DAA ASSIGNMENT#02

Saturday, 20 September 2025 6:14 PM

NABIRA KHAN 23K-0914

QUESTION#01

PSEUDOCODE:

```
function Partition(A, lo, hi):
    pivot = A[hi]          // last element as pivot
    i = lo-1
    for j=lo to hi-1:
        if A[j] >= pivot:  // for descending
            i = i+1
            swap A[i], A[j]
    swap A[i+1], A[hi]
    return i+1

function QuickSort(A, lo, hi):
    if lo < hi:
        p = Partition(A, lo, hi)
        QuickSort(A, lo, p-1)
        QuickSort(A, p+1, hi)
```

TIME COMPLEXITY:

Best Case ⟶ $T(n) = 2T(\frac{n}{2}) + O(n)$
$a=2, b=2, d=1$
$T(n) = O(n \log n)$

Avg Case ⟶ $T(n) = O(n \log n)$

Worst Case ⟶ already sorted + last element
$T(n) = T(n-1) + O(n)$
$T(n) = O(n^2)$

QUESTION#02

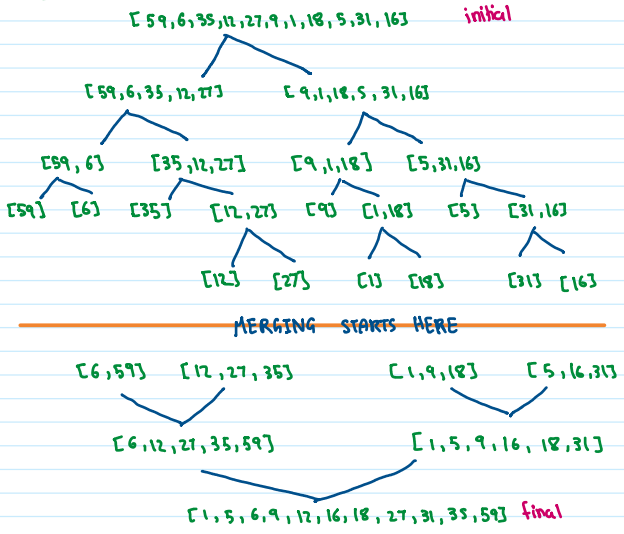
Unsorted List = [59, 6, 35, 12, 27, 9, 1, 18, 5, 31, 16]

① PSEUDOCODE:

```
function Merge(L, R):
    result = empty list
    i=0, j=0
    while i < len(L) and j < len(R)
        if L[i] ≤ R[j]:
            append L[i]; i = i+1
        else:
            append R[j]; j = j+1
    append remaining elements from L or R
    return result

function MergeSort(A):
    if len(A) ≤ 1:
        return A
    mid = floor(len(A)/2)
    L = MergeSort(A[0 .. mid-1])
    R = MergeSort(A[mid .. end])
    return Merge(L, R)
```

② STEP-WISE SORTING



③ MERGE SORT WITH 4 PARTS

recurrence relation ⟶ $T(n) = 4T(\frac{n}{4}) + cn$ cost to merge 4 sorted parts

asymptotic running time ⟶ $a=4, b=4, d=1$ $O(n)$

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

QUESTION NO3

a) PSEUDOCODE

```
function MatMulNaive(A, B, n):
    initialize C as nxn zero matrix
    for i = 0 to n-1:
        for j = 0 to n-1:
            sum = 0
            for k = 0 to n-1:
                sum = sum + A[i][k] * B[k][j]
            C[i][j] = sum
    return C
```

TIME COMPLEXITY
loop for i ⟶ O(n)
loop for j ⟶ O(n)
loop for k ⟶ O(n)
T(n) = O(n³)
for n = 4, number of scalar multiplications = $4^3 = 64$

b) ALGORITM
split 4x4 into 2x2 matrices

$$A = \begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix} \quad B = \begin{bmatrix} B11 & B12 \\ B21 & B22 \end{bmatrix}$$

then strassen defines 7 products (each is a multiplication of 2x2 matrices after linear combos)

M1 = (A11 + A22) * (B11 + B22)
M2 = (A21 + A22) * B11
M3 = A11 * (B12 - B22)
M4 = A22 * (B21 - B11)
M5 = (A11 + A12) * B22
M6 = (A21 - A11) * (B11 + B12)
M7 = (A12 - A22) * (B21 + B22)

then recombine to get 4 blocks of C

C11 = M1 + M4 - M5 + M7
C12 = M3 + M5
C21 = M2 + M4
C22 = M1 - M2 + M3 + M6

each M is computed by recursive matrix multiply on matrices of size n/2 ⟶ cost $O(\frac{n}{2})^2$
for 4x4 specifically, we split into 8 2x2 additions/subtractions and 7 2x2 X 2x2 multiplications
≈ 7*8 = 56 multiplications
$2^3 = 8$

c) T(n) = 7T(n/2) + O(n²)
7 subproblems + O(n²) additions/subtractions & recombinations

d) a = 7 a > b^d
b = 2
d = 2

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

e) Naive ⟶ O(n³)
Strassen ⟶ O($n^{2.81}$)
- strassen asymptotically faster than naive (2.81 < 3)
- however strassen has larger constants and overhead so for smaller matrices, naive is usually faster

