DATE

DAA Assignment-1

Deepak Godiyal
SEC-A

Ans 1) Asymptotic notation means towards infinity i.e our input is very large.
The notations are used to tell the complexity of an algorithm when input is very large.

(i) Big O (O)
(ii) Big Omega (Ω)
(v) Small omega (ω)
(iii) Theta (θ)
(iv) Small O (o)

Ans 2)

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

$i = \underbrace{1, 2, 4, 8, \ldots\ldots, n}_{K \text{ steps}}$

it is a GP

n^{th} term $= ar^{n-1}$

$n = 1 \times 2^{K-1}$

$n = \frac{2^K}{2}$

$2n = 2^K$

taking $\log_2$ both side

$\log 2n = K \log_2^2$

$\log_2 2^1 + \log_2 n = K$

$\therefore \quad T.C = O(\log n)$

Ans 3) $T(n) = 3T(n-1)$ — ① if $n>0$, $T(0)=1$

using backward ~~forward~~ substitution

Put $n = n-1$ in eq^n 1

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

$T(n-1) = 3T(n-2)$ — ②

Put value of $T(n-1)$ in eq^n ①

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

$T(n) = 9T(n-2)$ — ③

Put $n = n-2$ in eq^n 1

$T(n-2) = 3T(n-3)$ — substitute in eq^n 3

$T(n) = 27T(n-3)$

PAGE No.
DATE

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

$T(0) = 1$

$n - k = 0$

$k = n$

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

$T(n) = 3^n$

$T(n) = 3^n$

$TC = O(3^n)$

Ans 4) $T(n) = \{2T(n-1) - 1\}$ $T(0) = 1$ → (1)

$n = n-1$

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

↳ Put in eq^n (1)

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

$T(n) = 4T(n-2) - 2 - 1$ - (2)

Put $n = n-2$ in eq^n 1

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

PAGE No.
DATE

$T(n) = 4*(2T(n-3)-1) - 2 - 1$

$T(n) = 8T(n-3) - 4 - 2 - 1$

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - 2^{k-3} \ldots$

$n - k = 0$

$n = k$

$T(n) = 2^n - 2^{n-1} - 2^{n-2} - \cdots - 2^{n-n}$

$T(n) = 2^n \left[1 - \frac{1}{2} - \frac{1}{4} \cdots \frac{1}{n} \right]$

$T(n) = 2^n \times a \left(\frac{1}{2}\right)^{n-1}$

$T(n) = 2^n \times \frac{1\left(\left(\frac{1}{2}\right)^n - 1\right)}{-\frac{1}{2}}$

$T(n) = -2^{n+1} \times \frac{1}{2^n} + 2^n$

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

PAGE No.
DATE

$\therefore$ T.C = $O(2^n)$ =

Ans5)

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

i	s
1	1
2	3
3	6
4	10
5	15
:	:
k	

$S = \underbrace{1, 3, 6, 10, 15, \ldots - n}_{k \text{ times}}$

$S = 1, 1+2, 1+2+3, 1+2+3+4, \ldots\ldots 1+2+\ldots n$

n^{th} term = $\frac{n(n+1)}{2}$

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

$n = \frac{k^2 + k}{2}$

$2n = k^2 + k$

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

Ans:)

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

$i = 1, 2, 3, 4, \ldots = \sqrt{n}$

$i^2 = 1, 4, 9, 16 \ldots n$

k steps

Series: $1^2, 2^2, 3^2, \ldots n$

let $\sqrt{n} = n$

Series = $1, 2, 3, 4, \ldots n$

AP

$T(c) = O(n)$

n^{th} term

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

$n = 1 + k$

$k = n - 1$

T.C = $O(n)$

Ans7)

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

K loop :-

$K = 1, 2, 4, 8, \ldots\ldots n$ (K times)

$n = 1 \times 2^{K-1}$

$T(c) = O(\log n)$

j loop :-

similarly like K loop

$\log(n)$

i loop :-

$i = \frac{n}{2}, \frac{n}{2}+1, \frac{n}{2}+2, \ldots\ldots \frac{n}{2}+n$

$T(c) = O(n)$

$\therefore T(c) = n(\log(\log n))$

Ans 8)

```
function (int n)
{
    if (n == 1)
        return;
    for (i = 1 to n)
    {
        for (j = 1 to n)
            { printf ("*");
    }
    function (n-3);
}
```

TC of i loop = O(n)
TC of i loop = O(n)

function calling:

n, n-3, n-6, n-9, ... n-(n-1)
AP

$n-(n-1) = n + (K-1)-3$

$n-K+1 = n-3K+3$

$1 = -3K+3$

$3K = 2$

$K = \frac{2}{3}$

TC = O(n)

∴ overall TE = $O(n^3)$

Ans 9)

```
void function (int n)
{
    for (i=1 to n)
    { for (j=1 ; j<=n; j=j+i)
        {
            printf ("*");
        }
    }
```

i loop

1, 2, 3, --- n

1, 3, 5, ----- n

1, 4, 7, --- n

→ 1, 5, 9, ...

