DAA Assignment 2 Syeda Rija Ali
23K-0057

Q1) QuickSort (A, low, high)
    if low < high
        pi = partition (A, low, high)
        Quicksort (A, low, pi-1)
        Quicksort (A, pi+1, high)

Partition (A, low, high)
    pivot = A [high]
    i = low-1
    j from low to high -1
        if A [j] >= pivot
            i ++
            swap A[i] to A[j]
    swap A[i+1] and A [high]
    return i+1;

i) Time complexity:-

• Best & Avg cases:

→ partition splits array into 2 halves → logn

→ n elements processed at each level

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

• Worst Case:

→ pivot is always the smallest or largest element, one side has n-1 elements while other has 0

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

2) Example:

A = [50, 20, 90, 10, 70]

i) pivot = 70; partition [90, 70, 20, 10, 50]

ii) left = [90] and right = [20, 10, 50]

iii) sort right → [50, 20, 10]

iv) result → [90, 70, 50, 20, 10]

Q2) 1)

↳ Divide : if array has more than one element, split it into two halves.

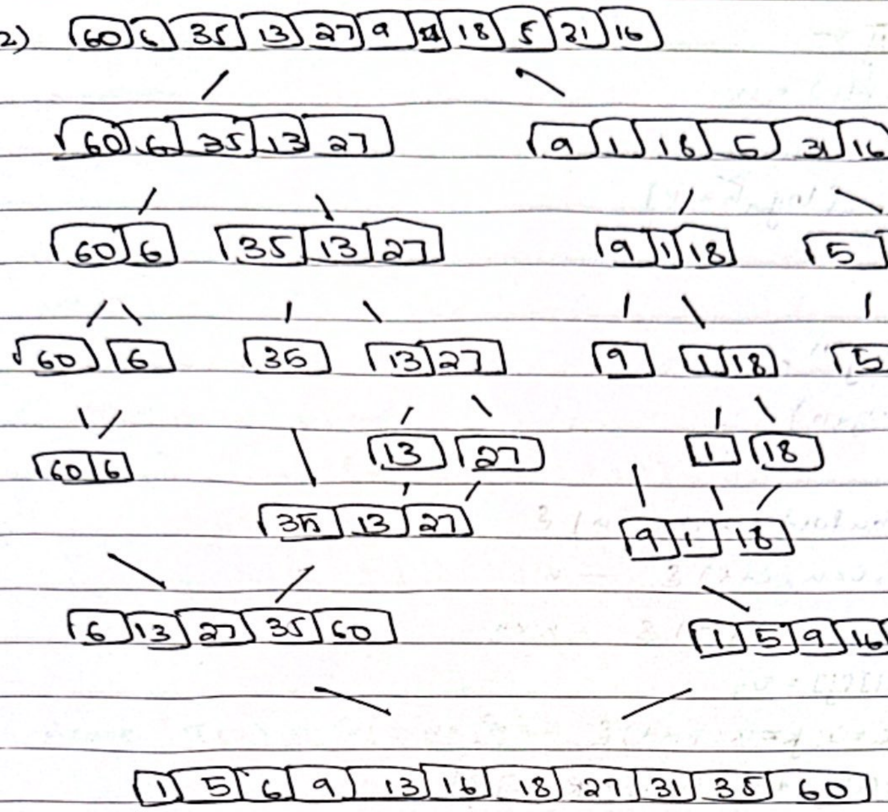
↳ Conquer : recursively sort each half using merge sort

↳ combine : merge the two sorted halves into one sorted array

```
MergeSort (scores, left, right) {
    if (left < right) {
    mid = floor ((left+right) /2)
     Mergesort (scores, left, mid)
     MergeSort (scores, mid+1, right)
      Merge (scores, left, mid, right)
    }}
Merge (scores, left, mid, right) {
   n1 = mid - left + 1 ;
   n2 = right - mid;
   for (i=0; i<n1; i++) {
       left Arr[i] = scores [left + i]; }
   for (j=0; j<n2; j++) {
       right Arr[j] = scores [mid+1 +j]; }
    i= 0 , j=0 , k= left;
     while (i<n1 && j<n2) {
       if ( left Arr [i] <= rightArr [j] ) {
          scores[k++] = leftArr [i++]
        }
     else {
       scores [k++] = right Arr [j++] ; }
      while (i<n1) {
        scores [k++] = left Arr [i++] }
       while (j<n2) {
        scores [k++] = rightArr [j++] }
```

