The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, FULL YEAR, 2016-2017

ALGORITHMS CORRECTNESS AND EFFICIENCY

Time allowed TWO HOURS

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer ALL FIVE questions.

Marks available for sections of questions are shown in brackets in the right-hand margin

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

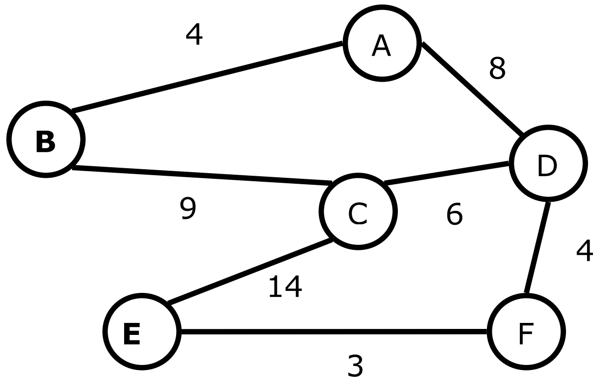
ADDITIONAL MATERIAL: none

INFORMATION FOR INVIGILATORS: Students should write all their answers in the answer book. The exam paper should be collected and placed inside the answer book.

1. This question concerns shortest paths in graphs. (20 marks total)

   (a) Explain, and give pseudo-code for, Dijkstra's algorithm to find the shortest path in an undirected graph when the distances for edges are given and all are non-negative. Your answer should include a brief explanation of why it always gives an optimal (shortest) path (if one exists).

   Then use Dijkstra's algorithm to find the shortest path from node **B** to node **E** in the graph below. Show your working, including the open and closed lists at each stage.

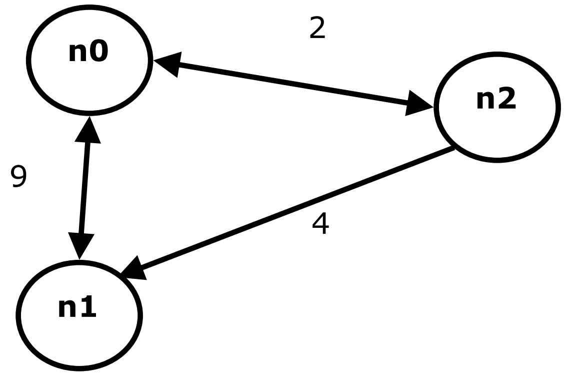


(10 marks)

   (b) Explain, and give pseudo-code for, the Floyd-Warshall algorithm to find the lengths of the shortest paths between each pair of vertices in a graph. The graph can have a mix of directed and bidirectional edges, but you can assume the distances are such that there are no negative cycles.

   Your answer should include a brief explanation of why it always gives an optimal path for each pair of vertices.

   Then use the algorithm on the graph below to find the matrix of all-pairs of shortest paths. Show your working.



(10 marks)

2. This question is concerned with string matching algorithms. (20 marks total)

Consider the algorithm `LastMatch` below, which returns the offset (shift) of the *last* occurrence of the pattern P in text T, or -1 if P does not occur in T:

```
LastMatch(T,P)
   for(s = T.length - P.length downto 0)
     j = 1
     while(j =< P.length and P[j] == T[s + j])
       j++
     if(j == P.length + 1)
       return s
   return -1
```

(a) State a suitable precondition for `LastMatch` in English, and a suitable postcondition in predicate calculus.

(6 marks)

(b) State suitable loop invariants for the **for** and **while** loops in LastMatch in predicate calculus. For the *outer* **for** loop invariant, explain how initialisation and maintenance of the invariant may be established.

(9 marks)

(c) In automata-based string matching, a string matching automaton is constructed that recognises the pattern. Draw the automaton that recognises the pattern *P* = abac, where the input alphabet is Σ = {a b c}. (Note: you don't need to show failing edges that return to state 0.)

(5 marks)

3. This question is concerned with the 'big-Oh' family and recurrence relations. (20 marks total)

   (a) Give the definitions of big-Oh, big-Omega, big-Theta, and little-oh by completing each of the following:

      i) 'big-Oh': f(n) is O(g(n)) ….

      ii) 'big-Omega': f(n) is Ω(g(n)) ….

      iii) 'big-Theta': f(n) is Θ(g(n)) ….

      iv) 'little-oh': f(n) is o(g(n)) ….

   (5 marks)

   (b) Give appropriate descriptions in terms of each of `O' (big-Oh) and `Ω' (big-Omega) of the function

   $$f(n) = 2 n^2 + 5 n \log(n^4)$$

   You should justify your answers, but do not need to give proofs. (5 marks)

   (c) Consider the function

   $$f(n) = 2 n + (n^2 \% 10)$$

   where "n^2 % 10" uses the usual modulus operator, e.g. 7^2 % 10 = 9, etc.

   From the definitions, prove or disprove that f(n) is O (n) (big-Oh)

   (5 marks)

   (d) Consider the recurrence relation

   $$T(n) = 3 T(n/2) \qquad \text{with} \quad T(1) = 1$$

   Solve the relation exactly.

   Hint:
   Compute T(2), T(4), etc., and use this to find the answer for T(2^k).
   Then prove your answer is correct using induction.

   (5 marks)

4. This question is concerned with MSTs in weighted undirected graphs. (20 marks total)

(a) Define a Minimum Spanning Tree (MST) for a weighted undirected graph.

(4 marks)

(b) Briefly describe Prim's algorithm for finding an MST.

(6 marks)

(c) Give an argument that Prim's algorithm is correct, i.e. that it will always find an MST.
Hint: consider partitioning the nodes into two sets V1 and V2 and consider the edges between them.

(5 marks)

(d) Suppose that all the weights on the graph are different. I.e. that no two edges have the same weight. In this case argue that the MST is unique.
Hint: consider the partition used in part c.

(5 marks)

5. This question is concerned with sorting algorithms. (20 marks total)

   (a) Explain, and give pseudo-code for, the mergesort algorithm to sort an array A[ ] of integers into ascending order.

   Show the working of the algorithm on the array A = [4 8 9 2 1 7 3 6]. Give the tree representing the execution of the algorithm.

   State the worst case runtime of mergesort, of an array of n entries, in big-Oh notation, and give a justification of your answer.

   (9 marks)

   (b) Explain, and give pseudo-code for, the quicksort algorithm to sort an array A[ ] of integers into ascending order.

   Both mergesort and quicksort are in the divide-and-conquer class of algorithms, but differ in their approach to this. Briefly, describe how they take a different approach to the 'divide' and to the 'conquer' steps.

   (7 marks)

   (c) Suppose that the pivot is selected randomly from within the array. Briefly outline a proof that the resulting average case complexity is O(n log n)

   (4 marks)