COMP26120

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Tuesday 19th May 2015

Time: 09:45 - 11:45

Please answer THREE Questions from the FOUR Questions provided

Use a SEPARATE answerbook for EACH Question

This is a CLOSED book examination

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

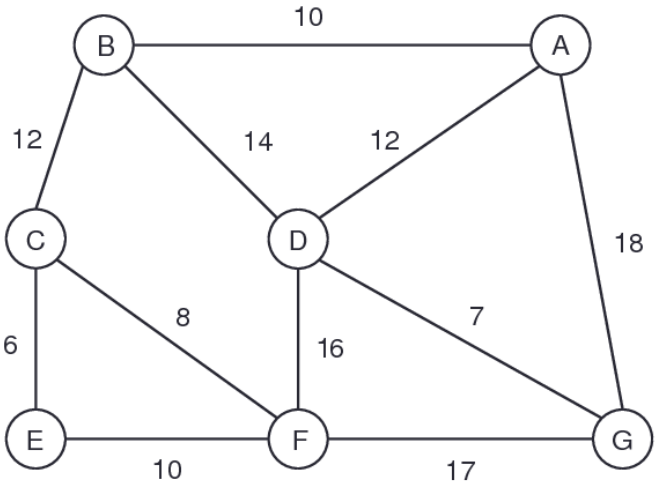
[PTO]

1. Graph algorithms.

a) Write a pseudocode description of Dijkstra's algorithm for finding the shortest path between a given node and all other nodes in a weighted undirected graph. Generalise this approach to find the shortest paths between all pairs of nodes. Assume in your answer that a priority queue is used to keep track of the unvisited nodes (you are not required to give the code for the priority queue functions). (5 marks)

b) Give the complexity of Dijkstra's algorithm as described in Part a) for finding the shortest path between a given node and all other nodes, and the shortest paths between all pairs of nodes. For this purpose, assume that a graph has N nodes and E edges and that a priority queue is used to keep track of unvisited nodes. Based on this, on which type of connected graphs is Dijkstra's algorithm most effective? (3 marks)

c) Consider the following weighted undirected graph:



Apply Dijkstra's algorithm to find the shortest path between the nodes A and E. Show the progression of the algorithm, step by step, and draw the corresponding content of the priority queue at each step of the algorithm. (7 marks)

d) Suppose that the graph in Part c) represents a road network with nodes as towns, and the weights on edges representing distances along roads. Assume that a traveller can travel only 12 distance units per day and must not spend the night on the road (i.e. partway along an edge). Discuss heuristics that can be applied to finding the shortest path between the towns A and E in this case. The shortest path can be understood as the shortest distance between A and E or the shortest time spent on the road (i.e. the minimum number of days required to complete the journey). In this context, you can consider in your answer the cases of doing only one trip between towns per day or multiple hops per day if the total path length does not exceed 12. (5 marks)

2. Knapsack problems.

   a) Give a complete description of a branch-and-bound algorithm for the 0/1 Knapsack problem. You may include pseudocode. You should explain any functions in your pseudocode, stating the input and output, and describing how they work, and the meaning of any terms you use. The more precise your description of the algorithm, the more marks will be awarded. Examples of the output of the code on a simple knapsack problem instance are *not* needed and should *not* be given. (14 marks)

   b) You are to decide which algorithm, Dynamic Programming, or Branch and Bound, would be more efficient at solving each of the following two 0/1 Knapsack Problem instances (called X and Y). Give your choice in each case, and explain *why* the behaviour of your selected algorithm would be better than the other one.
   (6 marks)

   Instance X: A knapsack instance with 1000 items of weights in the range 1–10 (in integer steps), a capacity to fit around 500 items in, and where many of the items in the instance have the same (or very similar) ratio of value to weight.

   Instance Y: A knapsack instance with 200 items varying in value-to-weight ratio, a capacity to fit around 100 items in, and with weights of arbitrary integer amounts in the range 1–25000.

3. Hashing techniques.

   a) Consider the following sequence of keys:

   `2,17,24,8,11,4,6,19`

   We wish to insert these numbers in the given order (from left to right) into a hash table of size 13 (with the elements denoted 0 to 12). Show the insertion process step by step, using the following hash function:

   $$h_1(x) = (5x + 9) \bmod 13.$$

   You should perform the insertion process twice with the collisions handled using:

   i) linear probing; (8 marks)

   ii) double hashing with the secondary hash function $h_2(x) = 5 - x \bmod 5$. (8 marks)

   b) Give, in pseudocode, a precise algorithmic description of the function `RemoveKey(k)` that removes a key k from a hash table or returns the flag `NO_SUCH_KEY`. Assume that collisions are handled by linear probing. In your answer, consider that other keys may be removed from the hash table between the insertion of the key k and its removal. (4 marks)

4. Graph traversal techniques.

   a) Explain what is meant by a Depth-First Search (DFS) of

      - a finite rooted binary tree, and
      - a finite directed graph.

      In each case you should explain clearly the principles involved and give examples to illustrate your answer, but you need not give explicit algorithms. (4 marks)

   b) For the case of finite directed graphs, give an explicit algorithm for performing DFS which numbers the nodes in the order that they are first encountered. You should express the algorithm in pseudocode. (6 marks)

   c) How does DFS of *undirected* graphs differ from that of directed graphs?

      Explain clearly how you would modify the DFS algorithm you gave in answer to Part (b) above so that the algorithm calculates the number of connected components in a finite *undirected* graph. (4 marks)

   d) For finite undirected graphs, a 2-colouring of a graph is an allocation of one of two colours (say, red and green) to each of the nodes of the graph, so that, whenever two nodes are linked by an edge, they are allocated different colours. Some graphs have a 2-colouring and some do not. Give an example of an undirected graph which does and one which does not. (2 marks)

      How can a DFS algorithm be used to determine whether or not a finite undirected graph has a 2-colouring? Your answer should be a clear explanation of the principle involved and the steps of your DFS algorithm (you need not give a program or pseudocode). (4 marks)

END OF EXAMINATION