**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.  **Write** a correct asymptotic upper bound for each of the above and sort the functions in ascending order of their growth rate.  **Solution:** | **03** |
| --- | --- | --- | --- |
|  | **b.**  **CO2** | **Write** the asymptotic time complexity of the following code snippet. Show your works/reasoning.   | 1. **for i in range (1,n)** 2. **j= 1** 3. **while j < i\*i** 4. **j= j-1** | | --- |   **Solution: Infinite Loop** | **03** |
|  | **c.**  **CO2** | **Express** the following (either one of the two) running time with an asymptotic bound.  Any method is acceptable as long as you show calculations.  **Solution:** | **04** |
|  |  |  |  |
| **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.  ii. Can we modify the algorithm to multiply two N-bit binary numbers? **Explain** how or why not.  **Solution:**  **i.**  source: <https://en.wikipedia.org/wiki/Karatsuba_algorithm>  ii. Just set B for any base. | **03**  **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.  **Solution:**  Worst case complexity: O(N2)  Worst case input: all sorted in either ascending or descending order. | **04** |
|  | **b.**  **CO3**  **CO2** | 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**.  i) **Write** your algorithm with a code/pseudocode/flowchart/step-by-step instructions.  ii) **Write** the time complexity of your algorithm.  **Solution:**  Algorithm: Alike binary search, take mid from current left, right. If A[mid] < A[0], i is on the left side. Otherwise, either mid is i or i is on the right side. So do **i= mid** and then search on the right side.  Complexity: O(log2N)  Can also be done in linear time by comparing current index i with (i-1) or (i+1)  **Wrong solution: any attempt where original indices are lost** | **04**  **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:   1. **Give** the adjacency list representation of this graph. You can use either the whole name or a shorter version of each node. 2. **Write** the name of your preferred algorithm to solve the above mentioned problem (a navigator system). Explain your reasoning in brief. 3. **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.   **Solution:** | **03**  **02**  **05** |

**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.  Write a correct asymptotic upper bound for each of the above and sort the functions in ascending order of their growth rate. | **03** |
| --- | --- | --- | --- |
|  | **b.**  **CO2** | **Write** the asymptotic time complexity of the following code snippet. Show your works/reasoning.   | 1. **for i in range (1,n)** 2. **j= 1** 3. **while j < i\*i** 4. **j= j+1** | | --- |   **Solution :** **1^2+2^2+3^2+......+n^2 = O(n^3)** | **03** |
|  | **c.**  **CO2** | **Express** the following (either one of the two) running time with an asymptotic bound.  Any method is acceptable as long as you show calculations. | **04** |
|  |  |  |  |
| **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.  ii. Can we modify the algorithm to multiply two N-digit hexadecimal numbers? **Explain** how or why not.  **Solution: Same as set A.** | **03**  **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 first element is chosen as pivot.  **Solution:**  Worst case complexity: O(N2)  Worst case input: all sorted in either ascending or descending order. | **04** |
|  | **b.**  **CO3**  **CO2** | 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**.  i) **Write** your algorithm with a code/pseudocode/flowchart/step-by-step instructions.  ii) **Write** the time complexity of your algorithm.  **Solution:**  Algorithm: Alike binary search, take mid from current left, right. If A[mid] > A[0], i is on the left side. Otherwise, either mid is i or i is on the right side. So do **i= mid** and then search on the right side.  Complexity: O(log2N)  Can also be done in linear time by comparing current index i with (i-1) or (i+1)  **Wrong solution: any attempt where original indices are lost** | **04**  **02** |
|  |  |  |  |
| **4** | **CO1** | 1. **Give** the adjacency matrix representation of this graph. 2. **Determine** whether the Graph is Bipartite/Bicolorable. Show a valid grouping/coloring of the vertices. 3. **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.   **Solution:**  **I**   |  | A | G | V | W | U | Y | T | B | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | G | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | V | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | | W | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | | U | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | | Y | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | | T | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | | B | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |   **II.**  No, the Graph is not Bipartite due to the V - W and W - U edges.  **III.**    Queue : A G V W U T Y B  Predecessor Pointers :   | A | G | V | W | U | T | Y | B | | --- | --- | --- | --- | --- | --- | --- | --- | | NULL | A | A | A | A | G | V | T |   Shortest Path from A to B : A > G > T > B | **03**  **02**  **05** |