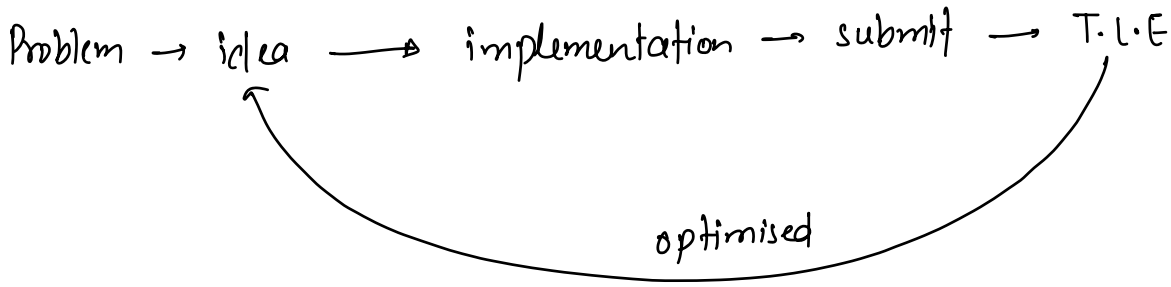
Good Morning / Evening everyone !! ☺

T.L.E. (Time Limit Exceeded)



Working of Online Editors -

Online editors $\xrightarrow[\text{runs}]{\text{code}}$ online servers

→ Processing speed (1 GHz)

→ 10^9 instructions / sec

declaring var
arithmetic operator
function calling
if-else

Time Limit : 1 sec

∴ Only 10^9 instructions can be executed.

```
int countFactors( N){
    count = 0;
    for( i = 1 ; i <= N; i++){
        if( N % i == 0){
            count += 1
        }
    }
    return count;
}
```

Approx.-1 1 iterations → 10 instructions

→ 10^9 instructions can be executed

→ $10^8 \times 10$ instructions can be executed

→ 10^8 iterations can be executed within Time Limit.

Approx.2. 1 iteration → 100 instructions

→ 10^9 instructions can be executed

→ $10^7 \times 10^2$ instructions can be executed

→ 10^7 iterations can be executed within Time Limit.

Conclusion : In order to submit a question, no. of iterations must be in $10^7 - 10^8$.

General Structure to solve a question →

- → Description
- → Constraints
- → I/p & O/p format
- → Examples

Q: constraints $1 \leq N \leq 10^5$

idea-1. → T.C → $O(N^3)$ → T.L.E

idea-2. → T.C → $O(N^2)$ → T.L.E

idea-3. → T.C → $O(N)$ → ✓

Sum of all elements in array

long
~~int~~ sum = 0

```
for( i=0; i < N; i++){
    sum = sum + arr[i];
}
return sum;
```

$1 \leq N \leq 10^6$

$1 \leq A[i] \leq 10^9$

int → 2×10^9

10^6

Sum → $10^9 \times 10^6 \rightarrow 10^{15}$

Space Complexity

↳ Max space (worst case) that is utilised by our algo/code at any point of time.

↳ Input & output space will never be included in Space complexity.

```
void func ( int N){
    int x = 10;      (4B)
    int y = 20;      (4B)
    long z = x+y;    (8B)
}
```

// 1 int → 4B , long → 8B

total extra space = 16B

S.C → O(1)

```
void func( int N){
    int x;           (4B)
    int y;           (4B)
    long z;          (8B)
    int [] arr = new int [N];  (4N B)
}
```

total extra space = (16+4N) B

S.C → O(N)

```
void func( int N){
    int a ;  (4B)
    int y ;  (4B)
    long z ;  (8B)
    int [] arr = new int [N];  (4N B)
    long[][] b = new long [N][N];  (8N^2 B)
}
```

total extra space = $16 + 4N + 8N^2$

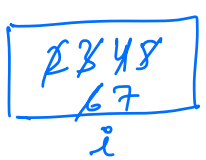
S.C → $O(N^2)$

```
int maxinArray( int[] arr, int N){
    int ans = arr[0];
    for( i = 1; i < N; i++){
        if ( arr[i] > ans){
            ans = arr[i];
        }
    }
    return ans;
}
```

total extra space → 4B

S.C → $O(1)$

```
for( i=1; i<100; i++){
    int j= 50;
}
```



4B.

Arrays.

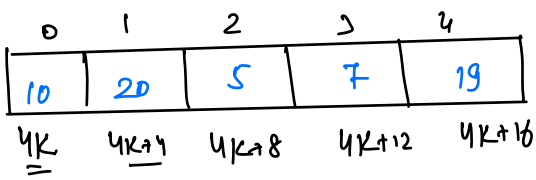
↳ collection of same type of data.

int arr[N];

↳ contiguous memory allocation will be there.

