Assignment 1

Q-1 Solve the following loops & calculate time complexity

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

- Here, i-loop will run n/2 times
  j-loop will run n/2 times

no. of times i-loop run $\Rightarrow n - n/2 = n/2$

and for j-loop $\rightarrow j + n/2 \leq n$
$j \leq n - n/2$
$j \leq n/2$

so, it will run n/2 times

and for k-loop

Step	k	
0	2	$= 2^1$
1	4	$= 2^2$
2	8	$= 2^3$

for k th step, it will be 2^n

Assuming $\rightarrow 2^k \geq n$

So, $\log 2^k \geq \log n$

$k \geq \log n$

So, total complexity $\rightarrow n/2 \times n/2 \times \log n$

$\frac{n^2}{4} \log n$

for asymtotic notation $\rightarrow n/2 \times n/2 \times \log n$

$= \frac{n^2}{4} \log n$

$= O^2 (n^2 \log n)$

2)
```
void function (int n) {
    if (n==1) return;
  for (int i=1; i<=n; i++) {
    for (int j=1; j<=n; j++) {
        printf ("K");
        break;
    } } }
```

- Here i-loop will run n-times and inner loop will be run only 1 time in every intration so,

$T(n) = 2n$ for $n > 1$

so, asympatic time complexity $= O(n)$

3)
```
void function (int n) {
      int i, count = 0;
       for (i =1; i * i <=n; i++)
       count ++
      }
```

- Here let's see the every iteration

Step	
0	1
1	2
⋮	
k	k

suppose, $k^2 > n$

$k > \sqrt{n}$

↳ So, the Asymptotic Time complexity - $O(\sqrt{n})$

4) function (int n) {

```
for (int i=1 ; i<=n ; i++) {
    for (int j = 1; j<=n ; j*=2) {
        Print ("*");
}}}
```

- Here, the outer loop will be execute n times

↳ and for inner loop :

Step	j
0	$1 = 2^0$
1	$2 = 2^1$
2	$4 = 2^2$
⋮	⋮
k	2^k

suppose

$2^k \geq n$

$k \geq \log n$

- So, final Asymptotic time complexity $O(n \log n)$

5] Function (int n){
    for (int i=1; i<=n/3; i++){
        for (int j=1; j<=n, j+=n){
            Printf ("*");
    }}}

→ Here, the outer i loop will be execute n/3 time and for inner j loop:-

Step	j
0	1
1	4
2	8
⋮	
k	4^k

suppose

$4k \geq n$

$k \geq n/4$

- so, final Asymptotic time complexity $O(n/3 \times n/4)$
  $O(n^2)$

6] function (int n){
    int sum=0;
    for (int i=0; i<n; i++){
        if (i>j){
            sum +=1;
        } else{
            for (int k=0; k<n; k++){
                sum -=1;
    }}}

- If we consider $j \to \infty$

- So, In that case, the outer-loop will run n-times and in each iteration the k-loop will run n loop.

- So, total iteration will be $n \times n$

- So, Asymptatic time complexity $O(n \times n) = O(n^2)$

Q-2) Calcute order of complxity using master's theorem:

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

Comparing with master's theorem

$a = 3$ $\quad$ $k = 2$

$b = 2$ $\quad$ $P = 0$

- So, $a < b^k$ ($\because 3 < 2^2$)

according to case 3-A

$T(n) = O(n^k \log^P n)$

$= O(n^2 \log^0 n)$

$= O(n^2)$

2) $T(n) = 4T(n/2) + n^2$

Comparing with master's theorem

$a = 4$, $b = 2$, $k = 2$, $P = 0$

So, $a = b^k$ ($\because 4 = 2^2$)

according to case 2-A

$T(n) = O(n \text{ to } n^{\log_b a} \log^{p+1} n)$

$= O(n^{\log_2 4} \log^1 n)$

$= O(n^2 \log n)$

3] $T(n) = T(n/2) + n^2$

- $a = 1, b = 2, k = 2, p = 0$

So, $a < b^k$ $(\because 1 < 2^2)$

$T(n) = O(n^2 \log^p n)$

$= O(n^2 \log^0 n)$

$= O(n^2)$

4] $T(n) = 16T(n/4) + n$

- $a = 16, b = 4, k = 1, p = 0$

So, $a > b^k$ $(\because 16 = 4^1)$

$T(n) = O(n^{\log_b a})$

$= O(n^{\log_4 16})$

$= O(n^2)$

5] $T(n) = 2T(n/2) + n \log n$

$a = 2, b = 2, k = 1, p = 1$

So, $a = b^k$ $(\because 2 = 2^1)$

$T(n) = O(n^{\log_b a} \cdot \log^{p+1} n)$
$= O(n^{\log_2 2} \log^{1+1} n)$
$= O(n \log^2 n)$

6) $T(n) = 2T(n/2) + n\log^{-1} n$

$a=2, b=2, k=1, p=-1$

so, $a=b^k$ ($\because 2=2^1$)

$T(n) = O(n^{\log_b a} \log \log n)$
$= O(n^{\log_2 2} \log \log n)$
$= O(n \log(\log n))$

