1

CS6363 DESIGN AND ANALYSIS OF ALGORITHMS

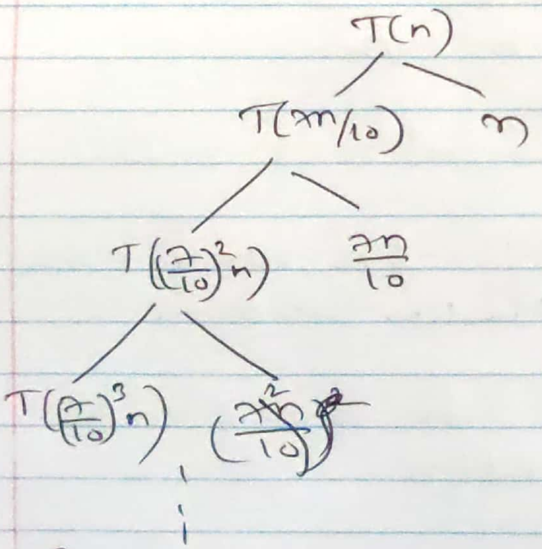
HOMEWORK-2

Name: HIMA SRI T
UTD ID: 2021624644

(1) Give asymptotic upper and lower bounds for the following recurrences

4.1b) $T(n) = T(7n/10) + n$.

Sol: Solving it using the recursion tree Method;



By continuing the tree, we arrive at the following step:

$$T(n) = n + \frac{7n}{10} + \left(\frac{7}{10}\right)^2 n + \left(\frac{7}{10}\right)^3 n + \cdots + \underbrace{\theta(1)}$$

since $T(n)$ is Constant for $n \leq 2$.

$$\Rightarrow T(n) = n\left(1 + \frac{7}{10} + \left(\frac{7}{10}\right)^2 + \cdots\right)$$

②

Solving this geometric progression, gives

$$T(n) = n\left(\frac{1}{1-\frac{7}{10}}\right) + \Theta(1)$$

$$= n\left(\frac{10}{3}\right) + \Theta(1)$$

$$\Rightarrow T(n) = \Theta(n).$$

Also, ~~we~~ solving it using Master's Theorem

$$T(n) = aT(n/b) + f(n)$$

Here, given

$$T(n) = T(7n/10) + n$$

Considering $a=1$, $b=10/7$.

$$\Rightarrow n^{\log_b a} = n^{\log_{10/7} 1} = n^0 = 1 = \Theta(1).$$

$$\Rightarrow f(n) = n = \Omega(n^{0+1}) \text{ for some } \epsilon = 1$$

Hence, from case 3 of Master's Theorem

If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$ &

$$af(n/b) \leq cf(n)$$

Then $T(n) = \Theta(f(n))$.

$$1 * f\left(\frac{7n}{10}\right) \leq c \cdot f(n). \quad \text{for } c < 1 \left[\text{Here } c = \frac{7}{10}\right]$$

$\therefore$ We can deduce that $\boxed{T(n) = \Theta(n)}$

(4.1c) $T(n) = 16T(n/4) + n^2$

Sol: We can solve this using Master's Theorem

$$T(n) = aT(n/b) + f(n)$$

Scanned with CamScanner

(3)

Here $a = 16$, $b = 4$, $f(n) = n^2$.

$\Rightarrow n^{\log_b a} = n^{\log_4 16} \Rightarrow n^{\log_4 4^2} = n^{2(1)}$
$= \underline{\underline{n^2}}$

$\Rightarrow f(n) = \Theta(n^{\log_b a}) = \underline{\Theta(n^2)}$

Case(2) of Master's Theorem states that

If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$

Therefore;
$T(n) = \Theta(n^{\log_b a} \lg n) = \boxed{\Theta(n^2 \lg n)}$

4.1d) $T(n) = 7T(n/3) + n^2$

Sol: solving this using Master's Theorem,
$T(n) = aT(n/b) + f(n)$

Here, $a = 7$, $b = 3$, $f(n) = n^2$

$\Rightarrow n^{\log_b a} = n^{\log_3 7} = n^{1.77}$

$\Rightarrow f(n) = n^2 = \Omega(n^{1.77+\epsilon})$ for some $\epsilon > 0$ $(\epsilon = 0.33)$
$7 \cdot f(n/3) = 7 \cdot \frac{n^2}{9} \leq c \cdot n^2$ for $c < 1$ $[\because c = \frac{7}{9}]$.

