Welcome ☺

Agenda: Constraints

Space Complexity

Arrays

2-3 questions

Importance of constraints

$1 \le N \le 10^5$

Complexity	Works?
$O(N^3)$	✗
$O(N^2 \log N)$	✗
$O(N^2)$	✗
$O(N \log N)$ → $10^5 \log(10^5)$ → 1.6×10^6	✓

($\log_2(10^5) \approx 16.6$)

$1 \le N \le 10^6$

Complexity	Works?
$O(N^3)$	✗
$O(N^2 \log N)$	✗
$O(N^2)$	✗
$O(N \log N)$ $10^6 \log(10^6) \Rightarrow 2 \times 10^7$	might or might not work

($\log_2(10^6) \approx 20$)

$\leq 10^6$	$10^7 - 10^8$	$> 10^8$
✓	might work / Not Sure	✗

Space Complexity

⇒ It is the amount of space additionally used by the algo., other than the input space to perform operations.

```
int func ( int arr[] , int n )
{
    . . . .
    . . . .
    . . . .
}
```

S.C ⇒ ~~O(N)~~ O(1)

⇒ I/p size is not considered in space complexity.

```
func ( int N)
{
    int x = N      ⇒ 4B
    int y = x+x    ⇒ 4B
    long z = x+y   ⇒ 8B
}
```

space = 16 B = O(1)

Q)

```
func ( int N)
{
    int x = N          ⇒ 4B
    int y = x+x        ⇒ 4B
    long z = x+y       ⇒ 8B
    int arr[10]        ⇒ 4*10 = 40
    int arr[N]         ⇒ 4*N
}
```

⇒ 56 B + 4*N B ⇒ O(N) space. comp.

Q)

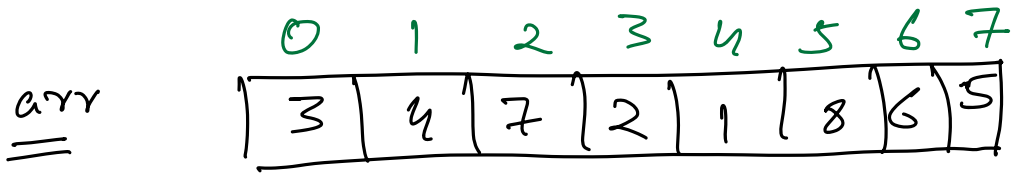
```
func ( int N)
{
    int x = N          ⇒ 4B
    int y = x+x        ⇒ 4B
    long z = x+y       ⇒ 8B
    int arr[10]        ⇒ 4*10 = 40
    int arr[N][N]      ⇒ 4*N*N
}
```

⇒ 56 B + 4*N*N ⇒ $O(N^2)$ space. comp.

Arrays

⇒ Collection of same types of data
   ↳ primitive / complex.

⇒ contiguous memory block.



⇒ Access index i ⇒ arr[i]

⇒ Declare array ⇒ int arr[N] (N → size of array)

first element → 0
last element → N-1

Q Print all the elements of the array

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

T.C ⇒ O(N)
S.C ⇒ O(1)

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

eg: A : { 1 2 3 4 5 }

o/p { 5 4 3 2 1 }

A : { 1 2 3 4 5 }

{ 5 2 3 4 1 }

{ 5 4 3 2 1 }

Code

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

T.C => O(N)

S.C => O(1)

Q Given an array of size N, rotate array from last to first by K times.

eg: { 1 2 3 4 5 }

K=1 { 5 1 2 3 4 }

K=2 { 4 5 1 2 3 }

Brute force

Rotate the array one distance at a time.

T.C O(N * K)

Optimized

{ 1 2 3 4 5 }

o/p K=2 { 4 5 1 2 3 }

1) Revers the array

2) { 5 4 3 2 1 }

reverse

3) { 4 5 3 2 1 }

reverse.

{ 4 5 1 2 3 }

{ 1 2 3 4 5 }

K=1 { 5 1 2 3 4 }

K=2 { 4 5 1 2 3 }

K=3 { 3 4 5 1 2 }

K=4 { 2 3 4 5 1 }

K=5 { 1 2 3 4 5 }

K=6 { 5 1 2 3 4 }

$K = K \% N$

H.W

T.C = $N + N = O(N)$

S.C = $O(1)$

Dynamic Arrays

→ array with no fixed size.

→ automatically resize