COMP 26120

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Thursday 30th May 2019

Time: 09:45 - 11:45

Please answer all FIVE Questions

Use a SEPARATE answerbook for each SECTION

This is a CLOSED book examination

The use of electronic calculators is permitted provided they
are not programmable and do not store text

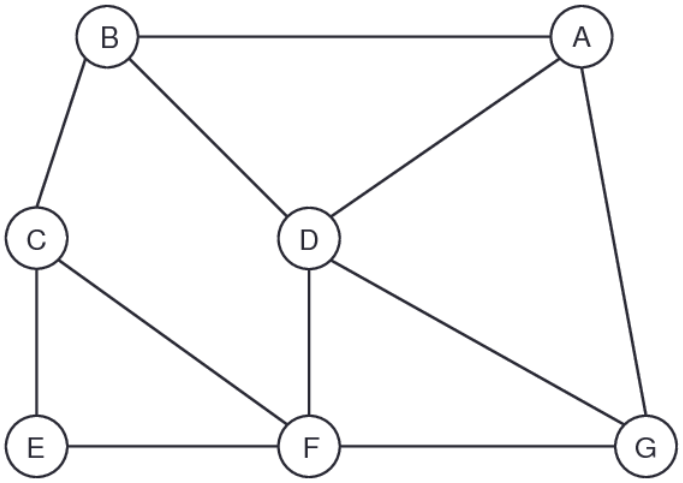
[PTO]

Section A

1. *Graphs and Graph Algorithms*

a) Write a pseudo-code description of the Depth First Search (DFS) algorithm for the traversal of an **undirected** graph G. Describe a data structure that is suitable for keeping track of the visited nodes during the course of the DFS algorithm when the algorithm is not implemented in a recursive manner. (5 marks)

b) Consider the following undirected graph:



Give a progression of the DFS algorithm, step by step, starting from the node A. Based on this, state which edges in the graph are the forward edges and which are the back edges. (10 marks)

c) Design an algorithm that checks whether a given undirected graph G contains a cycle. The algorithm should have asymptotic execution time $O(n)$, where n is the number of nodes in the graph G. Argue that your solution has this complexity. (5 marks)

Section B

2. Tractability and NP Completeness

Equivalence checking can be used to formally prove the equivalence of programs. Consider that B_1 and B_2 are Boolean programs and that B_2 can be generated from B_1 by an optimising compiler. Note that both Boolean programs B_1 and B_2 receive the same input (I) and must produce the same output (O). The problem of checking the equivalence between two Boolean programs is NP-complete. The decision problem EQUIVALENCE-CHECKING has the corresponding language:

$$EQUIVALENCE-CHECKING = \{\langle B_1, B_2, I, O\rangle : B_1 \text{ and } B_2 \text{ are Boolean programs,}$$
$$I \text{ and } O \text{ are the input and output, resp.}\}$$

Sketch a proof of NP-completeness for the decision problem EQUIVALENCE-CHECKING. In order to simplify that proof, you can consider that B_1 and B_2 contain only Boolean variables, assignments and *if-then-else* expressions.

No marks will be given for simply stating that EQUIVALENCE-CHECKING is NP-complete.

(10 marks)

Section C

Questions in this part relate to *Number-theoretic Algorithms and Algorithmic Design*. Throughout this section you may assume the existence of standard data structures (e.g. lists, queues, stacks, trees) but to get full marks you must state the complexity of the operations you perform on your data structure(s).

3. You are given a function $\mathsf{witness}(x,n)$ that takes a random variable x and a number n that works as follows:

   (a) If n is prime then $\mathsf{witness}(x,n)$ always returns *false*

   (b) If n is composite then $\mathsf{witness}(x,n)$ returns *false* with probability $q < 1$

   e.g. it sometimes incorrectly identifies a composite as prime with some probability.

