MAS341 GRAPH THEORY
PROBLEM SET 2 SOLUTIONS

QUESTION 1

The distances between seven towns are given in the table below.

A						
70	B					
62	57	C				
66	47	58	D			
78	66	59	95	E		
80	83	61	99	79	F	
75	64	71	65	40	42	G

Use the nearest neighbour algorithm, starting at A, to find an upper bound on the travelling salesman problem for these towns. By omitting A, give a good lower bound for the travelling salesman problem.

Proof. Using the nearest neighbour heuristic, we always go to the cheapest town we haven't visited. From A, the cheapest town is C costing 62. From C, the cheapest town is B costing 57. From B, the cheapest town is D costing 47. From D the cheapest town we haven't visited yet is G costing 65. From G, the cheapest town is E costing 40. From E, the only town we haven't visited yet is F costing 79. From F, we have to return to A costing 80.

This gives us a total of

$$62 + 57 + 47 + 65 + 40 + 79 + 80 = 430$$

For Part B, we first delete vertex A and then find a minimal cost spanning tree of the remaining graph. Using Kruskal's algorithm, the cheapest edges are $EG = 40, FG = 42, BD = 47, BC = 57$. The next cheapest edge, CD, would form a cycle, so we don't add it. CE costs 59, and now we have our minimal spanning tree, costing

$$40 + 42 + 47 + 57 + 59 = 245$$

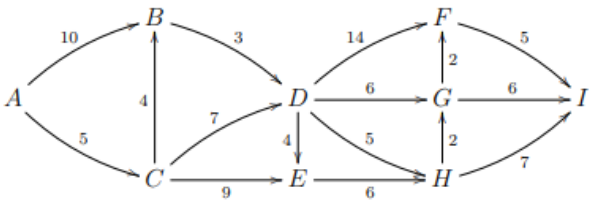
To this we have to add the lowest two edges out of A, which are AC and AD costing 62 and 66, giving 128 more, and a total lower bound of 373.

□

Comments: This question was the weakest by far – most common errors were taking the most expensive edge possible when running "nearest neighbour", and trying to get a lower bound by deleting a vertex and then running nearest neighbour. Both these seem to rest on not understanding importance of Traveling Salesperson being the *cheapest* Hamiltonian cycle. This means that ANY Hamiltonian cycle at all is an upper bound, so taking the most expensive edge at each step *does* give an upper bound, it's just a particularly *bad* upper bound. Similarly, to get a lower bound we have to do something substantially different than trying to find a Hamiltonian cycle...

QUESTION 2

(i) Consider the directed network below.



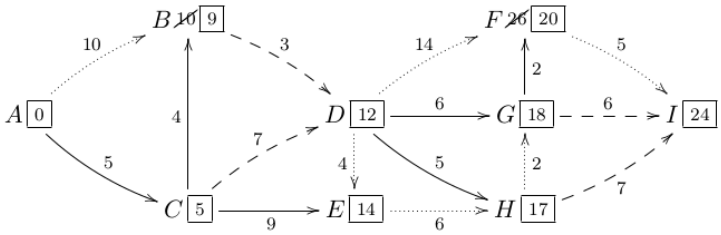
(a) Use the shortest and longest path algorithms to find all shortest and longest paths from A to I. Give the length s of the shortest path(s) and the length l of the longest path(s). ***(9 marks)***

(b) Which arc cannot be increased in length without changing s? Why is there no other arc with this property? ***(2 marks)***

(c) Is there an arc which cannot be decreased in length without changing both s and l? Justify your answer. ***(3 marks)***

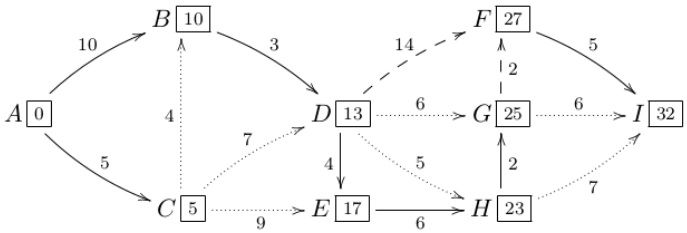
Question 4

(i)(a) Applying Dijkstra's algorithm gives the following. Solid lines are used for the shortest paths; dashed lines indicate a choice of shortest path; dotted lines are not used.



So $s = 24$ and there are four shortest paths: ACDGI, ACDHI, ACBDGI and ACBDHI.

Applying the longest path algorithm gives the following.



So $l = 32$ and there are two longest paths: ABDFI and ABDEHGFI.

(i)(b) Such an arc must be on every shortest path; the only such arc is AC.

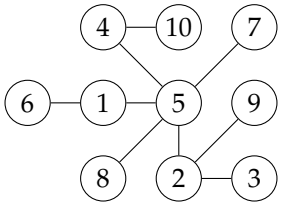
(i)(c) Such an arc must be on every longest path and at least one shortest path. There is a unique arc with this property, which is BD.

Comments: This question was done well. There are a couple of things you want to make sure are written/explained clearly, though, to make sure I don't miss them when marking 130 exams:

(1) Write out all the longest/shortest paths when/where it asks you to; sometimes these were somewhere in your work but not restated clearly in the final answer
(2) Make sure to explain your reasoning about which edges would length/shorten the shortest/longest path: "Such an edge would have to be in *all* longest paths – those edges are ..."

QUESTION 3

Find the Prüfer code of the following tree:



Proof. First we make a table of the edges, by pruning: iteratively deleting the leaf with the lowest label, and recording the leaf and the parent.

Parent	2	1	5	5	5	2	5	4	10
Leaf	3	6	1	7	8	9	2	5	4

The Prüfer code is the top row, with the last entry removed, so 2,1,5,5,5,2,5,4. □

Comments. The most common mistake was to include the last entry of the first row in the code, which is unnecessary: it can be figured out from all the previous entries! The point of the Prüfer code is not just to encode all labelled trees, but to do so without repetition or redunandant information, so that we have a bijection between codes and labelled trees, and hence a count of the labelled trees.