Ibrahim Johar Farooqi
23K-0074
BAI-5A

Design Analysis & Algorithms — Assignment 2.

Date ____ 20__

Q.1)

↳ Quicksort

↳ Pseudocode

```
QuickSort (A, low, high){
  if low < high:
     pivot_idx = Partition (A, low, high)
     QuickSort (A, low, pivot_idx - 1)
     QuickSort (A, pivot_idx+1, high)
  endif
}

Partition(A, low, high){
  pivot = A[high]
  i = low - 1

  for j = low to (high - 1)
     if A[j] >= pivot:
        i = i + 1
        swap( A[i], A[j])
     end if
  end for

  swap ( A[i+1], A[high])
  return i+1
}
```

↳ example : [10, 4, 30, 15, 20]

↳ first call (instance)

↳ QuickSort (A, 0, 4) → A = [10, 4, 30, 15, 20], low = 0, high = 4

↳ pivot = A[4] = 20 → partition around 20

[10, 4, 30, 15, 20]

idx: 0, 1, 2, 3, 4

Date ____ 20__

↳ second part → comparing
  ↳ 10 ≥ 20 → no swap possible
  ↳ 4 ≥ 20 → no swap possible
  ↳ 30 ≥ 20 → swap possible
    ↳ A[0] ↔ A[2]
  ↳ 15 ≥ 20 → no swap possible
  ↳ after loop: swap pivot w/ A[i+1]
    ↳ pivot now at A[1], pivot index = 1
    ↳ array now → [30, 20, 10, 15, 4]
↳ recursive part
  ↳ check left subarray
    ↳ left subarray: [30] → only one element → no further calls req.
  ↳ check right subarray
    ↳ right subarray: [10, 15, 4] → sorting to be done recursively
↳ partition [10, 15, 4] around pivot [4]
  ↳ 10 ≥ 4 → swap possible but no change since org. position will remain the same A[2] ↔ A[2]
  ↳ 15 ≥ 4 → swap possible but no change, A[3] ↔ A[3]
  ↳ swap pivot[4] w/ A[i+1] = A[4] → remains same
    ↳ pivot index = 4

~~partition sort~~
↳ sort [10, 15] → pivot will be [15]
  ↳ comparing: 10 ≥ 15 → not true
  ↳ swap pivot → A[2] ↔ A[3]
  ↳ final result → [30, 20, 15, 10, 4]
↳ Time Complexity
  ↳ cost per partition → $\Theta(n)$
    ↳ as 'Partition(A, low, high)' traverses subarray once, and for each element $O(1)$ work is done
  ↳ best case: pivot splits array always perfectly in half $(\frac{n}{2})$

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

$$T(n) = \begin{cases} 1 & , n=1 \\ 2T(\frac{n}{2}) + n & , n>1 \end{cases}$$

Date ____ 20__

↳ master's theorem

↳ $a=2, b=2, k=1, p=0$

$\log_b a$	k
$\log_2 2$	1
1	1

=

↳ case 2 applies → $p > -1 \rightarrow T(n) = \Theta(n^k \log^{p+1} n)$

$= \Theta(n^1 \log^{0+1} n)$

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

↳ worst case : when pivot is always the smallest or largest number in array, making the split completely unbalance

↳ one subarray's size will be $(n-1)$ and the other (0) since largest number will be picked as pivot.

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

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

↳ from guess and recall

↳ $\boxed{T(n) = \Theta(n^2)}$

↳ avg case

↳ if the pivot produces a moderately balanced splits on average, throughout the algo's working, then the recurrence will behave like in the balanced case; $\boxed{T(n) = \Theta(n \log n)}$

Q.2) list of scores: [59, 6, 35, 12, 27, 9, 1, 18, 5, 31, 16]

↳ Quicksort

↳ divide → splits the lists into 2 halves until each sublist/subarray has only one element

↳ conquer → it recursively sorts the halves

↳ combine → it merges 2 sorted halves into one sorted list

↳ $O(n \log n)$ time complexity

Date _____ 20__

1) ↳ pseudocode

```
MergeSort(scores, left, right){
  if (left < right){
    mid = (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

    leftArr[n1] , rightArr[n2]

    for i=0 to (n1-1){
        leftArr[i] = scores[left+i]
    }
    for j=0 to (n2-1){
        rightArr[j] = scores[mid+1+j]
    }
    i=0, j=0, k=left.

    while (i<n1 and j<n2){
      if leftArr[i] ≤ rightArr[j]{
          scores[k] = leftArr[i]
          i=i+1
      else
          scores[k] = rightArr[j]
          j=j+1
      }
      k=k+1
    }
```

