- Time complexity
- Big O notation
- TLE (time limit exceeded)

$\Big\}$ next class

How to calculate total no. of iterations.

1) Sum of N natural no.

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

2) [→ inclusive , (→ exclusive

[3 5] ⇒ 3 4 5

[0 4] ⇒ 0 1 2 3 4

[a b] ⇒ $b-a+1$

[4 7] ⇒ $7-4+1 = 4$

3) Geometric progression (GP)

$$3 \xrightarrow{\times 2} 6 \xrightarrow{\times 2} 12 \xrightarrow{\times 2} 24 \xrightarrow{\times 2} 48$$

4 7 10 13 19 ✗

2 4 8 16 32 ✓

4 16 64 256 ✓

first term : a

common ratio : r

1	2	3	4	5		n
a	ar	ar^2	ar^3	ar^4	....	ar^{n-1}

$$\boxed{\text{Sum of } n \text{ terms of GP} = \frac{a(r^n - 1)}{r - 1}}$$

$(r \neq 1)$

5 20 80 320 1280

$a = 5$

$r = 4$

$n = 5$

$$S_n = \frac{a(r^n - 1)}{r - 1}$$

$$= \frac{5(4^5 - 1)}{3} = \frac{5 \times \overset{341}{\cancel{1023}}}{\cancel{3}} = 1705$$

Q. 1

```
int fun (int N) {
    int s= 0;
    for (int i=0; i<= 100; i++) {
        s= s+i+i*i;
    }
    return s;
}
```

$i \rightarrow [0 \quad 100]$

101 iterations

Q. 2

```
int fun (int N) {
    int s= 0;
    for (int i=3; i<= 50; i++) {
        s= s+i+i*i;
    }
    return s;
}
```

$i : [\underset{a}{3} \quad \underset{b}{50}]$

$b-a+1 = 50-3+1=48$

48 iterations

Q. 3

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

$i : [1 \quad N]$

N iterations

Q.4
```
void fun (int N, int M) {
    for (int i=1; i<=N; i++) {
        if (i%2 == 0) {
            SOP(i);
        }
    }
    for (int i=1; i<=M; i++) {
        if (i%2 == 0) {
            SOP(i);
        }
    }
}
```

N

M

N+M itr

Q.5
```
int fun (int N) {
    for (int i=1; i*i <= N; i++) {
        S = S + i*i;
    }
    return s;
}
```

$i <= \sqrt{N}$

$i * i <= N$

$i^2 <= N$

take sqrt on both sides

$i <= \sqrt{N}$

$i \rightarrow [1 \quad \sqrt{N}]$

$\sqrt{N}$ iterations

Q.6
```
void fun (int N) {
    for (int i=1; i <= 2^N; i++) {
        SOP(i);
    }
}
```

$i \rightarrow [1 \quad 2^N]$

2^N iterations

Q.7
```
void fun(int N) {
    int i = N;
    while (i > 1) {
        i = i / 2;
    }
}
```

itr	i value after		
1	$\frac{N}{2}$	$\rightarrow$	$\frac{N}{2^1}$
2	$\frac{N}{4}$	$\rightarrow$	$\frac{N}{2^2}$
3	$\frac{N}{8}$	$\rightarrow$	$\frac{N}{2^3}$
4	$\frac{N}{16}$	$\rightarrow$	$\frac{N}{2^4}$

loop break at $i = 1$

// assume loop breaks after k itr

$i = \frac{N}{2^k}$

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

$N = 2^k$

take $\log_2$ on both sides

$\log_2 N = \log_2 2^k$

$\boxed{\log_2 N = k}$

$\log_2 N$ iterations

Q.8
```
void fun(int N) {
    int s = 0;
    for (int i = 0; i < N; i = i * 2) {
        s = s + i;
    }
}
```

infinite iterations

Q.9

```
void fun(int N) {
    int s=0;
    for(int i=1; i<N; i=i*2) {
        s=s+i;
    }
}
```

itr	i value after
1	2 → 2^1
2	4 → 2^2
3	8 → 2^3
4	16 → 2^4

loop breaks at $i=N$

// assume loop breaks after k itr

$i = 2^k$

$2^k = N$

$\log_2$ on both sides

$\log_2 2^k = \log_2 N$

$\boxed{k = \log_2 N}$

$\log_2 N$ iterations

Nested loops

Q.10

```
void func (int N) {
    for (int i=1; i<=3; i++) {
        for (int j=1; j<=4; j++) {
            SOP (i+" "+j);
        }
    }
}
```