⋮

1

n = $O(n)$

outside i loop = n

∴ TC = $O(n^2)$

Ans 10) relation $O(n^k) < O(a^n)$

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

let $k=1$ and $a=2$

$$o(n) \leq o(2^{n_0} + c$$

$$n = 2^{n_0} + c$$

$$\log_2 n = n_0 \log_2 2$$

$$\boxed{n_0 = \log_2 n}$$

$$c = n - 2^{n_0}$$

$$c = n - 2^{\log_2 n_0}$$

$$\boxed{c = n - n_0}$$

Ans 11)

```
int j=1, i=0;
While (i<n)
  i = i+j
  j++ ; }}
```

$i = 0, 1, 3, 6, 10, 15 \ldots$

$j = 1, 2, 3, 4, 5, 6 \ldots$

$i = 0, 1, 1+2, 1+2+3, 1+2+3+4, \ldots$

$\quad\quad 0 \quad 1 \quad 2 \quad 3 \quad 4$

PAGE No.
DATE

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

~~K steps~~

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

$2n = K^2 + K$

$n = \frac{K^2}{2} + \frac{K}{2}$

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

Ans 12 $T(n) = T(n-1) + T(n-2) \rightarrow (1)$

$T(1) = 1 \qquad T(0) = 0$

Put $n = n-1$ in 1

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

$\rightarrow (2)$

Put 2 in (1)

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

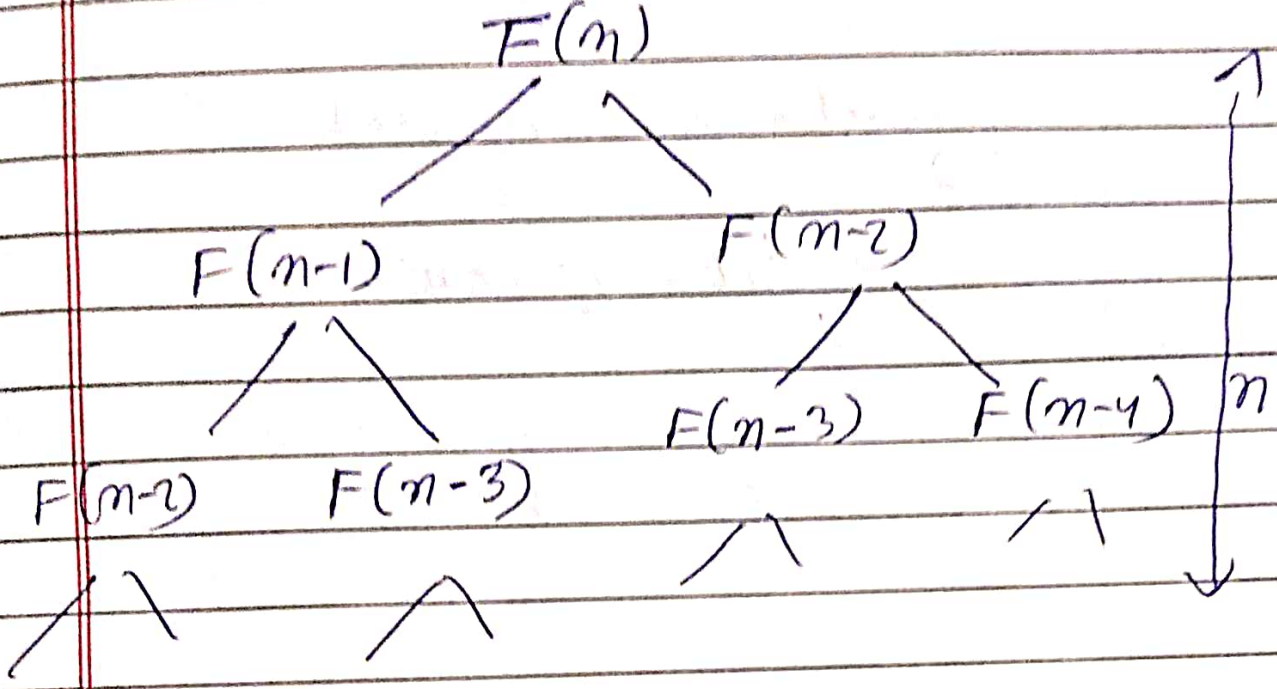
$T(n) = 2T(n-2) + T(n-3) \rightarrow (3)$

