Q: #iterations: $100N^0 = O(1)$

Time Limit Exceeded (TLE)

If your code takes more than the stipulated time (usually 1-2 secs)

1 sec $\Rightarrow$ 10^8 iterations/operations

	Algo1 $O(N)$	Algo2 $O(N^2)$	Algo3 $O(N^3)$
$N = 10$	✓	✓	✓
$N = 10^3$	✓	$(10^3)^2 = 10^6$ ✓	$(10^3)^3 = 10^9$ ✗
$N = 10^5$	✓	✗	✗
$N = 10^9$	✗	✗	✗

How to approach a problem?

- Read the **Question** and **Constraints** carefully.
- Formulate an **Idea** or **Logic**.
- Verify the **Correctness** of the Logic.
- Mentally develop a **Pseudocode** or rough **Idea of Loops**.
- Determine the **Time Complexity** based on the Pseudocode.
- Assess if the time complexity is feasible and won't result in **Time Limit Exceeded (TLE)** errors.
  - Note: In worst case we can only have **10^7 or 10^8 iterations**.
- **Re-evaluate** the **Idea/Logic** if the time constraints are not met; otherwise, proceed.
- **Code** the idea if it is deemed feasible.

Space Complexity

What is the extra space apart from input and output

Steps

1) #bytes used by our program
2) Ignore lower order terms
3) Ignore constant co-efficient

Ex:

```
fun (int N) {
    int x;        // 4 bytes
    double y;     // 8 bytes
    long z;       // 8 bytes
}
```

#bytes: 20 => O(1) space complexity

Ex:

Find the Space Complexity [Big(O)] of the below program.

```
func(int N) {      // 4 bytes
    int arr[10];   // 40 Bytes
    int x;         // 4 bytes
    int y;         // 4 bytes
    long z;        // 8 bytes
    int arr[N];    // 4 * N bytes
}
```

#bytes: 4N + 56 => O(N) S.C

Find the Space Complexity [Big(O)] of the below program.

```
func(int N) {          // 4 bytes
    int x = N;         // 4 bytes
    int y = x * x;     // 4 bytes
    long z = x + y;    // 8 bytes
    int arr[N];        // 4 * N bytes
    long l[N][N];      // 8 * N * N bytes
}
```

#bytes : $8N^2 + 4N + 16$ = $\boxed{O(N^2)}$

Question:

Inputs

```
function maxArr(int arr[], int N) {
    int ans = arr[0];
    for(i -> 1 to N-1) {
        ans = max(ans, arr[i]);
    }
    return ans;
}
```

Output

int

output

=> 4 bytes

$\boxed{S.C : O(1)}$

$N = 10^3$ ⟹ $O(N^2)$ S.C

$N = 10^5$ => $N^2 = 10^{10}$ => MLE

Introduction to Arrays (Static Arrays)

Array : Collection of elements of same data type

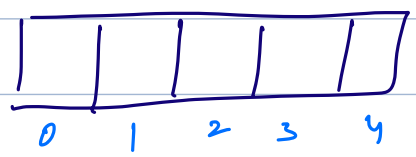
```
int arr[] = new int[10];
int[] arr = new int[10];
```

int[] → Datatype

arr → name of the array (Address of 1st element of array)

new int[10] →

arr2



size = 5

Question: Print all elements of an array

```
void printEle(int arr[]){
    for(i=0; i<arr.length; i++){
        print(arr[i]); → O(1)
    }
}
```

N

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

T.C to access an element at index i (arr[i]) => O(1)

Quiz:

1st ele => 0 => arr[0]

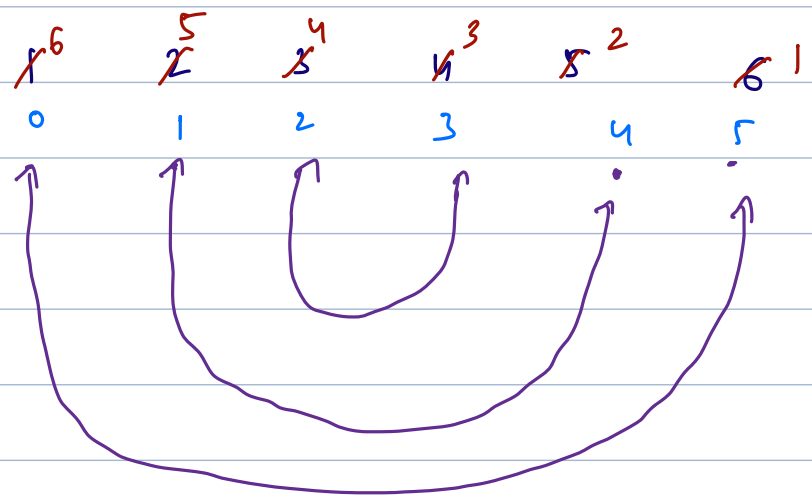
5th ele => 4 => arr[4]

