CS & IT ENGINEERING

Algorithms

Analysis of Algorithm

Lecture No.- 08

By- Dr. Khaleel Khan Sir

Recap of Previous Lecture

Topic Problem Solving with ASNs

Topic

Topic

Topic

Topic

Topic

Topics to be Covered

Topics

Framework for Analysing Recursive Algo

Framework for Analyzing Non- Recursive Algo

Topic: Asymptotic Notations

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$f(n) = \log_2 n \quad > \quad g(n) = \log_{10} n$$

(i) $n = 10$

$f(n) = \log_2 10$ $\qquad$ $g(n) = \log_{10} 10$

$= 3.32$ $\qquad >$ $\qquad = 1$

(ii) $\log_2 n > \log_{10} n$

$$\frac{\log_{10} n}{\log_{10} 2} \qquad \frac{\log_{10} n}{\log_{10} n}$$

$$\frac{1}{\log_{10} 2} \qquad 1$$

$\left(\frac{1}{0.3}\right) = 3.33 > 1$

$f(n) = \log_2 n > g(n) = \log_{10} n$

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

$\log_2 n$ $\qquad$ $\dfrac{\log_2 n}{\log_2 10}$

1 $\qquad$ $\dfrac{1}{\log_2 10}$

$1 > \dfrac{1}{3.3}$

① → The Time Complexity of Recursive Algorithm is derived / represented as a Recurrence Equation;

→ Engg. Maths / DAA (Mathematical Model)

② → Solve Recurrence to get the value of a Recurrence as a Mathematical fn (Poly; Log; Expo; Constant)

③ → Apply : $\tilde{O}$, Ω, $\tilde{\Theta}$

Solving recurrences

1) Back Substitution

<Universal Method>

Repeated Substitution

2) Master Method

<Short-Cut>

3) Recursion Tree

Topic : Time Complexity Framework for Recursive Algorithms

Develop (Recursion)

```
1. Algorithm What (n)
   {
      if (n = 1) return;        a
      else
      {
   →     What (n – 1);
         B(n);
      }
   }
```

→ function

(i) Assume $B(n) = O(1)$

Let $T(n)$ repr. Time complexity of What(n)

$$T(n) = c \quad , n = 1 \quad (c > 0)$$

$$T(n) = a + T(n-1) + b \quad , n > 1 \quad [a > 0; b > 0]$$

→ Recurrence

$$T(n) = T(n-1) + d \quad , n > 1 \quad -(1) \quad [d = a+b] > 0$$

PYQ: 2M

(ii) Assume $B(n) = O(n)$

$$T(n) = c \quad , n = 1$$

$$= a + T(n-1) + n \quad -(1)$$

(iii) Assume $B(n) = O(1/n)$

$$T(n) = c \quad , n = 1$$

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

$T(n) = T(n-1) + d$ — (1)

$T(n-1) = T(n-2) + d$ — (2)

$T(n) = T(n-2) + d + d$ — (3)

$= T(n-2) + 2d$ — (3)

$= T(n-3) + 3d$ — (4)

Generalization $= T(n-K) + K \cdot d$ — (5)

$= T(1) + K \cdot d$

$= C + K \cdot d$

$= C + d(n-1)$

$T(n) = C, \quad n = 1$

$n - K = 1$

$K = n - 1$

Soln / value of recurrence

$T(n) = C + dn - d, \quad n \geq 1$

$\rightarrow O(n)$

$\rightarrow \Omega(n)$

$\therefore \Theta(n)$ ✓

(ii) $T(n) = T(n-1) + a + n$ — ①

$T(n-1) = T(n-2) + a + (n-1)$ — ②

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

$= T(n-2) + (n-1) + n + 2a$ — ③

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

$\vdots$

$= T(n-K) + (n-(K-1)) + \cdots + (n-1) + n + K \cdot a$ — ⑤

$= T(1) + (n-(n-1-1)) + \cdots + n + K \cdot a$

$= c + (2+3+4+\cdots+n) + a(n-1)$

$2+3+4+\cdots+n = \frac{n(n+1)}{2} - 1$

Let $c = 1$

$n - K = 1$

$K = (n-1)$

$T(n) = (1+2+3+\cdots+n) + an - a$

$= \sum_{i=1}^{n} i + an - a$

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

$O(n^2)$

$\Omega(n^2)$

$\therefore \Theta(n^2)$

$$T(n) = T(n-1) + \frac{1}{n \cdot 0} + a \quad (1)$$

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

$$T(n) = T(n-2) + \frac{1}{(n-1)} + \frac{1}{n} + 2a \quad (3)$$

$$= T(n-K) + \frac{1}{(n-(K-1))} + \cdots + \frac{1}{n} + K \cdot a \quad (4)$$

$$= T(1) + \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right) + a(n-1) \quad (5)$$

