UNIVERSITY OF LONDON

BSc EXAMINATION 2023

For Internal Students of
Royal Holloway

DO NOT TURN OVER UNTIL TOLD TO BEGIN

CS2860: Algorithms and Complexity
CS2860R: Algorithms and Complexity – for FIRSTSIT/RESIT CANDIDATES

Time Allowed: **TWO hours**

Please answer **ALL** questions

1. (a) Describe the order of growth of the following functions in standard Θ notation as simply as possible. Sort the resulting Θ expressions by increasing speed of growth. [10 marks]

   i. $f(n) = 7(\log n)^2 + 3n^3 + 5$
   ii. $f(n) = 7n^2 + 4n \log n + 42$
   iii. $f(n) = 15n - 3 \log n + 7n \log n + 8$
   iv. $f(n) = 42 \log n - 24$

   (b) Explain what is the meaning of c and n_0 in the definition of Big-Oh notation. Given $5n - 1 = O(n^3)$, find the minimum possible value of c for $n_0 = 1$ and for $n_0 = 10$. [6 marks]

2. Consider the two functions `fnA` and `fnB` given below. What are the running times of `fnA` and `fnB` as functions of n? You may use Big-Oh notation in your answer. Justify your answer with complete computations. (Don't try to explain what the functions compute.) [12 marks]

```
1       int fnA(int n) {
2        int sum=0;
3        for (int i=0; i<n; i++) {
4         sum += n-i;
5        }
6        return sum;
7       }
8
```

```
1       int fnB(int n) {
2        int sum=0;
3        int size=n;
4        while (size > 1) {
5         sum += fnA(n-size);
6         size = size/2;
7        }
8        return sum;
9       }
10
```

3. Both parts of the question relate to the tree in Figure 1 below.

   (a) Perform the following operations on the binary search tree (BST) in the given order, and show the shape of the tree after each one: Delete 69. Delete 33. Insert 26. Delete 20. Show the ordinary binary search tree operations; do not rebalance the tree after insertions or deletions. [4 marks]

   (b) Consider the tree in Figure 1 again. The tree can also be seen as an AVL tree. Insert 35 into the tree and describe the rotation(s) that need to be performed to rebalance the tree. [8 marks]

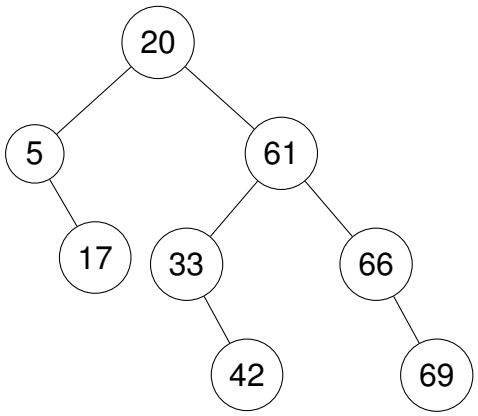

FIGURE 1: BST FOR QUESTION 3.

4. Consider the following array `C = [1, 18, 5, 25, 30, 48, 27, 41, 31, 49]` representing a binary min-heap.

   (a) Draw the tree representation of the binary min-heap. [4 marks]

   (b) Describe the algorithm from lectures for inserting a new element to a binary min-heap and illustrate it by inserting the number 10 in the heap `C`. [4 marks]

   (c) Describe the algorithm from lectures for deleting the minimum from binary min-heap and illustrate it on the heap `C` (i.e., the original min-heap `C`, not the outcome of (b)). [4 marks]

5. Consider inserting the keys 27, 24, 12, 46, 37, 20, 62, 48 into a hash table of length $m = 11$ using **open addressing** with the auxiliary hash function $h_0(k) = k$ (that is, the hash of a key is equal to the key). Illustrate the result of inserting these keys using **linear probing**. [10 marks]

6. (a) Show the order in which nodes are discovered in the graph below, if a depth-first search (DFS) is run starting from node 1 using the natural order on nodes (i.e., when it has the option, it selects lower-numbered nodes before higher-numbered ones). Draw the tree that this constructs. [4 marks]

   (b) Same question as 6a, but for a breadth-first search (BFS) instead. [4 marks]

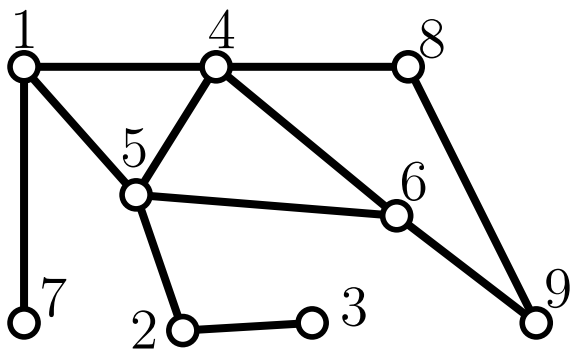

FIGURE 2: GRAPH FOR QUESTION 6.

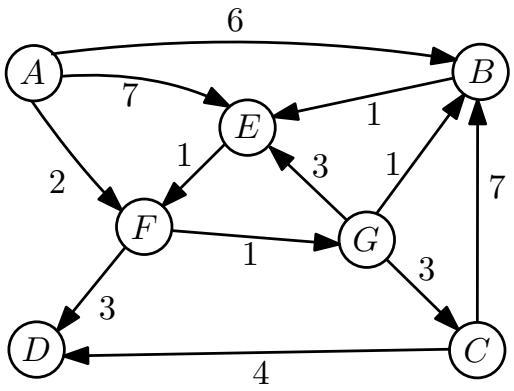

FIGURE 3: DIRECTED GRAPH FOR QUESTION 8.

7. (a) Describe Dijkstra's algorithm for finding single-source shortest paths in a graph, and illustrate it on the graph of Figure 3 by finding shortest paths to all destinations starting from vertex A. [8 marks]

   (b) Draw the resulting shortest paths tree. [3 marks]

   (c) Explain how to get an efficient implementation of Dijkstra's algorithm, i.e., an implementation that runs in time $O(m \log n)$ or better for a graph with m edges and n vertices. Justify your claim about the running time. [3 marks]

8. Propose a data structure to use for the situation described below. Justify your choices. You do not need to give complete code, but it should be clear how your solution is intended to work. You can use all data structures and operations on these data structures that have been described in lectures as “black-box” without giving an implementation.

   You want to write a simple "**unread email manager**". New emails arrive frequently and for each email you also have a deadline for handling the email. On arrival of an email you want to add the new email to your structure. You always want to handle the email with the closest deadline. Because of the huge amount of emails, you can miss deadlines on several of the emails at a time. In this case you want to be able to delete all such emails at once without reading them.

   Your "**unread email manager**" should be able to:

   - Add new unread email to your structure.
   - Find the email with the closest deadline, output it, and delete it from the structure.
   - Delete all the emails for which you missed the deadline.

   What is the time required for the three basic operations (adding a new email, finding the next email to read, and deleting all emails which have passed their deadline)? [16 marks]

END