COMP26120

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Tuesday 24th May 2011

Time: 14:00 - 16:00

Please answer THREE Questions
Use a SEPARATE answerbook for EACH Section

For full marks your answers should be concise as well as accurate.
Marks will be awarded for reasoning and method as well as being correct.

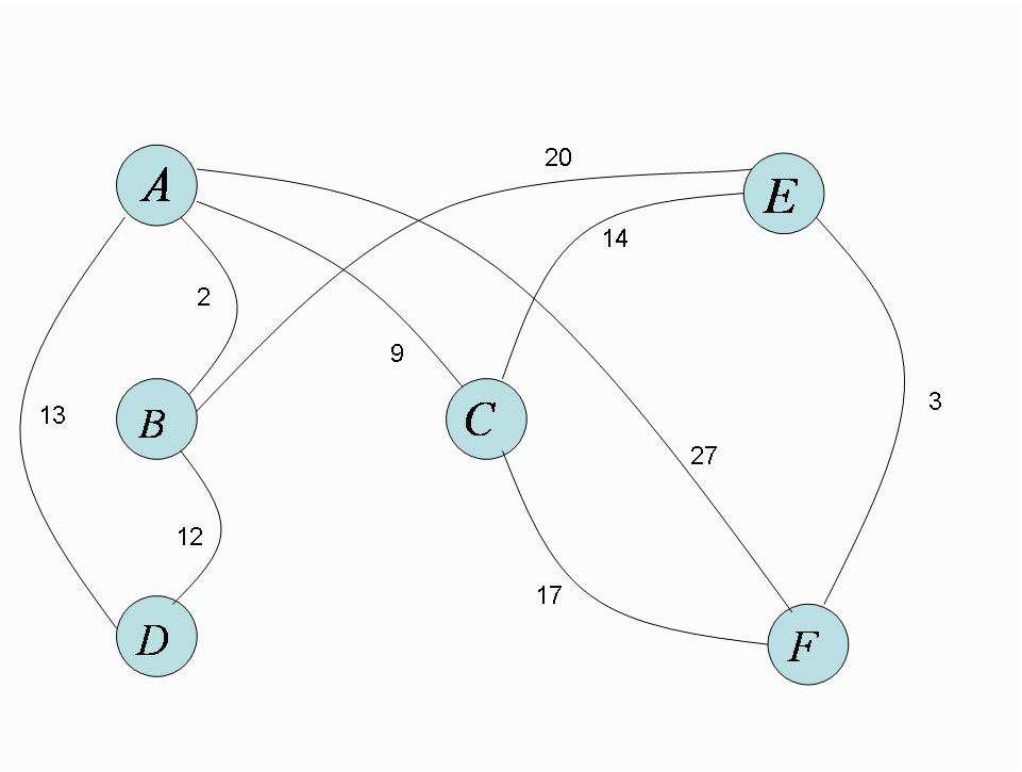
This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

Section A

1. Consider the following undirected weighted graph:

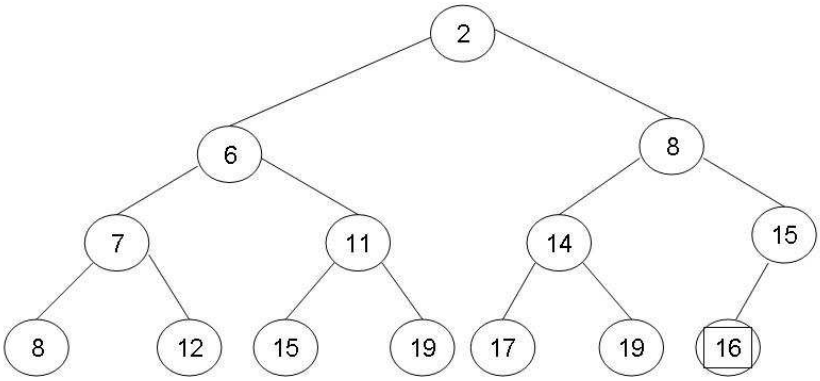


a) Give a pseudocode description of Dijkstra's algorithm for finding the shortest path between a given node and all other nodes in a graph *G*. Generalise this approach for finding the shortest path between all pairs of nodes in *G*. Assume in your answer that the functions `remove_smallest()`, `insert(int k)`, and `change(int k)` operating on a priority queue are given. (7 marks)