$$= c + \quad " \quad + an - a$$

Let $c = 1$

$n - K = 1$

$\Rightarrow K = n - 1$

$$T(n) = \sum_{x=1}^{n} 1/x + an - a$$

$$T(n) = \log n + an - a$$

$\rightarrow O(n)$

$\rightarrow \Omega(n)$

$\therefore \Theta(n)$

$$T(n) = \begin{cases} T(n-1) + \frac{1}{n}, & n > 1 \\ = c, & n = 1 \end{cases} \Longrightarrow O(\log n)$$

PYQ

Topic : Time Complexity Framework for Recursive Algorithms

2. Algorithm Do_It (n) : Expo $O(2^n)$

```
{
    if (n = 1) return;
    else
    return (Do_It (n – 1) + Do_It (n – 1));
}
```

$T(n-1)$ $T(n-1)$

V_1 V_2

$$T(n) = c, \quad n = 1$$

$$= a + 2 \cdot T(n-1) + b$$

$$T(n) = 2 \cdot T(n-1) + d \quad (d = a+b)$$

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

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

$2^1 \cdot T(n-1) + (2^1 - 1) \cdot d$

$$T(n) = 2\left[2T(n-2) + d\right] + d$$

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

$$= 2^2 \cdot T(n-2) + (2^2 - 1) \cdot d$$

$$= 2^3 \cdot T(n-3) + (2^3 - 1) \cdot d$$

$$= 2^K \cdot T(n-K) + (2^K - 1) \cdot d \quad - (4)$$

$$= 2^K \cdot T(1) + 2^K \cdot d - d$$

$$= 2^{n-1} \cdot c + 2^{n-1} \cdot d - d$$

$n - K = 1$

$K = n - 1$

$T(n) = e \cdot 2^n - d$

$O(2^n)$

$\Theta(2^n)$

$$f(n) = n * f(n-1)$$
$$f(1) = 1$$

$$f(5) = 5 * f(4)$$
$$= 5 * (4 * f(3))$$
$$= 5 * (4 * 3 * f(2))$$

Topic : Time Complexity Framework for Recursive Algorithms

PYQ : 2M

3. Algorithm A(n)
{
if (n = 2) return;
else
return($A(\sqrt{n})$);
}

$n>2$

$T(n) = c \quad , n = 2$

$= a + T(\sqrt{n}) \quad , n > 2$

$T(n) = T(n^{1/2}) + 1.a \quad -(1)$

$T(n^{1/2}) = T(n^{1/4}) + a \quad -(2)$

$T(n) = T(n^{1/4}) + 2a \quad -(3)$

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

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

$= T(n^{1/2^k}) + k \cdot a$

$= T(2) + a \cdot k$

$= c + a \cdot \log\log n$

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

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

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

$2^k = \log n$

$k = \log\log n$

PYQ : 2M

Topic : Time Complexity Framework for Recursive Algorithms

4. Algorithm A(n)

```
{
    if (n = 2) return;
    else
    return (A(√n) + A(√n));
}
```

$T(n) = c, \quad n = 2$

$= a + 2T(\sqrt{n}) + b, \quad n > 2$

$T(n) = 2T(\sqrt{n}) + d, \quad n > 2 \quad (d = a+b) > 0$

$T(n) = 2 \cdot T(n^{1/2}) + d \quad -(1)$

$T(n^{1/2}) = 2 \cdot T(n^{1/4}) + d \quad -(2)$

$T(n) = 2\left[2T(n^{1/4}) + d\right] + d$

$= 4 \cdot T(n^{1/4}) + 3d \quad -(3)$

$= 2^2 \cdot T(n^{1/2^2}) + (2^2 - 1) \cdot d$

$= 2^3 \cdot T(n^{1/2^3}) + (2^3 - 1) \cdot d$

$= 2^K \cdot T(n^{1/2^K}) + (2^K - 1) d$

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

$\Rightarrow 2^K = \log n$

$T(n) = 2^K \cdot T(n^{1/2^K}) + (2^K - 1) \cdot d$

$= \log n \cdot c + \log n \cdot d - d$

$O(\log n)$

$2^{\log_2 \log_2 n} = \log n^{\log_2 2}$

Topic : Time Complexity Framework for Recursive Algorithms

6. $T(n) = 2, n = 2$

$= \sqrt{n}.T(\sqrt{n}) + (n), n > 2$

(*)

