DAA - Test 1

Name: Tanbin Infan

Student No: 20240161

Tutorial Tim: Wed (10-12 PM)

QUESTION ONE

a

i) $T(n) = O(n^2)$

$T(n) \leq cn^2$ where $T(n) = n^2/5 - 2n$

$n^2/5 - 2n \leq cn^2 \quad n \geq 1$

$n^2/5 - 2n < \frac{n^2}{5} \leq cn^2 \quad n \geq 1$

$\frac{n^2}{5} \leq cn^2$ $\therefore T(n) = O(n^2)$ where $c \geq \frac{1}{5}$, $n \geq 1$

(ii) • $T(n) = O(n)$

$n^2/5 - 2n \leq cn$ $\therefore$ FALSE because the highest Polynomial factor is $n^2 > n$

• $T(n) = O(n^2 \log n)$

TRUE because $n^2 \log n \geq n^2$

• $T(n) = O(\sqrt[2]{n})$

FALSE because $\sqrt[2]{n}$ is a polynomially less than n^2

• $T(n) = O(2^n)$

TRUE. Because 2^n is larger than n^2, So it is True.

(b)(i) abc(5)

⇒ abc(5) = 5 × abc(4)

⇒ abc(4) = 4 × abc(3)

↳ abc(5) = 5 × 4 × abc(3)

⇒ abc(3) = 3 × abc(2)

↳ abc(5) = 5 × 4 × 3 × abc(2)

⇒ abc(2) = 2 × abc(1)

↳ abc(5) = 5 × 4 × 3 × 2 × abc(1)

⇒ abc(1) = 1

↳ abc(5) = 5 × 4 × 3 × 2 × 1

= 120

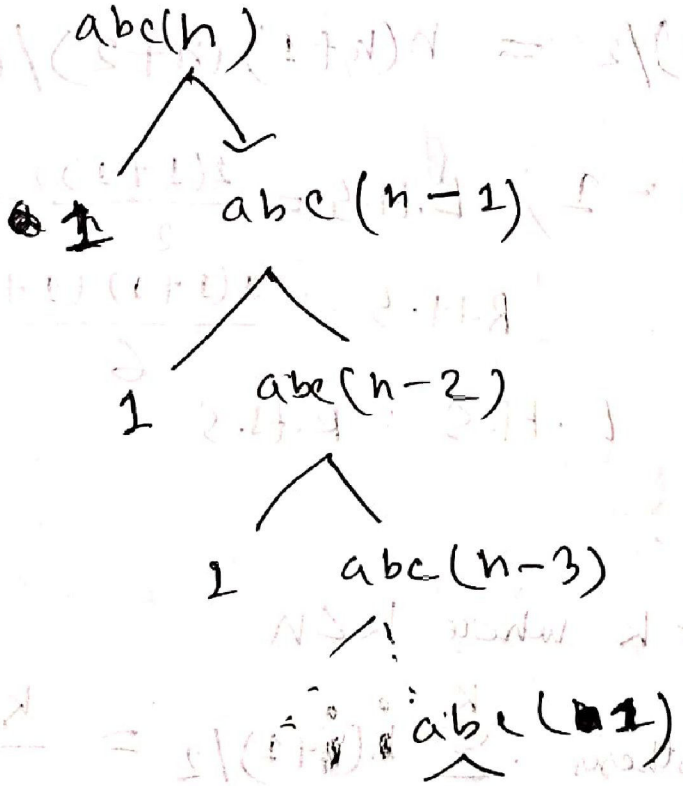
∴ abc(5) = 120 (Ans)

b) (ii)

$$abc(n) = abc(n-1) + O(1)$$

Here in if block, there are one condition checking and return the 1 if it is TRUE which take $O(1)$, Since in every recursive call the size is decreased by 1. That is why $abc(n-1)$ is used.

b) iii)



∴ Here we can see, in every call $O(1)$ amount of work is being done. The total recursive call is the number of input size. So, time complexity is $O(n)$

b) iv) $abc(n) = abc(n-1) + \Theta(1)$

It is possible to prove the above algorithm using master method. To prove using master method, the recurrence solution has to be like below,

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

c) $\sum_{i=1}^{n} i(i+1)/2 = n(n+1)(n+2)/6$

Best case, $i = 1$, $L.H.S = \frac{1(1+1)}{2} = 1$

$$R.H.S = \frac{1(1+1)(1+2)}{6} = \frac{6}{6} = 1$$

$$L.H.S = R.H.S$$

Inductive Step.