b) Show the progression of Dijkstra's algorithm, step by step, and draw the content of the priority queue at each step when finding the shortest path between the nodes *D* and *F* in the above graph. (7 marks)

c) Consider a heuristic algorithm for finding the shortest path between two nodes of a graph, based on the following rule: consider a node *n* and all the nodes (target nodes) linked to it by an edge. For each target node, we sum the weights of all edges incident at that node. Then we select the target node with the least sum as the next node to be visited from *n* . Compare the path found in the above graph between nodes *D* and *F* using this approach with that using Dijkstra's algorithm. What are the potential disadvantages of this particular heuristic approach? Under what circumstances is it appropriate, in general, to use a heuristic approach for path finding in graphs? (6 marks)

2. Consider the following heap, where the boxed node containing the element 16 is the last element of the heap.



a) Assuming a total order relation on the keys of a dataset is given (e.g. by a comparator), define the heap property of a binary tree.

(3 marks)

b) Show the steps of removing the minimal element from the heap given above. You should picture the heap at each step of the process. (6 marks)

c) Give the asymptotic time complexity of the following heap operations: `insert(a)`, `remove_min()`, and `replace(a,b)`. Your answer should be in terms of the Big O notation. (5 marks)

d) Give an efficient algorithm (using pseudocode or a clear step-by-step explanation) for finding all the keys in a heap that are smaller then, or equal to, a given key k. The value of k is not necessarily equal to any element in the heap. In your answer do not use the `remove_min()` operation. Your answer should leave the initial heap structure unchanged. What is the asymptotic time complexity of the proposed algorithm? Explain your answer. (6 marks)

Section B

3. Knapsack problems.

In a Knapsack problem, we are asked to pack N items into a knapsack of capacity (i.e. maximum allowable weight) C. Each item with index i has a weight w_i and a value v_i. An optimal solution to a knapsack problem is a set (or selection) of items (or fractions of items) that maximizes the total value in the knapsack, subject to the constraint that the total weight of the selected items must not exceed capacity C.

a) Explain clearly the difference between the fractional knapsack problem and the 0/1 knapsack problem. (2 marks)

b) Consider the following knapsack problem instance, in which the items are already sorted in decreasing order of their ratio of value to weight. The capacity (maximum allowable weight) C is 9.

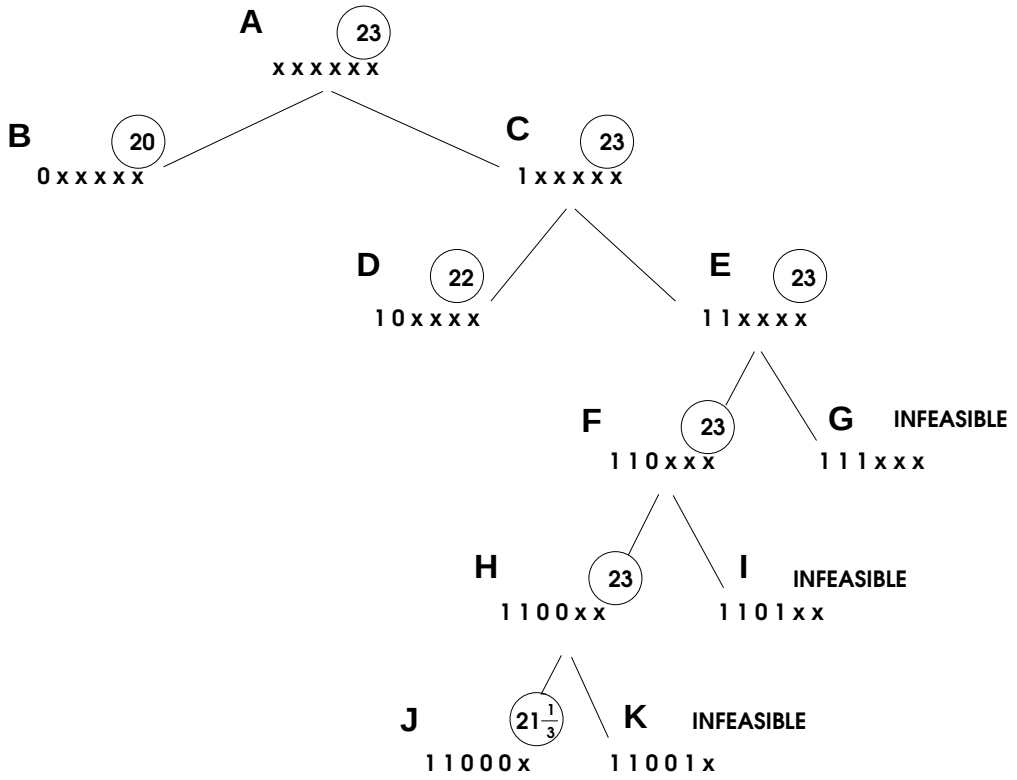
	Items					
Index i	1	2	3	4	5	6
Value v_i	12	9	12	6	4	2
Weight w_i	4	4	6	3	2	3
Ratio v_i/w_i	3	$2\frac{1}{4}$	2	2	2	$\frac{2}{3}$