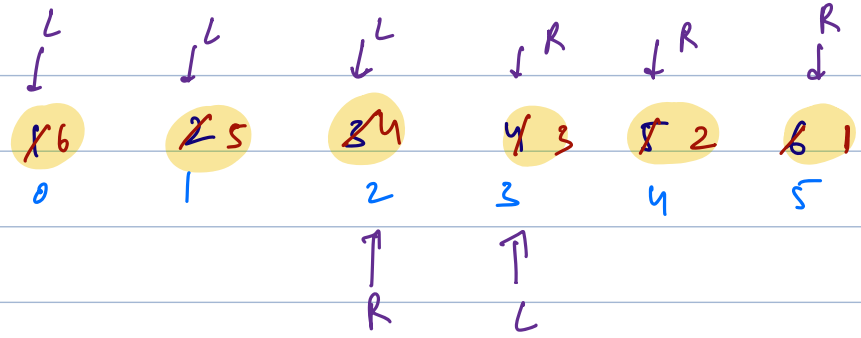
8:26

Question: Given an array of size N, reverse the array in-place

(Modify the given array itself)

Approach:






```
void reverse(int A[], int N){
O(1) {  int L = 0;
        int R = N-1;

        while( L < R )
N/2 : O(N)  {
            // swap A[L] and A[R]
            int temp = A[L];
            A[L] = A[R];
            A[R] = temp;
            L++;
            R--;
        }
}
```

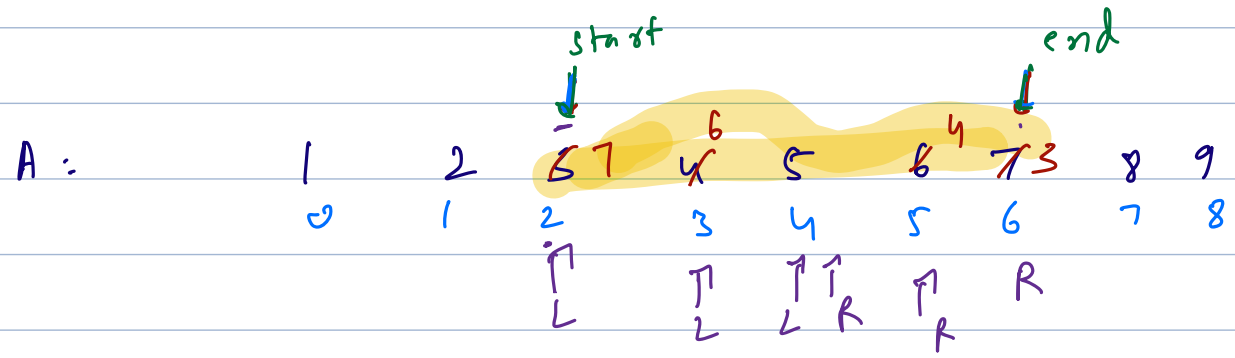
T.C:	O(N)
S.C:	O(1)

#iterations $= O\left(\frac{N}{2}\right) \Rightarrow$

$[a,b]$: $b-a+1$

$[2,6] \Rightarrow 6-2+1 = 5$ (end − start + 1)

Question: Reverse a subarray




```
void reverse( int A[], int N, int start, int end ){
    int L = start
    int R = end

    while( L < R ){
        // swap A[L] and A[R]
        int temp = A[L];
        A[L] = A[R];
        A[R] = temp;
        L++;
        R--;
    }
}
```

-> 1
-> 1
-> 1
-> 1
-> 1

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

Iterations : $\frac{(e - s + 1)}{2}$

In worst case,
start = 0
end = N - 1

start end

1 2 ~~3~~ 8 4 5 6 7 ~~8~~ 3 9 10

Question: Rotate an array by K times in clockwise direction in place $(K \le N)$

$N \le 10^6$

A: 1 2 3 4 5

0 1 2 3 4

K = 3

R_1 : 5 1 2 3 4

R_2 : 4 5 1 2 3

R_3 : 3 4 5 1 2

Brute Force

Is it easy to rotate the array once? YES

A: ~~1~~ 5 ~~2~~ 1 ~~3~~ 2 ~~4~~ 3 ~~5~~ 4

0 1 2 3 4

temp = 5

```
void rotate (int A[], int N) {

    temp = A[N-1]

    for (i = N-1; i ≥ 1; i--) {
        A[i] = A[i-1];
    }
    A[0] = temp;
}
```

T.C: $O(N)$

S.C: $O(1)$

$A = \underset{0}{1} \quad \underset{1}{\cancel{2}\,1} \quad \underset{2}{\cancel{3}\,1} \quad \underset{3}{\cancel{4}\,1} \quad \underset{4}{\cancel{5}\,1}$

```
for (i = 1; i < N; i++) {
    A[i] = A[i-1];
}
```

call rotate method k times

T.C: $O(K \cdot N)$

S.C: $O(\ )$

Worst case $= 10^6 \times 10^6$

$= 10^{12}$ iterations

TLE

Efficient Approach :