12 iterations

i	j	itr
1	[1 4]	4 +
2	[1 4]	4 +
3	[1 4]	4
		12

Q.11

```
void func (int N) {
    for (int i=1; i<=10; i++) {
        for (int j=1; j<=N; j++) {
            SOP (i+" "+j);
        }
    }
}
```

10N iterations

i	j	itr
1	[1 N]	N +
2	[1 N]	N +
3	[1 N]	N
⋮		⋮ +
10	[1 N]	N
		10N

Q.12

```
void func (int N) {
    for (int i=1; i<=N; i++) {
        for (int j=1; j<=N; j++) {
            SOP (i+" " + j);
        }
    }
}
```

N*N iterations

i	j	itr
1	[1 N]	N +
2	[1 N]	N +
3	[1 N]	N
⋮		⋮ +
N	[1 N]	N
		N*N

Q.13

```
void func (int N) {
    for (int i=1; i<=N; i++) {
        for (int j=1; j<N; j=j*2) {
            SOP (i+" " + j);
        }
    }
}
```

N $\log_2 N$ iterations

i	j	itr
1	[1 N) j=j*2	$\log_2 N$ +
2	[1 N) j=j*2	$\log_2 N$ +
3	[1 N) j=j*2	$\log_2 N$ +
⋮		⋮ +
N	[1 N) j=j*2	$\log_2 N$
		$N * \log_2 N$

Q.14
```
void func (int N) {
    for (int i=0; i< N; i++) {
        for (int j=0; j<=i; j++) {
            SOP (i+" " + j);
        }
    }
}
```

i	j [0 i]	itr
0	[0 0]	1 +
1	[0 1]	2 +
2	[0 2]	3 +
⋮		⋮ +
N−1	[0 N−1]	N

$1 + 2 + 3 + 4 + \dots + N$

$\Rightarrow$ Sum of N natural no. $= \dfrac{N(N+1)}{2}$

Q.15
```
void func (int N) {
    for (int i=1; i<=N; i++) {
        for (int j=1; j<=2^i; j++) {
            SOP (i+" " + j);
        }
    }
}
```

i	j $[1 \ 2^i]$	itr
1	[1 2]	2 +
2	[1 4]	4 +
3	[1 8]	8 +
⋮		⋮ +
N	$[1 \ 2^N]$	2^N

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

Sum of t terms = $\frac{a(r^t - 1)}{r - 1}$

$a = 2$

$r = 2$

terms = N

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

Comparing terms

$\log_2 N < \sqrt{N} < N < N\log_2 N < N\sqrt{N} < N^2 < 2^N$

(specially for large value of N)

Time complexity

↓

Big O notation — what, why } next class

how to find Big O

1) find total no. of iterations.

2) discard lower order terms (keep the highest order term)

3) discard constant coefficient.

itr : $10N^2 + 5N\log_2 N + 6N$

Big O $\rightarrow$ $O(N^2)$

itr : $100N\sqrt{N} + 50\,2^N + 60N^2$

Big O $\rightarrow$ $O(2^N)$

itr : $59N^2 + 64N^3 + 38N\sqrt{N} + 490N\log_2 N$

Big O $\rightarrow$ $O(N^3)$

itr : 101

Note: when no. of iterations are constant (independent of N)

Big O : $O(1)$

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

Big O $\rightarrow$ $O(N^2)$

itr : $4N^2 + 3N + 5 * 2^N$

Big O $\rightarrow$ $O(2^N)$

itr : $\dfrac{N(N+1)}{2} \Rightarrow \dfrac{N^2 + N}{2} = \dfrac{N^2}{2} + \dfrac{N}{2}$

Big O $\rightarrow$ $O(N^2)$

Doubts

Range sum query

A = [10 2 5 9 1 0 8]

0 1 2 3 4 5 6

B = [[0, 3],
[1, 4],
[2, 5]
]

ans = [26 17 15]

0 to 3
1 to 4
2 to 5

```
for ( int i=0; i < B.length; i++) {
    int s = B[i] [0];
    int e = B[i] [1];

    for ( s to e) {
        // find sum of Array
    }
}
```

B = [[0, 3],
[1, 4],
[2, 5]
]

i=0, s = 0 e = 3

i=1, s = 1 e = 4

i = 2, s = 2 e = 5