Space Complexity

How much extra memory the program uses

```
func(int N) {
    int x
    int y
    int z
}
```

4 Bytes
4
4 = 12 bytes
4 = O(1)

We do NOT include the space taken by N since it is part of input

Quiz

```
func(int N) {
    int arr[10]
    int x
    int y
    long z
    int arr[N]
}
```

40
4
4
8
4N
= 4N + 60
= O(N)

Quiz

```
func ( int N) {
    int x = N                    4
    int y = x * x                4
    long z = x + y               8
    int arr (N]                  4N
    long l [N] [N]               8N^2
}
```

$8N^2 + 4N + 16$

$O(N^2)$

Q

```
int calc (int ar[], int N) {
    int ans = ar [0]             // 4B
    for ( i = 1 ; i < N ; i++) {  // 4B
        ans = max (ans, ar [i])
    }
    return ans
}
```

$O(N)$ $O(1)$

Space = $4 + 4 = 8$

$O(1)$

Array

int arr[5] // 5 integers ⇒ 20 bytes

First element → arr[0]

0 1 2 3 4 5 ..... n-1

Q) First & last index of array

first → 0 last → n-1

Print all numbers of array

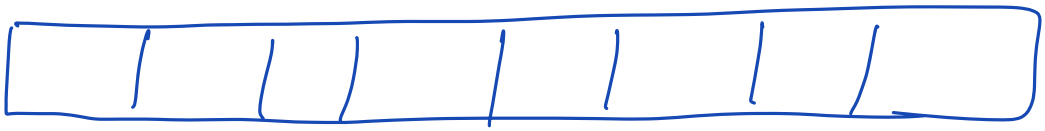
```
void print (int arr[], int n) {
    for (i=0; i<n ; i++) {
        print (arr[i])
    }
}
```

Quiz

arr[i] access time is O(1)

arr[1000]

print(arr[534])



$i^{th} \Rightarrow 656 + 4 \times i$

Quiz

int arr[5] = { 5, -4, 8, 9, 10 }

Sum of 1st & 5th elem

→ arr[0] + arr[4]

Q1 Given an integer array, reverse the entire array [O(1) SC]

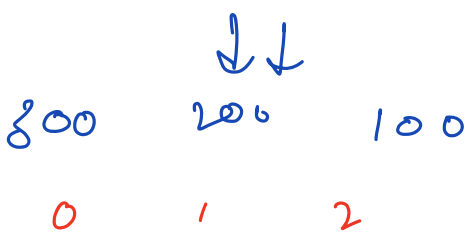
Eg- { -1 4 7 6 -2 7 8 10 }
0 1 2 3 4 5 6 7

ans= 10 8 7 -2 6 7 4 -1

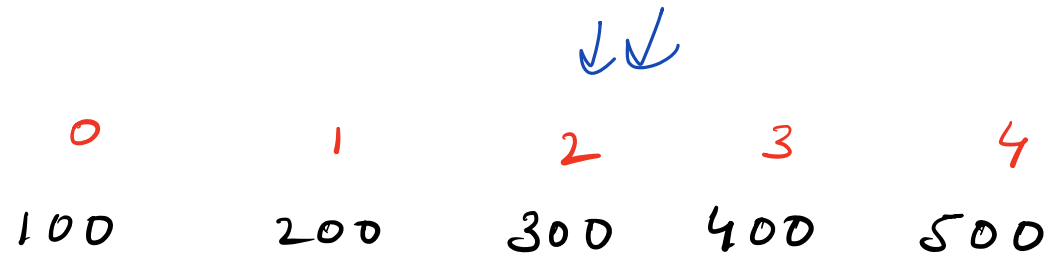
-1 4 7 -2 6 7 8 10

j i

i	j	
0	7	swap
1	6	swap
2	5	swap
3	4	swap
4	3	STOP!!!



