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.

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

arr[10]: { 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 60 50 40 30 20 10

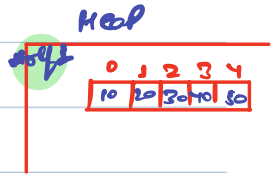
Swap (0, 7)
Swap (1, 6)
Swap (2, 5)
Swap (3, 4)

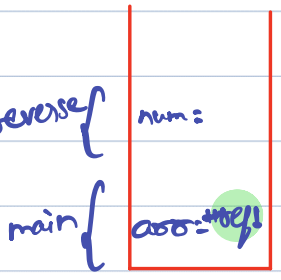
sp: 0+1 = 1+1 = 2+1 = 3
ep: 7-1 = 6-1 = 5-1 = 4

→ reverse an array = Combination of multiple swap.

//Pseudo code

```
int main() {
    //input arr[N]
    reverse(arr);
    for (int i=0; i<N; i++) { System.out.println(arr[i]); }
}
```






```
Public static void reverse(int[] num) {
    int sp = 0;
    int ep = num.length - 1;

    while (sp < ep) {
        int temp = num[sp];
        num[sp] = num[ep];
        num[ep] = temp;

        sp++;
        ep--;
    }
}
```

T.C: $O(N/2)$
$\approx O(N)$

S.C: $O(1)$

```
int sp = 0;
int ep = num.length - 1;

while (sp<ep) {
    int temp = num[sp];
    num[sp] = num[ep];
    num[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 60 10

sp		ep
0	<	7
1	<	6
2	<	5
3	<	4
4		3

3

Break

Q) Reverse a Part of array

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

[3,7]

arr[10] = { -3 4 2 8 3 9 6 2 8 10 }
(indices: 0 1 2 3 4 5 6 7 8 9)

-3 4 2 2 6 9 3 8 8 10

Swap (3,7)
Swap (4,6)

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

    int sp = s;
    int ep = e;

    while (sp < ep) {
        int temp = num[sp];
        num[sp] = num[ep];
        num[ep] = temp;

        sp++;
        ep--;
    }
}
```

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

↓ 1 rot.

{ 8 3 -2 1 4 6 9 }

↓ 2 rot.

{ 9 8 3 -2 1 4 6 }

↓ 3rd rot.

{ 6 9 8 3 -2 1 4 }

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

↓ After 3 rot.

{ 6 9 8 3 -2 1 4 }

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 }

↓ Reverse the first k elements

{ 6 9 8 | 4 1 -2 3 }

↓ Reverse the elements after k

{ 6 9 8 | 3 -2 1 4 }

//Pseudo Code

```
int main () {
    //input -> arr[N], K

    K = K % N;
    //Step1 -> reverse whole array
    reverse (arr, 0, N-1);

    //Step2 -> reverse the first K elements.
    reverse (arr, 0, K-1);
    //Step3 -> reverse the elements after Kth element.
    reverse (arr, K, N-1);
}
```

T.c: O(3*N) ≈ O(N)

S.c: O(1)

O(N)

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

    int sp = s;
    int ep = e;

    while (sp<ep) {
        int temp = num[sp];
        num[sp] = num[ep];
        num[ep] = temp;

        sp++;
        ep--;
    }
```

K = 10

arr[4] = { 4 1 6 9 }
0 1 2 3

↓ rotate 1

9 4 1 6 → you will get same array after doing multiples of arr.length rotation.

8

9, 10 → K = 10 : 2

↓ rotate 2

6 9 4 1

↓ rotate 3

1 6 9 4

↓ rotate 4

4 1 6 9 → 4 more rotation

Q) arr.length K

5 50 → same array

5 45 → same array

5 48 → 3 rotations

K = K % N

Q) arr.length K effective rotation

7 13 - 7 = 6 ⇒ 6

7 31 - 7 = 24 - 7 = 17 - 7 = 10 - 7 = 3

31 % 7 = 3

8 34 - 8 = 26 - 8 = 18 - 8 = 10 - 8 = 2

K = K % N

8 3 → 3 % 8 = 3

→ Break till 9:25PM

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

//idea

obs 1: max elements of the array won't be counted.

obs 2: except for max element, all the elements are valid.

// find the max element
↳ count no. of max elements → maxCount

ans: No. of elements - maxCount.

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

//Pseudo code

```
int CountGreater(int arr[N]){

    int max = Integer.MIN_VALUE;

