A

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam
Duration: 1 Hour 15 Minutes

Semester: Fall 2023
Full Marks: 40

CSE 221: Algorithms

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

Name:	ID:	Section:

1. **a. CO2** Explain the time complexity of the following code snippet in regards of the Big-O notation: **04**

```
for (i=0; i<n; i+=4) {
    for (j=1; j<n; j*=2) {
        for (k=0; k<30; k++) {
            print(“Am I still not 30?!!”);
        }
        print(“Why, God, why? We had a Deal!”);
        for (m=n; m>0; m-=2) {
            print(“Could you BE more dramatic?”);
        }
    }
}
```

b. CO2 Consider the following functions.

$f_1(n) = (log\,n)^{2023}$

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

$f_3(n) = n^3 + 7n^2$

$f_4(n) = 2.023^n$

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

$f_6(n) = n * \sqrt[3]{n^2}$

Now do the followings:

a. Write a correct asymptotic upper bound for each of the above. **03**
b. Sort the functions in ascending order of their growth rate, assuming n is significantly large. Just write the sorted order, no need to show any simulation. **03**

2. **CO3**

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)**. An example array is given below.
Here **i=3**, it means the values are in increasing order from index 0 to 2, and then again from 3 to 7.

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

Given the value of **i**, propose an algorithm to search a *key_value* in the array. Complexity of your algorithm must be less than **O(N)**.

a) Present your solution idea as a code/ pseudocode/ flowchart/ step-by-step instructions/ logical explanations in short. **04**

b) Write the time complexity of your presented solution. **01**

c) Show a simulation of the Merge Sort algorithm to organize the whole array in increasing order. **05**

3. **CO1**

Your friend gave you a binary string B (meaning each character is either **0** or **1**). He wanted to find out how to calculate the maximum number of consecutive 0s in that particular string. For example,

String: 100100000111	Maximum consecutive 0s: 5
String: 1010101010101	Maximum consecutive 0s: 1

You, as an algorithm enthusiast, know that this can be solved in linear time. However, your friend asked you to propose a Divide and Conquer approach

a) Name a suitable Divide and Conquer algorithm for this task. **02**

b) Explain how you can apply that algorithm in this scenario. Present your idea in a pseudocode/programmable code/Flowchart/step-by-step instructions. **06**

c) Write the time complexity of your algorithm. **02**

4. **a.** **CO1** **06**

You have the following adjacency matrix for a graph. However, some of the entries are missing. Your job is to find these missing entries with the help of some clues. Then draw the graph.

$$\begin{array}{ccccccc} & A & B & C & D & E & F \\ A & 0 & 1 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 & 1 \\ C & 0 & 1 & 0 & 1 & 0 & \\ D & 0 & & 0 & 0 & 0 & 0 \\ E & 0 & 1 & 1 & 1 & 0 & 0 \\ F & 0 & 0 & & 0 & & 0 \end{array}$$

Clues:

- E can not be reached from B
- D can be reached from A
- 2 x |E| = 3 x |V|
  (twice the no. of edges is equal to thrice the no. of vertices)
- In-degree of B is not a prime number

b. **CO2** **04**

Is an adjacency list more memory efficient than an adjacency matrix?
Justify your reasoning with respect to directed vs undirected, weighted vs unweighted, sparse vs dense graphs.

B

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam　　　　Semester: Fall 2023
Duration: 1 Hour 15 Minutes　　　　Full Marks: 40

CSE 221: Algorithms

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

Name:	ID:	Section:

1. a. CO2 Explain the time complexity of the following code snippet in regards of the Big-O notation: **04**

```
for (i=0; i<n; i+=4) {
      for (j=1; j<n; j*=2) {
            for (k=0; k<20; k++) {
                  print("Am I still not 30?!!");
            }
            print("Why, God, why? We had a Deal!");
            for (m=n; m>0; m-=4) {
                  print("Could you BE more dramatic?");
            }
      }
}
```

b. CO2 Consider the following functions.

$$f_1(n) = (log\, n)^{2000}$$

$$f_2(n) = n^3\, log_n(n^n)$$

$$f_3(n) = n^3 + 7n^2$$

$$f_4(n) = 4^n$$

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

$$f_6(n) = n * \sqrt{n}$$

Now do the followings:

a. Write a correct asymptotic upper bound for each of the above. **03**

b. Sort the functions in ascending order of their growth rate, assuming n is significantly large. Just write the sorted order, no need to show any simulation. **03**

2. **CO3** 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)**. An example array is given below.
Here **i=4**, it means the values are in increasing order from index 0 to 3, and then again from 4 to 7.

index	0	1	2	3	4	5	6	7
value	9	12	15	20	4	5	7	8

Given the value of **i**, propose an algorithm to search a *key_value* in the array. Complexity of your algorithm must be less than **O(N)**.

a) Present your solution idea as a code/ pseudocode/ flowchart/ step-by-step instructions/ logical explanations in short. **04**

b) Write the time complexity of your presented solution. **01**

c) Show a simulation of the Merge Sort algorithm to organize the whole array in increasing order. **05**

3. **CO1** Your friend gave you a binary string B (meaning each character is either **0** or **1**). He wanted to find out how to calculate the maximum number of consecutive 0s in that particular string. For example,

String: 1001000000111	Maximum consecutive 0s: 6
String: 10101010100101	Maximum consecutive 0s: 2

You, as an algorithm enthusiast, know that this can be solved in linear time. However, your friend asked you to propose a Divide and Conquer approach

d) Name a suitable Divide and Conquer algorithm for this task. **02**

e) Explain how you can apply that algorithm in this scenario. Present your idea in a pseudocode/programmable code/Flowchart/step-by-step instructions. **06**

f) Write the time complexity of your algorithm. **02**

4. **a.** **CO1** You have the following adjacency matrix for a graph. However, some of the entries are missing. Your job is to find these missing entries with the help of some clues. Then draw the graph. **06**

$$\begin{array}{ccccccc} & A & B & C & D & E & F \\ A & 0 & 0 & & 0 & 0 & 0 \\ B & 1 & 0 & 1 & 0 & 0 & 1 \\ C & 0 & 0 & 0 & 0 & 1 & 0 \\ D & 0 & 0 & 1 & 0 & 0 & 0 \\ E & 0 & & 0 & 0 & 0 & \\ F & 1 & 0 & 1 & 0 & & 0 \end{array}$$

Clues:

- B can not be reached from C
- A can be reached from D
- 2 x |E| = 3 x |V|
(twice the no. of edges is equal to thrice the no. of vertices)
- In-degree of C is not a prime number.

b. **CO2** Is an adjacency list more memory efficient than an adjacency matrix? **04**
Justify your reasoning with respect to directed vs undirected, weighted vs unweighted, sparse vs dense graphs.