2)

2)



3) Merge Sort (scores, l, r) {

if (l<r) {

// left to p1 — p1 = l + (r − l)/4 ;

// p1+1 to p2 — p2 = l + (r−l)/2 ;

// p2 to p3 — p3 = l + 3 * (r−l)/4 ;

} T(1)

merge Sort (scores, l, p1) ; T(n/4)

MergeSort (scores, p1+1, p2) ; T(n/4)

mergesort (scores, p2+1, p3) ; T(n/4)

merge sort (scores, p3+1, r) ; T(n/4)

Merge (scores, l, p1, p2, p3, right)

↳ Recurrence; since each level divides in 4 subproblem each of size n/4

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

→ Time Complexity :-

$a = 4, b = 4, f(n) = n$

$\log_a b = \log_4 4 = 1$

$n^1 = n^1 = [\log_a b = k]$

→ $p > -1$

→ $O(n^k \log^{p+1} n)$

→ $O(n \log_4 n)$

Q 3) a) MatrixMulStandard (W, R, C, n) {

```
for (i=0; i<n; i++) {              — n
    for (j=0; j<n; j++) {          — n x n
        C[i][j] = 0;
        for(k=0; k<n; k++) {       — n^2 x n
            C[i][j] += W[i][j] * R[k][j]
}}}
```

Time complexity : $n \times n^2 = O(n^3)$

3b) Strassen's Algo:

splitting 4x4 into 2x2 matrix.

$W = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \quad R = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$

$M_1 = (A+D)(E+H)$

$M_2 = (C+D)(E)$

$M_3 = (A)(F-H)$

$M_4 = (D)(G-E)$

$M_5 = (A+B)(H)$

$M_6 = (C-A)(E+F)$

$M_7 = (B-D)(G+H)$

$C_{11} = M_1 + M_4 - M_5 + M_7$

$C_{12} = M_3 + M_5$

$C_{21} = M_2 + M_4$

$C_{22} = M_1 - M_2 + M_3 + M_6$

3c) Recurrence Relation:

↳ 7 recursive multiplication of size n/2 done including $\Theta(n^2)$ of additional matrix additions / subtractions & reassembly

↳ So saying the relation as:

$\rightarrow T(n) = 7T\left(\frac{n}{2}\right) + n^2$

d) Master Theorem:

$a = 7,\ b = 2,\ f(n) = n^2 \rightarrow (k = 2)$

$\log_b a = k$

$\log_2 7 = 2.8$

$2.8 > 2$

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

e) Comparison:

↳ standard matrix multiplication does n row & n column inner product of length n : $[\Theta(n^3)]$

↳ Strassen as derived $[\Theta(n^{\log_2 7})]$, Hence, asymptotically strassen method is faster as it reduces the number of multiplications at each level. However it has higher constant factors and more additions so better for larger matrices.

4) Master Theorem Cases:

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

where $a \geq 1,\ b > 1,\ f(n)$ is +ve and is in form,

$f(n) = n^k \log^p n$

case 1:

when $\log_b a > k$ Then:

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

case 2:

when $\log_b a = k$

1) if $p > -1$ then $T(n) = \Theta(n^k \log^{p+1} n)$

2) if $p = -1$ then $T(n) = \Theta(n^k \log(\log n))$

3) if $p < -1$ then $T(n) = \Theta(n^k)$

case 3: when $\log_b a < k$ then:

1) if $p \geq 0$, then $T(n) = \Theta(n^k \log^p n)$
2) if $p < 0$, then $T(n) = \Theta(n^k)$

Q5) if algorithm splits into 4 equal parts, recursively ~~solve~~ sorts each part & merges them into one again.