If $i = k$ where $k < n$

Induction hypothesis $\sum_{i=1}^{k} i(i+1)/2 = \frac{k(k+1)(k+2)}{6}$

It is also True. $\sum_{i=1}^{k+1} i(i+1)/2 = \frac{(k+1)(k+2)(k+3)}{6}$

So, it is also TRUE

$$\text{L.H.S} = \sum_{i=1}^{k+1} i(i+1)/2 = k+1 + \sum_{i=2}^{k} i(i+1)/2$$

$$= k+1 + \frac{k(k+1)(k+2)}{6}$$

$$= \frac{6(k+1) + (k^2+k)(k+2)}{6}$$

$$= \frac{6k+6+k^3+2k^2+k^2+2k}{6}$$

$$= \frac{k^3+3k^2+8k+6}{6}$$

$$\text{R.H.S} = \frac{(k+1)(k+2)(k+3)}{6}$$

$$= \frac{(k^2+2k+k+2)(k+3)}{6}$$

$$= \frac{k^3+2k^2+k^2+2k+3k^2+6k+3k+6}{6}$$

$$= \frac{k^3+3k^2+8k+6}{6}$$

L.H.S = R.H.S ∴ Proved

d) $T(n) = T_1(n/2) + n^2$

Here, $a = 1, b = 2, f(n) = n^2$

$\log_b a = \log_2 1$

$n^{\log_2 1} = n^{\log_2 2^0} = n^0 = 1 < f(n) \rightarrow$ Case 3

Polynomail condition

$f(n)/1 = n^2$ here $f(n)$ is asymptotically larger by factor of n^ε where $\varepsilon > 2$,

$T(n) = \Theta(n^2)$ [Ans].

(e) $T(n) \leq T(n-2) + \log n$

Inductive

$T(n) \leq cn \log n$

$T(n-1) \leq c(n-1) \log(n-1)$

$T(n) = c(n-1)\log(n-2) + \log n$

$= c(n-1)\log n + \log n \quad n > 1$

$= cn\log n - c\log n + \log n$

$\leq cn\log n + n\log n$

$\leq n\log n(c+1)$

$T(n) \leq k\, n\log n\,(c+1)$

where $k \geq (c+1)$ and $n > 1$

Best case:

$T(1) = T(0) + \log 1 = 0 + 0 \leq cn\log n$ not possible

$T(2) = T(1) + \log 2$

$= T(1) + 1 \leq cn\log n$

$= 0 + 1 \leq cn\log n$ [next page]

Best case

$1 \leq c n \log n$ $n \geq 2$ and $c > 1$

[Proved]

Question - 2

(a) i)

STEP

```
Sum ← 0          → 1
for i = 2 to k   → k
    Sum ← sum + i → k - 1
return (sum)     → 1
```

$T(k) = O(1) + O(k) + O(k-1) + O(1)$

So, the upper bound time complexity is $O(k)$, Because $O(1)$ and $O(k-2)$ are ignoreable compared to $O(k)$

$\geq 1 + k + k - 1 + 1 \quad n \geq 2$

$2k + 1 \leq ck \quad n \geq 1$

$2k + 1 \leq 2k + k \leq ck$

$3k \leq ck$ when $c \geq 3$ $n \geq 1$

(a) (ii)

(b) Here ~~Quicksort~~ Mergesort is preferable. because a large number of inputs Mergesort performs well than Quicksort. For mergesort time complexity is always $n \log n$ where for Quick Sort time complexity depends on the partitioning if partitioning is bad time complexity for quick gets worst.

(c) (i) Because, in Strassen's algorithm the total number of Multiplicati is 7 with 18 addition and subtraction process. So, Total 7 recursive call are needed. That is, why

$$T(n) = 7\,T(n/2) + 18(n/2)^2$$

(c) (ii) The main idea is to reduce the number of multiplicatio and increase addition and subtraction. Because, Multiplication takes more time than addition and subtraction. That is why Strassen's algorithm takes less time than the normal algorithm

(d) (i)

A[n] ← All numbers

Match = 0.

for i = 0 to A.length

{