National University of Computer & Emerging Sciences, Karachi

Fall-2022 Department of Computer Science

Solution Mid Term-1

26th September 2022, 10:00 AM – 11:00 AM

Course Code: CS2009	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Dr. Farrukh Saleem, Dr. Waheed Ahmed, Anum Hamid, Aqsa Zahid and Sohail Afzal	
Student Roll No:	**Section:**

Instructions:

- Return the question paper
- Read each question completely before answering it. There are **6 questions** on **2 pages**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper

Time: 60 minutes. **Max Marks: 12.5**

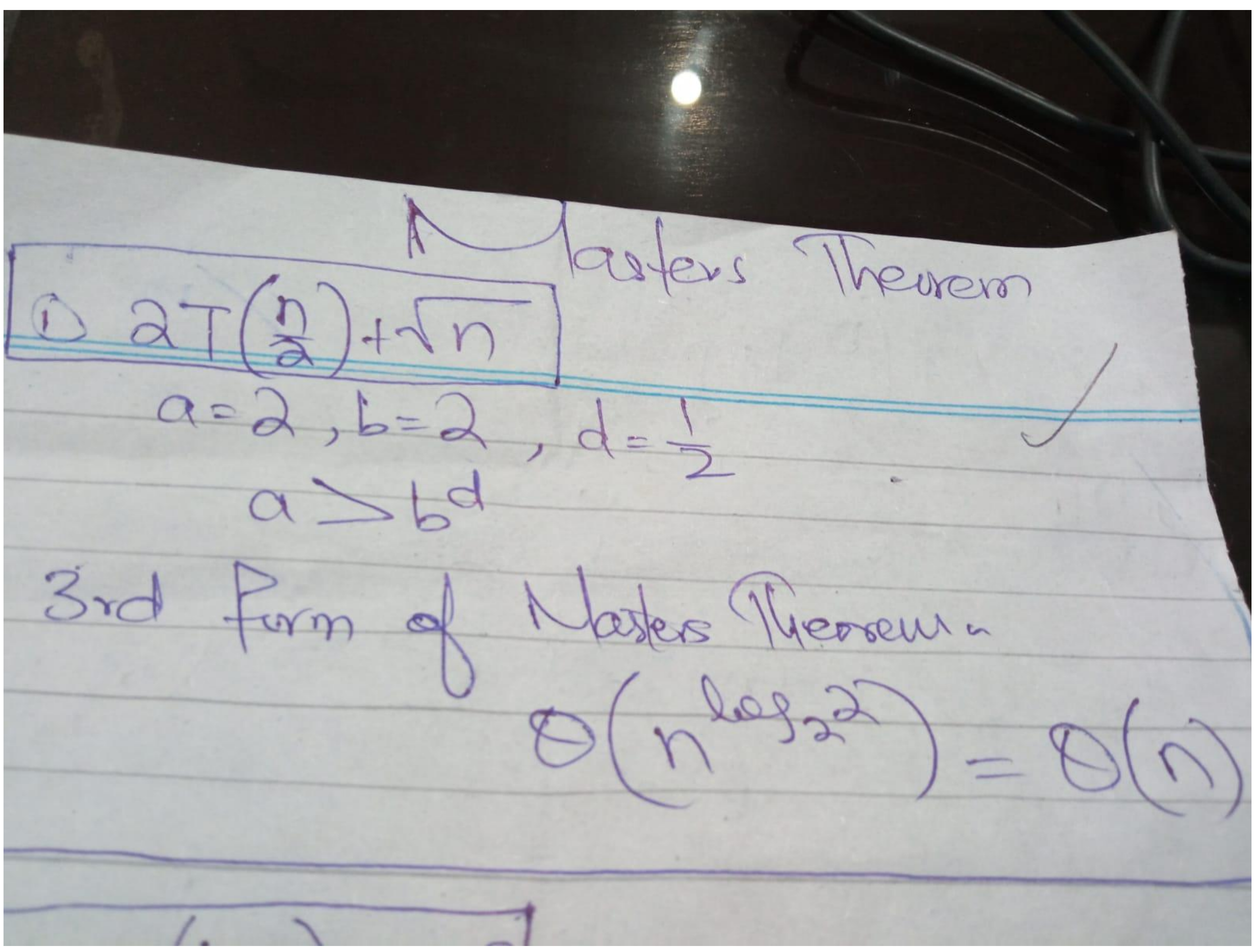
Question # 1 **[0.5*3 = 1.5 marks]**

Solve the following recurrences using **Master's Method.** Give argument, if the recurrence cannot be solved using Master's Method. [See appendix for Master's method 4th case if required]