QUESTION NO4

- CASE 1: a < b^d ⟶ O(n^d)
non recursive work grows slower than n^d
- CASE 2: a = b^d ⟶ O($n^d \log n$)
both parts contribute equally (non recursive matches n^d)
- CASE 3: a > b^d ⟶ O($n^{\log_b a}$)
non recursive work dominates

QUESTION NO5

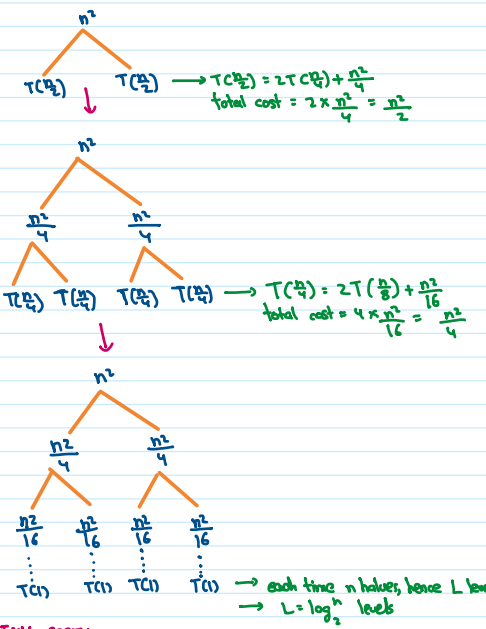
QUESTION # 06

i) $T(n) = 2T(\frac{n}{2}) + n^2$

using recurrence tree

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

subproblems | cost



→ each time n halves, hence L levels → $\frac{n}{2^L} = 1$ as base T(1)

→ $L = \log_2 n$ levels

TOTAL FULL COST:

total cost at each level = $\frac{n^2}{2^i}$

total cost of all levels = $\frac{n^2}{2} + \frac{n^2}{4} + \frac{n^2}{8} + \cdots \frac{n^2}{2^{\log_2 n}}$ geometric series, hence sum converges to $\frac{a}{1-r}$

$= \frac{n^2}{1-\frac{1}{2}} = 2n^2 = \Theta(n^2)$

per leaf node cost as each leaf base case $T(1) = c$ (c a constant)

total leaves → $2^{\log_2 n} = n$

total leaf cost: $cn \rightarrow \Theta(n)$

$T(n) = \Theta(n) + \Theta(n^2) = \Theta(n^2)$

using iterative

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

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

substituting

$T(n) = 2[2T(\frac{n}{4}) + \frac{n^2}{4}] + n^2$

$= 4T(\frac{n}{4}) + \frac{n^2}{2} + n^2$ — ①

$T(\frac{n}{4}) = 2T(\frac{n}{8}) + \frac{n^2}{16}$

substituting into ①

$T(n) = 4[2T(\frac{n}{8}) + \frac{n^2}{16}] + \frac{n^2}{2} + n^2$

$= 8T(\frac{n}{8}) + n^2 + \frac{n^2}{2} + \frac{n^2}{4}$

pattern after k iterations: $T(n) = 2^k T(\frac{n}{2^k}) + n^2 \sum_{i=0}^{k-1} \frac{1}{2^i}$

base case → $\frac{n}{2^k} = 1$

$\log_2 n = k$

substituting

$T(n) = 2^{\log_2 n} T(\frac{n}{2^{\log_2 n}}) + n^2 \sum_{i=0}^{\log_2 n - 1} \frac{1}{2^i}$

$T(n) = nT(1) + n^2 \cdot O(1) = O(n^2)$

ii) $T(n) = 3T(\frac{n}{3}) + n\log^2 n$

using iterative substitution

$T(\frac{n}{3}) = 3T(\frac{n}{9}) + \frac{n}{3}\log^2\frac{n}{3}$

$= 3T(\frac{n}{9}) + \frac{n}{3}\log^2\frac{n}{3}$

substituting

$T(n) = 3[3T(\frac{n}{9}) + \frac{n}{3}\log^2\frac{n}{3}] + n\log^2 n$

$= 9T(\frac{n}{9}) + n\log^2\frac{n}{3} + n\log^2 n$ ①

$T(\frac{n}{9}) = 3T(\frac{n}{27}) + \frac{n}{9}\log^2\frac{n}{9}$

substituting

$T(n) = 9[3T(\frac{n}{27}) + \frac{n}{9}\log^2\frac{n}{9}] + n\log^2\frac{n}{3} + n\log^2 n$