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

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

    return N - count;
}
```

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[3] = 8 ↳ false

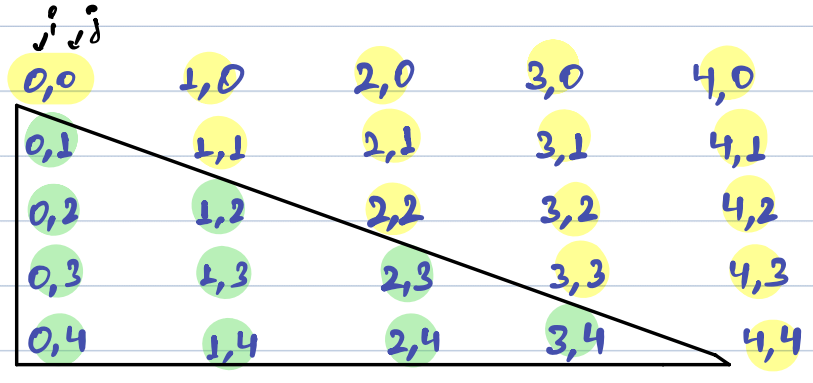
arr[6] : { 3 5 1 8 3 7 }

K=6 ↳ true

// Check all the combination of indexes

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

K=12



i → 0 1 2 3

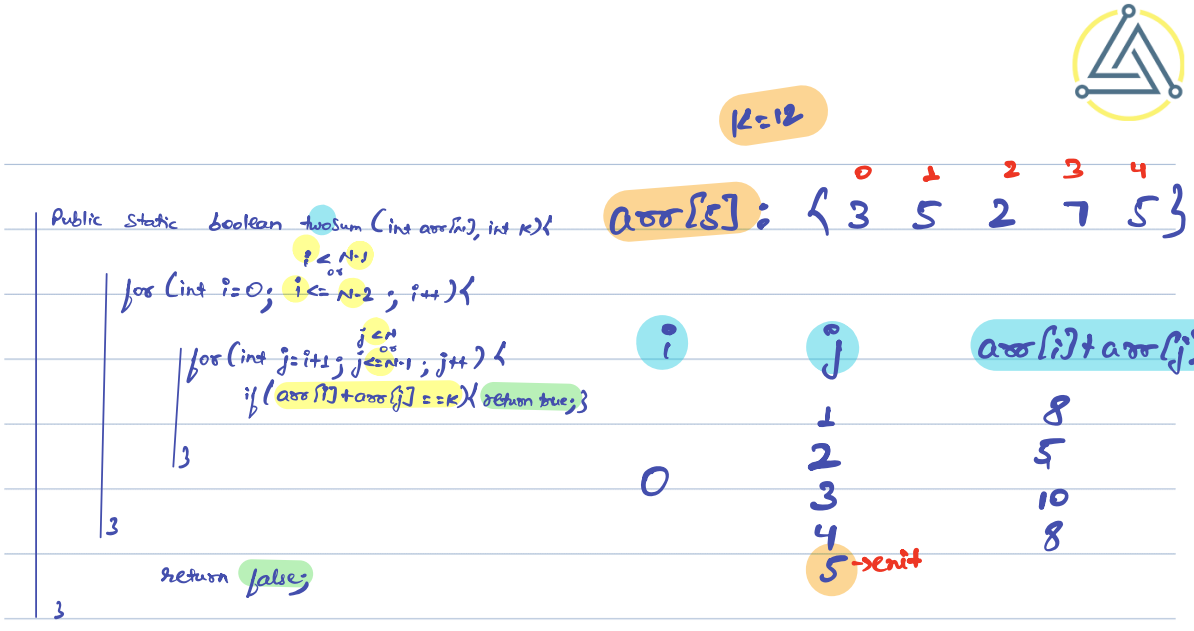
j → {1 2 3 4} {2 3 4} {3, 4} {4}

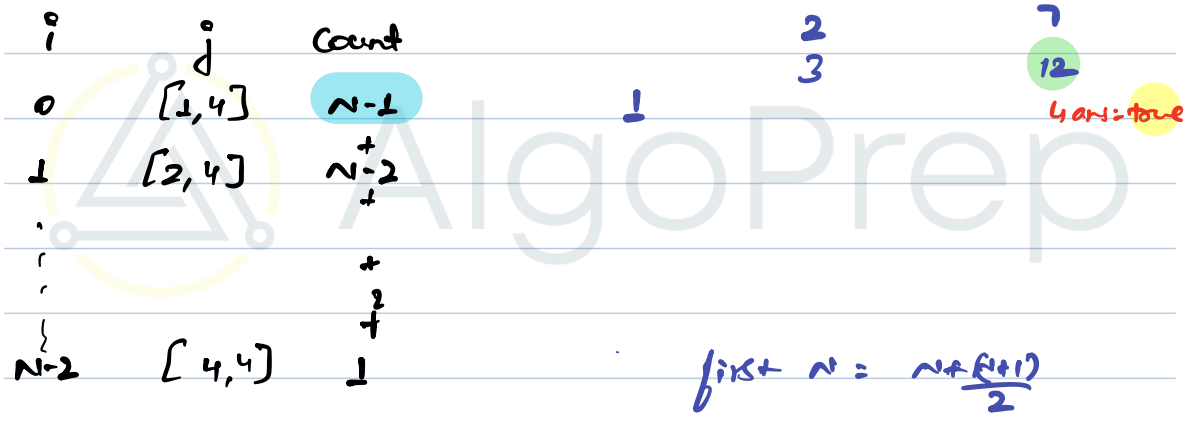
// Psuedo code

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

T.C: O(n²)

S.C: O(1)





No. of iteration: $\frac{N*(N-1)}{2} = \frac{N^2-N}{2}$

$N-1 : \frac{(N-1)*N}{2}$

$= O(N^2)$