i < j is only valid

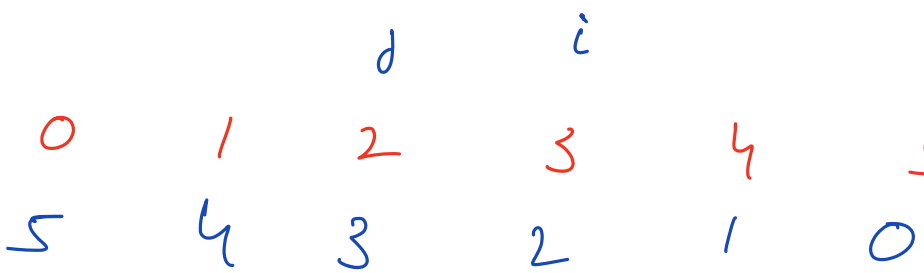


i	j		
0	4	swap	i<j
1	3	swap	valid
2	2	stop	

Code:

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

TC: O(N) SC: O(1)



Q Reverse the part of the array starting at idx s & ending of idx e .

100	200	300	400	500
0	1	2	3	4

s = 1 e = 3

ans: 100 400 300 200 500

```
reverse_part(arr[], N, s, e ) {
    int i=s   j=e
    while( i<j ) {
        swap( arr[i], arr[j] )
        i++        j--
    }
}
```

TC: O(N) SC: O(1)

Q3 Given an array, Rotate the array from first to last k times.

Eg	3	-2	1	4	6	9	8
K=1	8	3	-2	1	4	6	9
K=2	9	8	3	-2	1	4	6
K=3	6	9	8	3	-2	1	4
K=4	4	6	9	8	3	-2	1

1 rot → O(n)

k rot → O(nk)

Take K=3

3 -2 1 4 6 9 8 original

6 9 8 3 -2 1 4 rotated K=3 times

Last 3 elems ⇒ First 3 elem

- It is a good idea to reverse the whole array

3	-2	1	4	6	9	8	original
8	9	6	4	1	-2	3	reverse
6	9	8	3	-2	1	4	✓ the answer we want

Q Any relationship here you can observe ?

⇒ First k reversed
Last n-k reversed

Code

0) $k = k \% n$

1) reverse whole array $O(N)$

2) reverse first k
reverse (0, k-1) $O(N)$

3) reverse remaining
reverse (k, n-1) $O(N)$

⇒ $O(N)$

Imagine $K > N$

Eg. A : 100 200 $K = 5$

what happens ? —

$K=0$	a_0	a_1	a_2	a_3	a_4	a_5
$K=1$	a_5	a_0	a_1	a_2	a_3	a_4
$K=2$	a_4	a_5	a_0	a_1	a_2	a_3
$K=3$	a_3	a_4	a_5	a_0	a_1	a_2
$K=4$	a_2	a_3	a_4	a_5	a_0	a_1
$K=5$	a_1	a_2	a_3	a_4	a_5	a_0
$K=6$	a_0	a_1	a_2	a_3	a_4	a_5
$K=7$	a_5	a_0	a_1	a_2	a_3	a_4
$K=8$	a_4	a_5	a_0	a_1	a_2	a_3
$K=9$	a_3	a_4	a_5	a_0	a_1	a_2

Obs : Answer after rotating K times = Answer after rotating $K \% n$ times.

Thus before running code, $K = K \% n$

Dynamic Arrays

int arr[5] Size is fixed

Dynamic arrays in diff languages

C++	Java	Python
vector	ArrayList	List

- Arraylist : Part of java.util package

```
Arraylist<Integer> alist = new Arraylist<Integer>();
alist.add(10)  // Adds 10 to the end of arraylist
alist.add(20)  // 10,20
alist.size()   // Returns current size of arraylist
               ⇒ 2
alist.get(0)   // Element at 0th idx ⇒10
alist.get(i)   // Element at ith idx
```

done

0 1 2 ... 7