# Time Complexity

(i) `for(i=0; i<n; i++)` — $n+1$

```
{
  stm;
}
```

— $n$

Total $= n+1+n$
$= 2n+1$
$= O(n)$

(ii) `for (i=1; i<n; i=i+2)` → $\frac{n}{2}+1$

```
{ statement; → n/2
}
```

Total $= \frac{n}{2}+\frac{n}{2}+1$
$= \frac{n+n+2}{2} = \frac{2n+2}{2} = \frac{2(n+1)}{2} = n+1$

$O(n)$

(iii) Nested Loop:

```
for(i=0; i<n; i++) — n+1
{ for(j=0; j<n; j++) → n*(n+1)
  { statement; — (n*n)
  }
}
```

Total $= (n+1) + (n^2+n) + n^2$
$= n+1+n^2+n+n^2$
$= 2n^2+2n+1$
$= O(n^2)$

(iv)

```
for (i=0; i<n; i++) n
  for (j=0; j<i; j++)        n*(n-1)
    statement;  - n
```

| i | j | statement |
|---|---|---|
| 0 | 1x | x |
| 1 | 0<br>1x | 1 |
| 2 | 0<br>1<br>2x | 2 |
| 3 | 0<br>1<br>2<br>3x | 3 |
| : | : | : |
| n | (n-1) | n |

Total = $n + n^2 - n + n$

$= n^2 + n$

$= O(n^2)$ .

(v)

```
p=0;
for (i=1; p<=n; i++)
{
  p=p+1;
}
```

| i | p |
|---|---|
| 1 | 0 + 1 = 1 |
| 2 | 1 + 1 = 2 |
| 3 | 1+2+3 |
| 4 | 1+2+3+4 |
| : | : |
| K | 1+2+3+4+ ....+K |

Assum, $p > n$ (stopping)

$p = \frac{K(K+1)}{2}$

$\frac{K(K+1)}{2} > n$

$\Rightarrow K^2 > n$

$\therefore K > \sqrt{n}$

$O(\sqrt{n})$ times. ✓

(vi) `for(i=1; i<n; i=i*2)`
`{ statement; }`

| i |
|---|
| 1 |
| $2^0 = 1*2 = 2$ |
| $2^1 = 2*2 = 4$ |
| $2^2 = 4*2 = 8$ |
| $2^3 = 8*2 = 16$ |
| $2^4 = 16*2 = 32$ |
| ⋮ |
| $2^k$ |

$2^k >= n$

$\therefore i = 2^k$

$\therefore 2^k = n$

$k = (\log_2 n) \therefore O(\log_2 n)$

Comparison

`for(i=1; i<=n; i++)`
`{ stm; }`

$i = 1+1+1+1+1 \cdots = n$

$k = n$

যদি log আসে তবে তার double value নিবে। যেকারণে আমাদের যেখানে ceil value নিতে হবে। যেমন: একটা Code এ যদি

$n = 8$ হয়

$i = 1$
$2$
$4$
$8$ X

3 times

$\log_2 8 = \log_2 2^3 = 3 \log_2 2 = 3$

$n = 10$ হয়

$i = 1$
$i = 2$
$i = 4$
$i = 8$
$i = 16$ X

4 times

$\log_2 10 = 3.2$

যেহেতু 4 times হয়েছে

তাই, Ceil $\lceil 3.2 \rceil = 4$

(vii) $p=0$

```
for(i=1; i<n; i=i*2)
  {
  } p++;          — p = log n

for(j=1; j<p; j=j*2)
  {
  stmt;           — log p
  }
```

$= O(\log \log n)$

(viii)

```
for(i=0; i<n; i++)  — n
  {
  for(j=1; j<n; j=j*2)  — n log n
    {
    stmt;
    }
```

$n \times \log n$

$2n \log n + n$

$\Rightarrow O(n \log n)$

Ex 1:

Consider two Computer A = faster
B = Slower
(A is faster 100 times than B)

A run insertion sort
B a merge
input size 2 million numbers.
B execute only 100 million instruction Per second.
A " 1 billion "

$C_1 = 4, \quad C_2 = 50$

Insersion sort $= C_1 n^2$ times
Merge " $= C_2 * n * \log n$ "

Q-
Computer A takes $= \frac{4 * (2 * 1000000)^2}{1000 * 1000} = \frac{4 \times (2 \times 10^6)^2}{10^9}$ sec

$= \frac{4 * 4 * 12^{12}}{10^9} = 16 * 10^{12-9} \simeq 4000$ sec

$= 16 * 10^3 = 16$ min

B takes $= \frac{50 * 2 * 10^6 * \log(2 \times 10^6)}{10^7}$

$= 209$ second.

Math series

1, 2, 3, ----, n

$1+2+3+\cdots+n = \frac{n(n+1)}{2}$

$1^2+2^2+3^2+\cdots+n^2 = \frac{n(n+1)(2n+1)}{6}$

$k = 2^n, n>0$

$\Rightarrow \log k = \log 2^n$

$\Rightarrow \log k = n \log 2$

$\therefore n = \log k$

Types of Algorithm:

- Divide and Conquer
- Greedy Approach
- Dynamic programming.