Time Complexity / Space Complexity

Big O | Next class

Counting iterations

[1, N]

Sum of N natural numbers?

$$1 + 2 + 3 + 4 \dots N = \frac{N \times (N+1)}{2}$$

Yagyesh : (4th Class) Gauss

$$S = 1 + 2 + 3 + 4 \dots + 99 + 100 \quad (1)$$

$$S = 100 + 99 + 98 + 97 \dots 2 + 1 \quad (2)$$

$$2S = 101 + 101 + 101 \dots 101 + 101$$

$$2S = 100 \times 101$$

$$S = \frac{100 \times 101}{2} = 5050$$

Arithmatic progression

1	2	3	4	5	6	7

$$\underbrace{1, 5}_{4}, \underbrace{5, 9}_{4}, \underbrace{9, 13}_{4}, \underbrace{13, 17}_{4}, 21, 25$$

a (The first term of series) = 1

d (Common difference) = 4

n (No. of terms)

N^{th} term of series : $a + (n-1)d$

7^{th} term : $1 + (7-1) \times 4 = 25$

11^{th} term : $1 + (11-1) \times 4 = 41$

sum of N terms of AP: $\frac{N}{2}[2a + (n-1)d]$

Series

1, 2, 3, 4, 5, 6 ... AP (common dif = 1)

1, 3, 4, 5, 7, 11 ... Not AP (No common diff)

GP (Geometric progression)

Common ratio: Ratio of any 2 consec terms

$$\underbrace{\overset{1}{1}, \overset{2}{3}}_{\frac{3}{1}}, \underbrace{\overset{3}{9}}_{\frac{9}{3}}, \underbrace{\overset{4}{27}}_{\frac{27}{9}}, \underbrace{\overset{5}{81}}_{\frac{81}{27}} \quad 243$$

$$\frac{3}{1} \quad \frac{9}{3} \quad \frac{27}{9} \quad \frac{81}{27} = 3$$

a (first term) = 1

r (common ratio) = 3

n (No. of terms)

N^{th} term of GP = $a \times r^{n-1}$

6^{th} term = $1 \times 3^{6-1} = 3^5 = 243$

Sum of n term = $a\left[\frac{r^n - 1}{r-1}\right]$

$$\boxed{\log_a a^x = x}$$

$8 \xrightarrow{/2} 4 \xrightarrow{/2} 2 \xrightarrow{/2} 1 \quad = 3$

$$\boxed{\log_2 8 = 3}$$

$[1, 10] = 10$ numbers

$[3, 10] = 8$

$[7, 1001] = 1001 - 7 + 1 = 995$

$$\boxed{[a, b] = b - a + 1}$$

both included

Prob 1)

```
void func(int N) {
    s = 0
    for (i=1; i<=N; i++) {
        s = s+i;
    }

    return s;
}
```

$i : [1, N]$

$N - 1 + 1$

$= N$

Q2)

```
void func(int M, int N) {
    for (i=1; i<=N; i++) {
        print(i)
    }
    for (i=1; i<=M; i++) {
        print(i)
    }
}
```

$i : [1, N]$

iter : N

$i\ [1, M]$

iter : M

$N + M$

3) int func(N) {

```
    s = 0
    for(i=0;  i<=100 ; i++) {
        print(i)
    }
}
```

$i : [0, 100]$

$100 - 0 + 1$

$= 101$

Q4) void func (N) {

```
    for (i=1;  i*i <= N ; i++) {
        print(i)
    }
}
```

$i^2 \leq N$

$i \leq \sqrt{N}$

Taking sqrt both sides

$i : [1, \sqrt{N}]$

$\sqrt{N} - 1 + 1 = \sqrt{N}$

Q5) void func(N) {

i = N;

while (i > 1) {

i = i/2

}

}

N = 8

i = 8

↓ ←

i = 4

↓ ←

i = 2

↓ ←

i = 1

3

int($\log_2 N$)

