Today's Agenda:-

1. Space Complexity
2. Introduction To Arrays
3. Reverse the array.
4. Rotate array K times.
5. Dynamic Arrays.

Starting at 7:05 AM.

Space Complexity:

→ Max $\overset{\text{Extra}}{\wedge}$ space that is utilised at any point in time during running the algorithm.

→ Can be determined using Big O

$$\begin{tikzcd} \text{Input} \arrow[r] & \boxed{\text{my ALGORITHM}} \arrow[r] \arrow[d, Rightarrow] & \text{Output} \\ & \text{Auxillary / Extra space my algo is using other than Input / Output is called } \underline{\underline{\text{S.C.}}} & \end{tikzcd}$$

<u>Example:-</u>

```
void func (int N) {     // 4B  ✗
    int x;     // 4B  ✓
    int y;     // 4B  ✓
    long z;    // 8B  ✓
}
```

16B.

O(1).

<u>Quiz 1:-</u>

```
func (int N) {     // 4 B  ✗
    int arr[10];     // 40 B  ✓
    int x;     // 4B  ✓
    int y;     // 4B  ✓
    long z;    // 8B  ✓
    int [] a = new int[N];   // (4*N)B
}
```

$56 + 4N = O(N)$

Quiz 2 :-

```
func (int N) {        // 4B ✗
    int x = N;        // 4B ✓
    int y = x * x;    // 4B ✓
    long z = x + y;   // 8B ✓
    int[] arr = new int[N];   // 4N B ✓
    long[][] l = new long[N][N];
                      // 8 * N^2 B ✓
}
```

$$16 + 4N + 8N^2 = O(N^2)$$

Another Question :-

```
int maxArr ( int arr[] ✗, int N ✗) {
    int ans = arr[0];   4B ✗
    for ( i from 1 to N-1) {
      4B.
        ans = max(ans, arr[i]);
    }
    return ans;  // Part of output
}
```

$O(1)$.

int i = 1;

$i \nearrow \boxed{3}$

Introduction To Arrays :-

→ Array is a collection of same type of data.
ex:- int, float, char etc.
double.

int arr[N];

There is an array of size N.
'arr' is the name of that array.
int is the datatype of the values stored in that array.

Quiz 3 :-

Array of size N.

0	1	2	3	4	-	-	-	N-1
5	4	3	6	2	—	-	-	8

Printing elements of array.

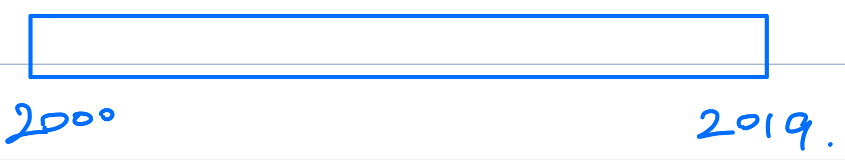
```
void printArray( int[] arr, int N) {
    for (i = 0; i <= N-1; i++) {
        print (arr[i]);
    }
```

T.C → O(N)

SC → O(1)

Quiz 4

int arr[5];



T.C is O(1) to access any element in my array.

$*(arr)$.
$*(arr+1)$.
$*(arr + i-1)$.

Quiz 5

```
                         0    1    2   3   4
int   arr[5]  =  { 5, -4, 8, 9, 10 }
```

print (arr[0] + arr[4]);

Problem 1:- Reverse the array.

Ex:- arr = {1, 2, 3, 4, 5}

Output = {5, 4, 3, 2, 1}.

Input :-

0	1	2	3	4	5
6	5	4	3	2	1
		j			
			i		

```
a[i] = a[j];     ✗
a[j] = a[i];
```

temp = 2;

```
temp = a[i];
a[i] = a[j];
a[j] = temp;
```

i++;
j--;

$i > j$

$i == j$

Until $i < j$; we should continue;

		i, j		
5	4	3	2	1
0	1	2	3	4.

Code:

```
void reverse (int arr[], int N){
    int i = 0, j = N-1;  int temp;
    while (i<j) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        i++; j--;
    }
}
```

T.C -> O(N).
S.C -> O(1)

Reverse in a range.

Ex:-

$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \end{array}$$

arr = { 1, 2, 3, 4, 5 }

l = 1

r = 3

$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \end{array}$$

Output: arr = { 1, 4, 3, 2, 5 }

```
void reverse ( int arr[], int N, int l, int r)
    int i = l; j = r;        int temp;
    while (i<j)  {
        temp   =  arr[i];
        arr[i] =  arr[j];
        arr[j] =  temp;
        i++;  j--;
    }
3.
```

T.C → O(N)

S.C → O(1)

8:30.

Problem 2:- Rotate K times.

Given an array 'arr' of size N. Rotate the array from right to left 'K' times.

Ex:- arr = {1, 2, 3, 4, 5}
K = 4.

