OLLSCOIL NA hÉIREANN, CORCAIGH
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

Examination Session and Year	**Semester 2 - Summer 2020**
Module Code	**CS2516**
Module Title	**Algorithms and Data Structures II**
Paper Number	I
External Examiner	Professor Omer Rana
Head of School/ Department	Professor Cormac Sreenan
Internal Examiners	Professor Ken Brown
Instructions to Candidates	Answer all questions Total Marks: **80** This is an Open Book exam
Duration of Paper	90 minutes
Special Requirements	The use of electronic calculators is permitted

Q1. **(40 marks)**

(i) What are the formal worst-case time complexities of the following algorithms, in terms of n, the length of the input?
(a) SelectionSort
(b) MergeSort
(c) Quicksort

(6 marks)

(ii) Sort the following list using in-place *SelectionSort*. Show the intermediate list that results after each main iteration on a separate line, with each item that is guaranteed by the algorithm to be in its final position marked in bold, and report the number of comparisons for each iteration at the end of the line in brackets. At the end, report the total number of comparisons.

17 23 15 31 49 30 52 46

(6 marks)

(iii) Sort the following list using bottom-up in-place *Mergesort*. Show the intermediate list that results after each main iteration on a separate line, with each item that is guaranteed by the algorithm to be in its final position marked in bold, and report the number of comparisons for each iteration at the end of the line in brackets. At the end, report the total number of comparisons.

17 23 15 31 49 30 52 46

(10 marks)

(iv) Sort the following list using in-place *Quicksort* (without randomisation). Show the intermediate list that results before each depth of recursive calls on a separate line, with each item that is guaranteed by the algorithm to be in its final position marked in bold, and report the number of comparisons for each iteration at the end of the line in brackets. Highlight in italics each element that you use as a pivot. At the end, report the total number of comparisons.

17 23 15 31 49 30 52 46

(10 marks)

(v) Based on the number of comparisons you reported for each of parts (ii), (iii) and (iv) above, which algorithm is fastest on the given input list, and which is slowest?

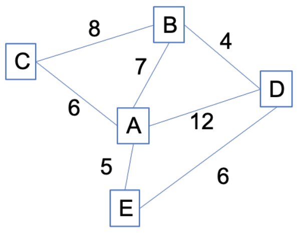
(2 marks)

(vi) Now suppose we must choose one of the three sort algorithms from part (i) to be used on lists of length n = 100,000. State which of the three algorithms would be most efficient. In your own words, explain any differences between this choice and the choice being made in part (v), and explain why and under what conditions your chosen algorithm would be most efficient.

(6 marks)

Q2. **(40 marks)**

(i) Show how the following graph would be represented using the *Adjacency Map* implementation of the *Graph* ADT (Abstract Data Type).



Note: any dictionary should be represented in the style

[*key1:value1*, *key2:value2*, ...]

and does not need to be ordered. Represent each vertex by its label, and you don't need to give any other information on the vertex objects. Give a brief description of any other class or ADT that you use, and briefly explain the notation you will use to represent them.

(8 marks)

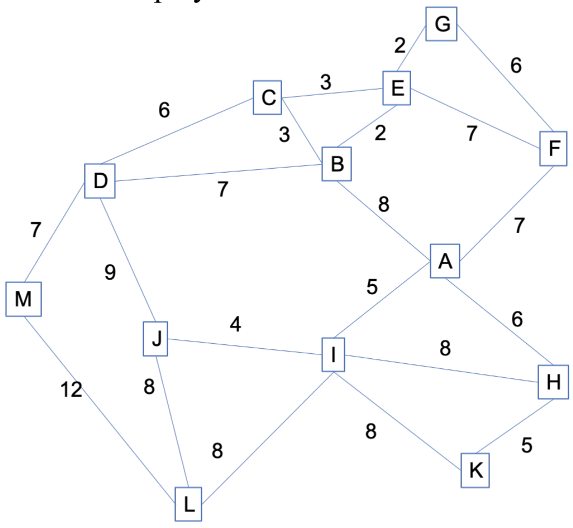
(ii) Explain how you would use your Graph representation of part (i) to obtain the following, specifying any method calls or pseudocode that you need:

(a) an edge between a pair of vertices *x* and *y*, if it exists, and

(b) the edge with lowest weight incident on a vertex *x*.

(6 marks)

(iii) For the following graph, show the progress of *Dijkstra's algorithm* as it searches for the shortest path between A and G. At each stage, show the revised APQ (adaptable priority queue) including all the relevant information for each item in it, and show which new vertex, with its associated information, has become closed. When you display the APQ, you can represent it as a sorted list. If you use any notation for data structures other than given in part (i), you must explain it, and make sure that enough details are shown that would allow someone else to trace through your execution. You can stop the algorithm when it has found the guaranteed shortest path to G. Explain how to re-construct the shortest path from the information you have displayed.



(15 marks)

(iv) For graphs of similar density to the one in part (iii), if run-time efficiency is the main goal, what data structure would you use to implement the APQ? Explain your choice.

(3 marks)

(v) Suppose now that the graph in part (iii) represents a graph in physical space, e.g. a road map, where the top of the page is North, and the edge weights represent on-the-road distances of the road segments connecting locations represented as vertices. The graph can be augmented with a table of the straight-line distances between any pair of vertices (i.e. the distance assuming free space, that could be computed from GPS coordinates without knowing anything about the road network). Below is the row of that table showing the straight-line distances to and from vertex G.

A	B	C	D	E	F	G	H	I	J	K	L	M
10	4	6	12	2	6	0	11	9	12	15	17	16

By considering the order in which the vertices are expanded in your solution to part (iii), and the vertices that are maintained in the APQ, describe an inefficiency in Dijkstra that would not be made by a person looking for a shortest path by inspection of the map, and explain why it happens. Devise an improvement we could make to the information we collect, maintain and process in Dijkstra's algorithm that would remove the inefficiency. What property would we have to ensure about the additional information to maintain Dijkstra's guarantee of generating the correct result?

(8 marks)

Total: 80 marks

END OF PAPER