$= 27T(\frac{n}{27}) + n\log^2\frac{n}{9} + n\log^2\frac{n}{3} + n\log^2 n$ ②

after k expansions

$T(n) = 3^k T(\frac{n}{3^k}) + n \sum_{i=0}^{k-1} \log^2(\frac{n}{3^i})$

base case → $\frac{n}{3^k} = 1$

$\log_3 n = k$

substituting

$T(n) = 3^{\log_3 n} T(\frac{n}{3^{\log_3 n}}) + n \sum_{i=0}^{\log_3 n - 1} \log^2(\frac{n}{3^i})$

$T(n) = nT(1) + \Theta(n\log^2 n)$

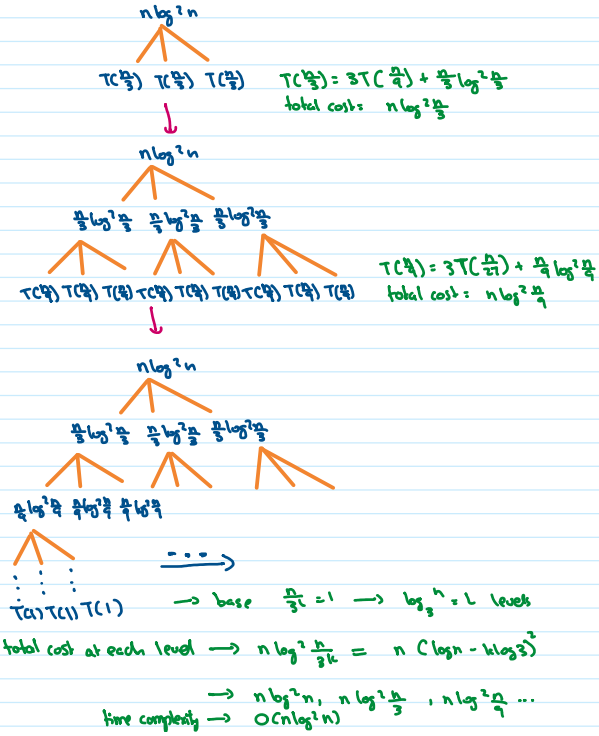
$= \Theta(n) + \Theta(n\log^2 n)$

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

using recurrence tree

$T(n) = 3T(\frac{n}{3}) + n\log^2 n$

subproblems | cost



→ base $\frac{n}{3^i} = 1$ → $\log_3 n = L$ levels

total cost at each level → $n\log^2\frac{n}{3^k} = n(\log n - k\log 3)^2$

→ $n\log^2 n$, $n\log^2\frac{n}{3}$, $n\log^2\frac{n}{9}$ ...