Case (3) of MT states that
If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$ &
$af(n/b) \leq c \cdot f(n)$
Then $T(n) = \Theta(f(n))$

4

Hence, here $T(n) = \Theta(f(n))$
$= \Theta(n^2)$

$\therefore \boxed{T(n) = \Theta(n^2)}$

4.1 e) $T(n) = 7T(n/2) + n^2$

Sol: Solving it using Master's Theorem, $T(n) = aT(n/b) + f(n)$
$a = 7,\ b = 2,\ f(n) = n^2$

$n^{\log_b a} = n^{\log_2 7} = n^{2.807}$

$f(n) = n^2 = O(n^{\log_b a - \epsilon}) = O(n^{2.807 - \epsilon})$ for $\epsilon > 0$.
[Here $\epsilon = 0.807$]

Case 1 of MT states that
If $f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$, then
$T(n) = \Theta(n^{\log_b a})$

Therefore, $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 7})$.

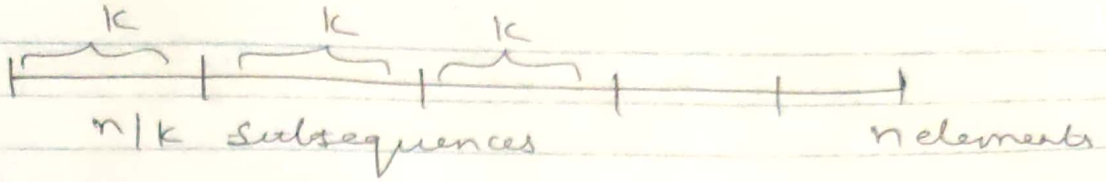
$\therefore \boxed{T(n) = \Theta(n^{\log_2 7})}$

2 Ex 8.1-4 on Page 194 from CLRS.

Sol: Given that
A sequence of n elements to sort.
n/k subsequences consisting k elements

elements in a given subsequence are smaller than elements in succeeding subsequence and greater than elements in preceding subsequence

(3)



Hence, sorting each subsequence gives the entire sorted array.

From the decision tree method.
Worst case no. of comparisions
= length of longest path from root to reachable leaf
= height (depth) of decision tree.

For each subsequence, there are K elements.
$\therefore$ There are $k!$ permutations that are associated to leaves. — (2)

For n/k subsequences,
Total no. of permutations $= (k!)^{n/k}$ — (1)

In a tree with height h, there are atmost 2^h leaves — (1)

From (1) & (2);

$$(k!)^{n/k} \leq 2^h$$

$$2^h \geq (k!)^{n/k}$$

Applying log on both sides;

$$\log 2^h \geq \log (k!)^{n/k}$$

$$h \geq \frac{n}{k} \log(k!)$$

(6)

$$h \geq \frac{n}{k} \log k!$$

$$\geq \frac{n}{k} \log(k*(k-1)*(k-2)* \cdots 1)$$

$$\geq \frac{n}{k}[\log k + \log(k-1) + \cdots + \log 1]$$

$$[\because \log(A*B) = \log A + \log B]$$

$$\geq \frac{n}{k}\left[\sum_{i=1}^{k} \log i\right]$$

$$\geq \frac{n}{k}\left[\sum_{i=k/2}^{k} \log i\right]$$

$$\geq \frac{n}{\cancel{k}}\left[\frac{\cancel{k}}{2} \log k/2\right]$$

$$\geq \frac{n}{2} \log k/2$$

We can show that $\boxed{1/2 \log k/2 \geq c \log k}$ (1) for some $c > 0$ & $n \geq n_0$

$$\frac{\frac{1}{2}\log k/2}{\log k} \geq c$$

$$\frac{1}{2}\frac{[\log k - \overset{1}{\log 2}]}{\log k} \geq c$$

$$\Rightarrow c \leq \frac{1}{2}\left[1 - \frac{1}{\log k}\right] \quad \left[\text{if } k \to \infty, \frac{1}{\log k} \to 0\right]$$

$$\Rightarrow c \leq 1/2.$$

Lets consider $c = 1/4$; then

$$\frac{1/2 \log k/2}{\log k} \geq 1/4.$$

$$\Rightarrow 2 \log k/2 \geq \log k.$$

$$2 \log k - 2 \geq \log k$$

$$\log k \geq 2 \quad \Rightarrow k \geq 4.$$

$\Rightarrow h \geq n * c \log k$ for $c \leq 1/2$ & $k \geq 4$. from (1).

