A

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam
Duration: 1 Hour 20 Minutes

Semester: Spring 2024
Full Marks: 35

CSE 221: Algorithms

Answer the following questions.
Figures in the right margin indicate marks.

Name:	ID:	Section:

1 **a.** **CO2** Consider the following functions. **03**

$f_1(n) = log\,(n!)$

$f_2(n) = n^2 log_n(n^n)$

$f_3(n) = n^4 + 10n^2$

$f_4(n) = 17.13^n$

$f_5(n) = n \log n$

$f_6(n) = n * \sqrt[5]{n^2}$

Write a correct asymptotic upper bound for each of the above and sort the functions in ascending order of their growth rate.

b. **CO2** **Write** the asymptotic time complexity of the following code snippet. Show your works/reasoning. **03**

```
1. for i in range (1,n)
2.     j= 1
3.     while j < i*i
4.         j= j-1
```

c. **CO2** **Express** the following (either one of the two) running time $T(n)$ with an asymptotic bound. **04**

$$T(n) = 8T(\tfrac{n}{4}) + n\sqrt{n} \quad Or \quad T(n) = 2T(n-2) + 1$$

Any method is acceptable as long as you show calculations.

2 **CO1** i. By showing necessary math, **explain** how Karatsuba's Fast Multiplication algorithm converts an N-digit multiplication to three N/2-digit multiplications. **03**

ii. Can we modify the algorithm to multiply two N-bit binary numbers? **Explain** how or why not. **02**

3 **a.** **CO1** **Write** the worst case time complexity of quick sort? **Illustrate** an array where the worst case of quick sort occurs if the last element is chosen as pivot. **04**

b. Consider an array containing **N** unique values where for some index **i**, the values are in increasing order from index **0** to **(i-1)**, and then again from **i** to **(N-1)**. Moreover, it is guaranteed that all the values from index **0** to **(i-1)** are greater than all the values from **i** to **(N-1)**.

An example array is given below.

index	0	1	2	3	4	5	6	7
value	9	12	15	2	4	5	7	8

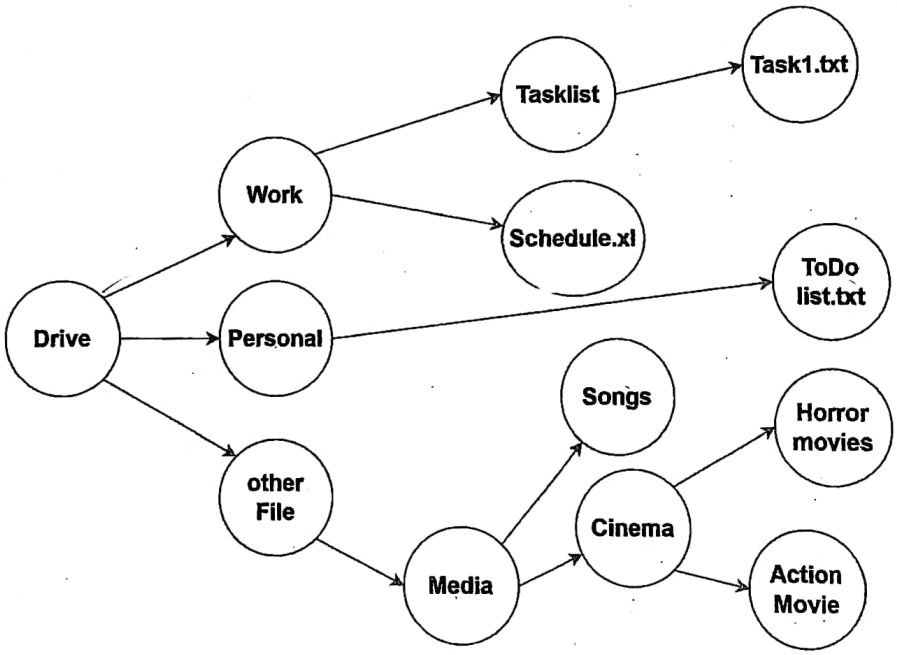
Here **i=3**, it means the values are in increasing order from index 0 to 2, and then again from 3 to 7. Also, all values from index 0 to 2 are greater than all values from 3 to 7 (it is guaranteed, no checking required).

Given such an array, propose an algorithm to find the index **i**.