Put $n = n-2$ in eq 1

$T(n-2) = T(n-3) + T(n-4) \rightarrow (4)$

Put 4 in (3)

Using tree method



n height each height have 2 divisor

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

Ans 13 $TC = n \log n$

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

PAGE NO.
DATE

TC = n^3

```
int count = 0;

for (int i=0; i<n; i+1)
  {
    for (int j=0; j<n; j++)
      {
        for(int k=0; k<n; k++)
          {  count ++;
          }
      }
  }
cout << count;
```

TC : $\log \log(n)$

```
int count = 0;
for (int i=1 ; i<=n; i=i*2)
  {
    for (int j=1; j<=n; j=j*2)
      {
        count++;
      }
  }
cout << count;
```

Ans
```
int fun (int n)
{
  for (int i=1; i<=n; i++)
  { for (int j = 1; j<n; j=j*1
    {
      // O(1) task
    }
  }
}
```

i = 1, 2, 3, ---- n

j = 1, 2, 3 ----

$O(n^2)$

Ans
```
for (int i=2; i<=n; i= pow(i, k) )
{
  ~
}
```

$i = (2^k)^0, 2^k, (2^k)^k, ((2^k)^k)^k \cdots n$

G.P.

$n = 2 \cdot (2^k)^{l-1}$

$\frac{n}{2} = (2^k)^{l-1}$

$\log n = l-1 \log_2 2^k$

$\log n = k(l-1)$ T.C = $O(k \log n)$

Ans) a) $100, \log\log n, \log n, \log n!, n\log n, \sqrt{n}, n, n^2, n!, 2^n, 2^{2n}, 4^n$

b) $1, \log\log n, \sqrt{\log n}, \log n, \log 2n, 2\log n, \log n!, n, 2n, 4n, n^2, n!, 2\cdot 2^n$

Linear search

Ans)

```
int key= input.
for (i=1   to n)
    if arr[i] = key
        Print("found");
        exit;
    else
        i = i+1;
```

insertion sort

Ans

```
for ( i=1 to array.length-1)
    current = array[i]
    pos = i
    while pos > 0 and Array[Pos-1] > current
        Array[Pos] = Array[Pos-1]
        Pos = Position-1
    END while
    Array[Position] = current
    i = i+1
```

Recursive insertion sort

~~if (array - long~~

```
if (n <= 1)
    Return
Call insertion_sort (arr, n-1)
current = arr[n-1]
pos = i-2
while (pos >= 0 .AND arr[pos] > current
    arr[pos+1] = arr[pos]
    pos = pos-1
}
arr[pos+1] = current
}
```

Insertion sort is online as we can change the size of array after inserting new values then also the algo will work. Whereas bubble sort and insertion sort fails here.

Ans Bubble sort :

Best = $\Omega(n)$ Average = $\theta(n^2)$

Best $O(n^2)$

Selection sort	$O(n^2)$
insertion sort	$O(n^2)$
Quick sort	$\Omega(n \log n)$

Algo	Stable	Place	Online
Bubble	✓	inplace	x
Insertion	✓	inplace	✓
Selection	x	inplace	x
Quick	x	outplace	✓
Merge	✓	outplace	x

Ans Binary Search

iterative

A ⊆ Sorted array
n ⊆ size of array
x ⊆ value to be searched

```
lowerbound = 0
Upperbound = n - 1
while lowerbound <= Upperbound
 mid = (lowerbound + Upperbound) / 2

 if A[mid] = x
        Print (found)
        exit
 if A[mid] < x
        lowerbound = mid + 1
 else
        Upperbound = mid - 1
 End while
```

PAGE No.
DATE

Recursive

A ← sorted array
n ← size
L ← lower bound
U ← upper bound
x ← element to be searched

Function binary_Search (A, L, U, x)

· while @ L <= U
mid = (L+U) / 2
If x == A [mid]
Return mid
else If x > A[mid]
·Return binary_Search(A, mid+1, U, x)
else
Return binary_Search(A, L, mid-1, x);
End while
End of function

Ans) Binary search reccurence relation

$$T(n) = T\left(\frac{n}{2}\right) + 1$$