Scanned with CamScanner

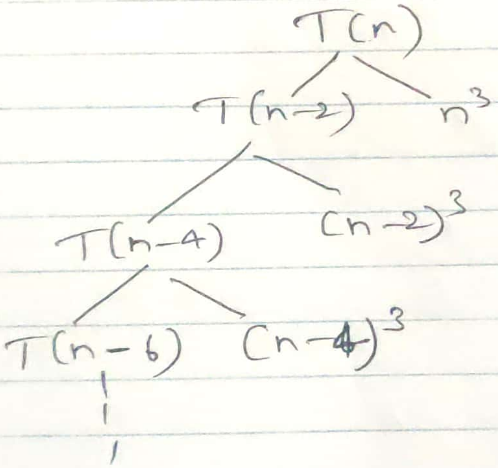
②

$\Rightarrow \quad h \geq cn \log k$

$\Rightarrow$ Worst No. of comparisions = $\boxed{\Omega(n \log k)}$

③ Solve the recurrence relation $T(n) = T(n-2) + n^3$ by using substitution method

Sol. To make a guess, let us check ~~the~~ using recurrence tree method.



$\Rightarrow T(n) = n^3 + (n-2)^3 + (n-4)^3 + \cdots + 2^3 + \underbrace{O(1)}_{\text{for 0 input}}$

$= 2^3 + 4^3 + \cdots + (n-4)^3 + (n-2)^3 + n^3$

↳ Sum of cubes of even numbers

$= 2(n(n+1))^2$

$= 2(n^2+n)^2 = 2(n^4+2n^3+n^2)$

$\Rightarrow T(n) = \cancel{\Theta(n^4)}\ O(n^4)$

8

Hence, our guess is $T(n) = O(n^4)$

We need to prove that

$$T(n) \le c \cdot n^4 \text{ for all } n \geq n_0, \exists\, c > 0, n_0 > 0$$

We can prove this by Induction

Basis step: Assume that $T(n) = 8$ for $n = 2$ when $T(0) = 0$.

$8 \le c \cdot (16)$

Hence base is true. for $c > 1/2$

Inductive hypothesis:

Assume that

$T(n) \le cn^4$ for all $k < n$

i.e $T(k) \le ck^4$

Lets us find out

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

$$\le c(n-2)^4 + n^3 \quad \text{[From Inductive hypothesis]}$$

$$= c(n^4 - 8n^3 + 24n^2 - 32n + 16) + n^3$$

$$= cn^4 - n^3 \underbrace{(8c-1)}_{\geq 0 \text{ for } c \geq 1/8} - c\,(\underbrace{32n - 24n^2 - 16}_{> 0 \text{ always since no real roots}})$$

$$= cn^4 - (\text{some quantity} > 0)$$

$$T(n) \le cn^4 \quad [\text{for } c \geq 1/2,\ n > 0].$$

$\therefore$ $T(n) = O(n^4)$

(4) Design a divide-and-conquer algorithm to compute the factorial of a positive integer n. Set up and solve the recurrence relation for the number of multiplications made by your algorithm.

Sol. Factorial of a positive integer

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

Divide and conquer algorithm for factorial:

Divide: Divide the calculating of n! as following

$n! = n * (n-1)!$ i.e $n! \begin{cases} n \\ (n-1)! \end{cases}$

Conquer: Recursively calculate $(n-1)!$

Combine: After obtaining $(n-1)!$ result multiply it to n.

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

(10)

Algo:

Fact(n):

i) If n=0 or n=1;
result = 1;

(ii) Else
result = n * Fact(n-1);

Recurrence relation for no. of multiplications:-

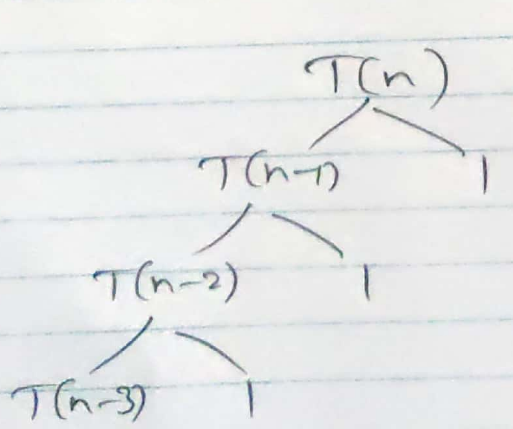
Let T(n): No. of multiplications for input n.

From the algo:

$$T(n) = \underbrace{T(n-1)}_{\text{for recursively calculating } (n-1)!} + 1 \rightarrow \text{multiplications in combine step.}$$

[Assuming comparing n to 0 or 1 & subtracting n-1 takes negligible time]

$$\Rightarrow T(n) = T(n-1) + 1$$



$$\Rightarrow T(n) = \underbrace{1 + 1 + 1 + \cdots}_{n-1 \text{ times}} + \underbrace{T(1)}_{1}$$

$$\Rightarrow T(n) = n$$

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

$\Rightarrow$ The no. of multiplications made $= \boxed{O(n)}$

Scanned with CamScanner