Space Complexity

The max space (worst case) that is utilised at any point in time during running the algorithm.

```
func(int N) {   // 4 bytes
    int x ;     // 4 bytes
    int y ;     // 4 bytes
    long } ;    // 8 bytes
}
```

The total memory used is 20B

The space complexity is O(1)

Quiz 1:

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

Total space : $40 + 20 + 4*N$

$= 60 + 4*N$

SC: O(N)

Quiz 2:


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

Total space: $20 + 4N + 8N^2$

SC: $O(N^2)$

Find Space Complexity:

```
int maxArr(int arr[], int N) {      // 4 + 4N Bytes
    int ans = arr[0];               // 4 Bytes
    for i -> 1 to N-1 {             // 4 Bytes for (i)
        ans = max(ans, arr[i]);
    }
    return ans
}
```

$4 + 4N + 4 + 4$

SC: $O(1)$ SC of whole function: $O(N)$

Should we consides input space?

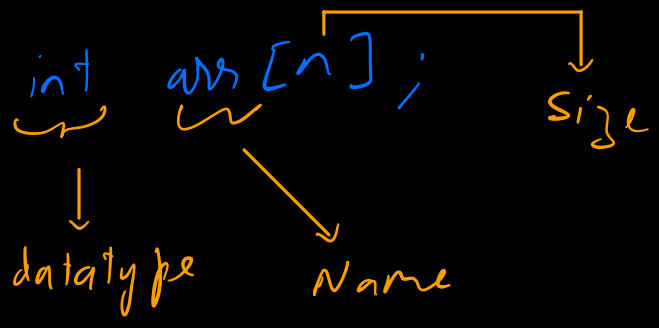
NO, we only consides auxiliary space.

→ arr[] is already given to us, we didn't create it, hence it'll not be counted in the Space Complexity.

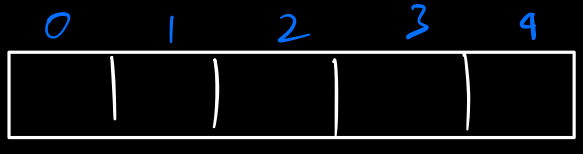
Arrays

→ Array is the collection of same types of data

→ The type can be anything: int, float, char, class etc.

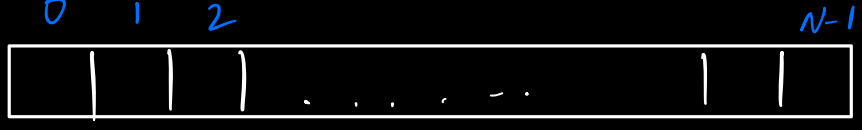


int arr[5];



⟶ Continous memory allocation

⟶ Linear data structure



[0, x] => # elements = N

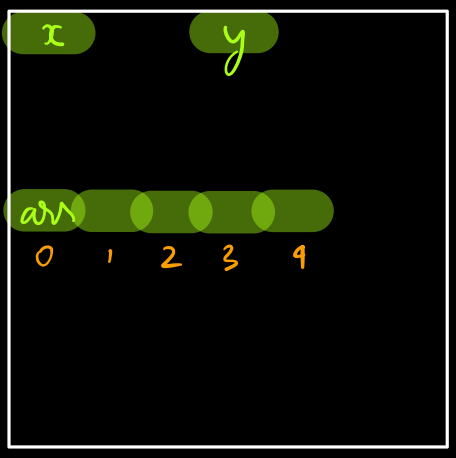
x =>

[a b] => b-a+1=N

x-0+1=N

x = N-1

Why index starts from zero?




```
int x
int y
int arr[5]
```

arr[0] = *(arr + 0)

arr[1] = *(arr + 1)

arr[4] = *(arr + 4)

Random Access -> You can access any index of a array in O(1) time.

Q: Print all elements of the array.

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

Quiz : Access time in arr[idx] => O(1) as Random Access

Quiz: `int arr[5] : {5, -4, 8, 9, 10}`

Print sum of 1st and 5th element

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

Q Given an array 'arr' of size N. Find the maximum element.

N = 5

arr = {9, 1, 3, 5, 7}

ans = 9

⟶ Consider 1st element as max and iterate and update if some better element found.

→ initialise ans = arr[0]

→ iterate from i → 1 to N-1

⟶ if (arr[i] > ans)

ans = arr[i]

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

→ We can also initialise ans with INT_MIN and iterate from 0 to N-1

Ques: Given an array 'arr' of size 'n'.

