Design and Analysis of Algorithm

Tutorial - 2

Priyanshu Bhadu
D-30

1)

Code :-

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

value of i

0
1
3
6
10
15
21
!
upto n

$i = 0, \underbrace{1, 3, 6, 10, 15, 21, \ldots\ldots n}_{K \text{ terms}}$

Let the sum of these k terms be S_K

$$S_K = 1+3+6+10+15+21+\ldots\ldots+T_n \quad (3)$$

$$S_{n-1} = 1+3+6+10+15+21-\ldots\ldots T_{K-1} \quad (2)$$

Subtracting (2) from (1)

$$T_K = S_K - S_{K-1} = 1+2+3+4+5+6+\ldots\ldots+K$$

we have $T_K = n$

$$\Rightarrow 1+2+3+4+5+6+\ldots\ldots+K = n$$

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

$$\Rightarrow K^2 + K - 2n = 0$$

$$K = -\frac{1 \pm \sqrt{8n+1}}{2}$$ taking only +ve

value get total no. times the loop for $i = K+1 = \frac{-1+\sqrt{8n+1}}{2} + 1$

$$= 1 - \frac{1}{2} + \frac{\sqrt{Q_{n+4}}}{2}$$

$$= \frac{1 + \sqrt{Q_{n+1}}}{2}$$

Time complexity:

$$T(n) = O\left(\frac{1 + \sqrt{Q_{n+4}}}{2}\right) = O\sqrt{n}.$$

2) pseudo code

```
int fib(int n)
{
    if (n<=1) ---- O(1)
      return n;
    return fib(n-1) + fib(n-2); ---- T(n-1)+T(n-2)
}
```

Time Complexity –

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

when $n=0$ and $n=1$

i.e $T(0) = T(1) = 1$

Here, $T(n-2) \approx T(n-1)$

$\rightarrow T(n) = 2 * T(n-1) + 1 = 2T(n-1) + 1$ ——(i)

put $n = n-1$ in eq^n(i)

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

put in (i)

$T(n) = 2[2T(n-2)+1] + 1 = 4T(n-2) + 2 + 1$ ——(2)

put $n = n-2$ in eq^n(i)

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

put in (2)

$T(n) = 4[2T(n-3)+1] + 2 + 1$ ——(3)

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

generalised form

$$T(n) = \underbrace{2^k T(n-k) + 2^{k-1} + 2^{k-2} + \dots + 1}_{k+1 \text{ terms}}$$

put $n-k=0$

$$T(n) = 2^n T(0) + 2^{n-1} + 2^{n-2} + \cdots + 1$$

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

$$= \underbrace{1 + 2 + 4 + 8 + 16 + \cdots + 2^n}_{k+1 \text{ terms}}$$

$a = 1$, $a_{k+1} = 2^n$, $r = 2$

→ $(k+1)^{th}$ term

$$a_{k+1} = a r^{(k+1-1)}$$

on comparing

$n = k$

$$T.C = O(k+1) = O(n+1) = O(n)$$

Space Complexity :-

Here n is the no. of entries in a stack ~~so space~~ and for each function call one

So space complexity for each case (call) is 1 i.e $O(1)$ and for n no. of case $i = n$

⇒ i.e $O(n)$

3))

```
for (int i = 0; i<n; i++) {
for (int j = 0 , j<m ; j = j+2)
    {
        O(1) // Statement
    }
```

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

```
for (int i = 0; i<n; i++) {
    for (int j = 0; j<n ; j++) {
```

```
for (int k=0; k<n; k++) {
    // O(1) - Statement
}
}
}
= O(n^3)
```

```
for (int i=0; i<n; i=i/2) {
    for (int j=0; j<n; j=j+2)
    {
        O(1) Statement
    }
}
```

$O(\log(\log n))$

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

on removing $T(n/2)$ as smaller term,

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

Applying Master's theorem,

$a = 0, b = 2, K = 2, P = 0$

$\log_b a = \log_2 0 = 0$

$0 < 3$; i.e., $\log_b a < K$, & $P \geq 0$

$\Rightarrow T.C = \Theta(n^K \log^P n)$

$TC = \Theta(n^2 \log^0 n)$

$\Theta(n^2)$

5)) Time complexity of the function fun() is $O(n \log n)$

= for $i = 1$, inner loop executed n time.

for $i = 2$, inner loop executed $n/2$ time

for $i = 3$, inner loop executed $n/3$ time.

for i=n, inner loop executed n/n = 1 time

$\Rightarrow n + \frac{n}{2} + \frac{n}{3} + \cdots\cdots + \frac{n}{n}$

$n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots\cdots + \frac{1}{n}\right)$

⇒ Particular form for time complexity

So. for, total time the loop executed,

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

6)) for (int i=2, i<n ; i = pow (i, k)) {

// O(1) - Expression

}

i take the value like, $2, 2^1, k^2, 2^{k^3} \cdots 2^{k^{\log_k (\log n)}}$

Last term must be less than or equal to n

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

8))

a) $100 < \log n < \log (n) < \log (\log n) < n < n! < n \log n$

$< \log^2 n < 2^n < 4^n < 2(2^n) < n^2$.

b) $1 < \sqrt{\log n} < \log n < \log (n!) < \log (\log n) < \log (2^n) < 2^n$.

$\log (n) < \log (n!) < n \log (n) < n < 2n < 4n < n! < 2^{2^n}$

c) $ab < \log_8 (n) < \log_2 (n) < \log (n!) < n! < n \log_e (n)$

$< n \log_2 (n) < 5n < 8n^2 < 8^{2n} < 7n^3$.