Day - 2

Sunday, 21 December 2025 12:08 PM

SC

Q.

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

[L → R]

⇒ TC = O(n)
SC = O(1)

Q.

```
printArr(int arr[], int n) {
    for (i=n-1 ; i>=0 ; i--) {
        print(arr[i])
    }
}
```

[R → L]

O(1)

TC = O(n)
SC = O(1).

Q.

```
printArr(int arr[], int n) {
    for (i=1 ; i<n ; i*=2) {
        print(arr[i])
    }
}
```

$TC = O(\log_2 n)$

SC = O(1)

Q.

```
for (i=n , i>1 ; i/=2) {
    print(i)
}
```

$i = n \quad n/2 \quad n/4 \quad n/8 \quad \dots \quad n$

$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$

$\frac{n}{2^0} \quad \frac{n}{2^1} \quad \frac{n}{2^2} \quad \frac{n}{2^3} \quad \dots \quad \frac{n}{2^{K-1}} > 1 \quad \frac{n}{2^K} \leq 1$

1st step 2nd step kth step

$$\underbrace{\overset{\Downarrow}{O(1)} \quad \overset{\Downarrow}{O(1)} \quad \overset{\Downarrow}{O(1)} \quad \overset{\Downarrow}{O(1)}}_{K\text{-times}}$$

$$\Rightarrow K \cdot O(1) \Rightarrow O(K)$$

$$\boxed{TC = O(\log_2 n)}$$

$SC = O(1)$ ✓

$$\frac{n}{2^K} \le 1$$

$$\Rightarrow \frac{n}{2^K} = 1$$

$$\Rightarrow \boxed{2^K = n}$$

$$\Rightarrow \boxed{K = \log_2 n}$$

Q

```
for (i=0 , i<n ; i++) {
    for (j=0 ,j<n ;j++) {
        print("Hi");
    }
}
```

$\rightarrow$ "n" times.

$\Rightarrow$ 'n' times $\Rightarrow O(n)$

$TC = n * O(n) \Rightarrow O(n^2)$

Q

```
for (i=0 , i<n ; i++) {
    for (j=0 ,j<m ;j++) {
        print("Hi");
    }
}
```

$TC = O(n.m)$
$SC = O(1)$

Q

```
for (i=0 , i<n ; i++) {
    for (j=1 ,j<n ;j*=2) {
        print("Hi");
    }
}
```

$\Rightarrow$ n times

$TC = O(\log_2 n) \cdot n$

$\Rightarrow O(\log_2 n)$

TC = O(n · $\log_2 n$)

SC = O(1)

Q. Given a no. "N".
Find the sum of 1st 'n' natural nos. (1, 2, 3 . . . N)

eg. N = 3 ⇒ (1, 2, 3) N = 5. (1, 2, 3, 4, 5)
O/P ⇒ 6 ✓ O/P ⇒ 15 ✓

M-1

```
sum = 0;                       O(1) ✓
for (i = 1; i <= n; i++)  {      DN
    sum += i;   — O(1)
}
print(sum)
```

i = 1	sum = 0
1	0 + 1 = 1
2	1 + 2 = 3
3	3 + 3 = 6 ✓
4	6 + 4 = 10
5	10 + 5 = 15 ✓

O(1) + N·O(1) ⇒ $\cancel{O(1)}$ + O(N) ⇒ O(N)

TC = O(N)
SC = O(1)

M-2. 1 + 2 + 3 + . . . N (AP)

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

⇓

$$\boxed{sum = \frac{n \cdot (n+1)}{2}}$$ ✓

n = 3

$sum = \frac{3 \cdot \cancel{4}^{2}}{\cancel{2}} = 6$ ✓

$sum = \frac{5 \cdot \cancel{6}^{3}}{\cancel{2}} = 15$ ✓

a, a+d, a+2d, . . . a+(N−1)·d

$\frac{n}{2}(\underset{1st}{a} + \underbrace{a+(N-1)\cdot d}_{last})$

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

```
sum = (n)·(n+1)/2
print(sum)
```
⇒ O(1) ⇒ TC.