i) What is the optimal solution to this problem when it is considered as a *fractional* problem? Your answer should indicate the amounts of each item taken and the total value. (2 marks)

ii) Explain the steps of the method you used to answer (b)(i) above. What is the time complexity of this algorithm, in general, including any sorting of the items that may be necessary? (3 marks)

c) Explain why the solution to a fractional knapsack problem instance is an upper bound to the solution of the 0/1 version of the same problem instance, i.e. the solution to the fractional instance has a value at least as high as the best value possible in the 0/1 instance. (3 marks)

d) A branch-and-bound algorithm is applied to the problem instance in part (b) above to find the optimal 0/1 solution. A tree of partial solutions that the algorithm has produced after several steps is shown below. The fractional knapsack upper bound value of each partial solution is indicated within a circle. The notation `1xxxxx` means that the first item is selected, and other items undecided, `0xxxxx` means that the first item is rejected and other items undecided. The algorithm is using a best-first tree traversal based on a priority queue.



i) Verify that node G represents an *infeasible* partial solution. (2 marks)

ii) Explain why the subtree rooted at node E is explored before that rooted at node D. (2 marks)

iii) Identify which node will be expanded next and explain why. (2 marks)

iv) Complete the computation of the optimal solution to this 0/1 knapsack problem. Give the final list of items and the total value. You do not need to show the other nodes of the tree you expanded. (2 marks)

v) Explain why the Branch-and-Bound algorithm terminates at this optimal solution. (2 marks)

4. Graphs and graph algorithms.

   a) Consider the following two representations of finite directed graphs

   - Adjacency lists
   - Adjacency matrices

   i) Explain clearly what these representations are.
   Choose an illustrative example of a directed graph with at least 4 nodes and show how it can be presented using these two representations. (3 marks)

   ii) Give ONE computational task on directed graphs that can be more efficiently undertaken using adjacency lists rather than adjacency matrices.
   Similarly, give ONE computational task on directed graphs that can be undertaken more efficiently using adjacency matrices rather than adjacency lists.
   In both cases you should explain clearly what the task is, what algorithm is involved and show why it is more efficient on the given representation by calculating the relevant time complexity measures (and giving the calculation).
   (4 marks)

   b) Explain what is meant by a Depth-First Search (DFS) and a Breadth-First Search (BFS) 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)

   c) For the case of finite *undirected* graphs, give an explicit algorithm for performing DFS of such graphs which numbers the nodes in the order that they are first encountered. You may either present a program or express the algorithm in pseudocode.
   (5 marks)

   d) A *spanning tree* in an undirected graph is a subset of the edges which forms a rooted tree and which includes all the nodes of the graph.

   Explain clearly why a DFS of a finite, connected, undirected graph produces a spanning tree. (4 marks)

END OF EXAMINATION