A :	1	2	3	4	5	6	7
	0	1	2	3	4	5	6
R_1 =	7	1	2	3	4	5	6
R_2 :	6	7	1	2	3	4	5
R_3 :	5	6	7	1	2	3	4
R_4 =	4	5	6	7	1	2	3
R_5 :	3	4	5	6	7	1	2

Observation:

When we rotate by K times, the last K elts will come to the front

$N-K$ $\qquad\qquad\qquad\qquad$ K

A : $a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9 \quad a_{10} \quad a_{11}$

Ans = $a_7 \quad a_8 \quad a_9 \quad a_{10} \quad a_{11} \quad a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6$

K $\qquad\qquad\qquad\qquad$ $N-K$

A : $a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9 \quad a_{10} \quad a_{11}$

rev(A) = $[\, a_{11} \quad a_{10} \quad a_9 \quad a_8 \quad a_7 \,]\,[\, a_6 \quad a_5 \quad a_4 \quad a_3 \quad a_2 \quad a_1 \quad a_0 \,]$

0 ... K-1 K ... N-1

K elements

Reverse Reverse

$a_7 \quad a_8 \quad a_9 \quad a_{10} \quad a_{11} \qquad a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6$

0 . . . . . K-1 K N-1

Steps

1) reverse (A, 0, N-1)
2) reverse(A, 0, K-1)
3) reverse(A, K, N-1)

```
void rotate (int A[], int K) {
    K = K%N
1   reverse (A, 0, N-1)        -> N
2   reverse( A, 0, K-1)        -> K
3   reverse( A, K, N-1)        -> N-K
}
```

$\#\text{iterations} = N + \cancel{K} + N - \cancel{K}$

$= 2N$

$= \boxed{O(N)}$

S.C: $O(1)$

HW: Rotate anticlockwise direction

What if $K \geq N$?

$A = a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4$

$R_1 = a_4 \quad a_0 \quad a_1 \quad a_2 \quad a_3$

$R_2 = a_3 \quad a_4 \quad a_0 \quad a_1 \quad a_2$

$R_3 = a_2 \quad a_3 \quad a_4 \quad a_0 \quad a_1$

$R_4 = a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_0$

$R_5 = a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4$

$R_6 = a_4 \quad a_0 \quad a_1 \quad a_2 \quad a_3$

$R_7 = a_3 \quad a_4 \quad a_0 \quad a_1 \quad a_2$

$N = 5$

$\Rightarrow$ $K = 0, 5, 10, 15, 20 \ldots$ $\Rightarrow$ $\% N = 0$

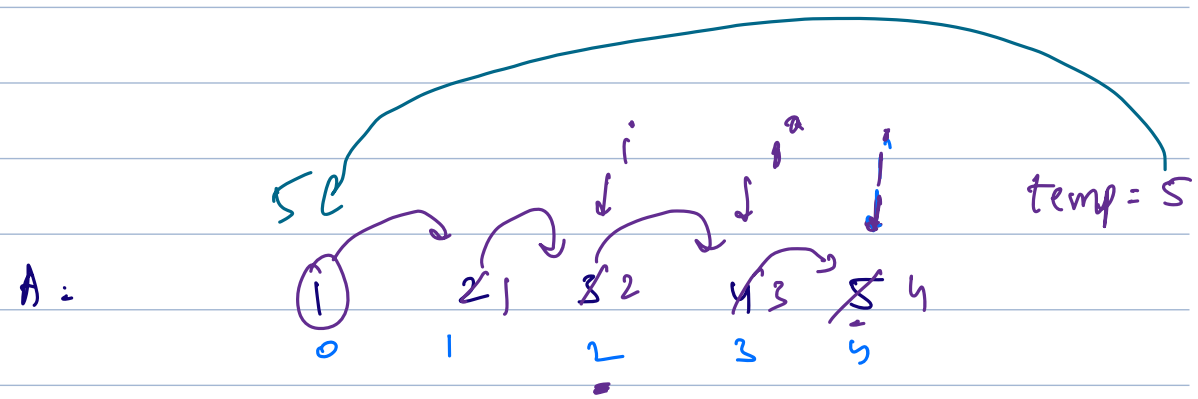
$K = 1, 6, 11, 16, 21, 26 \ldots$ $\% N = 1$

$K = 2, 7, 12, 17, 22 \ldots$ $\% N = 2$

$K = 3, 8, 13, 18, 23 \ldots$ $\% N = 3$

$K = 4, 9, 14, 19 \ldots$ $\% N = 4$

$K = K \% N$

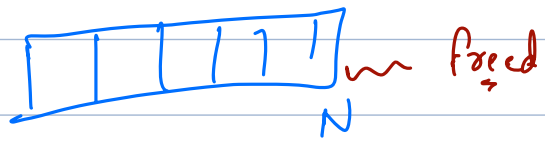


$A[i] = A[i-1]$

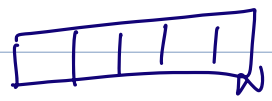
```
for (i = 1; i<N; i++) {
    A[i] = A[i-1]
}
```

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

I1



I2



I3: