02/08/2022 — Tutorial-2

Pikku Maity
IRV21MC074

① Find the time complexity of an algorithm which has two executable ~~Path~~ Part.

(a) Sorting Part that takes $\frac{1}{2}n(n-1)$ comparisons.

(b) To check consecutive element, it requires n comparison.

Sol^n →

(a) With the help of property of asymptotic notation, order of growth for Sorting Path is

$$\frac{1}{2}n(n-1)$$

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

∴ It is approximately ($\approx$) n^2

∴ Quadratic in Nature.

(b) order of growth checking consecutive element is n comparison which is approximately ($\approx$) n.

∴ Linear in Nature

∴ let, $f_1(n) = n^2$

& $f_2(n) = n$

$f(n) = f_1(n) + f_2(n) = \max\{n^2, n\}$

$f_2(n) \approx n^2$

(2) Order the following functions according to their order of growth from the lowest to the highest.

$(n-2)!$, $5 \log (n+100)^{10}$, 2^{2n}, 10.

∴ order of growth $(n-2)! \rightarrow (n-2)!$ (Factorial in nature).

order of growth $5 \log (n+100)^{10} \rightarrow \log (n)^{10}$ (Logarithmic)

order of growth $2^{2n} \rightarrow 2^{2n}$ (Exponential in nature)

order of growth $10 \rightarrow 10$ (constant in nature)

∴ According to order of growth from the lowest to the highest is $10 \leq 5 \log (n+100)^{10} \leq 2^{2n} \leq (n-2)!$

(3) Solve the following recurrance relation

$$X(n) = X(n/3) + 1 \text{ for } n > 1, \; X(1) = 1$$

∴ $$X(n) = \begin{cases} 1 & \text{if } n = 1 \rightarrow \text{Initial Condition.} \\ X(n/3) + 1 & \text{Otherwise.} \end{cases}$$

∴ Consider $X(n) = X(n/3) + 1$ — (1)

Replace n by $n/3$

$$X(n/3) = X\left(\frac{n/3}{3}\right) + 1$$

$$X(n/3) = X(n/9) + 1$$

So, $X(n) = [X(n/9) + 1] + 1$

∴ Replace n by $n/3$

$$X\left(\frac{n/3}{3}\right) = X\left(\frac{n/3}{9}\right) + 1$$

$$X(n/9) = X(n/27) + 1$$

$$X(n) = [X(n/27) + 1] + 2$$

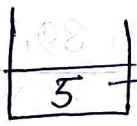
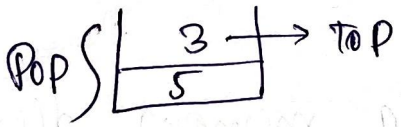
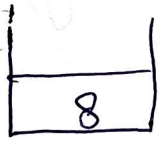
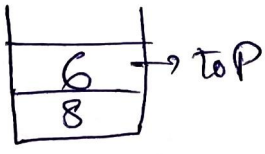
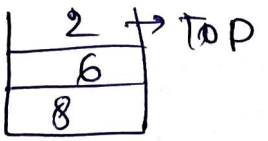
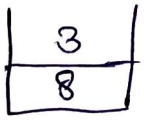
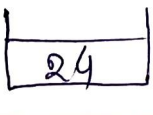
$$x(n) = \left\lceil x\left(\frac{n}{27}\right) + 1 \right\rceil + 2$$

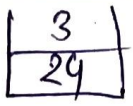
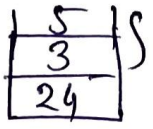
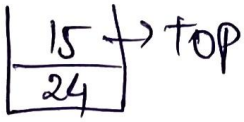
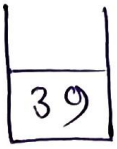
$$= x\left(\frac{n}{3^i} + 3\right)$$

$$= x\left(\frac{n}{3^i} + i\right).$$

(4) Evaluate the given Postfix Expression.

53 + 62 / * 35 * +

Scanned character	Operand Stack	Operation
5		
3		
+		OP2 = 3, OP1 = 5 5+3 = 8
6		
2		OP2 = 2, OP1 = 6 6/2 = 3
/		
*		OP2 = 3, OP1 = 8

3		
5		
*		OP2 = 5, OP1 = 3 3 * 5 = 15
+		OP2 = 15, OP1 = 24 24 + 15 = 39

- No character left out in Postfix exp^n. So, the top of the stack is the result. So, 39.

:) Implement the Dynamic memory allocation using Stack.