Date
Page

Tutorial sheet - 01

Q- what is the time complexity of foll. piece of code.

1.
```
int a=0, b=0;
for (i=0; i<N; i++)
    {
        a = a + rand();
    }
for (j=0; j<M; j++)
    {
        b = b + rand();
    }
```

① O(N * M) time, O(1) space
② O(N+M) time, O(N+M) space
③ O(N+M) time, O(1) space
④ O(N * M) time, O(N+M) space

2.
```
int sum = 0, i;
{
    for (i=0; i<n; i=i+2)
    {
        sum += i;
    }
}
```

0, 2, 4, 6, 8 . . . n
A.P.
$a = 0 \quad d = 2$
k^{th} term $= a + (k-1)d$
$n = 0 + (k-1)2$
$\left(\frac{n}{2} - 1\right) = k$
$k = n$
$T(n) = O(n)$

$T(n) = O(n)$, space $= O(1)$

3.
```
int sum=0, i;
for (i=ϕ; i<n; i=i*2)
  {
     sum+=i;
  }
```
// even if it is $i*10$ or $i*100$ ans. is same asymptotically.

1, 2, 4, 8 - - - - n = G.P.

$a=1 \quad r = t_2/t_1 = 2/1 = 2$

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

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

$n = \frac{2^k}{2^1}$

$2n = 2^k$

$k = \log_2(2n)$

$k = \log_2 2 + \log_2 n$

$k = 1 + \log_2 n$

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

space $\rightarrow O(1)$

4. int sum=0, i;

```
for(i=0; i*i<n; i++)
{
    sum+=i;
}
```

$i^2 < n$

$i < \sqrt{n}$

0*0 1*1 2*2 3*3

$= n + (n-1) + (n-4) + (n-9) + \dots + (n-k)$

$= n + (n * k) - (1^2 + 2^2 + 3^2 + \dots k^2)$

$= \sqrt{n}$

$T(n) = O(\sqrt{n})$, space $= O(1)$

5.

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

0, 1, 3, 6, 10, 15, ... n

k^{th} term $= \left(\frac{k*(k+1)}{2}\right)$

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

$n = k^2$

$k = \sqrt{n}$

$T(n) = O(\sqrt{n})$

space $= O(1)$

6. void recursion (int n) $\rightarrow T(n)$

```
{
    if (n==1) return;
    recursion (n-1);      → T(n-1)
    print (n);            → T(1)
    recursion (n-1);      → T(n-1)
}
```

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

$T(n) = 2T(n-1)+1$ — 1

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

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

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

$T(n) = 4T(n-2)+2+1$ — 2

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

$T(n) = 8T(n-3)+4+2+1$ — 3

$T(n) = 2^k(n-k) + (1+2+4+ \cdots k \text{ times})$

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

$n-k=1$

$n-1=k$

$T(n) = 2^{n-1}(n-n-1) + (1+2+4+ \cdots (n-1) \text{ times})$

$T(n) = \frac{2^n}{2} + (1+2+4+8+ \cdots (n-1) \text{ times})$

Date
Page

$a = 1 \quad r = 2$
$n = n-1$

sum of n terms of a G.P. $S_n = \frac{a(r^n - 1)}{r-1} = \frac{1(2^{n-1} - 1)}{2-1}$

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

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

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

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

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

7.
```
int recursion (int what[], int thisone,
               int thatone , int x)
{
  if (thatone >= thisone)
  {
    int something = thisone + (thatone - thisone)/2

    if (what [something]==x )
      return something;
    else if (what [something ] > x )
      return recursion (what, thisone,
                        something-1 , x);
    return recursion (what, Something+1,
                      thatone , x);
  }
  return -1;
}
```

It is a binary search algorithm-

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

~~$T(n) = T(n/2)$~~

At first iteration-

Length of array $= n$

At iteration 2-

Length $= n/2$

At iteration 3-

Length $= (n/2)/2 = n/2^2$

At iteration k-

Length $= n/2^k$

Also, we know that after k divisions, length of array becomes 1.

Therefore, $\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2 n$

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

8. Solve the foll. recurrence relation-

$T(1) = 1$

1. $T(n) = T(n-1) + 1$ — (1)

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

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

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

$T(n) = T(n-2) + 2$ — (2)

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

$T(n) = T(n-3) + 3$ — (3)

⋮

$T(n) = T(n-k) + k$ — (4)

$n-k = 1$

$n-1 = k$

$T(n) = O(n)$

2. $T(n) = T(n-1) + n$ — (1)

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

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

$T(n) = T(n-2) + (n-1) + n$ — (2)

$T(n) = T(n-2) + (n^2 - n)$ — (2)

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

⋮

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

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

$n - k = 1$

$k = n-1$

$T(n) = T(n-n-1) + (n + (n-1) + (n-2) + \dots \dots (n-n-1))$

$T(n) = T(1) + (n + (n-1) + (n-2) + \dots 1)$

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

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

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

3. $T(n) = T(n/2) + 1 \rightarrow$ (1)