0	1	2	3	4	K = 1.
5	1	2	3	4.	

temp = arr[4];
= 4.

i = ~~4~~. ~~3~~. ~~2~~. ~~1~~ 0

a[0] = temp;

0	1	2	3	4	K = 2;
4.	5	1	2	3.	

Code

```
rotate (int arr[], int N, int K)  {
                    int temp;
    for (j = 1; j <= K; j ++)  {
                   temp = arr [N-1];
        for (i = N-1; i > 0; i--) {
                   arr [i] = arr [i-1];
        }.
        arr [0] = temp;
    }
}.
```

T.C → $O(K * N)$

S.C → $O(1)$

Can I optimise it ?

1 2 3 4 5 6 7.

K = 3.

5 6 7 1 2 3 4

1 2 3 4 5 6 7.

Step 1:- Reverse entire array.

7 6 5 4 3 2 1

Step 2 :- Range reverse from 0 to K-1 index

&

from K to N-1.

Output :-

5 6 7. 1 2 3 4.

Expected Output

Code :-

```
rotate ( int arr[], int N, int K) {
    K = K % N;
    reverse (arr, N, 0, N-1);

    reverse ( arr, N, 0, K-1 );
    reverse ( arr, N, K, N-1);
}
```

T.C → O(N).

S.C → O(1).

N = 10;   K = 215

215 % 10 = 5.

10 + 10 + 10 + ...

210

Dynamic Arrays:

Q:- What is the drawback of static arrays?

A:- It has a fixed size.

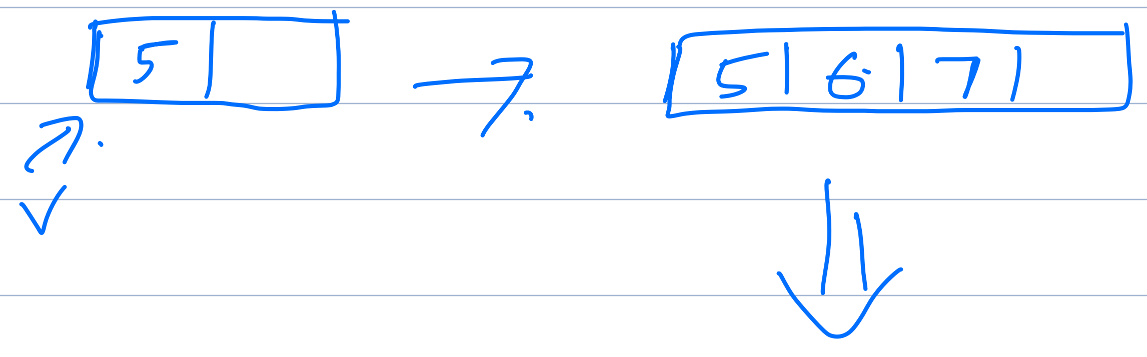
→ Automatic resizing.

Strength:- → Fast lookup
→ Size is flexible

Weakness:- → Comparatively slower.

Java.	ArrayList.
C++:	Vector.
Python	List.

```
vector <int> v;
```



v. push_back (5);

1;

v. push_back (6);

5	6	7	8	9	10	11

↓

7 : $\boxed{1}$

↓

8 : $\boxed{4}$; 9 $\boxed{1}$ 10 $\boxed{1}$ 11 $\boxed{1}$

12 → $\boxed{8}$,

Sometimes while doing an insertion, it takes O(N) iterations.
other time it takes just 1 iteration..

N = 2, 4, 8, 16, 32, 64.

N=64: 58 times it is taking 1 iteration
Remaining 6 times it is taking N iterations.

⇒ Amortized O(1).

Doubts :-

```
K = 100;
while ( K > 0) {
   if ( K % 5 == 0 ) {
      for ( i = 0; i < N; i++ ) {
      }
   K--;
}
```

N
+1

```
void solve (int N) {
   for ( i = 0 ; i < pow(2, N); i++ ){
      j = i;
      while (j > 0) {
         j -= 1;
      }
   }
```

$0 + 1 + 2 + 3 + 4 + 5 + \dots 2^N$

$\frac{N(N+1)}{2}$.

$2^N \frac{(2^N+1)}{2} = \frac{2^{2N}}{2} + \frac{2^N}{2}$.

$= 4^N \times \frac{1}{2}$.

$$= (2^2)^N$$

$N = 7$

$K = 9.$

$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7$$

$$\downarrow 1$$

$$7 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6$$

$$\downarrow 2$$

$$6 \quad 7 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$

$$\downarrow 3$$

$$5 \quad 6 \quad 7 \quad 1 \quad 2 \quad 3 \quad 4$$

$$\downarrow 4$$

$$4 \quad 5 \quad 6 \quad 7 \quad 1 \quad 2 \quad 3$$

$K \% N$ = K − Highest multiple of N which is less than K.