CS & IT ENGINEERING

Algorithms

Analysis of Algorithms

Lecture No.- 09

By- Aditya sir

Recap of Previous Lecture

Topic — PYQs

Topic — Imp Practice Questions

Topic — Framework for Time Complexity of Non-Recursive Algo.

Topics to be Covered

Covered

Topic Non-Recursive TC

Topic Determine TC of Recursive Algo (Imp)

Topic

[NAT]

#Q. An element in an Array is called Leader if it is greater than all elements to the right of it. The time complexity of the most efficient algorithm to print all Leaders of the given Array of size 'n' is _____.

H.W

→ Efficient Solution Logic

elem is greater than all elements to its right

⇒ (element is greater than the max elem on its right)

Efficient Algo:-

```
Algo AJ_Leader(A, n)
{
    L = A[n]  ⟶  O(1)
    for (i = (n-1); i >= 1; i--)   O(n)
    {
        if (A[i] > L)              O(1)
        {
            L = A[i]
            print(A[i])
        }
    }
}
```

Code walkthrough:-

A:

29	5	16	14	8	9
✓	×	✓	✓	×	×

L = 9

14 > 9 ⟶ 14 ✓ L = 14

16 > 14 ⟶ 16 ✓ L = 16

5 > 16 ⟶ 5 ✗ L = 16

29 > 16 ⟶ 29 ✓ L = 29

* Time Complexity Analysis:

(1) Best Case: Inc order $\longrightarrow O(n) \longrightarrow \Omega(n)$ LB

(2) Worst Case: Decr order $\longrightarrow O(n) \longrightarrow O(n)$ UB

$\Omega(n), O(n) \Rightarrow \Theta(n)$

Imp: Irrespective of the Input order, the Algo will take same amount of time $\longrightarrow O(n)$

TC $\Rightarrow$ efficient Algo $\longrightarrow \Theta(n)$.

Summary:-

	Best Case	Worst Case
① Brute Force :-	$O(n)$	$O(n^2)$
② Efficient Algo :-	$O(n)$	$\boxed{O(n)}$ ✓

$\rightarrow \underline{\underline{\Theta(n)}}$

67%

(Q) Arrange the following functions in increasing order of their growth rate

$f_1 \to n\log n$, $f_2 \to n^2$, $f_3 \to e^n$, $f_4 \to n^{3/2}$, $f_5 \to n$, $f_6 \to (1/n)$, $f_7 \to 2^n$

A) $f_2, f_1, f_4, f_3, f_5, f_7, f_6$

B) $f_6, f_5, f_1, f_4, f_2, f_7, f_3$

C) $f_6, f_5, f_1, f_4, f_2, f_3, f_7$

D) $f_6, f_5, f_1, f_4, f_7, f_3, f_2$

Soln :-

Polylog $\leftarrow f_1 \rightarrow n\log n$ $\quad$ $f_5 \rightarrow n \rightarrow$ poly

Poly $\leftarrow f_2 \rightarrow n^2$ $\quad$ $f_6 \rightarrow 1/n \rightarrow$ Decr

expo $\leftarrow f_3 \rightarrow e^n$ $\quad$ $f_7 \rightarrow 2^n \rightarrow$ Expo

poly $\leftarrow f_4 \rightarrow n^{3/2}$

$n, n^{3/2}, n^2, \underline{\underline{n\log n}}$

$n < n^{3/2} < n^2$

$\uparrow \quad \uparrow$

$1 \quad 2$

$n\log n > n$ ✓

$\cancel{n}\sqrt{n} \quad \cancel{n}\log n$

$\sqrt{n} > \log n$

$f_6 < \underline{f_5} < \underline{f_1} < \underline{f_4} < \underline{f_2}$

$< \underline{f_7} < \underline{f_3}$

2^n vs e^n

$e \approx 2.71$

$e > 2$

$\underline{\underline{e^n > 2^n}}$

74%

(Q.2) Arrange in decreasing order of growth.

$$f_1 \rightarrow n^{1/3},\quad \overset{f_2}{n(\log n)^{10}},\quad \overset{f_3}{n^{7/4}},\quad \overset{f_4}{(2.002)^n},\quad f_5 \rightarrow e^n$$