Date ____ 20 ____

```
//copying rem. elements
while(i<n1){
    scores[k] = leftArr[i]
    i=i+1
    k=k+1
}
while(j<n2){
    scores[k] = rightArr[j]
    j=j+1
    k=k+1
}
}
```

1 2 3 4 5 6 7 8 9 10 11

↳2) [59, 6, 35, 12, 27, 9, 1, 18, 5, 31, 16]

↳ split array, at mid=5,

leftArr[0→5] = [59, 6, 35, 12, 27]

rightArr[6→10] = [1, 18, 5, 31, 16]

↳ left side → continue splitting: [59, 6, 35, 12, 27]

↳ [59, 6, 35] [12, 27]

↳ [59, 6] [35] [12] [27]

↳ [59] [6] [35] [12] [27]

↳ right side → continue splitting [9, 1, 18, 5, 31, 16]

↳ [9, 1, 18] [5, 31, 16]

↳ [9, 1] [18] [5, 31] [16]

↳ [9] [1] [18] [5] [31] [16]

↳ merge pairs from smallest level (in ascending order)

↳ [59] + [6] → [6, 59]

↳ [35] → [35]

↳ [12] + [27] → [12, 27]

↳ [9] + [1] → [1, 9]

↳ [18] → [18]

↳ [5] + [31] → [5, 31]

↳ [16] → [16]

Date ____ 20__

↳ merge further

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

↳ [6,59] + [35] → [6,35,59]

↳ [12,27]

↳ [1,9] + [18] → [1,9,18]

↳ [5,31] + [16] → [5,16,31]

↳ merge further this level: [6,35,59] [12,27] [1,9,18] [5,16,31]

↳ [6,35,59] + [12,27] → [6,12,27,35,59]

↳ [1,9,18] + [5,16,31] → [1,5,9,16,18,31]

↳ finally, merging both halves

↳ left: [6,12,27,35,59] right: [1,5,9,16,18,31]

↳ final sorted merged list: [1,5,6,9,12,16,18,27,31,35,59]

3) 4 way merge sort variation

↳ divide → splits array of size n, into 4 parts $\left(\frac{n}{4}\right)$

↳ conquer → recursively sorts each part

↳ combine → merging the 4 sorted lists into 1 sorted list

```
MergeSort(scores, left, right) {
    if (left < right){
        part1 = left + (right-left)/4
        part2 = left + (right-left)/2
        part3 = left + 3(right-left)/4
        part4 = part3 + 1
        MergeSort(scores, left, part1)        → T(n/4)
        MergeSort(scores, part1+1, part2)     → T(n/4)
        MergeSort(scores, part2+1, part3)     → T(n/4)
        MergeSort(scores, part4, right)       → T(n/4)

        Merge(scores, left, part1, part2, part3, right) → n
    }
}
```

T(1)

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

↳ time complexity:

Date ____ 20__

↳ $a=4, b=4, k=1, p=0$

$\log_b a$	k
$\log_4 4$	1
1	1

=

↳ case 2 applies → $p > -1$ → $T(n) = \Theta(n^k \log^{p+1} n)$

$T(n) = \Theta(n^1 \log^{0+1} n)$

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

Q.3)

a)
```
StandardMatrixMul(W,R,n,A){
    for(i=0 to n){                          ---> n
        for(j=0 to n){                      ---> n x n
            A[i][j] = 0
            for(k=1 to n){                  ---> n^2 x n
                A[i][j] = A[i][j] + W[i][k] * R[k][j]
            }
        }
    }
}
```

↳ $n^2 \times n$

$\boxed{\text{time Complexity} = \Theta(n^3)}$

b) Strassen's Algo

↳ split the 4x4 matrix into 2 2x2 matrices, where each letter is a 2x2 submatrix

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

↳ applying the 7 formulae of Strassen $(M_1 - M_7)$ ↳ after formulae, combining results