time complexity → $O(n\log^2 n)$

iii) $T(n) = 2T(\frac{n}{2}) + \log n$

using recurrence tree

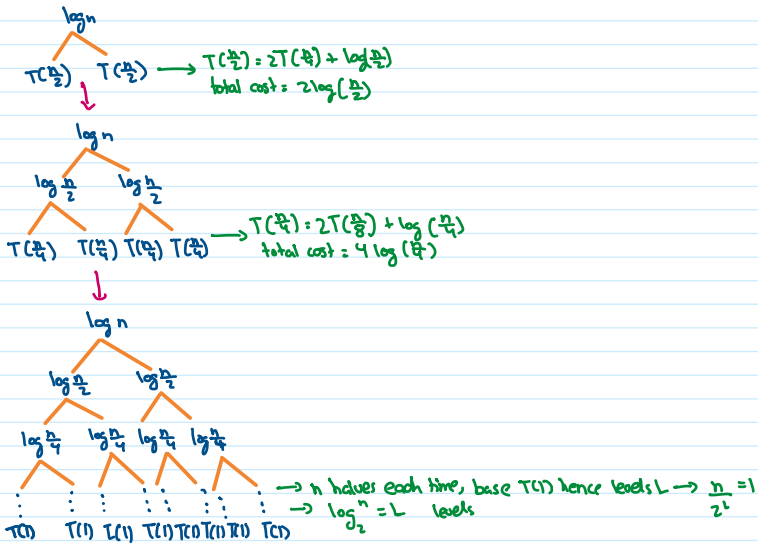
$= \Theta(n \log^2 n)$

iii) $T(n) = 2T(\frac{n}{2}) + \log n$

using recurrence tree

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

↓ subproblems ↓ cost



total cost at each level → $n \log^2 \frac{n}{2^k} = n(\log n - k \log 2)$

→ $n \log^2 n,\ n \log^2 \frac{n}{2},\ n \log^2 \frac{n}{4} \dots$

time complexity → $O(n \log^2 n)$

time complexity for leaves → $O(n)$

$T(n) = O(n \log^2 n) + O(n)$

$T(n) = O(n \log^2 n)$

using iterative substitution

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

$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \log \frac{n}{2}$

substituting

$T(n) = 2[2T(\frac{n}{4}) + \log \frac{n}{2}] + \log n$

$= 4T(\frac{n}{4}) + 2\log \frac{n}{2} + \log n$ — ①

$T(\frac{n}{4}) = 2T(\frac{n}{8}) + \log \frac{n}{4}$

substituting into ①

$T(n) = 4[2T(\frac{n}{8}) + \log \frac{n}{4}] + 2\log \frac{n}{2} + \log n$

$= 8T(\frac{n}{8}) + \log n + 2\log \frac{n}{2} + 4\log \frac{n}{4}$

after k iterations, pattern → $T(n) = 2^k T(\frac{n}{2^k}) + \sum_{i=0}^{k-1} 2^i \log(\frac{n}{2^i})$

base case → $\frac{n}{2^k} = 1$

$\log_2 n = k$

substituting

$T(n) = 2^{\log_2 n} T(\frac{n}{2^{\log_2 n}}) + \sum_{i=0}^{\log_2 n - 1} 2^i \log(\frac{n}{2^i})$

$T(n) = nT(1) + O(n)$

$T(n) = O(n)$

FULL TOTAL COST

cost at each level : $2^i \log(\frac{n}{2^i})$

total cost for all levels : $\log n + 2\log \frac{n}{2} + 4\log \frac{n}{4} + 8\log \frac{n}{8} + \dots + 2^{\log_2 n} \log \frac{n}{2^{\log_2 n}}$ → geometric progression $\frac{a}{1-r}$

$= \sum_{i=0}^{\log_2 n - 1} 2^i \log(\frac{n}{2^i})$ $\log_2 n - 1$ to separate leaf nodes

$= \sum_{i=0}^{\log_2 n - 1} 2^i (\log n - \log 2^i)$ → taking log 2

$= \sum_{i=0}^{\log_2 n - 1} 2^i (\log n - i)$

$\sum_{i=0}^{\log_2 n - 1} 2^i \log n$

$= \log n \sum 2^i$

↓ geometric series sum

$= 2^{\log_2 n} - 1$

$= n - 1 = \Theta(n)$

$= \Theta(n \log n)$

$\sum_{i=0}^{\log_2 n - 1} 2^i \cdot i$

largest term = $\log_2^n - 1$

$2^{\log_2 n}(\log_2 n) = n \log n$

sum grows linearly with n → $\Theta(n \log n)$

total cost of leaves = $cn = \Theta(n)$

$T(n) = O(n \log n) - O(n \log n)$

$T(n) = O(n)$

QUESTION #07

1) $T(n) = 4T(\frac{n}{2}) + n^2$

$a = 4$	$a = b^d$
$b = 2$	$4 = 2^2$
$d = 2$	$4 = 4$

$T(n) = \Theta(n^2 \log n)$

2) $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$

$a = 2$	$a = b^d$
$b = 4$	$2 = 4^{\frac{1}{2}}$
$d = \frac{1}{2}$	$2 = \sqrt{4}$
	$2 = 2$

$T(n) = \Theta(\sqrt{n} \log n)$

3) $T(n) = 4T(\frac{n}{2}) + n^2 \log n$

$a = 4$	$a = b^d$
$b = 2$	$4 = 2^2$
$d = 2$	$4 = 4$

$T(n) = \Theta(n^2 \log n)$

QUESTION #03

1) $T(n) = 4T(\frac{n}{2}) + n^2$

guess 1 → $T(n) = O(n)$ ✗

$T(n) \leq cg(n)$ for $c > 0$

$4T(\frac{n}{2}) + n^2 \leq cn$

$\frac{4cn}{2} + n^2 \leq cn$

$2cn + n^2 \leq cn$ ✗ wrong

guess 2 → $T(n) = O(n^2)$

$T(n) \leq cg(n)$ for $c > 0$

$4T(\frac{n}{2}) + n^2 \leq cn^2$

$4 \cdot \frac{cn^2}{4} + n^2 \leq cn^2$

$cn^2 + n^2 \leq cn^2$

$n^2(c+1) \leq cn^2$ $\quad c+1 > c$

✗ wrong