$\boldsymbol{a})\ \boldsymbol{T(n) = 2\,T\left(\frac{n}{2}\right) + \sqrt{n}}$

$\boldsymbol{b})\ \boldsymbol{T(n) = 5T\left(\frac{n}{2}\right) + 2^{\log_2 n}}$

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



Masters Theorem

$\boxed{① \ 2T\left(\frac{n}{2}\right)+\sqrt{n}}$

$a=2,\ b=2,\ d=\frac{1}{2}$

$a > b^d$

3rd Form of Masters Theorem

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

(3) $\boxed{5T\left(\frac{n}{2}\right) + 2^{\log_2 n}}$

$$5T\left(\frac{n}{2}\right) + n^{\log_2 2} \Rightarrow 5T\left(\frac{n}{2}\right) + n$$

$a=5,\ b=2,\ d=1$

$a > b^d$

Third form → $\Theta\left(n^{\log_2 5}\right)$

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

4th case → $a=8$

$b=2$

$$f(n) = n^{\log_2 8} \log^3 n$$

$$k=3$$

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

$$T(n) = n^{\log_2 8} \log^{3+1} n$$

1

$$T(n) = n^3 \log^4 n$$

Question # 2 **[0.25*2 + 1.5*2 = 3.5 marks]**

Part 2A) Write the recurrence relation for the following Algorithm statements (don't solve them)

a) Algorithm A solves problems by dividing them into five sub problems of half the size, recursively solving each sub problem, and then combining the solutions in $O(n^2)$ time.

$$Answer\ T(n) = 5\,T\,\frac{n}{2} + O\,(n^2)$$

b) Algorithm B solves problems of size n by dividing them into nine sub problems of size n/3, recursively solving each sub problem, and then combining the solutions in linear time.

$$Answer\ T(n) = 9T(n/3) + n$$

Part 2B) Compute the time complexity of the following recurrence relations by using **Iterative Method or Recurrence-Tree Method**. [See appendix for formulas if required]

$\boldsymbol{a})\ \boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + nlogn}$, Assume $T(1) = 1$

$\boldsymbol{b})\ \boldsymbol{T(n) = 2T(n-1) + n^2,}$ Assume $T(1) = 1$

Solution:

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

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

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

T(n) = 2 { 2 T (n/4) + n/2 log n/2} + n log n

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

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

T(n/8) = 2 T(n/16) + n/8 log n/8

T(n) = 4 { 2 T (n/8) + n/4 log n/4 } + n log n/2 + n log n

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

T(n) = 8{2T(n/16) + n/8 log n/8} + n log n/4 + n log n/2 + n log n

T(n) = 16T(n/16) + n log n/8+ n log n/4 + n log n/2 + n log n

T(n) = 2^4T(n/ 2^4) + n log n / 2^3 + n log n/2^2 + n log n/2 + n log n

……..

T(n) = 2^k T(n/2^k) + n log n / 2^{k-1} + n log n/2^{k-2} + n log n/2 + n log n

Lets n/ 2^k = 1 → 2^k = n → k =log n

n . T(1) + n (log n / 2^{k-1} + log n/2^{k-2} +log n/2^{k-3} … + log n – 1 + log n)

