Assignment 1

Kavish Chaturvedi
Section: ML
Roll No: 2014669

Ans-1 ⇒ Asymptotic notations are methods/ languages using which we can define the running time of a specific algorithm based on input size.
To represent the upper and lower bounds, we need some kind of syntax. and this is represented in the form of function $f(n)$

⇒ Linear = n
⇒ Logarithmic: $\log n$
⇒ Quadratic: n^2
⇒ Exponential: a^n

Ans-2 ⇒ for ($i=1$ to n) { $i = i*2$ }
'i' is doubling everytime.

for k^{th} step → $2^k = n$ & for $(k+1)$ we are out of loop
taking log both sides

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

$$k = \log_2 n$$

Time Complexity: $O(\log n)$

Ans3 ⇒ $T(n) = 3T(n-1)$ if $n>0$, otherwise 1
$T(n) = aT(n-b) + f(n)$ [Master Theorem]
$a=3, \ b=1$
$\because f(n) = 0, \ k=0$

$T(n) = O(n^k a^{n})$

$\Rightarrow T(n) = O(n^0 a^n)$

$T(n) = 3^n$

Ans-4 $\Rightarrow$ $T(n) = 2T(n-1) - 1$

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

$a = 2, b = 1$

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

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

$= 4T(n-2) - 3$

Similarly

$T(n) = 8T(n-3) - 7$

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

let $n-1 = 1$

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

$2^k (T(1) - 1) + 1$

$\Rightarrow T(n) = O(2^k) = O(2^n)$

Ans 5 $\Rightarrow$

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

Since s is increasing by 1,

$\therefore O(n)$

Ans-6 $\Rightarrow$

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

When $n = 5$

1, $i = 1$, $1 \times 1 <= n$

$i = 2$, $2 \times 2 <= n$

out of loop.

loop is working for $n/2$ times only

$\therefore O(n/2) \Rightarrow O(n)$

Ans-7 ⇒

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

$O(n \log^2 n)$

Ans-8 ⇒

```
fun (int n) {
    if (n == 1) return;
    for (i = 1 to n) {
        for (j = 1 to n) {
            print (n)
        }
    }
    fun (n-3)
}
```

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

Ans 9 ⇒ $O(n \log n)$

Ans-10 ⇒ $f(n) = n^k$, $k >= 1$

$g(n) = a^n$, $a > 1$

is $f(n) = O(g(n))$

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

take log

$k \log n = n \log a$

$$\frac{\log n}{\log a} = \frac{n}{k}$$

$$\log_a n = \frac{1}{k} n$$

let $1/k = c$

$O(cn)$ Time Complexity

Ans-11 =>

i	j	loop run
0	1	1
1	2	2
3	3	3

$\because$ loop is running $n/2+1$ times

$= O(n/2+1)$

$= O(n)$

Ans-12 => We know that the first statement takes $O(1)$ time, while else (2nd statements) takes $T(n-1)+T(n-2)$

$\therefore$ recursive eq^n $= T(n-1) + T(n-2) + O(1)$

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

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

Space $= O(n)$

Ans-13 => $n\log n$

```
for (i=1 ; i<n ; i++)
    for (j=1 ; j<=n ; j=j+1)
        printf("#")
```

n^3

```
for (i=1 ; i<n ; i++)
  for (j=1 ; j<n ; j++)
    for (k=1 ; k<n ; k++)
      printf("#")
```

loglog n

```
int fun (int n) {
    if (n<22) return 1;
    else return (fun(floor(sqrt (n))) + n);
}
```

Ans-14 => $T(n) = T(n/4) + T(n/2) + cn^2$

We can assume

$T(n/2) > T(n/4)$

$Tn = 2T(n/2) + cn^2$

Applying master's method

$a=2, b=2$

$k = \log_b a = \log_2 2 = 1$

$n^k = n$

$f(n) = n^2$

It is $\Theta(n^2)$

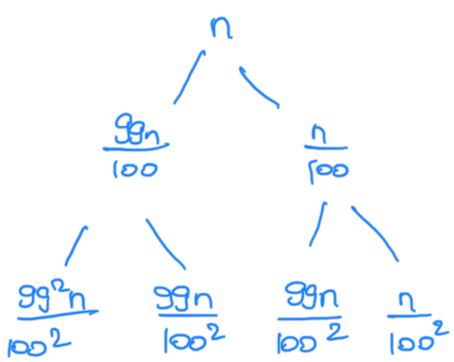
But as $T(n) <= \Theta(n^2)$

$T(n) = O(n^2)$

Ans-16 => If k is a constant greater than 1

Then $TC = O(\log\log n)$

Ans-17 => $T(n) = T\left(\frac{99n}{100}\right) + T\left(\frac{n}{100}\right)$



On taking longer branch $\left(\frac{99n}{100}\right)$

$TC = \log_{100/99} n \approx \log n$

We can say that the base of log doesnot matter as it is only a matter of constant.

Ans-18 ⇒ a) $100 \; \log\log n \; \sqrt{n} \; n \; \log n! \; n\log n \; n^2 \; 2^n \; 2^{2n}/4^n \; n!$

b) $1 \; \log\log n \; \sqrt{\log n} \; \log n \; 2\log n \; \log 2n \; n \; 2n \; 4n \; \log n! \; n\log n \; n^2 \; 2(2^n) \; n!$

c) $96 \; \log_8 n \; 5n \; \log n! \; n\log_6 n \; n\log_2 n \; 8n^2 \; 7n^3$

Ans-19 ⇒ linear Search (array, key)
for i in array
if value == key
return i

Ans-20 ⇒ Iterative Insertion Sort
insertionSort (arr, n)
loop from i=1 to i=n-1
pick element arr[i] and insert
it into sorted sequence arr [0 -- i-1]

Ans 21, 22 ⇒

	Best	Avg	Worst	SC	Stable	Inplace	On
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	✓	✓	×
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	×	✓	×
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	✓	✓	✓

Ans-24 ⇒ $T(n) = T(n/2) + c$