CO3 i) **Write** your algorithm with a code/pseudocode/flowchart/step-by-step instructions. **04**

CO2 ii) **Write** the time complexity of your algorithm. **02**

4 **CO1** You are a computer science student and you are given a file structure encoded in a graph. You want to make a navigator system where if a person wants to find a file they can enter the name of the file and your program will return the exact location of the file.
For example, consider the following graph:



If the user wants to find task1.txt; your program will give the following output:
Drive->Work->Tasklist->task1.txt
If the user wants to find Horror Movies; your program will give the following output:
Drive->other file->Media->Cinema->Horror Movies

Now answer the following questions:

i. **Give** the adjacency list representation of this graph. You can use either the whole name or a shorter version of each node. **03**

ii. **Write** the name of your preferred algorithm to solve the above mentioned problem (a navigator system). Explain your reasoning in brief. **02**

iii. **Show** a simulation of your presented solution with proper use of data structure and other necessary details to give the desired answers as the sample input shows. **05**

B

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam
Duration: 1 Hour 20 Minutes

Semester: Spring 2024
Full Marks: 35

CSE 221: Algorithms

Answer the following questions.
Figures in the right margin indicate marks.

Name:	ID:	Section:

1 **a.** **CO2** Consider the following functions. **03**

$f_1(n) = log\,(n!)$

$f_2(n) = n * \sqrt[4]{n^2}$

$f_3(n) = n^3 + 10n^2$

$f_4(n) = 17.13^n$

$f_5(n) = n\,log\,n$

$f_6(n) = n^2\,log_n(n^n)$

Write a correct asymptotic upper bound for each of the above and sort the functions in ascending order of their growth rate.

b. **CO2** **Write** the asymptotic time complexity of the following code snippet. Show your works/reasoning. **03**

```
1. for i in range (1,n)
2.     j= 1
3.     while j < i*i
4.         j= j+1
```

c. **CO2** **Express** the following (either one of the two) running time $T(n)$ with an asymptotic bound. **04**

$$T(n) = 4T(\tfrac{n}{4}) + \sqrt{n} \quad Or \quad T(n) = T(n-3) + n$$

Any method is acceptable as long as you show calculations.

2 **CO1** i. By showing necessary math, **explain** how Karatsuba's Fast Multiplication algorithm converts an N-digit multiplication to three N/2-digit multiplications. **03**

ii. Can we modify the algorithm to multiply two N-digit hexadecimal numbers? **Explain** how or why not. **02**

3 a. **Write** the worst case time complexity of quick sort? **Illustrate** an array where the worst case of quick sort occurs if the first element is chosen as pivot. **04**

CO1

b. Consider an array containing **N** unique values where for some index **i**, the values are in decreasing order from index **0** to **(i-1)**, and then again from **i** to **(N-1)**. Moreover, it is guaranteed that all the values from index **0** to **(i-1)** are smaller than all the values from **i** to **(N-1)**.

An example array is given below.

index	0	1	2	3	4	5	6	7
value	5	4	1	12	10	9	7	6

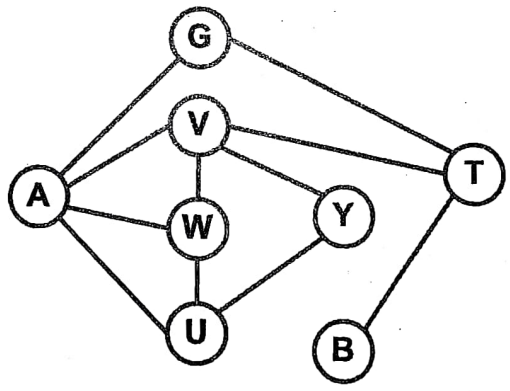
Here **i=3**, it means the values are in decreasing order from index 0 to 2, and then again from 3 to 7. Also, all values from index 0 to 2 are smaller than all values from 3 to 7 (it is guaranteed, no checking required).

Given such an array, propose an algorithm to find the index **i**.

CO3 i) **Write** your algorithm with a code/pseudocode/flowchart/step-by-step instructions. **04**

CO2 ii) **Write** the time complexity of your algorithm. **02**

4 CO1



i. **Give** the adjacency matrix representation of this graph. **03**

ii. **Determine** whether the Graph is Bipartite/Bicolorable. Show a valid grouping/coloring of the vertices. **02**

iii. **Show** a simulation of BFS algorithm with proper use of data structure and other necessary details to find the shortest path from A to B **05**