log n / 2^{k-1} = log n/ 2^k . 2^{-1} = 1

log n / 2^{k-2} = log n/ 2^k . 2^{-2} = 2

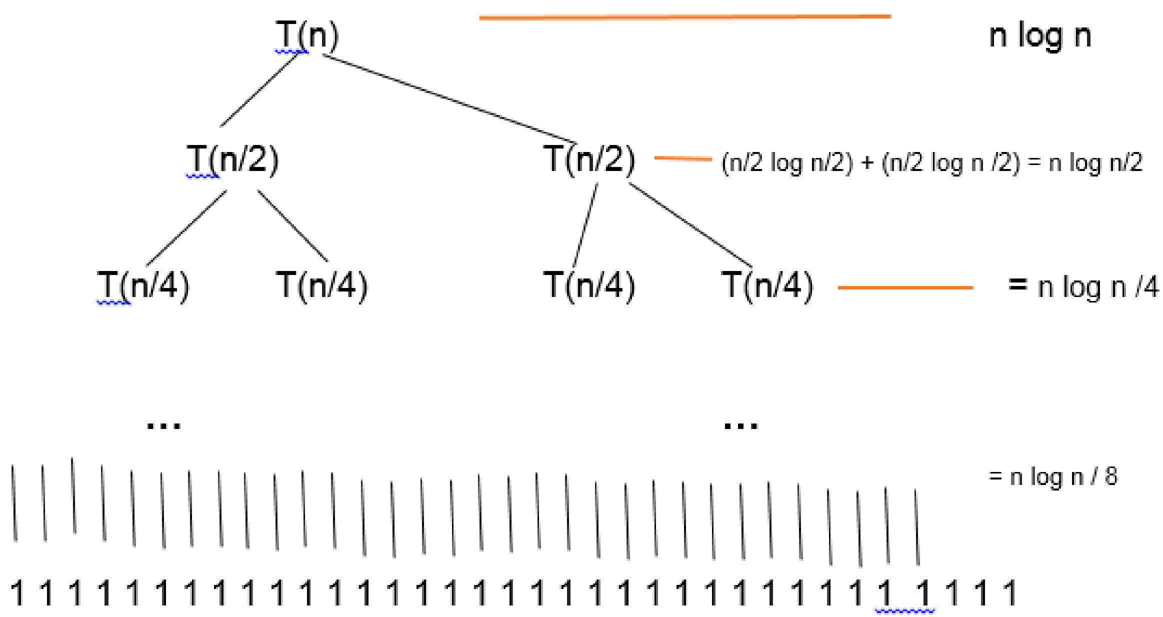
n . T(1) + n (1 + 2 + 3+ … + log n – 1 + log n)

1 + 2 + 3 + … + log n = log n (log n – 1) / 2

n + n (log^2 + log n)

O (n $\log^2$n)

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



Number of leaves n

n + (n log n + n log n/2 + n log n/4 + n log n/8 + ...)

O(n) + n (log n / 2^{k-1} + log n/2^{k-2} +log n/2^{k-3} ... + log n – 1 + log n)

log n / 2^{k-1} = log n/ 2^{k} . 2^{-1} = 1

log n / 2^{k-2} = log n/ 2^{k} . 2^{-2} = 2

n . T(1) + n (1 + 2 + 3+ ... + log n – 1 + log n)

1 + 2 + 3 + ... + log n = log n (log n – 1) / 2

n + n (log^2 + log n)

O (n $\log^2$n)

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

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

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

T(n) = 4T(n-2) + 2n – 2 + n

T(n) = 4(2T(n-3) + n - 2) + 3n - 2

T(n) = 8T(n-3) + 4n -8 + 3n -2

T(n) = 8(2T(n-4) + n -3) + 7n - 10

T(n) = 16T(n-4) + 8n – 24 + 7n -10

T(n) = 2^4T(n-4) + (2^4 – 1)n - c

....