Recurrence Relation,

$$T(n) = \begin{cases} \Theta(1) & n \leq 1, \\ 4T(n/4) + \Theta(n) & , n > 1 \end{cases}$$

→ using master's theorem:

$a = 4$, $b = 4$, $k = 1$, $p = 0$

b^k, $4^1 = 4$, $(a = b^k)$

case 2:

$p = 0$, $p > -1$

$$T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$$

$$\Theta(n^1 \log^1 n)$$

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

Q6i) substitution method,

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

$$T(n/2) = 2T(n/4) + (n/2)^2$$

$$T(n) = 2[2T(n/4) + (n/2)^2] + n^2$$

$$T(n) = 4T(n/4) + n^2 + n^2/2$$

$$T(n/4) = 2T(n/8) + (n/4)$$

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

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

general formula, $\frac{n}{2^k} = 1 \rightarrow n = 2^k$

$\log_2 n = k$

$$T(n) = 2^k - T\left(\frac{n}{2^k}\right) + n^2 \sum_{i=0}^{k-1} (1/2)$$

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

$$= n + 2n^2$$

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

Recurrence Trees:

$$T(n)$$

$$T(n/2) \qquad T(n/2)$$

$$T(n/4) \quad T(n/4) \quad T(n/4) \quad T(n/4)$$

$$T(n/8) \quad T(n/8)$$

$$\vdots$$

k depth $\log_2 n$ times

$$\Sigma \text{ cost} = n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \frac{n^2}{8}$$

$$n^2 \left[\frac{1}{1-1/2} \right] \Rightarrow n^2 \times 2$$

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

ii) $T(n) = 3T(n/3) + n\log^2 n$

$$T(n/3) = 3T(n/9)$$

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

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

$$= 3^k T\left(\frac{n}{3^k}\right) + \frac{n}{3^k}\left(\log\frac{n}{3^k}\right)^2 + \frac{n}{3^{k-1}}\left(\log\frac{n}{3^{k-1}}\right)^2 + \dots n(\log n)^2$$

Assuming $\frac{n}{3^k} = 1$

$$n = 3^k$$

$$k = \log_3 n$$

$$3^{\log_3 n} T(1) + n(\log n - k\log 3)^2 \left[1/3^k + 1/3^{k-1} + 1/3 + 1 \right]$$

$$n^{\log_3 3}(1) + n(\log n - k)^2 (1)$$

$$n + n(\log n)^3$$

$$T(n) = \Theta(n(\log n)^3)$$

6 ii) $T(n) = 3T\left(\frac{n}{3}\right) + n\log n$ — ①

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

↳ subs in eq 1.

$\Rightarrow 3T\left[\left(3T\cdot\frac{n}{3^2}\right) + \frac{n}{3}\log\frac{n}{3}\right] + n\log n$

$3^2T\cdot\frac{n}{3^2} + n\left[\log\frac{n}{3}\right] + \log n$ — ②

$T\left(\frac{n}{3^2}\right) \Rightarrow 3T\left(\frac{n}{3\cdot 9} + \frac{n}{9}\log\frac{n}{9}\right)$

↳ subs in eq 2.

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

$\Rightarrow 3^3T\left(\frac{n}{3^3} + \frac{n}{3^2}\log_2\left(\frac{n}{3^2}\right)\right)$

After k expansions:

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

$3^i\cdot\frac{n}{3^i} = n$

$n\log\left(\frac{n}{3^i}\right) = n\left(\log n - \log(3^i)\right)$

$= n(\log n - i\log 3)$

$= n\sum_{i=0}^{k-1}(\log n - i\log 3) = n\left(k\log n - \log 3\sum_{i=0}^{k-1} i\right)$

$= n\left(k\log n - \log 3\cdot\frac{k(k-1)}{2}\right)$

$\frac{n}{3^k} = 1$, $3^k = n$, $k = \log_3 n$

$\log_3 n = \frac{\log_2 n}{\log_2 3}$

$$k \log n = (\log_3 n) \log n \Rightarrow \frac{\log n}{\log 3} \cdot \log n = \frac{(\log n)^2}{\log 3}$$

