Tutorial → 1

Q1

These are the notations that tells about the complexity of an algorithm.

i) Time complexity

It tells how much ~~space~~ time is going to be taken by our algorithm.

ii) Space complexity.

It tells how much space is taken by our algorithm in main memory.

Q2

```
for(i=1 to n)
{ i = i * 2; }
```

$i \to 1, 2, 4, 8, 16 \ldots$

$2^0, 2^1, 2^2, 2^3, 2^4 \ldots 2^k$

let $i > n$

$i = n$

$2^k = n$

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

Q3

$$T(n) = \begin{cases} 1 & , n \le 0 \\ 3T(n-1) & , n > 0 \end{cases}.$$

$T(n) = 3T(n-1)$

Master theorem for decreasing func.

$T(n) = aT(n-b) + f(n)$

$f(n) = n^0$

$a > 1 \Rightarrow O(n^0 \cdot 3^{n/1})$

$\Rightarrow O(3^n)$.

(4) $T(n) = \begin{cases} 1 & n <= 0 \\ 2T(n-1) & n > 0 \end{cases}$

$a = 2, \quad b = 1, \quad f(n) = n^0$

$O(2^n)$

(5)

```
i=1, s=1
while (s<=n)
{   i++;
    s=s+i;
    Pf ("#");
}
```

i	s
1	1
2	2
3	2+3
4	2+3+4
5	2+3+4+5
⋮	⋮
k	2+3+4+5 ... k

let, $s > n$

$\therefore s = \frac{k(k+1)}{2}$

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

$k^2 > n \Rightarrow k = \sqrt{n} \Rightarrow O(\sqrt{n})$

(6)

```
void function(int n)
{ int i, count = 0;
  for(i=1; i*i<=n; i++)
      count++;
}
```

$i^2 <= n$

let $i^2 > n$

$i^2 = n$

$\Rightarrow i = \sqrt{n} \Rightarrow O(\sqrt{n})$

(7)

```
void function(int n)
{ int i, j, k, count = 0;
  for(i=n/2; i<=n; i++)          ---> n
      for(j=1; j<=n; j=j*2)      ---> log2n
          for(k=1; k<=n; k=k*2)  ---> log2n
              count++;
}
```

$O(n \log_2^2 n)$

(8) function(int n)
{ if (n == 1) return; ⟶ O(1)
for (i = 1 to n) { ⟶ O(n)
for (j = 1 to n) { ⟶ O(n)
pf ("*"); ⟶ O(n²)
}
}
function(n-3); ⟶ T(n-3)
}

$T(n) = T(n-3)$

$a = 1, \; b = 3, \; f(n) = n^0$

$\Rightarrow O(n)$

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

(9) void function(int n) ⟶ n
{ for (i = 1 to n) {
for (j = 1; j <= n; j = j + i)
pf ("*");
}
}

i	j
1	1+2+3+4...n → n(n+1)/2
2	1+2 ... → n(n+1)/2
	i

$O(n^2)$

(10) n^k e^n

$n^k = O(e^n)$