Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Friday 23rd May 2014

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

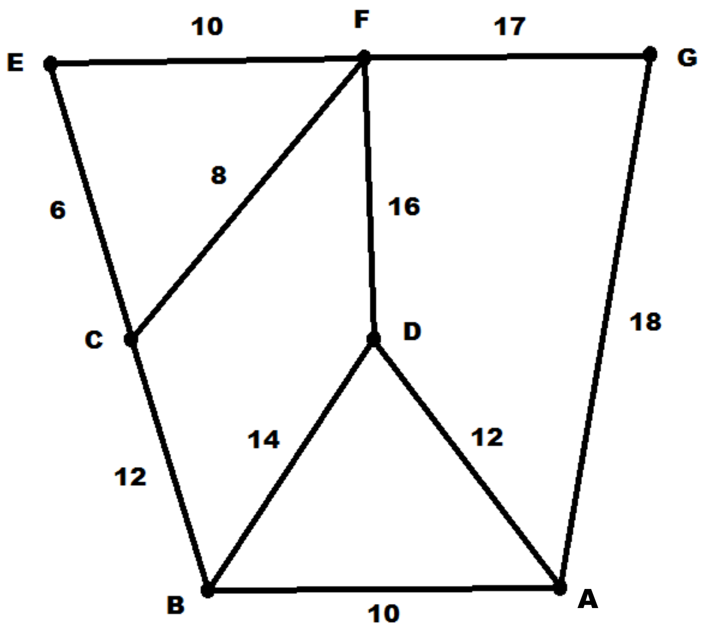
The use of electronic calculators is permitted provided they are not programmable and do not store text

[PTO]

1. Graph algorithms.

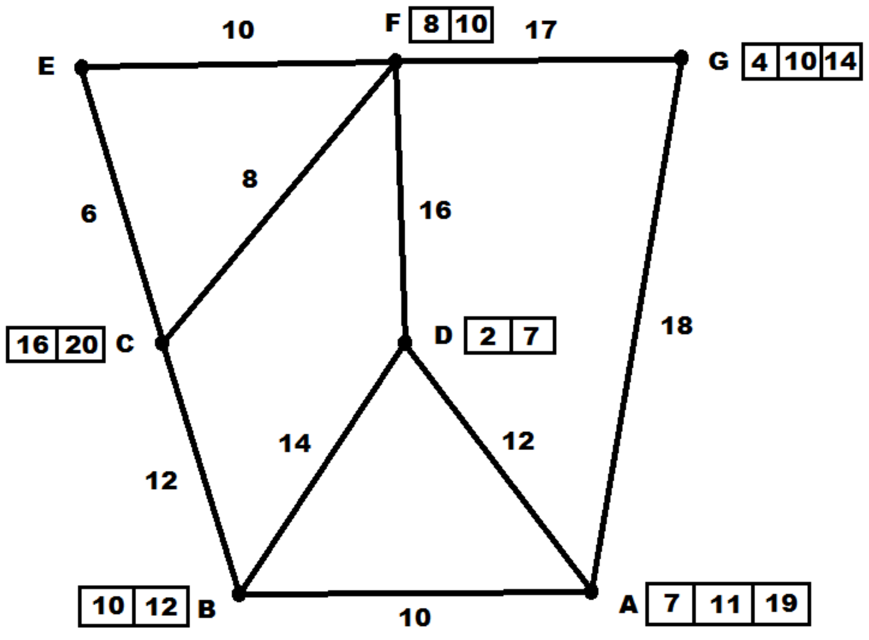
   a) Give a pseudocode description of Dijkstra's algorithm for finding the shortest path between a given node and all other nodes in a weighted undirected graph. Suggest how Dijkstra's algorithm can be generalised to find the shortest paths between all pairs of nodes. Find a modification of this approach which cuts the execution time by a half. In your answer assume that the priority queue and the functions operating on it are defined. (6 marks)

   b) Consider the following weighted undirected graph:



      i) The numbers on the edges denote distances. 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. (9 marks)

ii) Consider now the following graph.



Assume that a lorry driver has to drive from the town A to the town E, collecting goods that need to be delivered to E along the way. The weights w_i of the goods available for transport at each town (except E) are given in the boxes next to the town names in the above graph (for example, at A there are three items with weights 7, 11, and 19). The capacity of the lorry is C_0. The cost of running the lorry empty per unit distance is S_0, and the cost of transporting one weight unit per one distance unit is S_ℓ (thus, for example, transporting the item with weight 7 from A to B will cost $7 \cdot 10 \cdot S_\ell + 10 \cdot S_0$). The payout for each unit of load at the destination E is given only if the goods are transported using the shortest distance between the two towns. The payout is V_ℓ per one weight unit and one unit distance (thus, for example, the payout for transporting an item of weight 2 from D to E generates a payout of $2 \cdot (16+10) \cdot V_\ell$). It is assumed that $V_\ell > S_\ell$.