↳ index starts from 0. [0, N-1]

int arr[] = new int[5];



address of i^{th} idx element → $4K + 4*i$

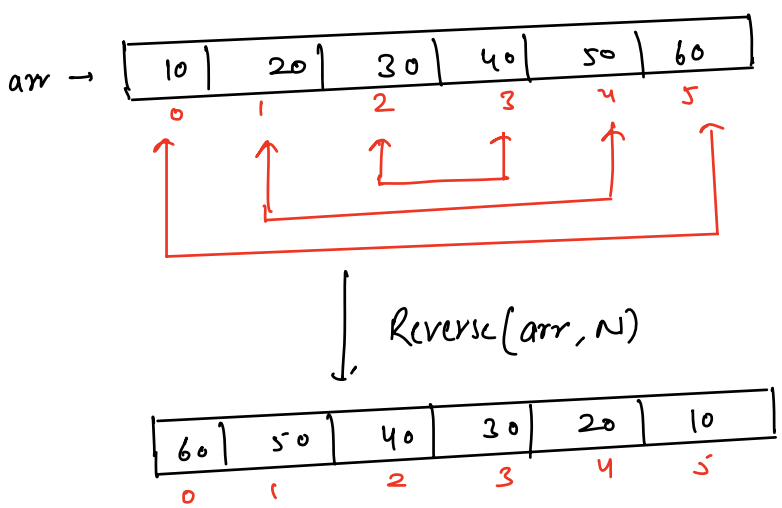
To access i^{th} idx element → arr[i] T.C → O(1)

Print An Array →

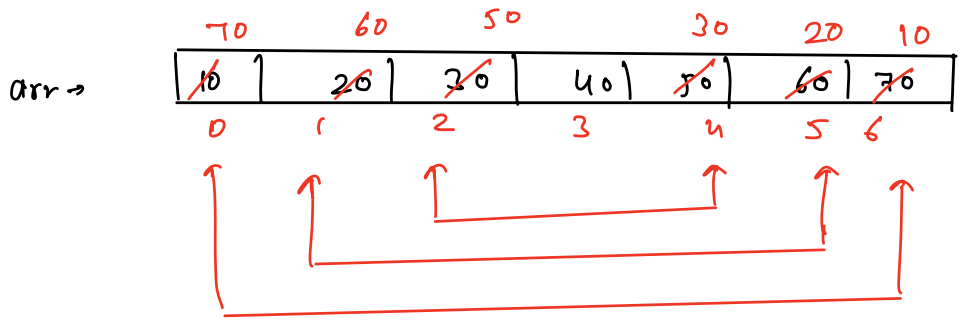
```
void printArr( int arr[], int N){
    for( i=0; i<N; i++){
        print( arr[i]);
    }
}
```

T.C → O(N)
S.C → O(1)

Reverse An Array



left	right
0	5
1	4
2	3



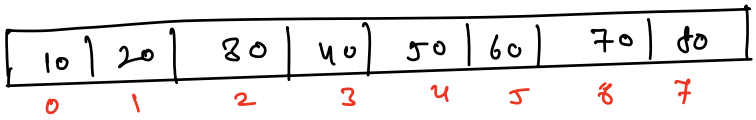
left	right
0,	6
1,	5
2,	4

code:-

```
void Reverse( arr[N], N){
    left = 0, right = N-1;
    while (left < right){
        // swap arr[left] with arr[right]
        temp = arr[left];
        arr[left] = arr[right];
        arr[right] = temp;
        left++ ;
        right-- ;
    }
}
```

T.C → O(N)
S.C → O(1)

Q Reverse part of an array



l = 2
r = 6

```
void reverse( arr[N], l, r) {
    while( l < r) {
        // swap arr[l] with arr[r]
        temp = arr[l];
        arr[l] = arr[r];
        arr[r] = temp;

        l++;
        r--;
    }
}
```

T.C → O(N)
S.C → O(1)

Rotate an Array =>

[Google]

arr → | 10 | 20 | 30 | 40 | 50 |

$1 \leq N \leq 10^6$

$0 \leq K \leq 10^9$

K=1. [50 10 20 30 40]

K=2. [40 50 10 20 30]

K=3. [30 40 50 10 20]

K=4. [20 30 40 50 10]

arr → | 31 | 17 | 21 | 19 | 16 |

K=3.

