PRIFYSGOL CYMRU; UNIVERSITY OF WALES

M.Sc. AND DIPLOMA EXAMINATIONS

May/June 2002

SWANSEA

Computer Science

CS M22 Algorithms and their Applications

Attempt 2 questions out of 3

Time allowed: 2 hours

Students are permitted to use the dictionaries provided by the University

Students are NOT permitted to use calculators

CS M22
ALGORITHMS AND THEIR APPLICATIONS

(Attempt 2 questions out of 3)

Question 1

(a) Let $(D, <)$ be a linearly ordered domain.
Consider sorted arrays $A = A[1 \dots N]$ over $(D, <)$.

(i) Describe the principle underlying binary search, with emphasis on how it exploits – and relies on – the fact that the input A is sorted.

(ii) Sketch pseudocode for recursive procedures $\text{SeqSearch}(A, p, q; d)$ and $\text{BinSearch}(A, p, q; d)$ for searching the sorted array $A[1 \dots N]$ in the search interval $[p, \dots, q] \subseteq [1, \dots, N]$ for a target value $d \in D$.

(iii) Describe the complexities of BinSearch versus SeqSearch, in terms of the length of the search interval $n = q - p + 1$.

[9 marks]

(b) Sketch the main stages in sorting the array

$$A = (\mathrm{g}, \mathrm{o}, \mathrm{o}, \mathrm{d}, \mathrm{l}, \mathrm{u}, \mathrm{c}, \mathrm{k})$$

with respect to the alphabetical ordering of letters using

(i) InsertionSort. (Stages after each completed insertion loop.)

(ii) HeapSort. (Tree lay-out after completion of each main stage of the transformation into a heap, and in the main iteration of HeapSort.)

Briefly explain the sorting strategy in HeapSort in words.

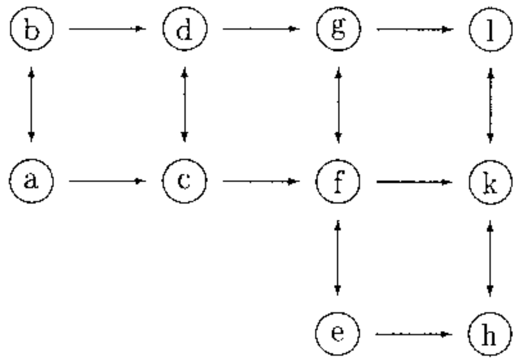
[10 marks]

(c) (i) State precisely in which sense the growth rates of n^2 and $n \log n$ characterise the complexity of InsertionSort and HeapSort, respectively.

(ii) Which of InsertionSort or HeapSort is to be preferred in the following circumstances, and why?
Some long sorted array A is to be maintained as a sorted array and repeatedly needs to be re-sorted in response to updates on individual array entries.

[6 marks]

G



Question 2

(a) Consider the directed graph G depicted above.

(i) Give its adjacency list representation with respect to the alphabetical enumeration of its vertices.

(ii) Indicate the sequence in which a run of BFS(G, a) discovers the vertices of G, give a table of the final assignments to the auxiliary functions d and π, and draw the resulting BFS tree.

(iii) Account for the main steps in a run of DFS(G), by indicating in a diagram of the graph the time stamps, and marking those edges along which vertices are discovered. Give a table for the final assignments to the auxiliary functions π, t_1 and t_2.

[**12 marks**]

(b) For each of the following decision problems indicate why they are feasibly solvable, giving reasonable bounds on the complexity of feasible algorithms where possible.

(i) REACHABILITY: given a directed graph $G = (V, E)$ and two vertices $s, t \in V$, decide whether t is reachable from s (i.e., $s \rightsquigarrow t$).

(ii) DISTANCE: given a directed weighted graph (G, ω), two vertices $s, t \in V$, and a number $m \in \mathbb{N}$, decide whether t is reachable from s on a path of weight less than m (i.e., $\delta(s, t) < m$).

(iii) 2-COLOURABILITY:
given an undirected graph G, decide whether G has a 2-colouring.

[**7 marks**]

(c) Discuss the significance of the following two decision problems. This involves brief characterisations of the problems, as well as short explanations of the relevant concepts of computability and complexity involved [like computability, feasibility, NP-completeness, reductions, ..., as appropriate].

(i) the Halting Problem.

(ii) the 3-Colourability Problem.

[**6 marks**]

Question 3

Consider Dijkstra's algorithm for finding shortest paths from source vertex s in a weighted directed graph (G, ω).

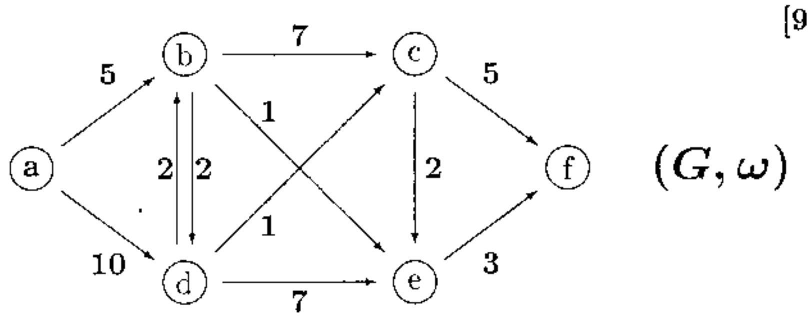
(a) (i) Give the definition of the distance function $\delta(u, v)$ induced by ω, and define the notion of a shortest path from u to v.

(ii) Explain the notion of a shortest paths tree as generated by Dijkstra's algorithm.

[7 marks]

(b) Display the main intermediate stages (assignments to π and d) during a run of DIJKSTRA(G, ω, a) on the weighted graph (G, ω) below, with source **a**. Also draw the resulting shortest paths tree.

[9 marks]



(c) (i) Give pseudocode for the relaxation procedure in Dijkstra's algorithm and explain its role: how is RELAX(u, v) used to possibly improve the current value of $d[v]$? [Draw a sketch to explain.]

(ii) Why is it essential that the sequence of relaxation steps is determined by a priority queue with respect to d-values?

(iii) Explain how DIJKSTRA is feasible, as opposed to a brute force algorithm that would inspect all (loop-free) paths from s to u in order to find $\delta(s, u)$.

[9 marks]