$T(n/2) = T(n/4) + 1$

$T(n/4) = T(n/8) + 1$

$T(n) = T(n/4) + 1 + 1$

$T(n) = T(n/4) + 2$ — (2)

$T(n) = T(n/8) + 2 + 1$

$T(n) = T(n/8) + 3$ — (3)

$\vdots$

$T(n) = T(n/2^k) + k$ — (4)

Page

$$n/2^k = 1$$
$$n = 2^k$$
$$k = \log_2 n$$

$$T(n) = T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n$$

$$T(n) = T(1) + \log_2 n$$

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

4. $T(n) = 2T(n/2) + 1$ — 1

$T(n/2) = 2T(n/4) + 1$

$T(n/4) = 2T(n/8) + 1$

$T(n) = 2\,(2T(n/4) + 1) + 1$

$T(n) = 4T(n/4) + 2 + 1$ — 2

$T(n) = 4\,(2T(n/8) + 1) + 2 + 1$

$T(n) = 8T(n/8) + 4 + 2 + 1$ — 3

⋮

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

$$\frac{n}{2^k} = 1$$
$$n = 2^k$$
$$k = \log_2 n$$

$$T(n) = 2^{\log_2 n}\, T\left(n/_{2^{\log_2 n}}\right) + 2^{\log_2 n - 1} + 2^{\log_2 n} \dots$$

$$= \log_2{}^2 n + T(1) + \frac{2^{\log_2 n}}{2^1} + \frac{2^{\log_2 n}}{2^2} + \dots$$

$$= n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \dots + \frac{n}{2^k}$$

$$= n\left[1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \dots + \frac{1}{n}\right]$$

Here, $0 - k \rightarrow (k+1)$ terms

$$r = \frac{1/2}{1} = 1/2$$

So,

$$T(n) = n\left[\frac{1 - (1/2)^{k+1}}{1 - 1/2}\right]$$

$$\Rightarrow n\left[\frac{\left(1 - 1/2 \cdot 2^k\right)}{1/2}\right]$$

$$\Rightarrow n\left[2 \cdot \left(\frac{2 \cdot 2^k - 1}{2 \cdot 2^k}\right)\right] \qquad 2^k = n$$

$$\Rightarrow n\left[\frac{2n - 1}{n}\right] = 2n - 1 = O(n)$$

or

We can also solve it by using Master's method.

i.e. $\rightarrow$

Date

Page

$$\cdots + 2 + 1$$

$$\cdots + \frac{2^{\log_2 n}}{2^k}$$

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

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

Solution $\rightarrow T(n) = n^{\log_b a}\,[U(n)]$

$= n^{\log_2 2}\,[U(n)]$

$= n\,[U(n)]$

$$h(n) = \frac{f(n)}{n^{\log_b a}} = \frac{1}{n^{\log_2 2}} = \frac{1}{n} = n^{-1}$$