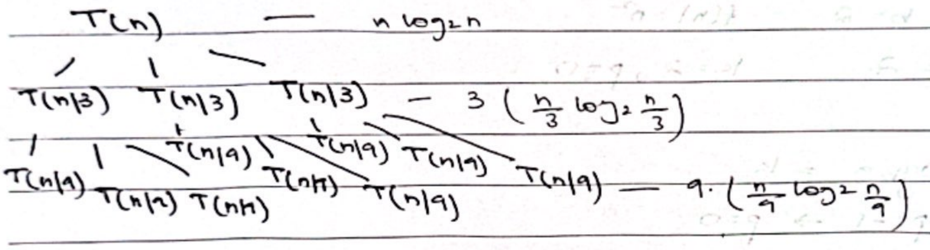
$$\frac{\log 3}{2} k(k-1) \approx \frac{\log 3}{2} \cdot k^2 = \frac{\log 3}{2}\left(\frac{\log n}{\log 3}\right)^2 = \frac{(\log n)^2}{2\log 3}$$

So,

$$k \log n - \frac{\log 3}{2} k(k-1) \approx \frac{(\log n)^2}{\log 3} - \frac{(\log n)^2}{2\log 3} = \frac{(\log n)^2}{2\log 3}$$

Multiplying by n gives: $i(k) = \Theta(n \log n)^2$

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



$$S = n\left(\frac{(\log_2 n)^2}{\log_2 3} - \frac{(\log_2 n)^2}{2\log 3} + \cdots\right) = \Theta(n(\log_2 n)^2)$$

3) Substitution method

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

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

$T(n) = 2[2T(n/4) + \log(n/2)] + \log n$

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

$T(n/4) = 2T(n/8) + \log(n/4)$

$T(n) = 4[2T(n/8) + \log(n/4)] + 2\log(n/2) + \log n$

$= 8T(n/8) + 4\log(n/4) + 2\log(n/2) + \log n$

General formula:-

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

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$ & $k = \log_2 n$

$n(T(1)) + \sum n$

T.C $= \Theta(n)$

- Recurrence Tree :

$$T(n)$$
$$T(n/2) \quad T(n/2)$$
$$T(n/4) \quad T(n/4) \quad T(n/4) \quad T(n/4)$$
$$T(n/8) \quad T(n/8) \quad T(n/8) \quad T(n/8) \quad \vdots \quad \vdots$$

K depth : $\log_2 n$

$\Sigma\, 2^k \log(n/2^k)$

$\Theta(n)$.

Q7) i) $T(n) = 4T(n/2) + n^2$

$a=4$, $b=2$ $\quad f(n) = n^2$

$\log_2 4 = 2$ $\quad k=2, p=0$

$\log_b a = k$

$p > -1 \rightarrow p=0$

Hence, $\Theta(n^k \log^{p+1} n)$

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

ii) $T(n) = 2T(n/4) + n^{1/2}$

$a=2$, $b=4$ $\quad k = 1/2$ & $p=0$

$\log_4 2 = 1/2$

$\rightarrow \log_b a = k$

$p > -1 \rightarrow p=0$

Hence ; $\Theta(n^{1/2} \log n)$.

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

$a=4$, $b=2$, $k=2$ & $p=1$

$\log_2 4 = 2 = k$

$p > -1$ as $p=1$

Hence ; $\Theta(n^2 \log^2 n)$

Q8) $T(n) = 4T(n/2) + n^2$

$T(n) = O(n)$, $T(n) \leq c \cdot n$

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

$T(n) = 2cn + n^2$? $T(n) \leq cn$

$cn \geq 2n + n^2$

↳ not possible since n^2 dominates.

$\therefore T(n) \neq O(n)$ false.

$T(n) = O(n^2)$, $T(n) \leq cn^2$

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

$T(n) = n^2(c+1)$? $T(n) \leq cn^2$

$\cancel{n^2}(c+1) = c\cancel{n^2}$

$c+1 \leq c$ → not possible.

$\therefore T(n) \neq O(n^2)$

↳ false