$T(n) = n^{1/2}.T(n^{1/2}) + n$ — (1)

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

$T(n) = n^{1/2}\left[n^{1/4}.T(n^{1/4}) + n^{1/2}\right] + n$

$= n^{3/4}.T(n^{1/4}) + 2n$ — (3)

$\left(3/4 = 1 - \frac{1}{2^2}\right)$

$T(n) = n^{1-\frac{1}{2^2}}.T(n^{1/2^2}) + 2n$ — (4)

$= n^{1-\frac{1}{2^3}}.T(n^{1/2^3}) + 3n$ — (5)

$= n^{1-\frac{1}{2^k}}.T(n^{1/2^k}) + k.n$ — (6)

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

$k = \log\log n$

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

$= \frac{n}{2}.2 + n.\log\log n = n + n.\log\log n$

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

Topic : Time Complexity Framework for Recursive Algorithms

```
5. Algorithm Recur (n)
    {
        if (n = 1) return;
        else
        {
            Recur(n/2);
            Recur(n/2);
            B(n);
        }
    }
```

(i) Assume $B(n) = O(1)$

(ii) Assume $B(n) = O(n)$ (*)

$$T(n) = c, \quad n = 1$$
$$= a + 2 \cdot T(n/2) + b, \quad n > 1$$

$$T(n) = 2T(n/2) + d, \quad -(1)$$
$$T(n/2) = 2T(n/4) + d \quad -(2)$$
$$T(n) = 2\left[2T(n/4) + d\right] + d$$
$$= 4 \cdot T(n/4) + 3d \quad -(3)$$
$$= 2^2 \cdot T\left(n/2^2\right) + \left(2^2 - 1\right) \cdot d$$
$$= 2^3 \cdot T\left(n/2^3\right) + \left(2^3 - 1\right) \cdot d$$

$$T(n) = 2^K \cdot T\left(n/2^K\right) + \left(2^K - 1\right) \cdot d$$
$$= n \cdot T(1) + (n-1) \cdot d$$
$$= cn + dn - d$$

$$\frac{n}{2^K} = 1$$
$$\Rightarrow n = 2^K$$

$\rightarrow O(n)$ ✓

$$T(n) = 2 \cdot T(n/2) + n + a \quad -(1)$$

$$T(n/2) = 2\,T(n/4) + n/2 + a \quad -(2)$$

$$T(n) = 2\left[2T(n/4) + n/2 + a\right] + n + a$$

$$= 4 \cdot T(n/4) + 2n + 3a \quad -(3)$$

$$= 2^2 \cdot T(n/2^2) + 2n + (2^2 - 1) \cdot a \quad -(4)$$

$$= 2^3 \cdot T(n/2^3) + 3n + (2^3 - 1)a \quad -(5)$$

$$\vdots$$

$$= 2^K \cdot T(n/2^K) + K \cdot n + (2^K - 1) \cdot a \quad -(6)$$

$$= n \cdot c + n \cdot \log n + an - a$$

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

$$\Rightarrow n = 2^k$$

$$\Rightarrow k = \log n$$

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

$$\Omega(n \cdot \log n)$$

$$\Theta(n \cdot \log n)$$

```
Algo DO_it(n)
{
  if(n==1) return;
  else
  {
    DO_it(n/2);
    B(n);
  }
}
```

(i) $B(n) \longrightarrow O(1)$

$T(n) = c, \quad n = 1$

$= a + T(n/2) + b, \quad n > 1$

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

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

$T(n) = T(n/4) + 2d \quad -(3)$

$= T(n/2^2) + 2d \quad -(4)$

$\vdots$

$= T(n/2^K) + K \cdot d \quad -(5)$

$= T(1) + d \cdot \log n$

$= c + d \cdot \log n$

$= O(\log n)$ ✓

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

$\Rightarrow k = \log n$

(ii) $B(n) = O(n)$

$$T(n) = c, \quad n = 1$$
$$= a + T(n/2) + n$$

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

$$T(n) = T(n/2) + n \quad -(1)$$
$$T(n/2) = T(n/4) + n/2 \quad -(2)$$
$$T(n) = T(n/4) + \frac{n}{2^1} + \frac{n}{2^0} \quad -(3)$$
$$= T\left(n/2^2\right) + \frac{n}{2^1} + \frac{n}{2^0} \quad -(3)$$
$$= T\left(n/2^3\right) + \frac{n}{2^2} + \frac{n}{2^1} + \frac{n}{2^0} \quad -(4)$$

