Agenda:

- Space Complexity
- Introduction To Arrays.
- Reverse the array.
- Rotate the array k times.
- Dynamic Arrays.

$2^k = N$

$2 \times 2 \times 2 \times \ldots \times 2 = N$ (k times)

$\log_2 N = k$

$4N^2 + 3N + 1$

$4N^2$ $O(N^2)$

$O(N^2)$

$1 \leq N \leq 10^5$

10^{10} 10^8

TLE

Space Complexity:

```
func (int N)
{
    int  x;  → 4 bytes (32 bits)
    long y;  → 8 bytes (64 bits)
    int  Z:    int [] A: new int [N]
}                          16 Bytes.   O(1)
```

1 byte = 8 bit

0 or 1

long → 8 bytes

Input Space (X)

Output Space (X)

O(1)

```
fun() (A[], N)
{   ans = A[0];
    for (i → 1 to N-1)
    {   ans = max(ans, A[i])
    }
    return ans;
}
```

Introduction To Arrays:

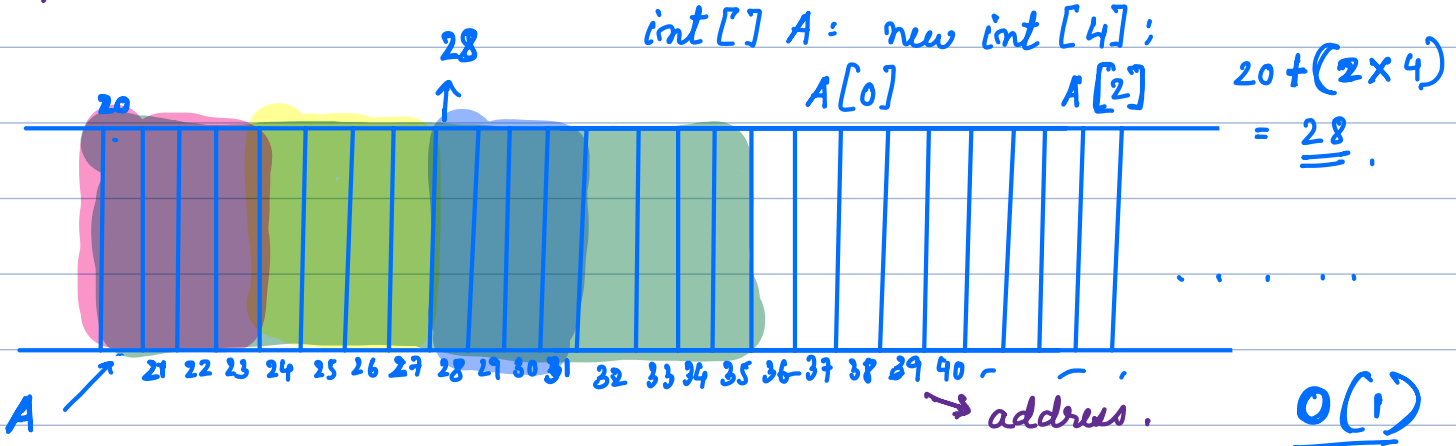
A[0] A[N-1]

int arr[N]; Indexing starts from 0.
ends at N-1

int → 4 bytes.

int[] → 4

Time Complexity for accessing random element in the array. → O(1)



O(1)

int arr[] : new int[5];

Print all elements in the array.

```
fun (int[] A)
{  int N = A.length;
   for (i → 0 to N-1)
   {  println(A[i]);
   }
}
```

A.length();

Break : 8:07 AM

Swapping.