| 16 | 31 | 17 | 21 | 19 |

| 19 | 16 | 31 | 17 | 21 |

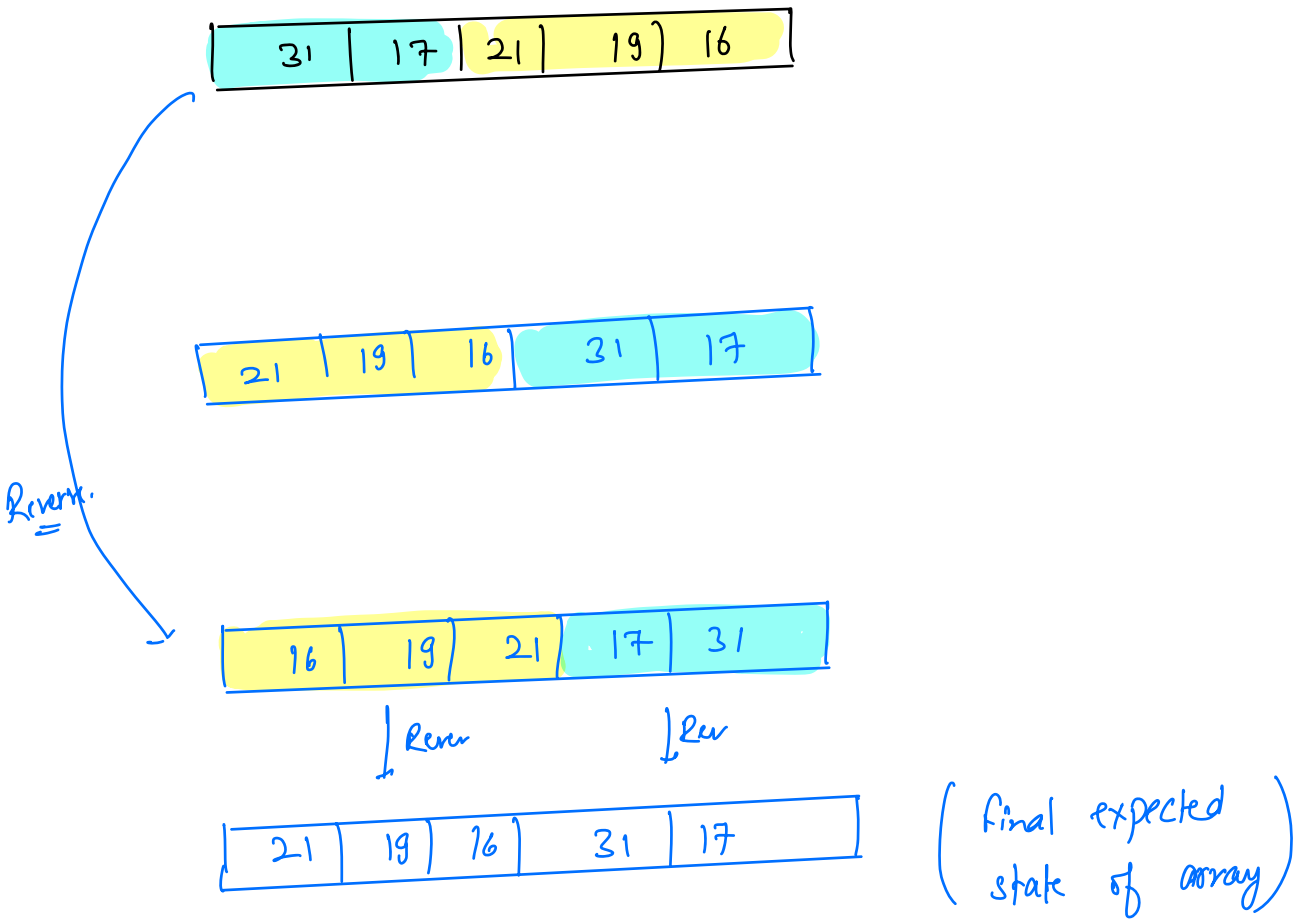
| 21 | 19 | 16 | 31 | 17 |

T.C → O(N*K)

⇓

T.L.E

idea-2.



(final expected state of array)

```
void rotate( arr[N], N, K){

    K = K % N

    ① reverse( arr, 0, N-1) ;

    ② reverse( arr, 0, K-1);

    ③ reverse ( arr, K , N-1);

}
```

T.C → O(N)
S.C → O(1)

arr → | 1 | 2 | 3 |

k=0 → [1 2 3]

k=1 → [3 1 2]

k=2 → [2 3 1]

k=3 → [1 2 3]

k=4 → [3 1 2]

k=5 → [2 3 1]

k=6 → [1 2 3]

k=7 → [3 1 2]

k=8 → [2 3 1]

k=9 → [1 2 3]

k=10 → [3 1 2]

0	3	6	9
1	4	7	10
2	5	8	11

$k = k \% N$

pSum concept

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

i	j	iterations
$i=0$	-	-
$i=1$	$1 \to 0$	1
$i=2$	$2 \to 0$	2
$i=3$	$3 \to 0$	3
$i=4$	$4 \to 0$	4
|		|
|		|
$i=2^N-1$	$2^N-1 \to 0$	2^N-1

iterations = $1+2+3+4+5+\cdots(2^N-1)$

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

$$\Rightarrow \frac{4^N-2^N}{2}$$

$$\Rightarrow \frac{4^N}{2} - \frac{2^N}{2}$$

T.C → $O(4^N)$

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

$[a,b] \Rightarrow b-a+1$

$[2,N]$

$\Rightarrow N-2+1$

$\Rightarrow N-1$

i	j	iterations
i = 0	N → 1	N
i = 1	N → 2	N-1
i = 2	N → 3	N-2
i = 3	N → 4	N-3
.		
.		
.		
i = N-1	N → N-1	1

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

$\Rightarrow O(N^2)$

→ Head first Java / Python / C