Amazon

Infinite iteration

Q6)
```
void func (N) {
    for (i=0; i<N ; i=i*2) {
        print (i)
    }
}
```

i=0 → 0×2 → 0×2 → 0×2 ... 0

Q7)
```
void func (N) {
    for (i=1; i<N ; i=i*2) {
        print (i)
    }
}
```

N = 8

$$i=1 \xrightarrow[1]{} 2 \xrightarrow[2]{} 4 \xrightarrow[3]{} 8$$

iter: $\log_2(N)$

N=10

$$1 \xrightarrow[1]{} 2 \xrightarrow[2]{} 4 \xrightarrow[3]{} 8 \xrightarrow[4]{} 16$$

Break (10:12 – 10:22)

Q8) void func (N) {

```
for (i=1 ; i<=10 ; i++) {
    for (j=1 ; j<=N ; j++) {
        print (i, j)
    }
}
}
```

i	j	iteration
1	[1,N]	N + 1
2	[1,N]	N + 1
3	[1,N]	N + 1
⋮		
10	[1,N]	N + 1
		10N + 10
		10N

N +

Q9) for (i=1 ; i< N ; i++) {

prin(i)

for (j=1 ; j<=i ; j++) {

print (i, j)

}

}

i	j	iteration
1	[1,1]	1 + 1 [i loop]
2	[1,2]	2 + 1
3	[1,3)	3 + 1
⋮		
N-1	[1,N-1]	N-1 + 1

$\overline{1+2+3+4 \ldots (N-1)}^{\,N-1}$

a = 1

d = 1

n = N-1

sum of AP

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

$= \frac{N-1}{2}[2\times 1 + (N-1-1)\times 1]$

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

$\frac{N \times (N-1)}{2} + N$ (i loop)

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

$\frac{N^2}{2} + \frac{N}{2}$

TC: $O(N^2)$

Q12)
```
void func (N){
    for (i=1; i<= N; i++){
        for (j=1 ; j< N; j=j*2) {
            print (i,j)
        }
    }
}
```

N=8

$1 \to 2 \to 4 \to 8$

i	j	iteration
1	[1,N,2]	$\log_2 N$ + 1 (i loop)
2	[1,N,2]	$\log_2 N$ + 1 (i loop)
⋮		
N	[1,N,2]	$\log_2 N + 1$
		$N\log_2 N + N$

TC: $O(N\log N)$

Q13) void func(N) {

```
void func(N) {
    for (i=1 ; i<= 2^n ; i++) {
        print(i)
    }
}
```

Iterations: 2^N

Q14) HARD

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

i	j	iteration
1	[1,2]	2 + (1)
2	[1,4]	4 + (1)
3	[1,8)	8 + 1
⋮		
N	[1,2^N]	2^N + 1

$2 + 4 + 8 \ldots 2^N$

$a = 2 \quad r = 2 \quad n = N$

$$\frac{a(r^n - 1)}{r-1} = 2\left(\frac{2^N - 1}{2-1}\right)$$

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

$O(2^N)$

N (Input size)

$N = 10000$

constant $<$ $\boxed{\log_2 N < \sqrt{N} < N < N\log N < N^2 < 2^N < N!}$ Higher order terms

value increases

$N!$

$100! = 100 \times 99 \times 98 \ldots 1$

$\quad\quad\quad\ \ \vee \quad\ \vee \quad\ \vee \quad \vee$

$2^{100} = 2 \times 2 \times 2 \times 2 \ldots 2$

Big O

✓ 1) What is big O Monday

2) How to find big O Today

✓ 3) What is its significance Monday

1) Count no. of iterations
2) Take the highest order term
3) Remove constant coefficient

Total iter : $5N^2 + 3N$

$\downarrow$

$O(N^2)$

lowest order te..

$4N\log N + 3N\sqrt{N} + 10^6$

highest order term

$O(N\sqrt{N})$

lowest order

$4N^2 + 3N + 10^6$

$O(N^2)$

$4N + 3N\log N + 10^6$

$O(N\log N)$

Done!