Time Complexity & Space Complexity — 2.5 hr (10-15 min)

Assymptotic analysis

Big-O

Time limit exceeded

Basic Maths

Q1 Sum of first N natural no. : $\frac{N(N+1)}{2}$

Q2 [3, 10]

3, 4, 5, 6, 7, 8, 9, 10 ⇒ 8 no.

Q3 [a, b] : b − a + 1

Arithmetic Progression (AP)

4, 7, 10, 13, 16, 19, 22 ...

3 3 3 3 3 3

a, a+d, a+2d, a+3d, a+4d, ...

d d d d

a → first term

d → common diff

N → no of terms

$$S_N = \frac{N}{2}[2a + (N-1)d]$$

Geometric Progression (GP)

10/5 20/10 40/20 80/40

5, 10, 20, 40, 80, 160, 320 ....

×2 ×2 ×2 ×2

$a, ar, ar^2, ar^3, ar^4 \ldots\ldots$

$a \rightarrow$ first term

$r \rightarrow$ common ratio

$N \rightarrow$ no of terms

$$S_N = \frac{a(r^N - 1)}{r - 1} \qquad r > 1$$

log

$\log_a b \longrightarrow$ The no of times we need to divide b by a to make it 1.

$b = a^{*}$ ← $\log_a b$

$\log_a a^x = x$

$\log_2 1024 = 10$

$1024 = 2^{*}$ → ans

$\log_{10} 1000000 = 6$

$N = 2^K$

$\log_2 N = \frac{\log_2 2^K}{\log_2 N \quad K}$

Q

```
void fn(N){
    for(i=0; i<=100; i++){
        ....
    }
}
```

$i : [0, 100]$

$\Rightarrow$ 101 iterations

$101 \times N^0$

$O(1)$

Q

```
void fn(N){
    for(i=1; i<=N; i++){
        ____
    }
}
```

$i : [1, N]$

$\longrightarrow N$

iterations : N

$O(N)$

Q

```
void fn(int N, int M)
    for(i=1; i<=N; i++){
        if(i%2==0){
            Print(i);
        }
    }
    for(j=1; j<=M; j++){
        if(j%2==1){
            Print(j);
        }
    }
}
```

$i : [1, N]$

iteration : N

$j : [1, M]$

iterations : M

iterations : $N+M$

$O(N+M)$

Q

```
void fn(int N){
    for(i=1; i<=N; i=i+2){
        ---
    }
}
```

i
1
3
5
7
9
11
13
⋮
N / N-1

iterations : No of odd no. in the range [1, N]

$\rightarrow \frac{(N+1)}{2}$ (int div) — Hw

$N=7$: {1, 2, 3, 4, 5, 6, 7} ⟶ 4

$N=6$: {1, 2, 3, 4, 5, 6} ⟶ 3

$N=8$: {1, 2, 3, 4, 5, 6, 7, 8} ⟶ 4

$N/2$ ↳ $\frac{N}{2} + \frac{1}{2}$

$O(N)$

Q

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

$i \times i <= N$

$i^2 <= N$

$i <= \sqrt{N}$

$i_{max} \rightarrow \sqrt{N}$

iterations = $\sqrt{N}$

$O(\sqrt{N})$

$x <= 60$; $x_{max} \rightarrow 60$

$x <= 100$; $x_{max} \rightarrow 100$

$x <= N$; $x_{max} = N$

Q
```
void fn(N){
    int i = N;
    while( i > 1) {
        i = i/2;
    }
}
```

i_{Before}	Iteration	i_{after}
N	1	$N/2 - \frac{N}{2^1}$
$N/2$	2	$N/4 - \frac{N}{2^2}$
$N/4$	3	$N/8 - \frac{N}{2^3}$
$N/8$	4	$N/16 - \frac{N}{2^4}$
⋮	⋮	⋮
	K	$1 \rightarrow \frac{N}{2^K}$

$N \longrightarrow \frac{N}{2} \longrightarrow \frac{N}{4} \longrightarrow \frac{N}{8} \cdots\cdots 1$