$SC = O(1)$

Time Complexity

Prob^m

$A1 = O(n)$ $\quad$ $A2 = O(\sqrt{n})$ $\quad$ $\boxed{A3 = O(1)}$ ✓

Q

```
int i;
for (i=0 ; i*i < n ; i++) {      => 'K' times
                                   'K' step
      print(i)  => O(1)
}
```

$i = 0 \quad 0.0 = 0 < n$

$i = 1 \quad 1.1 = 1 < n$ ✓

$i = 2 \quad 2.2 = 4 < n$

$\vdots$

$i = K-1 \quad (K-1)(K-1) < n$ ✓

$i = K \quad \boxed{(K)\cdot(K) \geq n}$ ✗

$K \cdot O(1) \Rightarrow O(K)$

$\Rightarrow O(\sqrt{n}) \Rightarrow TC$

$SC = O(1)$

$K^2 \geq n$

$K^2 = n$

$K = \sqrt{n}$

Q

```
for (i=0; i <= n ; — ) {
      print("hi")
      i = i*2;
}
```

$i = 0 < n$ ✓

$i = 0 * 2 = 0 < n$

$i = 0 * 2 = 0 < n$ ✓

$\vdots$

∞

Q

```
for ( i = 1 ; i <= n ; i *= 2) {     => "log2 n" times
      // work() => TC = O(n)
      print("hi")                    => O(n)
}
```

$TC = \log_2 n * O(n) \Rightarrow O(n \cdot \log_2 n)$

$SC = O(1)$

Q. for (i = 1 ; i ≤ n ; i = i * 3) {
   print ("Hi")
}

$\Rightarrow O(\log_3 n)$

Q. for (i = 0 ; i ≤ 99 ; i++) { — "100" times.

   for (i = 1 ; i ≤ n ; i++) {
      print ("Hi")
   } $\Rightarrow O(n)$
}

$TC = 100 \cdot O(n) \Rightarrow O(100 \cdot n)$

$\Rightarrow O(\cancel{100} \cdot n)$

$\Rightarrow O(n)$

(*) Drop constants.

$O(n)$ ✓

Q. i = 0 ; sum = 0

"k" times ⇐ for (i = 1 ; sum < n ; i++)

   $\boxed{sum += i} \Rightarrow O(1)$ 'k' steps

}

i = 0	sum = 0	
1	0 + 1 = 1	< n ✓
2	1 + 2 = 3	< n ✓
3	3 + 3 = 6	< n ✓
4	6 + 4 = 10	< n ✓
⋮		
(k−1)	◯ + (k−1)	< n ✓
k	$\left[\frac{k \cdot (k+1)}{2}\right] \geq n$	→ Condⁿ failed.

$TC = k \cdot O(1) \Rightarrow O(k)$

$\Rightarrow O(\sqrt{n})$ ✓

$SC = O(1)$

$$\frac{k \cdot (k+1)}{2} \geq n \Rightarrow k^2 + k \geq 2n$$

$$\Rightarrow k^2/k + k/k = 2n/k$$

$$\Rightarrow k + 1 = 2n/k$$

$$\Rightarrow k = \frac{2n}{k} (-1) \Rightarrow k = \frac{n}{k} \Rightarrow k^2 = n$$

$$\Rightarrow \boxed{k = \sqrt{n}}$$

Rules of Big-O Notation

1. If f(n) is sum of several terms, then one with the largest growth rate (largest degree) will be kept and others can be removed.

$$TC = N^3 + \underbrace{N^2 + N + \log n}_{\times}$$

$$= O(N^3)$$

2. If f(n) is product of several terms, any constants (that don't depends upon 'n') can be removed.

$$TC = \cancel{5} \cdot N^3 + \cancel{\frac{3N}{2}} + \cancel{2 \cdot \log n}$$

$$= O(N^3)$$

$$TC = \cancel{3}(\log n) \cdot N^3$$

$$= O(\log n \cdot N^3).$$

$$TC = n + \cancel{\log n} \Rightarrow O(n)$$

$$\cancel{K} \cdot n > \cancel{K} \cdot \log n$$

$$TC = n^2 + n \cdot \log n \Rightarrow O(n^2)$$

$$TC = \cancel{10^3} \cdot 2^n \cdot n^2 + n \log n$$

$$2^n \cdot n^2 + n \cdot \log n.$$

2^{10}

$$\Rightarrow O(2^n \cdot n^2) \times$$

Q Compare the following

1. $N \cdot 1 < N \log n$

2. $N > \log n$

3. $N^2 > \sqrt{n}$

4. $n^2 > n\log n$

5. $n\log n > n\log(\log n)$

Dominance Relⁿ

$$\underbrace{O(1) < O(\log n)}_{\text{Best}} < O(\sqrt{n}) < O(n) < O(n\log n) < O(n\cdot\sqrt{n}) < O(n^2) < O(n^3) \ldots < \underbrace{O(2^n) < O(n^n)}_{\text{worst}}$$

"8972411533" — Rahul