   Describe (using pseudo-code) how to use this witness function to construct a probabilistic primality testing algorithm *RandomizedPrimalityTesting* that takes two inputs (a number n and a confidence parameter k) and decides whether n is prime with error probability 2^{-k}. You may assume a function $\mathsf{random}()$ that produces random numbers. Argue that your algorithm achieves the required error probability.

   (4 marks)

4. Use the *Simplex Method* to solve the following optimisation problem:

$$\begin{array}{ll} \text{Maximize:} & z = f(x,y) = x + 2y \\ \hline \text{Subject to:} & x + y \leq 120 \\ & -2x + 3y \leq 0 \\ & x \geq 0, y \geq 0 \end{array}$$

   You should state the solution e.g. maximum value for z and the corresponding values for x and y. (6 marks)

5. It is your first day as the new Head of Computer Science after the previous Head luckily won the lottery. However, there are a few issues you need to fix before you can enjoy yourself.

   a) The first problem is to find out whether your lecturers can teach the classes assigned to them. You are given a list classes of N triples $(\mathsf{id}, \mathsf{start}, \mathsf{end})$ describing the classes of L lecturers. Each triple gives the start and end time of a class given by a lecturer identified by their id.

   You should assume that $N > L$ e.g. some lecturers will teach more than one class. You cannot assume anything about the given order of classes. Lecturer ids are integers in the range 0 to $10,000$ (you may assume $L < 10,000$).

   Design an algorithm (in pseudo-code) that determines whether any lecturer has two or more overlapping classes and *explain its complexity*. There is a straightforward $O(N^2)$ solution but to receive full marks you need to do better than this.

   (6 marks)

   b) There is a concern that there are not enough printers in the school to print all course notes in time. You ask the *Head of Printers* to check this and she provides the following algorithm assigning course notes to printers, which can also be used to check how many printers are required. This takes as input a list of course information of the form (r_i, t_i) where r_i is the time that the notes will be ready and t_i is the amount of time required to print them. Due to the disorganisation of lecturers, course notes are only ready just in time e.g. they are required at $r_i + t_i$.

      1. Let P be a list containing the end time of each printer's current job. Initialise P to be the empty list.
      2. Sort course information by r_i from earliest to latest
      3. For each course (r_i, t_i) in the sorted list:
         a. If P contains an entry e such that $e < r_i$ then replace e by $r_i + t_i$
         b. Otherwise, extend P by $r_i + t_i$
      4. The number of required printers is the size of P

   Identify the algorithmic technique being used in this solution and briefly explain the general idea behind the technique with reference to the given problem.

   (3 marks)

c) Next you notice that students are taking too long to get to class so you decide to build a new super-corridor through the building. However, this will require you to move some important professors out of their offices. To keep the professors happy you have decided to give them extra vacation days but you want to minimise the number of vacation days you have to give. Thankfully the building is designed in a grid structure and the last Head of School left you a table indicating how many vacation days the professor in each office will need to stay happy. The table does not capture information about existing corridors as their location and size are irrelevant due to the space required for the super-corridor.

The super-corridor can go through adjacent offices only (it cannot go diagonally as there is not enough room). For example, in the following table the ideal solution is highlighted in grey.

4	7	8	6	4
6	7	3	9	2
3	8	1	2	3
7	1	7	3	7
2	9	8	9	3

Design an algorithm (in pseudo-code) that takes a $m \times n$ grid of numbers bribes and finds the weight of minimal path from $(0,0)$ to (m,n). You do not need to return the path itself (for now you just want to check if you can afford your super-corridor). State the complexity of your solution. In order to fit with the University's unceasing appetite for efficiency, you should avoid an exponential solution (such solutions will only receive partial marks).

(7 marks)

d) Finally, the University has launched a new online survey and have promised to give the School free cakes for the first 120 students filling in the survey. The email says that you will get one cake for an undergraduate student and two cakes for a postgraduate student. However, for every 2 postgraduate students filling in the survey there must be at least 3 undergraduates filling it in. Describe a general method (you do not need pseudo-code) to work out how many of each student should fill in the survey to get the most free cakes.

(4 marks)

END OF EXAMINATION