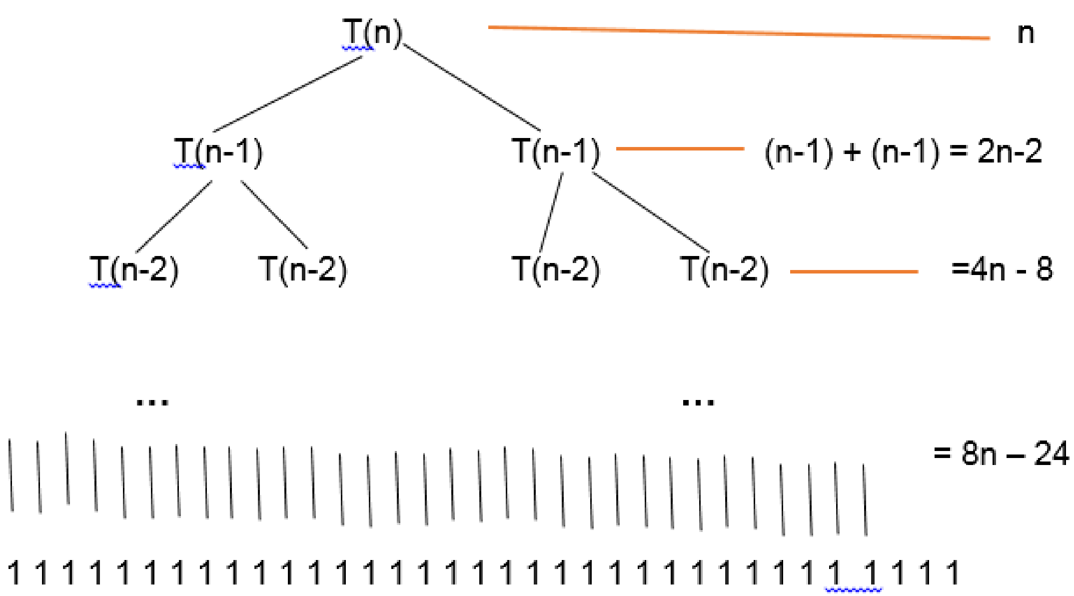
T(n) = 2^k T(n-k) + 2^k . n

Lets k =n-1

2^k . 1 + 2^k . n

2^n-1 + n . 2^n-1

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



Number of leaves (subproblem) 2^n

O (2^n) + (n + 2n + 4n + 8n+ … + 2^k) -2 – 8 -24

O (2^n) + n ($2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^k$) Apply geometric Series

O (2^n) + n (2^n – 1) / 2-1

O(2^n) + n 2^n

O (n. 2^n)

Question # 3 **[1.5 mark]**

Consider following pseudo code to find maximum number from array and prove given loop invariant :

Algorithm Computing the maximum of the elements of an array

Require: Array *A* of length *n*
M ← *A*[0]
for *i* ← 1...n-1 **do**
  if M < *A*[*i*] **then**
    M ← *A*[*i*]
  end if
end for
return *M*

Loop Invariant Property: At the beginning of iteration i, M = max{A[j] : 0 <= j <= i-1}

Solution:

1. *Initialization* ($i = 1$): Observe that M is initialized as $A[0]$. The loop invariant claims for $i = 1$ that $M_1 = \max\{A[j] : 0 \leq j \leq 0\} = \max\{A[0]\} = A[0]$. The loop invariant hence holds for $i = 1$, since M is initialized with $A[0]$.

Maintenance: Assume that the loop invariant holds in the beginning of iteration i, i.e., $M_i = \max\{A[j] : 0 \leq j \leq i-1\}$. We need to show that $M_{i+1} = \max\{A[j] : 0 \leq j \leq i\}$. Observe that the body of the loop consists of an IF operation. We thus need to distinguish two cases: when the IF evaluates to true and when the IF evaluates to false.

Suppose first that the IF evaluates to false. Then $M \geq A[i]$ holds and M is not updated. In this case we thus have $M_{i+1} = M_i$. Recall that $M_i = \max\{A[j] : 0 \leq j \leq i-1\}$. We thus need to show that in this case we have $\max\{A[j] : 0 \leq j \leq i-1\} = \max\{A[j] : 0 \leq j \leq i\}$. This is of course true since the fact that the IF evaluates to false implies $M_i \geq A[i]$. Hence $\max\{A[j] : 0 \leq j \leq i-1\} \geq A[i]$ which in turn implies $\max\{A[j] : 0 \leq j \leq i-1\} = \max\{A[j] : 0 \leq j \leq i\}$.

Next, we need to see what happens if the IF evaluates to true. Then $M < A[i]$ and M is updated to $A[i]$. Observe that in this case $M_{i+1} = A[i]$. Observe that $M < A[i]$ means that $\max\{A[j] : 0 \leq j \leq i-1\} < A[i]$ and hence $\max\{A[j] : 0 \leq j \leq i\} = A[i]$. Since $M_{i+1} = A[i]$, the loop invariant thus holds.