Check if there exist pair (i, j) such that arr[i] + arr[j] = K and i != j

N = 5

arr = {9, 1, 3, 5, 9}

K = 12

ans = True

Quiz:

	0	1	2	3	4	
arr[5] = {	3	5	2	7	3 }	K = 6

Ans = True (0, 4)

Quiz:

	0	1	2	
arr[3] = {	4	2	7 }	K = 8

Ans = False

Brute Force => Consider all the pairs and check.

i \ j	0	1	2	3	4
0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)

4 (4,0) (4,1) (4,2) (4,3) (4,4)

```
funct (int arr[], int N, int K) {
    bool answer = false;
    for i -> 0 to N-1 {
        for j -> 0 to N-1 {
            if (i != j && arr[i] + arr[j] == K) {
                answer = True;
            }
        }
    }
    return answer
}
```

TC: $O(N^2)$ SC: $O(1)$

Optimised Brute Force

i ↓ \ j →	0	1	2	3	4
0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
4	(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

```
funct (int arr[], int N, int K) {
    bool answer = false;
    for i -> 0 to N-1 {
        for j -> i+1 to N-1 {
            if (arr[i] + arr[j] == K) {
                answer = True;
            }
        }
    }
    return answer
}
```

i	j [i+1 N-1]	# of iteration
0	[1 N-1]	N-1
1	[2 N-1]	N-2
2	[3 N-1]	N-3
⋮		
N-2	[N-1 N-1]	1
N-1	[N N-1]	

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

TC: $\frac{N^2}{2} + \frac{N}{2} - N = O(N^2)$ SC: $O(1)$

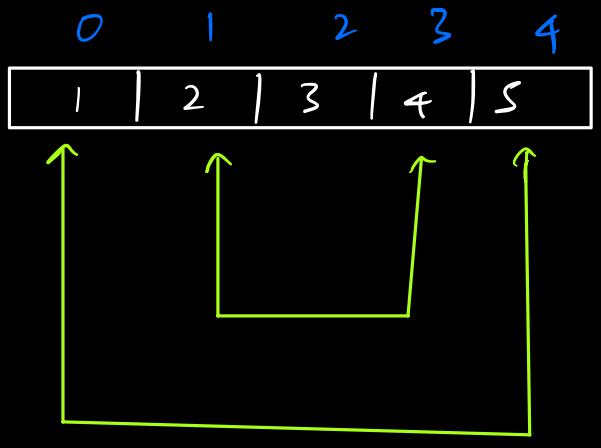
Q Given an array of size N. Reverse the array.

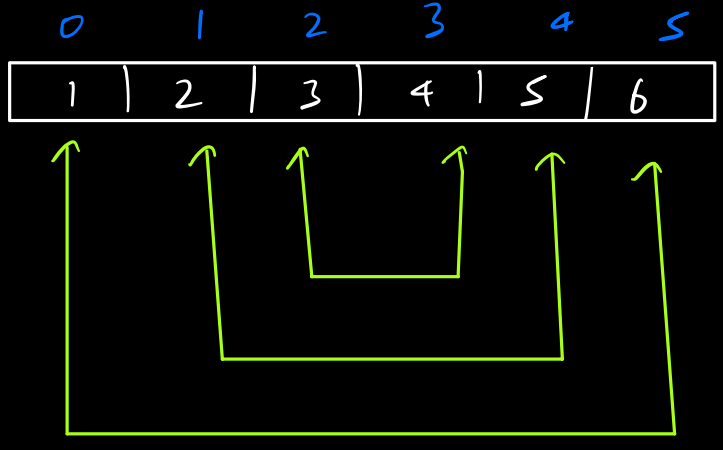
N = 5

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

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

Direct Approach





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

TC: O(n)
SC: O(1)

Q: Given an array of size n and integers 'l' 'r'. Reverse array from 'l' 'r'