A) $f_5 > f_4 > f_1 > f_3 > f_2$

B) $f_4 > f_5 > f_3 > f_2 > f_1$

C) $f_4 > f_5 > f_2 > f_3 > f_1$

D) $f_5 > f_4 > f_3 > f_2 > f_1$

Soln: $f_1 \to n^{1/3} \longrightarrow$ Poly
$f_2 \to n(\log n)^{10} \to$ Polylog
$f_3 \to n^{7/4} \longrightarrow$ Poly
$f_4 \to (2.002)^n \to$ Expo
$f_5 \to e^n \longrightarrow$ Expo

$$f_5 > f_4 > \underline{f_3} > \underline{f_2} > f_1$$

$f_1 < f_3$

$7/4 \to 1 + 3/4$

$n^{1/3}$ vs $n^{7/4}$ ($<$)
$\downarrow$
$n^{0.3}$ vs $n^{1.75}$

$n^{1/3} \to n^{0.3}$ vs $n^1(\log n)^{10}$
$<$

$n(\log n)^{10}$ vs $n^{7/4}$ ($<$)

$\cancel{n}(\log n)^{10}$ vs $\cancel{n} * n^{3/4}$

$(\log n)^{10}$ $\quad$ $n^{3/4}$

Taking log.

$\log((\log n)^{10})$ $\quad$ $\log(n^{3/4})$

$$10 * \log(\log n) < 3/4 * \log(n)$$

Framework To Determine the Time Complexity of a Recursive Algorithm

★ Recursion :-

$$n! = n*(n-1)*(n-2) \cdots 1$$

$$5! = F(5) = 5*\overset{24}{F(4)}$$

$$4*\overset{6}{F(3)}$$

$$3*\overset{2}{F(2)}$$

$$2*F(1) \rightarrow 2*1 = 2$$

$$1$$

$4*6 = 24$

$3\times2 = 6$

$5\times24 = 120$

$\boxed{120}$

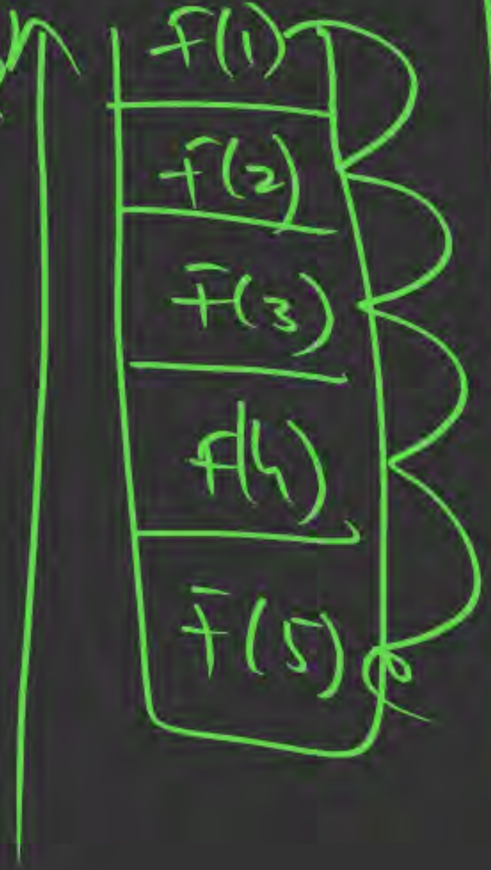
Base Condition

Terminating Condition

BC

$\boxed{F(1)=1}$

Recursion Stack



Recursive Code for $n!$

Time Complexity = ?

```
Algo F(n):
{
    if(n==1)
        return n

    return n*F(n-1)
}
```

Process/
* Framework to Determine TC of Recursive Algo.

3 steps:-

Step1:- Find (Recursive equation for TC) for the Recursive Code.

Recurrence

Step2:- Solve this Recursive equation to get a mathematical expression / value

Step3:- Apply Asymptotic notation on this value.

TC Recurrance equation

① Back Substitution mtd

- → general approach
- → gives value of Recurrence
  - → also gives TC → O, Θ, Ω

② Master's mtd / Theorem

D&C loc

- works in Specific cases
  - → only gives Final TC but not value

③ Recursion Tree Appr

end

- → TC (Final) but no value

Example:-

Algo $\underline{F(n)} \rightarrow T(n)$

```
{ if(n==1)
      return 1

  return n*F(n-1)
}
```

$F(n-1) \rightarrow T(n-1)$

Let $T(n)$ Represent the Time Complexity of $F(n)$.

$$T(n) = T(n-1) + c \quad , \quad n \geq 1$$

$$T(n) = c_1 \quad , \quad n = 1$$

eg 1:-

Algo AJ1(n): $\to T(n)$

{

if (n==1)
return } $\to x$

$\boxed{AJ2(n)}$ $\to$ const

AJ1(n-1) $\to T(n-1)$

}

① **Case 1**: Assume AJ2(n) takes Constant time $\to$ **O(1)**.