↳ $M_1 = (A+D) \times (E+H)$

↳ $M_2 = (C+D) \times (E)$

↳ $M_3 = (A) \times (F-H)$

↳ $M_4 = (D) \times (G-E)$

↳ $M_5 = (A+B) \times (H)$

↳ $M_6 = (C-A) \times (E+F)$

↳ $M_7 = (B-D) \times (G+H)$

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

$C_{12} = M_3 + M_5$

$C_{21} = M_2 + M_4$

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

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

c) recurrence relation

Date ____ 20__

↳ since each submatrix is half the size in each dimension → $\frac{n}{2}$

↳ strassen (recursive) call is made 7 times

↳ $\Theta(n^2)$ for additions & subtractions of submatrices in the combination step as wa

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

d) master's theorem

↳ $a=7, b=2, k=2, p=0$

$\log_b a$	k
$\log_2 7$	2
2.81	2

↳ $2.81 > 2$

↳ case 1 applies

↳ $T(n) = \Theta(n^{\log_b a})$

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

$$\boxed{T(n) = \Theta\left(n^{\log_2 7}\right)}$$

e) comparision

↳ naive - triple loop (standard matrix mult.)

↳ recurrence : $T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) \rightarrow$ T.C. : $\Theta(n^3)$

↳ strassen

↳ recurrence : $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2) \rightarrow$ T.C : $\Theta(n^{2.81})$

↳ Strassen's Algo is faster asymptotically than the naive-triple loop method. As Strassen's Algo reduces the number of recursive multiplications from 8 rec. calls in naive-triple loop method to 7 calls (recursive) in strassen method, only at the cost of a few extra addition and subtractions in the $M_1 - M_7$ operations.

Q.4) Master Theorem

↳ applies to : $\boxed{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$

↳ a → num of recursive function calls, $a \geq 1$

↳ b → factor by which input size is reduced, $b > 1$

↳ f(n) → runtime of each function (dividing & combining) non recursive

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

Date ____ 20__

↳ **case 1:** $\log_b a > k$ → $T(n) = \Theta(n^{\log_b a})$

↳ here the recursive terms grows at a faster rate than the non-recursive term n^k, as a result of recursive term dominating the operation

↳ **case 2:** $\log_b a = k$

- ↳ $p > -1$: $\Theta(n^k \log^{p+1} n)$
- ↳ $p = -1$: $\Theta(n^k \log \log n)$
- ↳ $p < -1$: $\Theta(n^k)$

↳ here the recursive & non-recursive terms, grow at the same rate

↳ **case 3:** $\log_b a < k$

- ↳ $p \geq 0$: $\Theta(n^k \log^p n)$
- ↳ $p < 0$: $\Theta(n^k)$

↳ here the non recursive term grows faster than the recursive term.

Q.5) ↳ going to assume that the question is refering to Q2

↳ checking if modified mergesort recurrence from Q2 can be resolved w/ master's theorem

↳ $T(n) = 4T(\frac{n}{4}) + \Theta(n)$

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

↳ $\log_b a$ | k

$\log_4 4$ | 1

1 | 1

$=$

↳ yes

↳ case 2 applies

↳ $p \geq -1$: $T(n) = \Theta(n^k \log^{p+1} n)$

$T(n) = \Theta(n^1 \log^{0+1} n)$

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

↳ yes, the modified recurrence can be solved using master's theorem.

Q.6) 1) $T(n) = 2T(\frac{n}{2}) + n^2$

↳ substitution

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

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

Date ____ 20__

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

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

$= 2^3\left[T\left(\frac{n}{8}\right)\right] + \frac{n^2}{4} + \frac{n^2}{2} + n^2$

↳ pattern: $2^k T\left(\frac{n}{2^k}\right) + n^2\left[\frac{1}{2^k} + \frac{1}{2^{k-1}} + \dots + \frac{1}{4} + \frac{1}{2} + 1\right]$

↳ geometric series (converges to 1)