n = 5

	0	1	2	3	4
arr = {	1,	2,	3,	4,	5 }

l = 1

r = 3

output => { 1 4 3 2 5 }

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

TC: $O(r-l+1)$

$\approx O(N)$

SC: $O(1)$

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

N = 5

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

K = 2

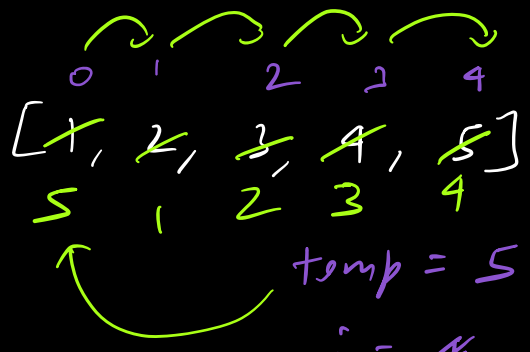
1 2 3 4 |5

↓ 1 rotation

5 1 2 3 4

↓ 2nd rotation

ans = 4 5 1 2 3

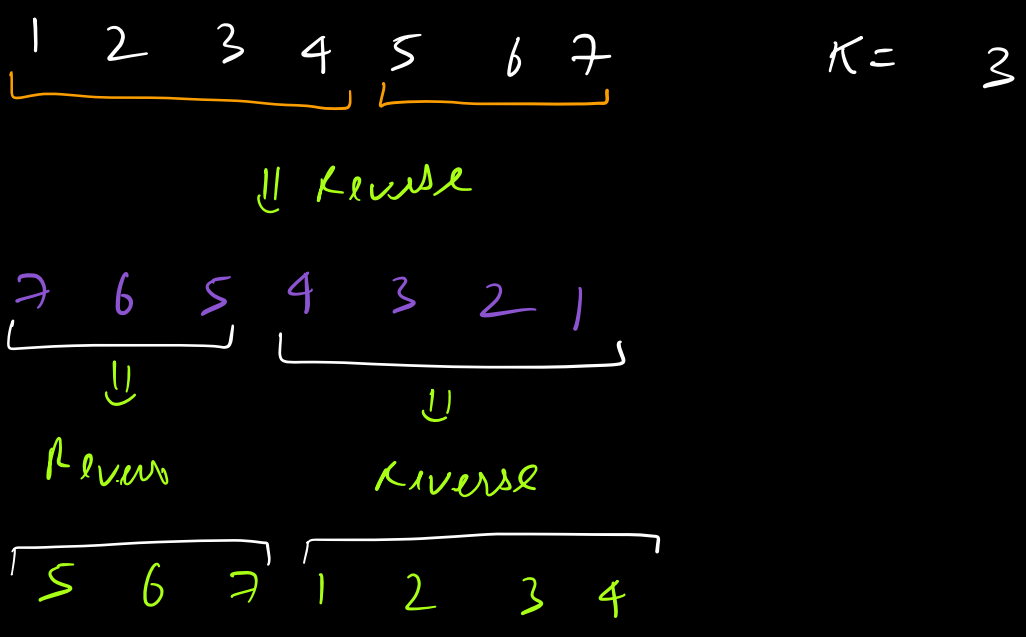



```
rotateArray(int arr[], int N, int K){
    for(int i=0; i<K; i++){
        int temp = arr[N-1];
        for(j=N-1; j>=1; j--){
            arr[j] = arr[j-1]
        }
        arr[0] = temp
    }
}
```

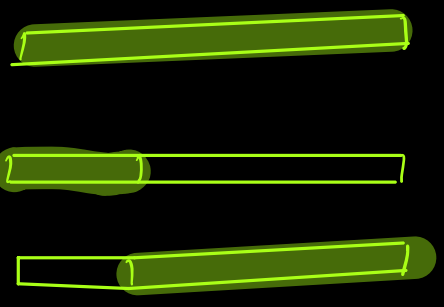
TC: O(N * K)

SC: O(1)

Optimal:



```
rotateArray(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)
}
```



TC: $O(3N) \approx O(N)$

SC: $O(1)$

Corner case:

$5 \% 9 = 5$

arr = [1 2 3]

K = 5

K = K % N

= 5 % 3

= 2

1 2 3

↓ 1st not

3 1 2

↓ 2nd not

2 3 1

↓ 3rd not

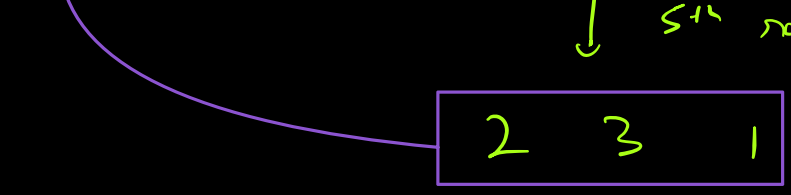
1 2 3 ⇒ After N rotation I get original array

↓ 4th not

3 1 2

↓

same



Arr = [5 10 2 1 6] K = 2

ans = [1 6 5 10 2]

Q What are the drawback of arrays ?

→ It has fixed size

Dynamic Arrays

→ A array with improvement => automatic resizing

→ It expands as you add elements.
So you don't need to know size beforehand.

Strengths :

1) Variable Size

2) Fast lookup O(1) similar to normal array

Weakness :

1) Usually it has O(1) insertion time.
2) But when the array is full it actually copy the whole array to a new array
3) It wastes some memory.

Java, C#: ArrayList<String>

C++ : vector<string>

Python : thislist = [ ]

Time Limit Exceeded Error

→ It means the program did not completed in given time.

What to choose time vs Space.

→ Time is given priority
  → lesser response time
  → User experience ⇒ revenue
  → Space can be increased by buying more
  → Space can be handeled by scalling

(adding more resources).

