1.10 - Complexity Analysis

Wednesday, September 10, 2025 8:07 PM

- → Time Analysis
- → Space complexity

Time Complexity Analysis

→ A normal PC can do 10^8 operations in 1 sec [lines of code]

→ Our goal in DSA is to write a code that can execute in approx 1 sec.

3 ops / loc
```
sout ("Hello");
sout ("Hello");
sout ("Hello");
```

10 ops ←
```
for(i=0; i<10; i++) {
    sout ("Hello");
```

10^{10}

$1 \rightarrow 10^8$

10^{11} 1000

10^{12} 10000

$10^8 \rightarrow 1$ sec

$1 \rightarrow \frac{1}{10^8}$

$10^{10} \rightarrow 100$ sec

$1 \rightarrow \frac{1}{10^8}$

$10^{10} \rightarrow \frac{1}{10^8} \times 10^{10} \Rightarrow 10^{2}$

$10 \rightarrow 20$

$1 \rightarrow \frac{20}{10}$

$40 \rightarrow \frac{20}{10} \times 40$

$10^8 \rightarrow 1$

$1 \rightarrow \frac{1}{10^8}$

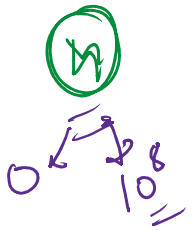
$10^{10} \rightarrow \frac{1}{10^8} \times 10^{10}$

$\Rightarrow 10^2$

Big O

→ Notation

→ It tells you the worst case complexity of an algo.




```
for (i=0; i<n; i++)
    sout ("Hello");
```

→ O(n)

$n \leq 10^8$

```
for (i=0; i<10; i++)
    sout ("Hello")
```

→ O(10) → O(1)

constant

Rules of Big O

1) $O(10) = O(50) = O(k) \Rightarrow \boxed{O(1)}$ constant

2) $O(n) = O(2n) = O(3n) = O(k \times n) \Rightarrow O(n)$

$[O(n^2)] \rightarrow 10^8$

$n^2 <= 10^8 \Rightarrow \boxed{n <= 10^4}$ ✓

$O(n^4)$ $n^4 <= 10^8$, $n <= 10^2$

$n^2 <= 10^8 \Rightarrow \boxed{n <= 10^4}$ ✓

[order of speed] →

$O(1) < O(\sqrt{n}) < O(n) < O(n^2) < O(n^3) < O(n^4) \ldots$

$O(\sqrt{n})$ $\sqrt{n} <= 10^8$

$\boxed{n <= 10^{16}}$ ✓

i → i
1 → 2 → 4

$1 + 2^1 + 2^2 + \dots 2^{n-1}$

slow off bro. → $(\log_2 n)$

log

$\log_a b$ ⇒ a ki power kya kare ki answer b ho

$\log_2 8$ → 3

$\log_2 2^3$ ⇒ 3×1 ⇒ 3

$\log_2 N <= 10^8$ ⇒ $N <= 2^{10^8}$

$O(1) <= O(\log n) < O(\sqrt{N}) < O(n) \dots$

How to analyse an entire program

```
for(i=0; i<n; i++){
    sout("Hello")
}

for(i=0; i<n; i++){
    for(i=0; i<n; i++){
        sout("Hello")
    }
}
```

O(n) + O(n²) ⇒ O(n²)

n <= 10^8

n <= 10^4

$n <= 10^8$ $n <= 10^4$

$O(n + n^2) \Rightarrow O(n) + O(n^2)$

* Its the worst component that decides the overall complexity of a prog.

$O(n + \sqrt{n} + n^2 + n + n^3) \Rightarrow O(n^3)$

$\Rightarrow O(n + n) = O(2n) \Rightarrow O(n)$

Space Complexity → Big O

→ Worst case space that my program is going to take.

int a; → 4 Byte → $O(4) \Rightarrow O(1)$

int b; → 400 → $O(1)$

int arr[] = new int[n];

$O(n)$

int arr[] = new int[100];

```
int arr[] = new int [100];
```

$O(1)$

4 Byte

```
int a = sc.nextInt();
```

S.C $O(n)$

```
int arr[] = new int[n];
for(i=0; i<n; i++)
    arr[i] = sc.nextInt();
```

[3, 7, 5, 4, 1]

=>

4 1 3 7 5

k = 2

1 3 7 5 4

4 1 3 7 5

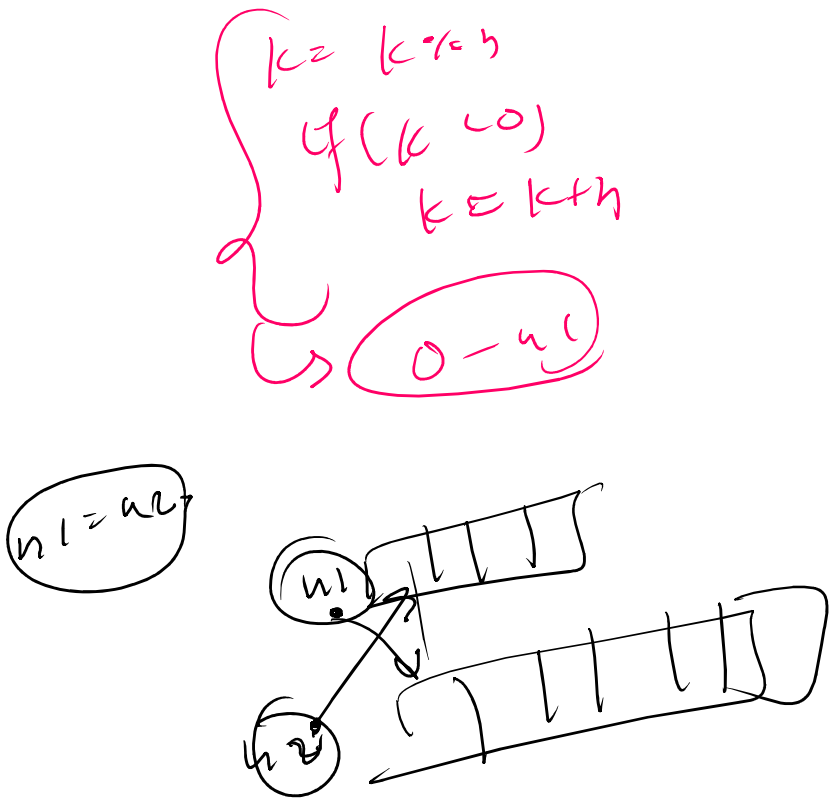
```
reverse(0, n-1)
reverse(0, k-1)
reverse(k, n-1)
```

4 1 | 5 7 3

4 1 3 7 5

```
reverse(arr, 0, n-1)
```

1 3


k = k%n
if(k<0)
k = k+n
0-n-1
n1 = n2
n1
n2