3. *Termination*: We have that after the last iteration (or before the nth iteration that is never executed) $M = \max\{A[j] : 0 \leq j \leq n-1\}$. M is thus the maximum of the elements in A.

Question # 4 **[1 mark]**

Apply **Substitution Guess & Test method** on given recurrence relation to identify if given guess is true :

$\boldsymbol{T(n) = T(n-2) + n^2}$ Guess $\boldsymbol{T(n) = O(n^3)}$

Solution:

Inductive Case: For n>2, we show that P(n-2)⟹P(n).

Assume that P(n-2) holds.

Then

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

$\leq c(n-2)^3+n^2$

$< cn^2(n-2)+n^2$

$=n^2(c(n-2)+1)$

$\leq n^2(c(n-2)+2c)$ for $c \geq 0.5$

$=cn2.$

<u>Question # 5</u> **[2 + 1.5 = 3.5 marks]**

Part 5A) Given a sorted array arr[] and a number x, Modify the below AlgoS to find the 'first' occurrence of the number x.

Part 5B) Dry run the algorithm which you modified, to show the steps to search for the first occurrence of number x = 2 in the array arr[] = {1, 2, 2, 3, 3}

```
AlgoS (arr, x, low, high)
  if high >= low
    mid = (low + high) / 2
    if x == arr[mid]
      return mid
    else if x > arr[mid]
      return AlgoS (arr, x, mid + 1, high)
    else
      return AlgoS (arr, x, low, mid - 1)

  return -1
```

Solution (a):

In the condition: if x == arr[mid]

Add further condition: if (**(mid == 0 || x > arr[mid-1])** && x == arr[mid])

So the updated algorithm will be:

```
AlgoS (arr, x, low, high)
  if high >= low
    mid = (low + high) / 2
    if ( ( mid == 0 || x > arr[mid-1]) && x == arr[mid])
      return mid
    else if x > arr[mid]
      return AlgoS (arr, x, mid + 1, high)
    else
      return AlgoS (arr, x, low, mid - 1)

  return -1
```

Solution (b):

<u>Assuming first index of array to be 1.</u>

Let x=2, so for arr[] = {1, 2, 2, 3, 3};

<u>First Iteration</u>

```
AlgoS (arr, 2, 1, 5)
  if 5 >= 1
    mid = (1 + 5) / 2
    if ( ( mid == 0 || 2 > 2) && 2 == 2)
      return mid
    else if 2 > 2
      return AlgoS (arr, x, mid + 1, high)
    else
      return AlgoS (arr, 2, 1, 2)

  return -1
```

Second Iteration

```
AlgoS (arr, 2, 1, 2)
  if 2 >= 1
    mid = (1 + 2) / 2
    if ( ( mid == 0 || 2 > 1) && 2 == 2)
      return mid
    else if 2 > 2
      return AlgoS (arr, x, mid + 1, high)
    else
      return AlgoS (arr, x, low, mid - 1)

  return -1
```

So the index 2, which is the first occurrence of number 2, will be returned.

Question # 6 **[1 + 0.5 = 1.5 marks]**

a) Apply below algorithm for SomeMethod(A,1,7,4), where A = {3,-1,-1,10,-3,-2,-4}. Clearly show the values of left_sum and right_sum for each iteration.

b) What is the time complexity of 'SomeMethod'.

```
int SomeMethod(int arr[], int l, int h, int m)
{
    int sum = 0;
    int left_sum = INT_MIN;
    for (int i = m; i >= l; i--) {
        sum = sum + arr[i];
        if (sum > left_sum)
            left_sum = sum;
    }

    sum = 0;
    int right_sum = INT_MIN;
    for (int i = m; i <= h; i++) {
        sum = sum + arr[i];
        if (sum > right_sum)
            right_sum = sum;
    }
return max(left_sum + right_sum - arr[m], left_sum, right_sum);
}
```

Solution:

Left Sum = 11, Right Sum = 10

Appendix

Masters Theorem 4th Case

If $f(n) \in \Theta(n^{\log_b a} \log^k n)$ *for some* $k \geq 0$ *then*

$$T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$$

$$\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r} \quad \text{(if r<1)}$$

$$\sum_{k=0}^{n} 2^k = 2^{k+1} - 1$$