Online Judges Why TLE?

→ 1 Ghz processing speed

10^9 instructions

We assume that 1 iteration will include 10 operations.

How many iterations allowed = 10^8

We assume that 1 iteration will include 100 operations.

How many iterations allowed = 10^7

Mostly we can perform $10^7 - 10^8$ iteration in 1sec and pass the time limit

Doubts

```
for(int i = 1; i * i * i <= N; i++){

}
```

$$i^3 <= N$$

$$\sqrt[3]{i^3} <= \sqrt[3]{N}$$

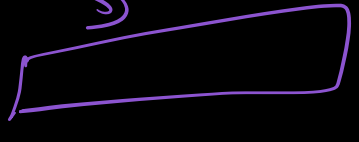
$$i <= \sqrt[3]{N}$$

$$O(\sqrt[3]{N}) \simeq O(N^{1/3})$$

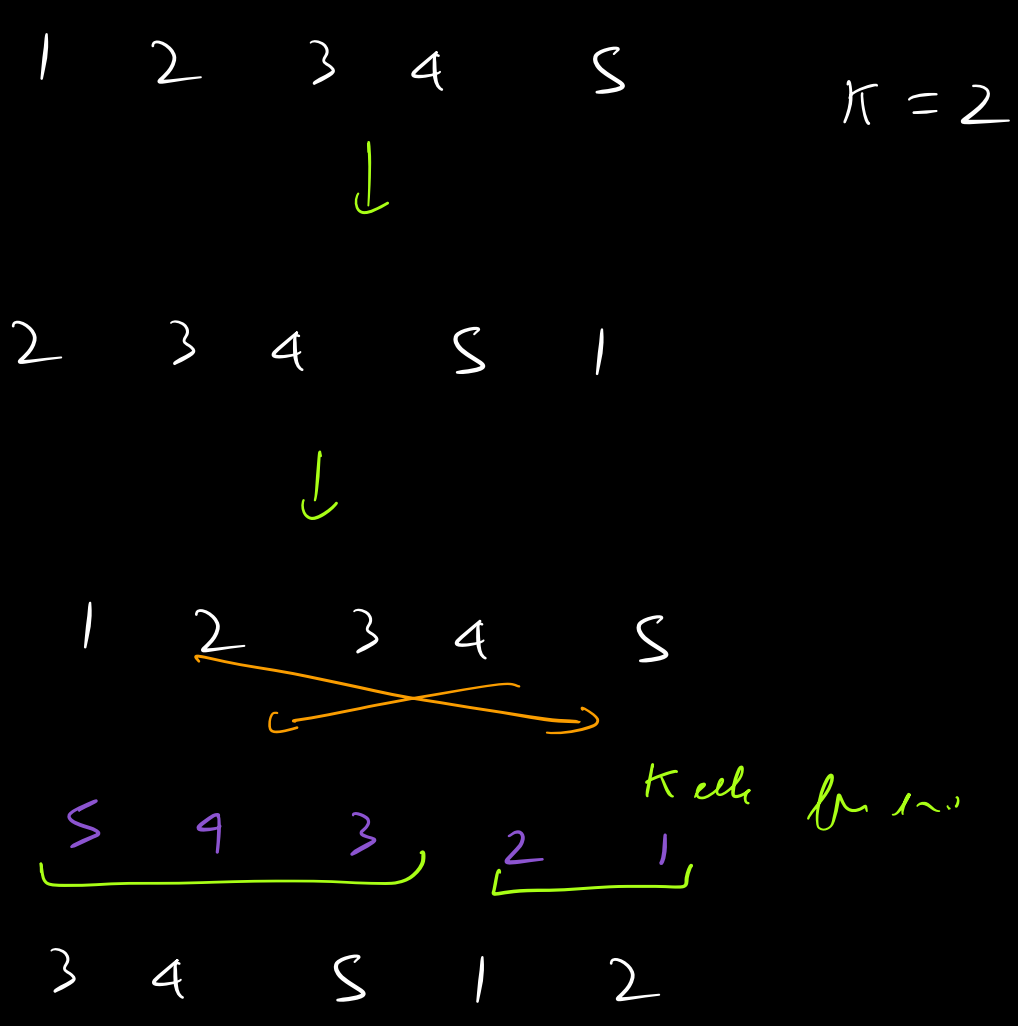
~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ $K = 2$

4 5 1 2 3

4 5 1 2 3



1 2 3 $K = 5$



Alok

N
N/2
N/4
N/8
⋮

1
2
4
8
⋮
N

$\log_2 N$

$\frac{N}{5}$ 5

$\frac{N}{25}$ 25

$\frac{N}{125}$ 125

$\vdots$

k $\}$ l

1