② **Case 2**:- Assume AJ2(n) $\to O(n)$

③ **Case 3**:- Assume AJ2(n) $\to O(1/n)$

$\hookrightarrow$ **PYQ**

Algo

Case 1: Soln:

Step 1:- Recurrence

$T(n) = x \quad , \quad n=1, \; x>0$ — B.C

$T(n) = a\,T(n-1), \; n>1, \; a>0$

Step 2:- Use Back Substitution

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

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

Here, $T(n) = \big(T(n-2) + a\big) + a$

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

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

$T(n) = \big(T(n-3) + a\big) + 2a$

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

$T(n) = T(n-4) + 4a$

$= \quad =$

Generalised eqn.

$T(n) = T(n-k) + k*a$ —— (1)

for Base Condition, $T(1)$

$n-k=1$

$\boxed{k=(n-1)}$

eqn 1 becomes

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

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

$T(n) = x + (n-1)*a,$

$x, a \rightarrow$ constants.

$\boxed{T(n) = a*n + (x-a)} \longrightarrow O(n)$

value of Recurrance

Step 3: Apply asymptotic Notation

TC of AJI $\rightarrow T(n) \longrightarrow O(n)$

$(cn \rightarrow n)$

Case 2 :- $AJ2(n) \longrightarrow O(n)$

Step 1 :- Recurrence eqn

$T(n) = b, \; n=1$

$T(n) = a + n + T(n-1), \; n>1$

Step 2 :- Solve Recurrence using Back substitution

$T(n) = T(n-1) + n + a$ — ①

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

$$T(n) = \underbrace{\left[T(n-2) + (n-1) + a\right]}_{T(n-1)} + n + a$$

$= T(n-2) + 2a + [n + (n-1)]$ — ②

$T(n-2) = T(n-3) + (n-2) + a$ — ③

eqn ② now become.

$T(n) = T(n-3) + 3a + [n + (n-1) + (n-2)]$

$= T(n-3) + 3a + [n + (n-1) + (n-2)]$

Generalised eqn

$T(n) = T(n-k) + k*a + [n + (n-1) + \dots (n-(k-1))]$

Use Base Condition,

for B.C,

$(n-k) = 1$

$k = (n-1)$

$T(n) = T(n-k) + k*a + [n + (n-1) \dots + (n-(k-1))]$

$= T(1) + (n-1)*a + [\underbrace{n + (n-1) + \dots (n-(n-1-1))}]$

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

$\left[\sum_{i=1}^{n} i\right] - 1$

Value

$T(n) = T(1) + (n-1)*a + \left[\frac{n(n+1)}{2} - 1\right]$

$T(n) = b + (n-1)*a + \left[\frac{n^2+n-2}{2}\right]$

Step3 →

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

Case 3 $AJ2(n) \rightarrow O(1/n)$

Step 1 :- Recurrance.

$$T(n) = b, \quad n=1$$
$$T(n) = a + \left(1/n\right) + T(n-1), \quad n>1$$

Step 2 :- Solve Recurrance using Back substitution

$$T(n) = T(n-1) + a + \left(1/n\right)$$

$$T(n-1) = T(n-2) + a + \left(1/(n-1)\right)$$

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

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

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

$$T(n) = T(n-3) + 3a + \left[\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2}\right]$$

$n-k$

$$\frac{1}{n-(k-1)}$$

Generalised form:

$$T(n) = T(n-k) + k*a + \left[\frac{1}{n} + \frac{1}{n-1} \cdots + \frac{1}{(n-(k-1))}\right] \quad \text{— (2)}$$

for B.C, $(n-k) = 1$

$k = (n-1)$

eqn (2) becomes,

$$T(n) = T(n-k) + k*a + \left[\frac{1}{n} + \frac{1}{n-1} \cdots \frac{1}{2}\right]$$

$$T(n) = T(n-k) + k*a + \left[\frac{1}{n} + \frac{1}{n-1} \cdots \frac{1}{2}\right]$$

$$* \quad \sum_{i=1}^{n} \frac{1}{i} = \log(n)$$

$$T(n) = \underbrace{T(n-k)}_{b} + k*a + [\log(n) - 1]$$

$$T(n) = b + (n-1)*a + [\log n - 1]$$

Step 3:

$$T(n) \rightarrow O(n)$$

* Homework

1

```
Algo AJ(n)
{
    if (n==2)
        return 2
    else
        return AJ(√n)
}
```

2

```
Algo AJ(n)
{
    if (n==1)
        return 1
    else
        return (AJ(n-1) + AJ(n-1))
}
```

2 mins Summary

Topic Framework for Determining TC of non-Recursive ✓

Topic Framework " " Recursive Algo TC ✓

THANK - YOU

Telegram Link: https://t.me/AdityaSir_PW