7) $T(n) = 2T(n/4) + n^{0.51}$

$a=2, b=4, k=51/100, p=0$

so, $a<b^k$ ($\because 2 < 4^{51/100}$)

$T(n) = O(n^{51/100} \log^0 n)$
$= O(n^{0.51})$

8) $T(n) = 6T(n/3) + n^2 \log n$

$a=6, b=3, k=2, p=1$

so, $a<b^k$ ($\because 6 < 3^2$)

$T(n) = O(n^2 \log^1 n)$
$= O(n^2 \log n)$

9) $T(n) = 7T(n/3) + n^2$

$a = 7,\ b = 3,\ k = 2,\ p = 0$

so, $a < b^k$ ($\because 7 < 3^2$)

$T(n) = O(n^2 \log^0 n)$
$= O(n^2 \log^0 n)$
$= O(n^2)$

10) $T(n) = 4T(n/2) + \log n$

$a = 4,\ b = 2,\ k = 0,\ p = 1$

So, $a > b^k$ ($\because 4 > 2^0$)

$T(n) = O(n^{\log_b a})$
$= O(n^{\log_2 4})$
$= O(n^2)$

11) $T(n) = 16T(n/4) + n!$

$a = 16,\ b = 4,\ k = n,\ p = 0$

so, $a < b^k$ ($\because 16 < 4^n$)

$T(n) = O(n^k \log^p n)$
$= O(n^n \log^0 n)$
$= O(n^n)$

12) $T(n) = 3T(n/2) + n$

$a = 3, b = 2, k = 1, P = 0$

so, $a > b^k$ $(\because 3 > 2^1)$

$T(n) = O(n^{\log_b a})$
$= O(n^{\log_2 3})$

13) $T(n) = 3T(n/4) + n \log n$

$a = 3, b = 4, k = 1, P = 1$

So, $a < b^k$ $(\because 3 < 4^1)$

$T(n) = O(n^k \log^p n)$
$= O(n^1 \log^1 n)$
$= O(n \log n)$

14) $T(n) = 3T(n/3) + n/2$

$a = 3, b = 3, k = 1, P = 0$

so, $a < b^k$ $(\because 3 < 4^1)$

$T(n) = O(n^{\log_b a} \log^{P+1} n)$
$= O(n^{\log_3 3} \log^{0+1} n)$
$= O(n \log n)$

15] $T(n) = 2T(n/2) + n$

$a=2,\ b=2,\ k=1,\ P=0$

so, $a = b^k$ ($\because 2 = 2^1$)

$T(n) = O(n^{\log_b a} \log^{P+1} n)$

$= O(n^{\log_2 2} \log^{0+1} (n))$

$= O(n \log n)$

Q-3) compare the fun^{ns} and determine which is greater

1) $f(n)\ n\log n$
$g(n) = n \log n$

$f(n) = n \log n$ $\qquad g(n) = n \log n$

$g(n) = n \log n = 2^{(\log n)^2}$

so, $f(n) = n \log n$
$g(n) = 2^{(\log n)2}$

$\therefore$ ~~A~~ $f(n) << g(n)$

2) $f(n) = 2^{\log n}$
$g(n) = n^{\sqrt{n}}$

$f(n) = 2^{\log n}$ $\qquad f(n) = 2^{\log_2 n} = n$

so, $f(n) = n$ $\qquad g(n) = n^{\sqrt{n}}$

$n^{\sqrt{n}} = e^{\sqrt{n}\cdot\log n}$ $\quad n = 1^{\log n}$

so, $f(n) = n << g(n) = n^{\sqrt{n}}$

$g(n) >> f(n)$

3) $f(n) = 2^n$
$g(n) = 2^{2n}$

$g(n) = 2^{2n} = (2^n)^2$

so, $g(n) = (f(n))^2$

$g(n) >> f(n)$

Q-4) Determine time complexity for

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

$T(n) = T(n-1) = T(n-2) = \text{-----} = T(0)$

$T(n) = O(n)$

2)
```
void Test (int n)
{
  if (n>0)
  {
  Test(n-1);
  Test (n-1);
  }
}
```

Page No.
Date: / /

↳ $T(n) = 2*T(n-1)$
$T(n-1) = 2*T(n-2)$
$T(n-2) = 2*T(n-3)$
⋮
So,
$T(n) = 2.2.2 \ldots 2T(*0)$
since $T(0) = 1$;

$T(n) = 2^n$

↳ Time complexity : $O(2^n)$

3) void test (int n)
{
if (n>1)
{
printf ("ICT");
Test (n/2);
}
}

→ $T(n) = T(n/2) + O(1)$, for $n>1$
$T(n) = O(1)$, for $n \le 1$

Solving the

↳ n
↳ $n/2$
↳ $n/4$
⋮
$n <= 1$

Page No.
Date: / /

→ Time complexity $O(\log n)$

4) $T(n) = \begin{cases} 2T(n/2) + n & ; n > 1 \\ 1 & ; n = 1 \end{cases}$

- The function divides the problem with 2 subproblem of size $n/2$

number of levels is $\log_2 n$

$$T(n) = n \log_2 n$$

Time complexity : $O(n \log n)$