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

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

$$= c + n\left[\sum_{i=0}^{k-1} 1/2^i\right]$$

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

$$S_n = \frac{a(1-r^n)}{1-r}$$

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

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

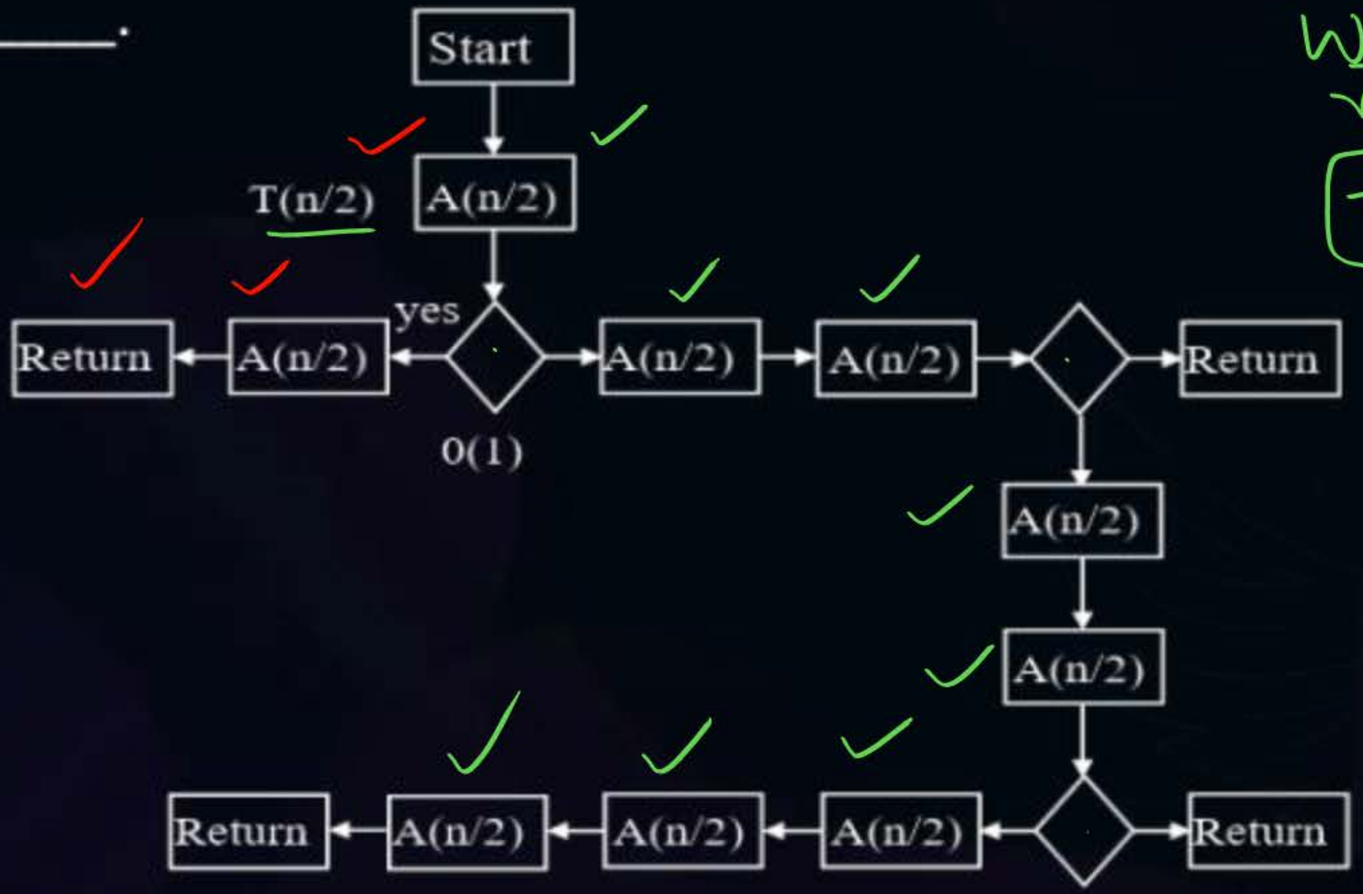
$$= c + 2n - \frac{2n}{2^k}$$

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

$$O(n)$$

7. The given diagram represents the flowchart of recursive algorithm A(n). Assume that all statements except for the recursive calls have order(1) time complexity. Then the best case & worst case time of this algorithm is _____.

PYQ



Worst Case recurrence:

$T(n) = 8 \cdot T(n/2) + c$

Best-Case:

$T(n) = 2 \cdot T(n/2) + c$

$\hookrightarrow O(n)$

$$T(n) = 2 \cdot T(n/2) + n \cdot \log n \quad n > 1$$

$$= c \quad , n = 1$$

H/w

THANK - YOU