$\downarrow$

$n^r,\ r < 0$

So, $O(1)$

So, $U(n) = O(1)$

Now, $T(n) = n\,[U(n)]$

$= n\,[1]$

$T(n) = O(n)$

5. $T(n) = 2T(n-1) + 1 \quad - (1)$

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

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

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

$T(n) = 4T(n-2) + 2 + 1 \quad - (2)$

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

$T(n) \Rightarrow 2^3 T(n-3) + 2^2 + 2 + 1 \quad - (3)$

$\vdots$

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

$\cdots 2^2 + 2 + 1$

$- (4)$

Page

$n-k=1$

$k=n-1$

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

$\Rightarrow 2^{n-1}T(1)+(1+2+2^2+\cdots\cdots 2^{k-1})$

$\Rightarrow 2^{n-1}+2^{k}-1$

$\Rightarrow 2^{n-1}+2^{n-1}-1$

$\Rightarrow \frac{2^n}{2^1}+\frac{2^n}{2^1}-1$

$\Rightarrow 2 \cdot 2^n$

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

6. $T(n)=3T(n-1)$, $T(0)=1$ — 1

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

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

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

$T(n)=9T(n-2)$

$T(n)\ 3^2T(n-2)$ — 2

$T(n)=9(3T(n-3))$

$T(n)=3^3T(n-3)$ — 3

$T(n)=3^kT(n-k)$ — 4

$n-k=0$

$n=k$

Scanned by TapScanner

Date
Page

$T(n) = 3^n\, T(n-n)$

$T(n) \Rightarrow 3^n\, T(0)$

$T(n) \Rightarrow 3^n \cdot 1$

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

7. $T(n) = T(\sqrt{n}) + 1$

$T(n) = T(n^{1/2}) + 1$ — ①

$T(n^{1/2}) = T(n^{1/4}) + 1$

$T(n^{1/4}) = T(n^{1/8}) + 1$

$T(n) = T(n^{1/4}) + 1 + 1$

$T(n) = T(n^{1/4}) + 2$ — ②

$T(n) = T(n^{1/8}) + 2 + 1$

$T(n) = T(n^{1/8}) + 3$ — ③

⋮

$T(n) = T(n^{1/2^k}) + k$ — ④

$n^{\frac{1}{2^k}} = 1$

$\frac{1}{2^k} \log n = 1$

$2^k = \log n$

$k = \log(\log_2 n)$

$\cancel{T(n) = T(n^{\frac{1}{2^{(\log_2 n)}}}) + \log_2 n}$

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

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

8. $T(n) = T(\sqrt{n}) + n$

$T(\sqrt{n}) = T(n^{1/4}) + n^{1/2}$

$T(n^{1/4}) = T(n^{1/8}) + n^{1/4}$

$T(n) = T(n^{1/4}) + n^{1/2} + n$

$T(n) = T(n^{1/8}) + n^{1/4} + n^{1/2} + n$

$\vdots$

$T(n) = T(n^{\frac{1}{2^k}}) + (n + n^{1/2} + n^{1/4} + \cdots$

$n^{1/2^k} = 1$

$\frac{1}{2^k} \log n = 1$

$2^k = \log n$

$k = \log(\log n)$

$T(n) = T(1) + (n + n^{1/2} + n^{1/4} + \cdots k \text{ terms})$

G.P.

$T(n) \Rightarrow 1 + \left(n \frac{((n^{1/2})^k - 1)}{(k-1)} \right)$ $a = n$, $r = n^{1/2}$

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

$T(n) = n \cdot \log(\log(n))$ By neglecting other values

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

Scanned by TapScanner

Page

9] int sum=0, i;
for (i=0; i<n; i++)
{
sum += i;
}

0, 1, 2, 3, - - - n

So, $T(n) = O(n)$, space = $O(1)$

10] int a = 0;
for (i=0; i<N; i++)
{
for (j=N; j>i; j--)
{
a = a + i + j;
}
}

① $O(N)$
② $O(N * \log(N))$
③ $O(N * \text{sqrt}(N))$
④ $O(N*N)$

$T(n) = O(N * (N, N-1, - - - 1))$

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

$T(n) = O(N * N)$

11] int i, j, k = 0;
for (i = n/2; i <= n; i++)
{
for (j=2; j<=n; j=j*2)
{
k = k + n/2;

$n/2$

$\log_2 N$

③ $O(N * sqrt(N))$ ④ $O(N*N)$

Date
Page

$$T(n) = O\left(\frac{n}{2} * \log_2 N\right)$$

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

12] What does it mean when we say that an algorithm X is asymptotically more effici[ent] than Y?

options:

1. X will always be a better choice for sm[all] inputs.
2. X will always be a better choice for larg[e] inputs.
3. Y will always be a better choice for sma[ll] inputs.
4. X will always be a better choice for all inputs.

13]
```
int a=0, i=N;
while (i>0)
{
    a += i;
    i /= 2;
}
```

options -

① $O(N)$
② $O(sqrt(N))$
③ $O(N/2)$
④ $O(\log N)$

14] Solve the foll. recurrence relation -
$T(n) = 7T(n/2) + 3n^2 + 2$

a) $O(n^{2.8})$ b) $O(n^3)$

c) $\Theta(n^{2.8})$ d) $\Theta(n^3)$.

FREEMIND

Date

Page

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

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

$a = 7$

$b = 2$

$c = \log_b a = \log_2 7 \approx 2.807$

$n^c = n^{2.8}$

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

So, $n^c > f(n)$

So, $T(n) = \theta(n^{2.8})$

15) Sort the foll. functions in the descending order of their asymptotic (big-O) complexity:

$f1(n) = n^{\sqrt{n}}$

$f2(n) = 2^n$

$f3(n) = (1.000001)^n$

$f4(n) = n^{10} * 2^{n/2}$

a) $f2 > f4 > f1 > f3$

b) $f2 > f4 > f3 > f1$

c) $f1 > f2 > f3 > f4$

d) $f2 > f1 > f4 > f5$

Scanned by TapScanner

16) $f(n) = 2^{2n}$
which of the foll. correctly represents the above function:
a) $O(2^n)$
b) $\Omega(2^n)$
c) $\Theta(2^n)$
d) None of these

$$f(n) = 2^{2n}$$
$$\log f(n) = 2n \log_2 2$$
$$\log f(n) = 2n$$
or
$$f(n) = 2^n \cdot 2^n$$
$$\Rightarrow \Omega(2^n)$$

17) $T(n) = 2T(n/2) + n^2$. $T(n)$ will be
a) $O(n^2)$
b) $O(n^{3/2})$
c) $O(n \log n)$
d) None of these

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2$$
$$c = 1$$
$$n^c = n$$
$$n^2 > n$$
$$f(n) > n^c$$
$$T(n) = \Theta(n^2)$$

Scanned by TapScanner

18)
```
int gcd (int n, int m)
{
    if (n%m ==0) return m;
    if (n < m) swap(n, m);
    while (m>0)
    {
        n= n%m;
    }
    return (n, m);
}
    return n;
}
```

This is a G.C.D. function where n keeps on decreasing by $n/2$.

$O(\log N)$

19)
```
int a=0, b=0;
for (i=0; i<N; i++)
{
    for (j=0; j<N; j++)
    {
        a= a+j;
    }
}
for (K=0; K<N ;K++)
{
    b=b+K;
}
```

N^2

N

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

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