Propose a greedy heuristic based on Dijkstra's algorithm and the 0/1 knapsack problem, which attempts to maximise the profit of a lorry driver. You need give only an outline solution, and no pseudocode or program is expected. How would your proposed strategy change if the payout at E is a flat fee V_ℓ^* per unit weight? (5 marks)

2. Knapsack problems.

a) Describe the 0/1 Knapsack Problem, concerning a set of items and a container of limited capacity. (Do NOT describe methods for solving it). (4 marks)

b) Briefly explain the standard Greedy method for solving the 0/1 Knapsack Problem, including any preprocessing step needed. (3 marks)

c) Consider the following 0/1 Knapsack Problem instance, which is going to be solved using a Branch-and-Bound method. The items are already sorted in the order in which the Branch-and-Bound will consider them.

i	1	2	3	4	5
v_i	400	200	220	240	279
w_i	40	20	22	24	31
v_i/w_i	10	10	10	10	9

$C = 55$

i) What is the fractional upper bound of the partial solution represented by 10***, where 1 means definitely pack, 0 means definitely don't pack, and * means "don't know"? (Show your working.) (2 marks)

ii) Make a list of all the different *feasible* partial solutions that would have the same fractional upper bound as 10***. (4 marks)

d) The Integer Knapsack Problem is similar to the 0/1 Knapsack Problem, with the exception that for each item i there is given an integer $n_i \geq 1$ indicating the available number (or stock) of that type of item, as in the following example instance:

i	1	2
v_i	12	16
w_i	3	4
n_i	5	2

$C = 11$

A solution to the problem is a vector, giving the numbers of each item i to pack. A valid solution is one where the total weight of items packed does not exceed the knapsack capacity C, and the number of items of each type i packed does not exceed n_i. For example, a valid solution to the above problem instance is (3,0) with value=36 and weight=9. An optimal solution is a valid solution that maximises the value.

Find an optimal solution to the instance above using a dynamic programming approach. Give the full table of optimal values to subproblems, and explain how your method works. (7 marks)

3. Hashing and trees.

   a) Consider the following sequence of the keys:

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

   We wish to insert these numbers in the above order (from left to right) into a hash table of size 13 (with indices denoted from 0 to 12), using the hash function $h(x) = (5x+9) \bmod 13$. Show the insertion procedure, step by step, assuming in your answer that collisions are handled by linear probing. (8 marks)

   b) Consider the following set of keys:

   `4,7,10,2,6,9,1,3,8,5`

   i. Show the process, step by step, of inserting the above keys in the given order (from left to right) into an AVL tree. (7 marks)

   ii. Give a pseudo-code description of an algorithm `findElement(k,T)` for finding an element with the key `k` in an AVL tree `T`. What is the complexity of this algorithm? How does this complexity compare to that for the same operation when a binary search tree is used instead? (5 marks)

4. Trees and tree searching.

   a) Explain clearly what is meant by the following search strategies for *finite rooted trees*:

      i. Depth-first search (DFS), and
      ii. Breadth-first search (BFS)

      Illustrate your explanations with suitable examples. (4 marks)

      Give an example of a real problem where a DFS strategy is an appropriate choice, and another example where a BFS is an appropriate choice. Justify your answers. (2 marks)

   b) Now suppose we consider trees whose nodes are assigned numerical priorities. Explain clearly what is meant by a *priority search* over such a tree. Illustrate your answer with an example.

      How may priority searches be used to perform *heuristic searching*? (3 marks)

   c) A *priority queue* is a linear data structure storing items, each item having a numerical priority. It has the same operations as a queue (*push*, *pop*, *top*, *empty*), except that the *top* operation returns the item with highest priority, and the *pop* operation removes this item.

      i. Explain how you may use the operations provided by a priority queue to implement a priority search of trees.
      ii. Give a pseudocode description of the algorithm you presented in part (c)(i). (5 marks)

      What is a *heap* data structure? You should explain what the structure is, what the heap property is, and what operations are available, but you need not describe how the operations are implemented. How may a heap be used to implement a priority search? (2 marks)

   d) Consider the following puzzle, using words in the English language. The aim is to transform a starting word into a target word. Each step of the transformation consists of choosing a position in the current word and replacing the letter at this position with another letter from the alphabet to form another word in the English language. Thus if the starting word is `head` and the target is `feet`, then one sequence of steps is:

      ```
      head -> bead -> beat -> beet -> feet
      ```

      Notice that a replacement letter need not be in the target word. The aim of the puzzle it to perform the transformation in a minimum number of steps.

      Explain clearly how you would use a priority search over a tree to solve this problem. In particular, explain how a tree is generated, what the priorities are and how this is a *heuristic search*. (4 marks)

END OF EXAMINATION