Name:- Harsimran Singh Dhillon

B.no:- B00983136

DAA - Homework
Theory Assignment 2.1

1] Use the iteration method or recursion tree method to solve the following recurrence equation.

a] $$T(n) = \begin{cases} T(n-1) + n & \text{if } (n>1) \\ 1 & \text{if } (n=1). \end{cases}$$

Sol^n We know that $T(n) = T(n-1) + n$.
lets solve for $T(n-1)$.

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

Substituting this value in the main eq^n.

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

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

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

Similarly we can find till a positive int k.

$$= T(n-k) + (n-(k-1)) + (n-(k-2)) + \ldots (n-2) + (n-1) + n$$

We know $T(1) = 1$
So lets make $n-k=1$ (Assumption)
then, $k = n-1$

Substituting this value of k in above eq^n.

$$T(n-(n-1)) + (n-(n-1-1)) + (n-(n-1-2)) + \ldots (n-1) + n$$

$T(n) = T(n-n+1) + (n-n+2) + (n-n+3) + \dots (n-1) + n.$

$T(n) = T(1) + 2 + 3 + \dots (n-1) + n.$

We know $T(1) = 1.$

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

$$\boxed{T(n) = \frac{n(n+1)}{2}}$$

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

b] $T(n) = \begin{cases} 0 & \text{if } n = 2 \\ T(\sqrt{n}) + 1 & \text{if } n > 2 \end{cases}$

(Can be assumed $n^{1/2^k} = 2$)

Sol^n We know that

$T(n) = T(\sqrt{n}) + 1$

lets solve for $T(\sqrt{n})$

$\sqrt{n} = n^{1/2}$

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

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

$[n^{1/4} = n^{1/2^2}]$

Substituting this value in the main eq^n.

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

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

$= T(n^{1/8}) + 1$

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

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

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

Similarly we can solve till a positive int k.

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

We have been told that we can assume.

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

lets substitute that value.

$$T(n) = T(2) + k.$$

$(\because T(2) = 0).$

$$T(n) = 0 + k$$
$$T(n) = k$$

lets solve for k.

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

taking log

$$\log n^{1/2^k} = \log 2.$$

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

$[\because \log 2 = 1]$

$$\log n = 2^k$$

again taking log.

$$\log(\log n) = \log 2^k$$
$$\log(\log n) = k \log 2$$

$(\log 2 = 1).$

$$\boxed{k = \log(\log n)}$$

$$\boxed{T(n) = \log(\log n)}$$
$$T(n) = \underline{\Theta(\log(\log n))}$$

Q2] Use Master method to solve

$$T(n) = 4T(n/2) + n^2 \quad \text{and } T(1) = 1.$$

Sol] We know that for master's theorem,

$$T(n) = aT(n/b) + cn^k.$$

Suppose the above function is non decreasing.

So according to theorem we can write

If :- $\log_b a > k$ then $\Theta(n^{\log_b a})$

Else if :- $\log_b a < k$ then $\Theta(n^k)$

Else $\log_b a = k$ then $\Theta(n^k \log n)$

We have a given Eq^n as :-

$$4T\left(\frac{n}{2}\right) + n^2$$

from this we can calculate,

$a = 4$, $b = 2$, $c = 1$, and $k = 2$.

lets check condition.

$$\log_b a = \log_2 4 = \log_2 2^2 = 2\log_2 2$$

$(\because \log_2 2 = 1)$

$$\log_b a = 2 \quad \text{——— (1)}$$

$$k = 2 \quad \text{——— (2)}$$

from eq^n 1 & 2.

$\log_b a = k$.

then according to the condition.

$T(n) = \Theta(n^k \log n)$

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

Q3] Professor Caesar wishes to develop a matrix multiplication algorithm that is asymptotically faster than strassen's Algorithm.

Soln

We ~~know~~ are given professor caesar's algorithm time complexity.

$T(n) = a\,T(n/4) + \Theta(n^2)$

We know strassens algorithm

$T(n) = 7\,T(n/2) + n^2$

from this we can get the values of

$a = 7$, $b = 2$ and $k = 2$ and $c = 1$

here $a > b^k$ then $T(n) = \Theta(n^{\log_2 7})$ —— 1

for professor Caesar's Eq^n.

$T(n) = a\,T(n/4) + \Theta(n^2)$

here $a = a$, $b = 4$, $c = 1$, & $k = 2$.

We can use these values in masters theorem to find the value.

We know there are 3 conditions in masters theorem.
Lets take a look at all 3 cases.

Case 1:-

If :- $a < b^k$ then we will get $T(n) = \Theta(n^k)$.

$a < 4^2$ $(\because b = 4 \text{ and } k = 2)$.

$\boxed{a < 16}$

~~So~~ So for $a < 16$ we will get a time complexity of $\Theta(n^2)$ which is less than strassen's algorithm.

Case 2:-

If $a = b^k$ then we will get $T(n) = \Theta(n^k \log n)$.

~~So~~ If $a = 4^2$ $(\because b = 4 \text{ and } k = 2)$

$\boxed{a = 16}$ $\Theta(n^2 \log n)$

So for $a = 16$ we will get a time complexity of $\Theta(n^2 \log n)$

And $\Theta(n^2 \log n)$ is smaller than strassens algorithm i.e $(n^{2.81})$ as n approaches to infinity.

Case 3 :- ~~$a > b^k$~~

If $a > b^k$ then we will get
$T(n) =$ ~~$\theta(n^k \log n)$~~
$= \theta(n^{\log_b a})$

$a > 4^2$
$a > 16$

$T(n) = \theta(n^{\log_b a})$

as a is increasing from 16 and going till infinity we get a time complexity of $\theta(n^{\log_b a})$.
but we need a finite value of a for which it is efficient than strassen's algo.

$T(n) = \theta(n^{\log_4 a})$

This value ~~so~~ should be less than the T(n) of strassens Eq^n.

$\therefore \theta(n^{\log_4 a}) < \theta(n^{\log_2 7})$ from eq ①

$\therefore n^{\log_4 a} < n^{\log_2 7}$

$n^{\log_{2^2} a} < n^{\log_2 7}$

$n^{\frac{1}{2}\log_2 a} < n^{\log_2 7}$

$n^{\log_2 a^{1/2}} < n^{\log_2 7}$.

$a^{1/2} < 7$

$\boxed{a < 49}$ $\boxed{\therefore a = 48}$

So Professor caesar's algorithm will be faster than strassen's algorithm for $\underline{a = 48}$