THE UNIVERSITY OF SYDNEY

CONFIDENTIAL EXAM PAPER

This paper is not to be removed from the exam venue.

Computer Science

EXAMINATION

Semester 1- Main, 2020

COMP2123/9123 Data Structures and Algorithms

EXAM WRITING TIME: 2 hours
READING TIME: 10 minutes

EXAM CONDITIONS:
This is a CLOSED book examination.
All submitted work must be **done individually** without consulting someone else's solutions, or extenal resources like the Internet or a textbook, in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

MATERIALS PERMITTED IN THE EXAM VENUE:

One A4 sheet of handwritten and/or typed notes double sided

MATERIALS TO BE SUPPLIED TO STUDENTS:
Nothing

INSTRUCTIONS TO STUDENTS:
Type your answers in your text editor and submit via Canvas. Handwritten responses will **not** be accepted.

Start by typing you student ID at the top of the first page of your submission. Do **not** type your name.

Submit only your answers to the questions. Do **not** copy the questions.

For examiner use only:

Problem	1	2	3	4	Total
Marks					
Out of	10	10	20	20	60

Problem 1.

a) Suppose we have a recursive function defined on a binary tree. The function takes as input a node in the tree, calls another function BAR that performs some work on the node and then makes a recursive call on its parent, provided it has one. [3 marks]

```
1: def FOO(u)
2:     BAR(u)
3:     if u.parent is not empty then
4:         FOO(u.parent)
```

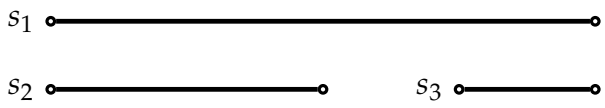
Suppose BAR(u) takes $O(\log n)$ time to execute. Analyze the time complexity of running FOO from a leaf of a tree with n nodes.

b) Solve the following recursion: [3 marks]

$$T(n) = \begin{cases} T(n) = 4T(n/2) + O(n) & \text{for } n > 1 \\ O(1) & \text{for } n = 1 \end{cases}$$

c) We have a set S of line segments on the real line (all having distinct length) and we want to pick a set $T \subseteq S$ of these line segments such that every segment s_i in S is covered by the union of the segments in T. In other words, T is a covering of S if the union of all segments in T is the union of all segments in S. [4 marks]

Example:

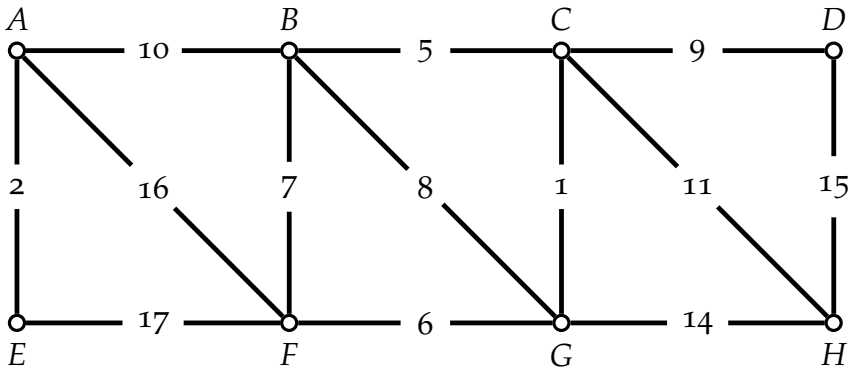


In the example above, we have three line segments: $s_1 = [0, 4]$, $s_2 = [0, 2]$, and $s_3 = [3, 4]$. The smallest cover T contains only s_1. Picking s_1 and any other line segment is still a cover, but not the smallest one. Not picking s_1 results in T not covering every line segment in S, since s_1 is not fully covered.

Construct a counterexample to show that the following algorithm doesn't compute the *smallest* set T: Sort the line segments in S by their length in decreasing order. For every segment s_i, if s_i isn't covered by T, we add s_i to T.

```
1: def SmallestCover(S)
2:     T ← [ ]
3:     Sort S by the length of the segments in decreasing order and renum-
   ber such that |s_1| ≥ |s_2| ≥ ... ≥ |s_n|
4:     for i ← 1; i ≤ n; i++ do
5:         if s_i is not covered by T then
6:             T ← T ∪ s_i
7:     return T
```

Problem 2. Consider the following edge weighted undirected graph G:



Your task is to:

a) Compute a minimum spanning tree T of G. List the edges in T. [7 marks]

b) Indicate the order in which the edges are added by Prim's algorithm starting from A. [3 marks]

(You do **not** have to explain your answer.)

Problem 3. Consider an Extended AVL Tree supporting the following operations:

- SEARCH(k): search for key k
- INSERT(k): insert key k
- REMOVE(k): remove key k, if it exists
- GET-MEDIAN(): return the median of all keys stored (do not modify the contents)

Recall that the median of n integers is defined as the $\frac{n+1}{2}$-th integer in sorted order (if n is odd), and as the average of the $\frac{n}{2}$-th integer and the $(\frac{n}{2}+1)$-th integer in sorted order (if n is even).

Your task is to come up with a data structure implementation for the Extended AVL Tree that uses $O(n)$ space, where n is the size of the tree. The SEARCH, INSERT, and REMOVE operations should run in $O(\log n)$ time. The GET-MEDIAN operation should run in $O(1)$ time. Remember to:

a) Describe your data structure implementation in plain English. [8 marks]

b) Prove the correctness of your data structure. [7 marks]

c) Analyze the time and space complexity of your data structure. [5 marks]

Problem 4. We are designing a new time-sensitive blockchain and we aim to maximize the number of transactions that we store. Let T be the set of all transactions. Every transaction t_i is a tuple (s_i, c_i), where s_i is the starting time and c_i is the computation time (indicating how long performing the calculations for this transaction takes). Time starts at $t = 0$ and all s_i and c_i are positive integers. Due to the time-sensitive nature of these transactions, a transaction is either started at its starting time, or not at all. Furthermore, the transactions stored in the blockchain are **not** allowed to overlap, i.e., the previous transaction needs to be finished when the next one starts (this start and end time is allowed to be the same, see example). We want to find the largest set of transactions that we can store such that no two overlap.

Example:
$t_1 = (1, 1)$
$t_2 = (0, 5)$
$t_3 = (2, 3)$
In this example, the largest set of non-overlapping transactions is $\{t_1, t_3\}$. Note that these two transactions do not overlap.

Your task is to give a greedy algorithm for this problem that runs in $O(n \log n)$ time. Remember to:

a) Describe your algorithm in plain English. [8 marks]

b) Prove the correctness of your algorithm. [7 marks]

c) Analyze the time complexity of your algorithm. [5 marks]