19.02.23

Assignment 1

—x—x—

(2)

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

i
1
2
$4 = 2^2$
2^3
2^4
⋮
2^k

Assume $i \geq n$

$\therefore i = 2^k$

$2^k = n$

$k = \log_2 n$

Ans → $O(\log n)$

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

$T(n) = 3T(n-1)$ — (i)

$T(n-1) = 3T(n-2)$

$T(n) = 3[3T(n-2)]$ — (ii) by substituting in eqn (i)

$T(n-2) = 3T(n-3)$

$T(n) = 3^3 T(n-3)$ — (iii)

⋮ for k times

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

Assume $n - k = 0$

$\therefore n = k$

$T(n) = 3^n T(0)$

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

Ans → $T(n) = 3^n$ $\quad O(3^n)$

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

$T(n) = 2T(n-1) - 1$ — (i)

$T(n-1) = 2T(n-2) - 1$

substituting $T(n-1)$ in eqⁿ (i)

$T(n) = 2(2T(n-2)-1) - 1$

$T(n) = 2^2 T(n-2) - 2 - 1$ — (ii)

$T(n-2) = 2T(n-3) - 1$

$T(n) = 2^2[2T(n-3)-1] - 2 - 1$

$T(n) = 2^3 T(n-3) - 2^2 - 2 - 1$ — (iii)

for k times

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - \dots - 2^2 - 2^1 - 2^0$

Assume $n - k = 0$

$\therefore n = k$

$T(n) = 2^n T(0) - 2^{n-1} - 2^{n-2} - 2^{n-3} - \dots - 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} - \dots - 2^2 - 2 - 1$

$= 2^n - (1 + 2 + 2^2 + 2^3 + \dots + 2^{n-2} + 2^{n-1})$

$= 2^n - \left(\frac{1(2^n - 1)}{1}\right)$ $\quad S_n = \frac{a(r^n - 1)}{r-1}, r>1$

$= \cancel{2^n} - \cancel{2^n} + 1$

$= 1$ $\quad \underline{\underline{O(1)}}$

(5)
```
int i=1, s=1;
while (s<=n)
{ i++;
  s=s+i;
  print("#");
}
```

i	s
1	1
2	3 = 1+2
3	6 = 1+2+3
4	1+2+3+4
5	1+2+3+4+5
⋮	⋮
m times	1+2+3+ - - - - +m

Assume $s > n$

$\because s = (1+2+3+ - - - m)$

$= \frac{m(m+1)}{2}$

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

$m^2 > n$

$m > \sqrt{n}$

Ans $O(\sqrt{n})$

(6)
```
void func(int n)
{
  int i, count=0;
  for(i=1 ; i*i <= n; i++)
  { count++;
  }
}
```

i: 1^2, 2^2, 3^2, ⋮ for K items, K^2

count

Assume ~~i*i > n~~ $i*i > n$

~~$\therefore$ i*i = K~~ $\therefore i*i = K^2$

$K^2 > n$

$[K > \sqrt{n}]$

Ans $O(\sqrt{n})$

(7)
```
void func(int n)
{ int i, j, K, count=0;
  for(i=n/2 ; i<=n ; i++)  — log n
  { for (j=1; j<=n ; j=j*2)  — log n x log n
    { for (K=1; K<=n ; K*=2)  — log n x log n x log
      { count++;
      }
    }
  }
}
```

~~i~~ ~~j~~ ~~K~~

~~n/2~~ ~~n/2+1~~ ~~n/2+1~~

$\frac{n}{2}+(1+1+ - - n)$

$O(\log^3 n)$

$\frac{n}{2}+$

8 func(int n) $T(n)$
{
if (n == 1) — 1
return;
for (i=1 to n) — $n+1$
{
for (j=1 to n) — $n \times (n+1)$
{
//
}
}
func(n-3); — $T(n-3)$
}

$T(n) = T(n-3) + n^2 + n + n + 1 + 1$

$T(n) = T(n-3) + \underline{n^2}$
Highest deg Term

$$\boxed{\begin{array}{l} T(n) = aT(n-b) + f(n) \\ a \geq 0 \quad \text{and } f(n) = O(n^K) \\ b > 0 \qquad\qquad \text{where } K \geq 0 \end{array}}$$ { MASTER'S THEOREM

$a = 1$
$b = 3$
$f(n) = n^2 = O(n^2)$
$K = 2$

Case 2: i.e. if $a=1$
$O(n^{K+1})$
$O(n^{2+1})$

Ans. $O(n^3)$

9 void func(int n)
{
for (i=1 to n) — $(n+1)$
{
for (j=1; j<=n; j=j+i)
{
//
}
}
}

$\downarrow$
$n \times \sqrt{n}$

i	j no. of
1	1
2	3
3	1+2+3
4	1+2+3+4
;	for m
m tim	[illegible]

Assume $j \geq n$

$\therefore j = 1+2+3+ \dots + m$

$j = \frac{m(m+1)}{2}$

$\therefore \frac{m(m+1)}{2} > n$ $\quad O(n\sqrt{n})$ $^{1+\frac{1}{2}}$

$m^2 > n$ $\quad O(n^{3/2})$

$m < \sqrt{n}$

(1) Asymptotic Analysis of an algo, refers to defining the mathematical ~~Analysis~~ boundation of its run time performance based on the input size.

There are generally 3 types of Asymptotic notations currently in use:

(1) Big oh (O) (2) Theta (Θ) (3) Big Omega (Ω)

upperbound — Avg. Bound — Lower Bound

(i) Big Oh (O):

The fn $f(n) = O(g(n))$ iff $\exists$ +ve constant c and n_0 such that $f(n) \leq c * g(n) \;\; \forall n > n_0$

eg. $f(n) = 2n+3$

$2n+3 \leq 2n^2+3n^2$

$2n+3 \leq 5n^2$

$f(n)$, c, $g(n)$

(ii) Big Omega (Ω):

The fn $f(n) = \Omega(g(n))$ iff $\exists$ +ve constants c and n_0 such that $f(n) \geq c * g(n) \;\; \forall n > n_0$

eg $f(n) = 2n+3$

$2n+3 \geq 1 * n$

$f(n)$, c, $g(n)$

(iii) Theta (Θ) – The f^n $f(n) = \Theta(g(n))$ iff

$\exists$ +ve constants c_1, c_2, n_1

such that $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$

eg $f(n) = 2n + 3$

$$1 * n \leq 2n + 3 \leq 2n^2 + 3n^2$$

$$\underset{c_1}{1} * \underset{g(n)}{n} \leq \underset{f(n)}{2n+3} \leq \underset{c_2}{5}\, \underset{g(n)}{n^2}$$

Ans 10

If K is a constant & $c \geq 1$

then,

c^n grows faster than n^K as $n \to \infty$.

because,

rate of growth of c^n is exponential while rate of growth of n^K is polynomial.

Or, we can say $c^n = O(n^K)$.