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

} next class

Agenda
- how to calculate total itr

Basics maths

i) Sum of N natural no.

$1 + 2 + 3 + 4 + 5 + \dots + N = \frac{N * (N+1)}{2}$

ii) Arithmetic Progression

4 7 10 13 16 19

3 3 3 3 3

first term : a

common diff: d

1	2	3	4		n
a	a+d	a+2d	a+3d	.....	a+(n-1)d

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

iii) Geometric progression

3 6 12 24 48 96

2 2 2 2 2

first term : a

common ratio : r

4 16 64 256

4 4 4

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

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

iv) How many times we need divide N by 2 till the N value reaches 1.

$N = 32$ $\quad 32 \xrightarrow{/2} 16 \xrightarrow{/2} 8 \xrightarrow{/2} 4 \xrightarrow{/2} 2 \xrightarrow{/2} 1$

$N = 24$ $\quad 24 \xrightarrow{/2} 12 \xrightarrow{/2} 6 \xrightarrow{/2} 3 \xrightarrow{/2} 1$

$\log_2 N$

$1024 \longrightarrow 512 \longrightarrow 256 \longrightarrow 128 \longrightarrow 64 \longrightarrow$

$32 \longrightarrow 16 \longrightarrow 8 \longrightarrow 4 \longrightarrow 2 \longrightarrow 1$

$\log_2 1024 = 10$ ✓ $\quad$ Sqrt(1024) => 32 ✗

Q.1

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

i → [1 N]

N iterations

O(N)

Q.2

```
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 itr

M itr

N + M itr

O(N+M)

Q.3

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

i : [0 100] 101 itr

[→ inclusive

) → exclusive

[a b] ⇒ b - a + 1

No. of itr are independent of input → O(1)

constant itr

Q.4

```
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

$i <= \sqrt{N}$

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

$O(\sqrt{N})$

Q.5

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

loop breaks at i = 1

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

// assume loop breaks after k itr

$i = \frac{N}{2^k}$, $\frac{N}{2^k} = 1 \Rightarrow N = 2^k$

$O(\log_2 N)$

$\log_2 N = \log_2 2^k$

$\log_2 N = k$

$\log_2 N$ itr

Q.6

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

infinite itr

Q.7

```
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 i = N

// assume loop breaks after K iterations

$i = 2^K$

$2^K = N$

take log base 2 on both sides

$\log_2 2^K = \log_2 N$

$\boxed{K = \log_2 N}$

$\log_2 N$ itr

Nested loop

Q.8

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

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

10N itr

O(N)

Q.9

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

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

N^2 itr

Q.10
```
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

$1 + 2 + 3 + 4 + 5 + .. + N$

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

$\Rightarrow \frac{N^2}{\cancel{2}} + \frac{\cancel{N}}{\cancel{2}}$ $O(N^2)$

Q.11
```
void func(int N) {
    for(int i=1; i<=N; i++) {
        for(int j=1; j<N; j=j*2) {
            SOP(i+" "+j);
        }
    }
}
```

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

$N * \log_2 N$ itr

$N * \log_2 N$

Q-12

```
void fun(int N) {
    for (int i=1; i<=2^N; i++) {
        sop(i);
    }
}
```

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

2^N iterations

Q-13

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

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

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

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

$a = 2$

$r = 2$

terms $= N$

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

Q.14

```
void func (int N) {
    for (int i=N; i>0; i=i/2) {
        for (int j=1; j<=i; j++) {
            Sop (i+" "+j);
        }
    }
}
```

i	j [1 i]	itr
N	[1 N]	N
		+
$\frac{N}{2}$	[1 $\frac{N}{2}$]	$\frac{N}{2}$
		+
$\frac{N}{4}$	[1 $\frac{N}{4}$]	$\frac{N}{4}$
		+ ... +
1	[1 1]	1

$$N + \frac{N}{2} + \frac{N}{4} + \ldots + 1$$

$a = N$, $r = \frac{1}{2}$

terms = $\log_2 N + 1$

→ to find total no. of terms (t)

$$ar^{t-1} = 1$$

$$N * \left(\frac{1}{2}\right)^{t-1} = 1$$

$$\frac{N}{2^{t-1}} = 1$$

$$N = 2^{t-1}$$

take log on both sides

$$\log_2 N = \log_2 2^{t-1}$$

$$\log_2 N = t - 1 \quad , \quad \boxed{t = \log_2 N + 1}$$

$a = N$, $r = \frac{1}{2}$

terms $= \log_2 N + 1$

$$S = \frac{a(r^t - 1)}{r - 1}$$

$$= \frac{N\left(\left(\frac{1}{2}\right)^{\log_2 N + 1} - 1\right)}{\frac{1}{2} - 1}$$

$$= \frac{+N\left(1 - \left(\frac{1}{2}\right)^{\log_2 N + 1}\right)}{+\frac{1}{2}}$$

$$= 2N\left(1 - \left(\frac{1}{2}\right)^{\log_2 N + 1}\right)$$

$$= 2N\left(1 - \frac{1}{2^{\log_2 N + 1}}\right)$$

$$2^{\log_2 N} = N$$

$$= 2N\left(1 - \frac{1}{2^{\log_2 N} \times 2}\right)$$

$$= 2N\left(1 - \frac{1}{N \times 2}\right)$$

$$= \cancel{2N}\left(\frac{2N - 1}{\cancel{2N}}\right) = 2N - 1$$

Comparing terms

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

Big O notation — what, why (next class)
                — how ✓

how to find Big O ?

i) find total no. of iterations

ii) discard lower order terms (keep the highest order terms)

iii) discard constant coefficients.

it$_1$: $10\,N\log_2 N + 50\,N\sqrt{N} + 100\,N^2$

$O(N^2)$ → TC

it$_2$: $90\,N\sqrt{N} + N\log_2 N + 64\,N$

$O(N\sqrt{N})$ → TC

it$_3$: $4\,N\log N + 3\,N\sqrt{N} + 10^6$

$O(N\sqrt{N})$ → TC

Doubts
=

Arrays.sort(arr); → arr will be sorted

		3	4	5	6	7
1	2	~~2~~	~~2~~	~~2~~	~~3~~	~~3~~

temp = ~~8~~ 7

cnt = 2

		3	4	5	6
1	2	~~2~~	~~2~~	~~4~~	~~4~~

temp = 4

cocunt = 2

3 4 4 4 5 6 6

		5	6	7	8	9	
3	4	~~4~~	~~4~~	~~5~~	~~6~~	~~6~~	temp = 7

cnt =

1 + 2 + 2 + 2 + 3

⇒ 10

Range sum query

A = { 10 20 5 -9 2 6 }

Q [[0, 2] , [1, 4]]

ans: [35 , 18]

for (on query) {
  s → Q[i][0]
  e → Q[i][1]
  for (sum of s to e)

5

List : 10 20 30 ~~40~~

40 10 20 30

val = 40

```
int val = list.remove(n-1);
list.add(0, val);
```

idx val

1st. N = 12 (i → 1 to N)

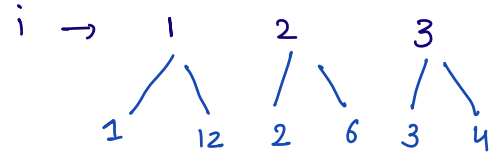
i → 1 2 3 4 5 6 7 8 9 10 11 12

count = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

2nd N = 12

i → (1 to $\sqrt{N}$)

(i, N/i)



count = ~~1~~ ~~4~~ 6