	0	1	2	3
A:	5	4	9	3

swap(1, 3)

```
int temp = A[1];
A[1] = A[3]
A[3] = temp;
```

Question

Reverse an array.

i	res
N-1 : 4	0
N-2	1

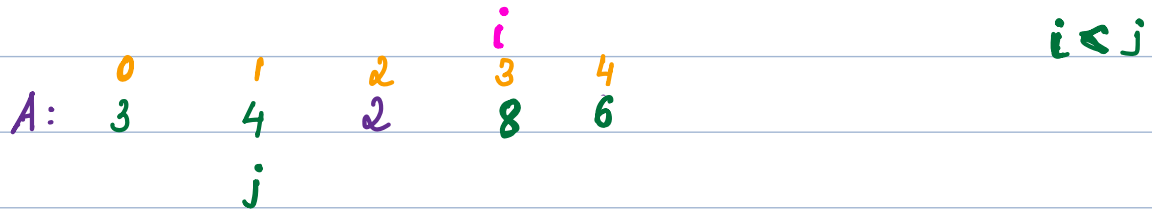
	0	1	2	3	4
A:	3	4	2	8	6
res:	-	-	-	-	-

	0	1	2	3	4
A':	6	8	2	4	3

```
int[] reverseArray(int[] N)
{  int[] res = new int[N.length];
   for (i = N.length-1 ; i ≥ 0 ; i--)
   {   res[N.length-1-i] = A[i];
   }
   return res;
}
```

TC: O(N)

SC: O(N)




```
int[]  reverseArray (int[] A)
{  int  i = 0;   int j = A.length - 1;
   while (i < j)
   {  swap (A[i], A[j]);
      i++; j--;
   }
   return A;
}


void  swap (int[] A, int i, int j)
{


}
```

TC: O(N)

SC: O(1)

Question: Reverse an array within a given range.

	0	1	2 (s)	3	4 (e)
A:	3	4	2	8	6

s = 2 e = 4

Ans →

	0	1	2	3	4
A:	3	4	6	8	2

initial value of i & j will be s & e

same as last program

Question: Given an array of size N, rotate the array from right to left k times. You can't use a new array.

	0	1	2	3	4
A:	1	2	3	4	5
R(1):	5	1	2	3	4
R(2):	4	5	1	2	3
R(3):	3	4	5	1	2
R(4):	2	3	4	5	1
R(5):	1	2	3	4	5

k = 3 → 5

3 % 5 → 3

k % N

k % N

4 → 3

k

6 → 1
7 → 2
8 → 3
9 → 4
10 → 0

11 → 1
12 → 2
13 → 3

temp = 5

```
void rotateK(int[] A, int K)
{   int N = A.length;
    K = K % N;
    for(i=0; i<K; i++)
    {  int temp = A[N-1];
       for(j=N-2; j>=0; j--)
       {  A[j+1] = A[j];
       }
       A[0] = temp;
    }

}
```

	j				
	0	1	2	3	4
A:	5	1	2	3	4

K * N ⇒ O(K*N)

TC → O(N²)

SC → O(1)

1 ≤ N ≤ 10^5

Optimized Approach:

A:

0	1	2	3	4	5	6
5	6	7	1	2	3	4

	0	1	2	3	4	5	6	
	1	2	3	4	5	6	7	k = 3
Reverse (A) →	7	6	5	4	3	2	1	
Reverse (A, 0, K-1) →	5	6	7	4	3	2	1	
Reverse (A, K, N-1) →	5	6	7	1	2	3	4	

Reverse (A) → O(N)

Reverse (A, 0, K-1) → O(N) +

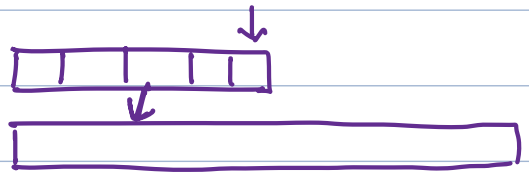
Reverse (A, K, N-1) → O(N) +

3 * O(N)

⇒ O(N)

Dynamic Arrays:

Array → length static.
Dynamic array → You can add



Next Lecture: Prefix Sum
Problems on Prefix Sum.

Doubts:

```
for(i=0; i<N; i++)
{
    int i
}
```