UNIVERSITY OF CAMBRIDGE

ENGINEERING TRIPOS PART IIA

Monday 26 April 2004 2.30 – 4.00

Module 3I1

DATA STRUCTURES AND ALGORITHMS

*Answer **all** of Section A and **two** questions from Section B.*

All questions carry the same number of marks.

*The **approximate** percentage of marks allocated to each part of a question is indicated in the right margin.*

There are no attachments..

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

(TURN OVER

SECTION A

Attempt all parts of this question

1. You should answer each of the following parts with a short-answer, quoting relevant results rather than proving them. Give names for algorithms you need to refer to and sketch methods or justifications of claims you make. Credit will be given for the clarity and succinctness of answers as well as for basic factual accuracy.

(a) Is it the case that every function of the form $f(n)=An^k$ (with A and k constant) satisfies $f(n)=O(2^n)$? [10%]

(b) G is a connected graph with n vertices and with a weight associated with each edge. F is a sub-graph of G consisting of the n-1 shortest edges in G. Is F necessarily a tree and does it necessarily span the vertices of G? Explain. [10%]

(c) A tree has a fan-out of 5. In other words each node in it has just five nodes as its immediate offspring. Every leaf-node in the tree is at the same distance, d, from the root, and d is fairly large. About what proportion of the nodes in the tree are leaf (rather than internal) nodes? [10%]

(d) Once a pivot has been selected a key part of ***quicksort*** involves partitioning the original data into a set of items less than (or equal to) the pivot and another set of items that are all greater than the pivot. Explain how to achieve this when the original values are stored in an array and you do not have any additional workspace to use. Ensure that your method will work in extreme cases, such as when all the values are the same, or when the pivot value is the largest or smallest value in the array. [10%]

(e) Explain the concept of a ***winding number*** and show how it can be used to determine if a point is inside a polygon. Are there extra issues if the polygon is concave? [10%]

(f) What is a ***garbage collector*** and what problem does it solve? Give a very brief sketch of the actions involved in some simple form of garbage collection. [10%]

(CONT.

(g) Why is it often said that sorting n values must necessarily cost $n \log(n)$ steps? Are there any cases where this assertion is false? [10%]

(h) Suppose that a priority queue is to be implemented. The normal implementation would probably be to use a heap, but for some reason a programmer has tried to use ***red-black trees***. How can the smallest item in the queue be found? Starting from an empty queue how much work is required to build a queue with n items in it? [Note that you are not required to discuss the process of removing an item from the queue.] [10%]

(i) What problem do ***splay trees*** address and what particular advantages might they have over other competing technology? [10%]

(j) Why is it not possible to say that ***simple insertion sort*** has cost $\Theta(n^2)$? Would it be safer to say it had cost $\Theta(n \log(n))$? [10%]

SECTION B

2. (a) A complete compression algorithm must use some statistical or predictive model to expose redundancy in the original input, and some packing procedure that exploits this information to create a bit-compact representation of the data. Explain how the Burrows-Wheeler transform works and what it achieves in this context. [30%]

(b) The output from the Burrows-Wheeler transform is generally expected to be readily compressible using a combination of run-length and Huffman encoding. Give a brief explanation of these two techniques and how the corresponding decoders work. You do not need to give elaborate details of how to construct a Huffman encoding – just an overview. [20%]

(c) Explain why it is possible to reverse the Burrows-Wheeler transform and hence recover the original data from its compressed form. [30%]

(d) A normal explanation of Burrows-Wheeler treats it as working with data in the form of blocks of characters. Discuss whether it would be possible to use the same procedure but to treat the blocks as blocks of bits rather than characters. Comment on whether this, if possible, would seems a good idea. You may wish to consider complexity of implementation, cost of performing a compression and any predictions about how the change would impact the ability of the algorithm to predict patterns in the data. [20%]

3. (a) Explain the general idea of a hash function. If the data you are working with involves strings of characters, suggest a scheme that might be used to compute reasonable hash values. [25%]

(b) How do you use hash functions to implement a table where you can record key-value pairs and access stored data efficiently? If your hash table has space for a maximum of n records in it but only k are in use at some stage estimate (assuming that the hash function you use has perfect statistical properties) the number of table accesses (probes) needed when inserting a new entry into the table. [25%]

(c) What is Larsen's method of dynamic hashing, and how can it be used to organise data stored on disc? Why might it be used? Explain how new data can be added and existing data retrieved using Larsen's method. [25%]

(d) Describe a method for using hashing to speed up the search for a particular target string in a very long sequence of characters. Discuss its typical and worst-case costs. [25%]

4. The normal implementation of a heap keeps data in an array, and a node at offset k in the array is interpreted as having its two child-nodes at offsets $2k$ and $2k+1$. The tree representation this gives is always very well balanced and so, if there are n items in the heap, the height of the tree is $\log_2(n)$. The main operations on heaps have costs that are proportional to the height of the tree.

(a) Show how to interpret offsets in an array to represent a balanced tree with fan-out 4 rather than 2. [10%]

(b) Explain the relationship between the height of your new 4-way branching tree and the original 2-way branching one. [10%]

(c) Since heap operation costs depend on the height of the tree, your new scheme might have different costs from the original binary heap. Describe algorithms to rearrange arbitrary initial data so they satisfy the heap property and to remove the top item from your 4-heap. Discuss the cost growth rates and implications of your procedures and compare them with the normal implementation of heaps. [80%]

END OF PAPER