17th April'23

Time and Space Complexity

Q1) Analyze the time complexity of the following Java code and suggest a way to improv it:

```
int sum = 0;
for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            sum++;
        }
}
```

Sol

$i=1$	$i=2$	$i=3$	$i=n$
$j=1$	$j=1, j=2$	$j=1, j=2, j=3$	$j=1, j=2, \ldots j=n$

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

As their are nested loops, by iteration we got time complexity = O(n^2).

improvement by using mathematical sum formula $\frac{n(n+1)}{2}$ directly the time complexity O(n^2) will be reduced to O(1).

Q2. Find the value of T(2) for the recurrence relation $T(n) = 3T(n-1) + 12n$, given that $T(0) = 5$.

Sol Given $T(n) = 3T(n-1) + 12n$ — ①

$\boxed{n=1}$ in ①

$T(1) = 3T(0) + 12n$

$T(1) = 3\times5 + 12 = 15 + 12 = 27$

$\cancel{\boxed{T(1) = 17}}$

$\boxed{n=2}$ in ①

$T(2) = 3T(1) + 12\times2$

$= 3\times27 + 24$

$= 81 + 24 \Rightarrow \boxed{T(2) = 105}$

$\cancel{\boxed{T(2) = 75}}$

Q3. Given a recurrence relation, solve it using a substitution method.

Relation: $T(n) = T(n-1) + C$

Sol $T(n) = T(n-1) + C$ — ①

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

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

$T(n) = T(n-2) + 2C$ — ②

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

$T(n) = T(n-3) + 3C$ — ③

$T(n) = T(n-k) + kc$ - (E)

$n-k=1$

$\boxed{n = k+1}$

$\boxed{k = n-1}$

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

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

$O(n-1) \Rightarrow \underline{O(n)}$ is time complexity of given recurrence relation.

Q4. Given a recurrence relation:

$T(n) = 16T(n/4) + n^2 \log n$

find the time complexity of this relation using the master theorem.

So) given $T(n) = 16T\left(\frac{n}{4}\right) + n^2 \log n$

master's theorem

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n)$$

$a >= 1 \rightarrow$ number of sub problems

$b > 1 \rightarrow$ size of sub problem

$k > 0 \rightarrow k >= 0$

$p \rightarrow$ real number

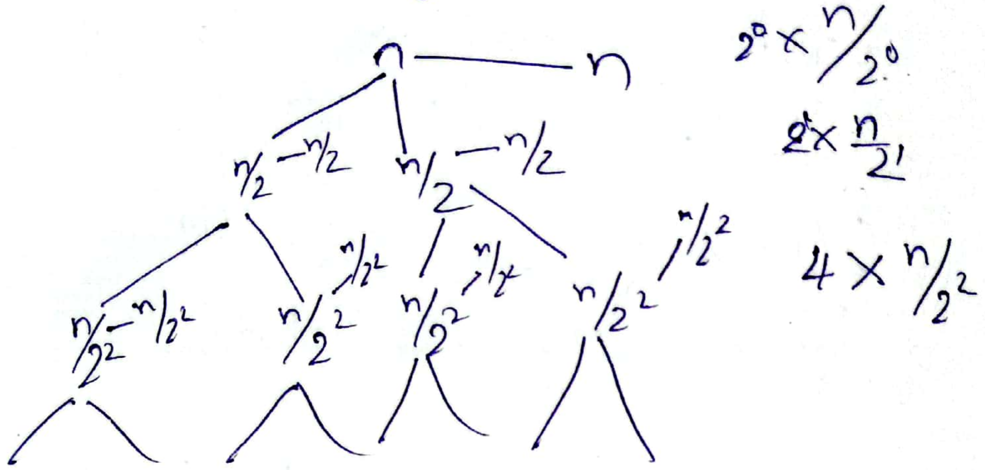
Case 1 $a > b^k$ $\therefore$ Time Complexity $= O(n^{\log_4 16})$

$16 > 4^1$ $= O(n^2)$ //

Q.5. Solve the following recurrence relation using recursion tree method

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

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



Summation of all levels

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

$$T(n) = n[1 + 1 + \cdots + 1] = n$$

$$T(n) = O(n) \rightarrow \text{Time complexity}$$

But height of the tree is $\log(n)$, since the problem size is halved at each level.

$$\therefore T(n) = n + n + n + \cdots + n(\log(n))$$

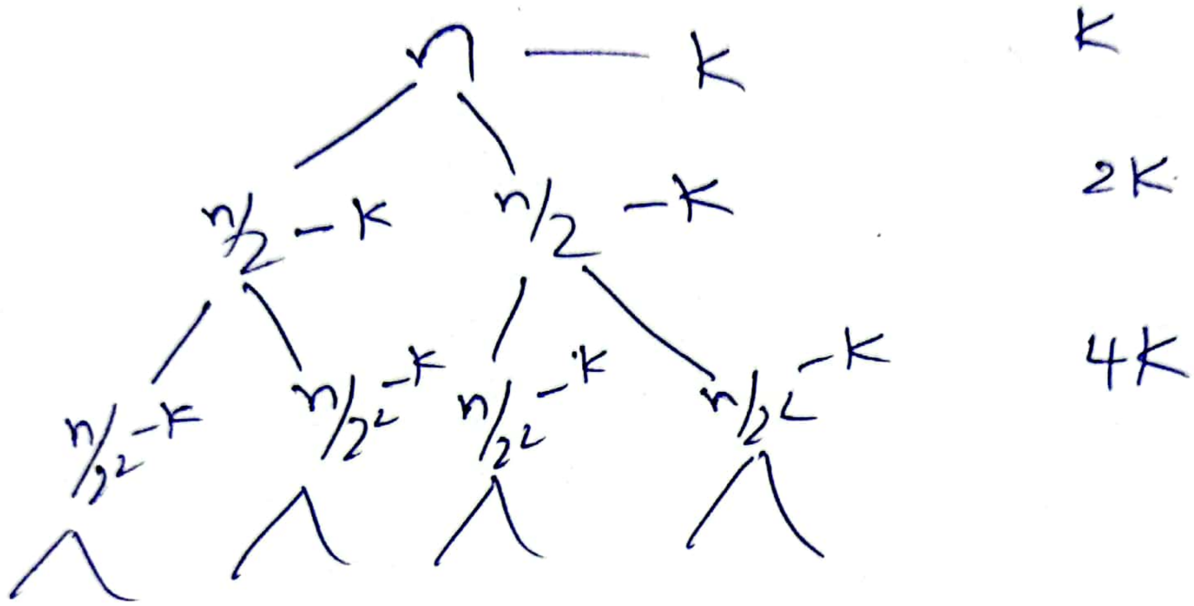
Time complexity $= O(n * \log(n))$ //

Q 6) $T(n) = 2T(n/2) + K$, solve using Recurrence tree method

sol

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

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



The height of the tree is $\log(n)$ because the problem size is halved at each level.

Therefore the total cost of all levels is:

$$T(n) = O(n)$$

$\therefore$ Time complexity $= O(n)$. //