Department of Computing and Information Systems
COMP20007 Design of Algorithms
Semester 1, 2015
Mid Semester Test

Instructions

- **Do not open this paper until instructed to do so.** You may read this page now.
- You may fill out your student number now.
- You must have your student card on display during this test.
- The test will start at 11:10pm and finish at 11:50pm.
- The total time allowed for this test is 40 minutes and 40 marks are available.
- This is a closed book exam. You should **not** have any study notes of any kind, including electronic (no calculators, phones, etc).
- Any student seen looking at their phone (or similar) or another student's paper during the test will have their paper removed immediately, and will be referred to the School of Engineering for a breach of academic honesty.
- Throughout you should assume a RAM model of computation where input and output items fit in a word of memory, and basic operations such as $+ - \times /$ and memory access are all constant time.
- Answer all questions on this paper.
- If you make a sensible assumption about an algorithm or data structure that you feel influences your answer, write it on the test.
- Remember, -1 mark for an incorrect answer in Question 1 True/False (advised not to guess).

Student Number: _____

Question 1 [10 marks, minimum 0 marks]

Answer True or False for each of these statements. You will gain one mark for a correct answer, and **lose** one mark for an incorrect answer.

1.	$f(n) = \log(n^2)$ is in $\Theta(\log n)$	
2.	$f(n) = (\log n)^2$ is in $O(\sqrt{n})$	
3.	$f(n) = \frac{n}{\log n}$ is in $\Omega(\sqrt{n})$	
4.	Dijkstra's SSSP algorithm can run in $O(n \log n)$ time using a heap data structure for the priority queue assuming that the number of vertices and edges in the graph is $O(n)$.	
5.	Inserting a new item into a heap of n items requires $\Omega(\log n)$ time.	
6.	DFS is preferable to BFS when exploring a graph because the stack in DFS uses less space than the queue in BFS for any graph.	
7.	The worst case height of the stack used to keep track of recursion in sorting n items with standard, recursive mergesort is $\Theta(n)$.	
8.	An algorithm that can divide in $O(\log n)$ time, and conquer in $O(n)$ time will have a worst case running time that is in $\Omega(n)$.	
9.	The worst case height of the stack used to keep track of recursion in quicksort when sorting n items is $\Omega(n)$.	
10.	Consider an edge from u to v in a directed graph such that u is in strongly connected component X and v is in strongly connected component and $Y \neq X$. For any DFS on the graph, the pre number of u will be higher than the pre number of v.	

Question 2.1 [4 marks]

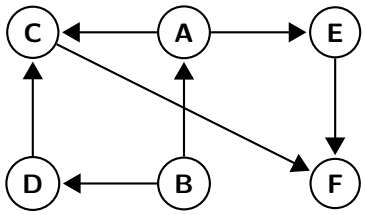
Write a recurrence relation that describes the worst case running time of Quicksort. Briefly justify why you have used this recurrence relation.

Question 2.2 [5 marks]

What is the solution to your recurrence in Question 2.1? Prove it with induction.

Question 3.1 [6 marks]

Label this graph with the pre and post numbers that would be assigned with Depth First Search. Assume the numbers begin at 1, and always prefer a lower labelled vertex if there is a choice (so start at `A`).



Question 3.2 [2 marks]

Are there any back edges or cross edges in the graph? If so, what are they?

Question 3.3 [2 marks]

Draw the graph with vertices of the graph in a single line in the order they would be sorted by a topological sort.

Question 3.4 [1 mark]

How many strongly connected components are there in the graph?

Question 3.5 [1 mark]

List all the source vertices of the graph?

Question 3.6 [1 mark]

List all the sink vertices of the graph?

Question 3.5 [1 mark]

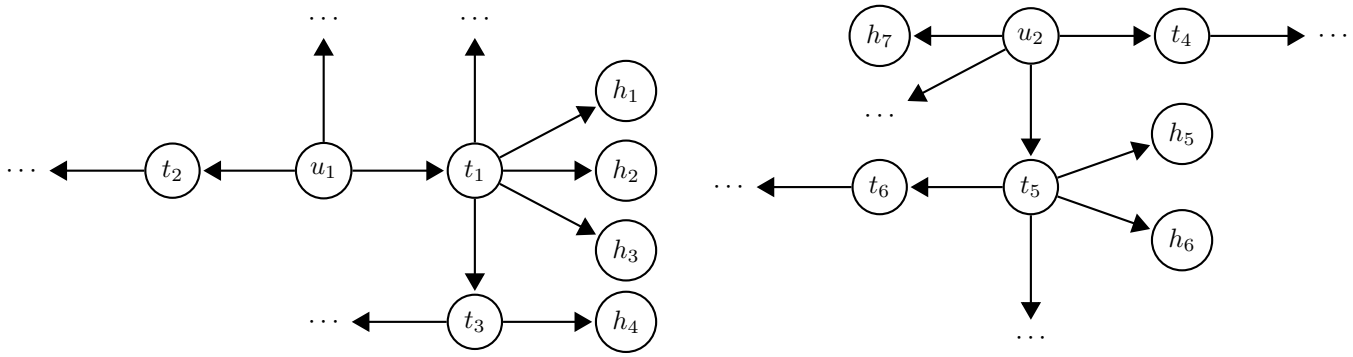
What single edge could be added so that there is only one strongly connected component in the resulting graph?

Question 4 [7 marks]

The Federal Government decides to alter the funding model of universities to a strict geographical equity model. Each university will get funding proportional to the number of eligible students living within k kilometers of travel time to the university, where k is a parameter set during an election campaign.
The consulting company Sneaky Go8 Inc has employed you to work out the value of k so that Unimelb and Monash get about the same funding without stealing each others students. You are given the input data as a graph $G(V, E)$, where

- $u_1 \in V$ is the vertex representing Unimelb;
- $u_2 \in V$ is the vertex representing Monash;
- $t_i \in V, i \in [1, T]$ represents a transport intersection, train station, and so on;
- $h_i \in V, i \in [1, H]$ represents the living location of one potential student; and
- w_e is the cost of traveling along edge $e \in E$.

Here is an example of a 2 small sections of the graph around u_1 and u_2 but not showing edge weights ("$\cdots$" represents more graph that is not shown).



Thus if U is the set of all h_i vertices whose path length to u_1 is $< k$ and M is the set of h_i vertices whose path length to u_2 is $< k$, then an algorithm to solve algorithm should find k such that the intersection of U and M is empty, and $|U|^3 + |M|^3$ is maximum. You may assume the input is a correct graph, and that $|U| > 1$ and $|M| > 1$. Give high-level pseudo code for such an algorithm.
Analyse the running time of your algorithm. Full marks will only be given for a correct algorithm that runs in $O((E + V) \log V + H \log H)$ time.