓
Swap (0 < 7)
       ↓ +1   -1
Swap (1 < 6)
       ↓ +1   j-1
Swap (2 < 5)
       ↓ +1   j-1
Swap (3 < 4)
```

//Psuedo code

```
int main() {
    //input → arr
                ↳ 2 idx
    reverse (arr);
    for(int i=0; i< arr.length; i++){ s.o.p (arr[i]);}
}
```

```
public static void reverse (int [] arr){
        int n= arr.length;

    int sp= 0;
    int ep = n-1;

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

T.C: O(N)

S.C: O(1)

```
Public static void reverse (int [] arr){
        int n= arr.length;

      int sp= 0;
      int ep = n-1;

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

n=7

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

sp	ep	sp< ep
0	6	T
1	5	T
2	4	T
3	3	F exit

Q) Reverse a Part of array

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

[3,7]

      0 1 2   3 4 5 6 7   8 9
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)
+
swap(4,6)

```
Public static void reverse (int [] arr, int s, int e) {
        int n= arr.length;

    int sp = ~~0~~; s;
    int ep = n~~-1~~; e;

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

worst case time complexity
↑
T.C: $O(\frac{e-s}{2}) \approx O(e-s)$
⇓
$O(n)$

S.C: $O(1)$

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

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

K<N

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

↓ 1st rot.

	0	1	2	3	4	5	6	
{	8	3	-2	1	4	6	9	}

↓ 2nd rot.

{9 8 3 -2 1 4 6}

↓ 3rd rot.

{6 9 8 3 -2 1 4}

expected T.C: O(N)

expected S.C: O(1)

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

↓

{6 9 8 3 -2 1 4}

K=~~3~~ 2

0 1 2 3 4 5 6

arr[7]: { 3 -2 1 4 6 9 8 }

↓

Reverse the array

↓

0 1 2 3 4 5 6

{ 8 9 6 | 4 1 -2 3 }

↓

Reverse the first K elements

↓

{ 6 9 8 | 4 1 -2 3 }

↓

Reverse the elements after Kth element

{ 6 9 8 | 3 -2 1 4 }

{ 6 9 8 | 3 -2 1 4 }

//PSuedo code

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

```
int main(){
    //input → arr[N], K
    K = K%N;
    //Step1: Reverse the array
    reverse (arr, 0, N-1);
    //Step2: Reverse the first K elements
    reverse (arr, 0, K-1);

    //Step3: Reverse the after K 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 n= arr.length;

    int sp= s;
    int ep = e;

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

K = 7

0 1 2 3

arr[4] = { 4 1 6 9 }

↓ ~~rot~~+1

9 4 1 6

↓ ~~rot~~+2

6 9 4 1

↓ ~~rot~~+3

1 6 9 4

↓ ~~rot~~+4

4 1 6 9

↓ rot+5

K = 10

0 1 2 3

arr[4] = { 4 1 6 9 }

↓ rot+1

9 4 1 6

↓ rot+2

6 9 4 1

↓ rot+3

1 6 9 4

↓ rot+4

4 1 6 9

rot+5
rot+6
rot+7
rot+8

4 1 6 9

effective no. of rotation = K % N

→arr.length

N	K	effective Rotation
7	22	$22 - 7 = 15 - 7 = 8 - 7 = 1$ $\overset{\downarrow}{22} \% 7 = 1$
7	31	$31 \% 7 = 3$
7	3	$3 \% 7 = 3$

Break till 10:50 pm

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

0 1 2 3 4 5 6

ex: arr[7]: {-4 -3 7 9 3 9 4}

↳ 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

//idea

obs1: max elements of the array are not valid.

obs2: except for max elements, all the elements are valid.

→ find max no. & count it. → maxcount

↳ ans = No. of elements − maxcount.

//Psuedo Code

```
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 : $O(2N) \approx O(N)$

S.C : $O(1)$

maxCount = ~~0~~ ~~1~~ 2

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

```
int   max = arr[0];

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

max = ~~3~~ ~~4~~ 11

↳ 6

```
int  maxcount = 0;
for(int i=0; i<N; i++ ){
      if(arr[i]== max){ maxcount ++; }
}
```

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

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

K:5

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

K:8

arr[6] : { 3 5 1 8 3 7 }

K=6