$\log_2 N$

$O(\log N)$

$\log_a b$

$$\frac{N}{2^K} = 1$$

$$N = 2^K$$

$$\log_2 N = \log_2 2^K$$

$$\Rightarrow K = \log_2 N$$

Q
```
void fn(N){
    for(i=1; i<N; i=ix2){
        ——
    }
}
```

i_b	iteration	i_a
1	1	$2 \rightarrow 2^1$
2	2	$4 \rightarrow 2^2$
4	3	$8 \rightarrow 2^3$
8	4	$16 \rightarrow 2^4$
⋮	⋮	⋮
	K	$N \rightarrow 2^K$

x

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \cdots\cdots \rightarrow 98 \rightarrow 99 \rightarrow 100 \longrightarrow 100$

$x \rightarrow 100$

$100 \rightarrow 99 \rightarrow 98 \rightarrow 97 \cdots\cdots \rightarrow 3 \rightarrow 2 \rightarrow 1 \longrightarrow 100$

$1, 2, 4, 8, \ldots, N \longrightarrow \log_2 N$

$1, \ldots, N/4, N/2, N$ $\quad O(\log N)$

Q
```
void fn(N){
    for(i=0; i<N; i=i*2){
        ——
    }
}
```

i_b	iterate	i_a
0	1	0
0	2	0
0	3	0
0	4	0
⋮	⋮	⋮

iterations : ∞

Q
```
void fn(N){
    for(i=1; i<=10; i++){
        for(j=1; j<=N; j++){
            Print(i*j);
        }
    }
}
```

i	j	Iterations in inner loop
1	[1, N]	N
2	[1, N]	N
3	[1, N]	N
⋮	⋮	⋮
10	[1, N]	N
11	×	×

iterations : $10N$

$O(N)$

Q
```
void fn(N){
    for(i=1; i<=N; i++){
        for(j=1; j<=N; j++){
            Print(i+j);
        }
    }
}
```

i	j	Iterations in inner loop
1	[1, N]	N
2	[1, N]	N
3	[1, N]	N
⋮	⋮	⋮
N	[1, N]	N
N+1	x	y

iterations : $N \times N$

N^2

$O(N^2)$

$3+3+3 \Rightarrow 3\times 3$

$5+5+5+5+5 \Rightarrow 5\times 5$

$4+4+4+4 \Rightarrow 4\times 4$

$N+N+N+\cdots N \Rightarrow N\times N$ (N times)

Q
```
void fn(N){
    for(i=1; i<=N; i++){
        for(j=1; j<=N; j=j*2){
            Print(i+j);
        }
    }
}
```

i	j	Iterations in inner loop
1	1 → N	$\log_2 N$
2	1 → N	$\log_2 N$
3	1 → N	$\log_2 N$
⋮	⋮	⋮
N	1 → N	$\log_2 N$

iterations : $N \log_2 N$

$O(N \log N)$

Q
```
void fn(N){
    K = 2^N;
    for(i=1; i<=K; i++){
        =
    }
}
```

$i : [1, 2^N]$

iterations → 2^N

$O(2^N)$

Q
```
void fn(N){
    for(i=1; i<=N; i++){
        for(j=1; j<=2^i; j++){
            Print(i+j);
        }
    }
}
```

i	j $[1, 2^i]$	Iterations in inner loop
1	[1, 2]	2
2	[1, 2^2]	4
3	[1, 2^3]	8
4	[1, 2^4]	16
5	[1, 2^5]	32
⋮	i	⋮
N	[1, 2^N]	2^N

iterations → $2(2^N - 1)$

$2 + 4 + 8 + 16 + 32 + \cdots + 2^N$

$2 \times 2^N - 2$

$O(2^N)$

$a \to 2$

$r \to 2$

No of terms $\to N$

$$S_N = \frac{2(2^N - 1)}{2 - 1}$$

$$= 2(2^N - 1)$$

How to calculate Big-O from no of iterations

- Neglect all lower order terms / Only keep the highest Power
- Neglect all constant coeffecits

$\#\text{iterations} = 4N^2 + 3N + 1$

$4N^2$

N^2 $\quad O(N^2)$

What is TC ?
What is Big-O ?
Why above steps ?

→ Next class