↳ assuming: $\frac{n}{2^k} = 1$

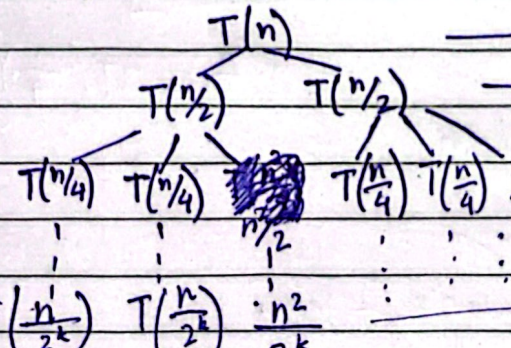
$n = 2^k$

$k = \log n$

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

$= n(1) + n^2 = n + n^2 \longrightarrow \boxed{T(n) = \Theta(n^2)}$

↳ tree



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

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

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

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

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

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

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

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

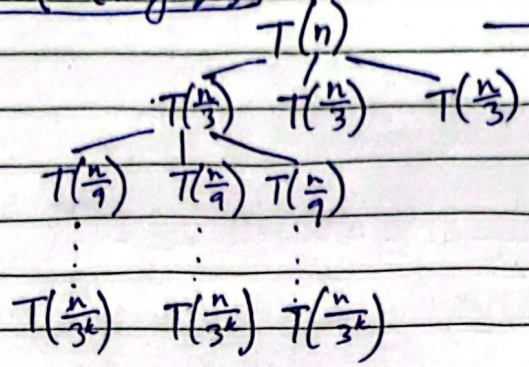
$n = 3^k \longrightarrow k = \log_3 n$

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

$= n^{\log_3 3}(1) + n(\log n - k)^2(1)$ Date ____ 20__

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

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



$\longrightarrow n(\log n)^2$

$\frac{n}{3^k}\left(\log \frac{n}{3^k}\right)^2$

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

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

$= n(1) + n(\log n)^2$

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

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

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

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

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

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

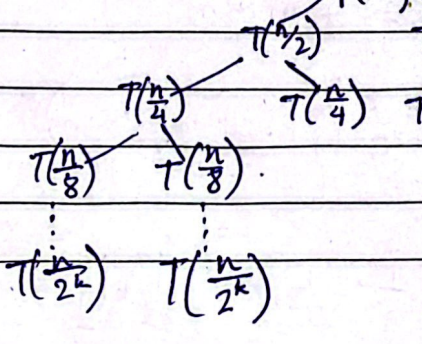
$= 2^3 T\left(\frac{n}{8}\right) + \log \frac{n}{4} + \log \frac{n}{2} + \log n$

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

$= 2^{\log_2 n} T(1) + (\log n - k)$

$= n + (\log n - k)$

$= \Theta(n)$



$\longrightarrow \log n$

$\frac{\log n}{2^k} \rightarrow \log n - \log 2^k$

$= \log n - k$

$T(n) = \log n + 2(\log n - 1) + 4(\log n$

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

Date ____ 20__

Q.7) 1) $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

$a=4, b=2, k=2, p=0$

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

↳ case 2 applies → $p > -1$:

↳ $T(n) = \theta(n^k \log^{p+1} n)$

$= \theta(n^2 \log^{0+1} n)$

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

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

$a=2, b=4, k=1/2, p=0$ ↳ case 2 applies → $p > -1$

$\log_b a$	k
$\log_4 2$	$1/2$
$1/2$	$1/2$
=	

↳ $T(n) = \theta(n^k \log^{p+1} n)$

$= \theta(n^{1/2} \log^{0+1} n)$

$\boxed{T(n) = \theta(n^{1/2} \log n)}$

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

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

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

↳ case 2 applies → $p > -1$

↳ $T(n) = \theta(n^k \log^{p+1} n)$

$= \theta(n^2 \log^{1+1} n)$

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

Q.8) 1. $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

↳ guess 1 : $T(n) = O(n)$

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

$T(n) \leq c.n$

$T\left(\frac{n}{2}\right) \leq c.\frac{n}{2}$

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

$T(n) = 4\left(c\left(\frac{n}{2}\right)\right) + n^2 \leq c.n$

$= 4c\left(\frac{n}{2}\right) + n^2 \leq c.n$

$= 2cn + n^2 \leq c.n$

$n^2 \leq c.n$

↳ guess 1 is wrong (as n^2 is dominating term)

QUALITY PAPER PRODUCTS

Date ____ 20__

↳ guess 2: $T(n) = O(n^2)$

$T(n) \leq cn^2$

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

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

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

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

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

↳ guess 2 is also wrong, instead of a constant we got $(c+1)$, indicating the inequality doesn't hold for constant

↳ hence incorrect