Data Structures and Algorithms

Mock Minor 2

Semester: III **Division: C** **Course Code: 19ECSC201**

Note: Answer any two full questions.

Question 01

1a. You divide a given array recursively into thirds, sort each third and combine using three way merge. What is the time complexity of this modified Merge Sort? Set up a recurrence relation and solve it. **04 marks**

1b. You have a text file containing a thousand zeros. You are searching for the patterns:
i. 00001
ii. 10000
iii. 01010
How many comparisons are made in each case to declare pattern not found using brute force string search algorithm? **06 marks**

1c. Prove that any weighted connected graph with distinct weights has exactly one minimum spanning tree. Also, How can we use Prim's algorithm to find a spanning tree of a connected graph with no weights on its edges? Explain your process. **10 marks**

Question 02

2a. Design good, bad and prefix table for the input text: 101001
Compare the shifts and table performance with respect to each. **04 marks**

2b. Given an adjacency-list representation of a directed graph, where each vertex maintains an array of its outgoing edges, how long does it take in the worst case to compute the in-degree of a given vertex? Discuss the efficiency of the method you have opted for. **06 marks**

2c. Answer the following with respect to a heap:
i. Is it always true that the bottom-up and top-down algorithms yield the same heap for the same input?
ii. Find the minimum and the maximum number of keys that a heap of height h can contain.
iii. Prove that the height of a heap with n nodes is equal to $\log_2 n$. **(1+1+2+3+3) 10 marks**

PH

iv. Outline an algorithm for checking whether an array H[1..n] is a heap and determine its time efficiency.
v. Design an efficient algorithm for finding and deleting an element of the smallest value in a heap and determine its time efficiency.

Question 03

3a. Consider the following version of insertion sort. **04 marks**

```
Algorithm InsertSort2 (A[0..n − 1])
for i ← 1 to n − 1 do
    j ← i − 1
    while j ≥ 0 and A[j] > A[j + 1] do
        swap(A[j], A[j + 1])
        j ← j − 1
```

What is its time efficiency? How is it different form the version we have studied?

3b. You are given a collection of N bolts of different widths and N corresponding nuts. You are allowed to try a nut and bolt together, from which you can determine whether the nut is larger than the bolt, smaller than the bolt, or matches the bolt exactly. However, there is no way to compare two nuts together or two bolts together. The problem is to match each bolt to its nut. Design an algorithm for this problem with average-case efficiency in $\Theta(n\log n)$. **06 marks**

3c. What changes, if any, need to be made in Kruskal's algorithm to make it find a minimum spanning forest for an arbitrary graph? **10 marks**
Write the modified algorithm using the base algorithm.

**** MAY THE FORCE BE WITH YOU ****