ENGINEERING TRIPOS PART IIA

Wednesday 5 May 2010 9:00 – 10:30

Module 3I1

DATA STRUCTURES AND ALGORITHMS

*Answer **all** of Section A (which consists of short questions) 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.*

STATIONERY REQUIREMENTS	SPECIAL REQUIREMENTS
Single-sided script paper	none

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

Version: final

SECTION A

Answer all parts of this question. The question in this section will be marked out of the same total as each question in section B, and each part carries the same weight.

1 (a) The expected cost C of accessing an item in a hash table can be expressed as

$$C = \frac{1}{2}1 + \frac{1}{4}2 + \frac{1}{8}3 + \ldots$$

What is the value of this series? [10%]

(b) Two bad people are constructing data intended to give pain to your implementation of Larsen's Dynamic Hashing. One of them has found a set of keys so that when you apply any of your first series of hash functions (controlling which disc block will be used) only a small number of distinct hashes arise. The other has selected keys so that you only ever get a small number of distinct signature or priority values. Which will disrupt you more rapidly? [10%]

(c) How do you expect the best, worst and typical costs of allocation under first-fit and best-fit to compare? [10%]

(d) The course covered 2-3-4-trees. One thing that it explained was that when necessary a 4-node could be split into two 2-nodes. One could perform that split every time a 4-node was about to arise, ending up with a tree consisting of only 2-nodes and 3-nodes. Using big-O notation explain what the cost of looking up data and inserting new data is liable to be. You do not need to give full details of algorithms — just enough overview to make it clear that you could construct them if necessary. [10%]

(e) It is commonly said that Huffman codes can waste half a bit for each symbol transmitted. Why is this? Give a small example to illustrate it and name a compression scheme that does not suffer from this waste. [10%]

(f) What is a minimum spanning tree? If all the edges in a graph have different lengths will any minimum spanning tree be unique? [10%]

(g) Why might one modify a simple implementation of Quicksort by selecting a random element of the array at each stage to be the pivot? Why might one terminate Quicksort's processing before it manages to split data into partitions of size at most 2? [10%]

(h) You have a binary search tree, and a pointer to one particular item stored in it. Explain how you would find the item in the tree that comes just before your selected one in terms of the sorting order, or determine that there is no such element. [10%]

(i) What operations must be specified to describe an Abstract Data Type for a "stack". What merit is there in introducing abstract data types rather than just using concrete data structures? [10%]

(j) On current real computers memory access times depend strongly on the extent to which an algorithm has a pattern of memory use that exhibits locality. Quote examples of algorithms that do this and hence run well, and ones that do not and hence behave less well than one might have hoped based on just an asymptotic analysis. [10%]

SECTION B

Attempt ***two*** *questions from this section. Each question has the same weight in marks.*

2 Suppose that a positive integer is represented in binary, with the binary number then stored as a list with the least significant bit as the first item in the list. Thus the number ten is `1010` in binary and so would become the list `[0, 1, 0, 1]`. Find the maximum and minimum costs of incrementing a number that is represented this way. If m and n are two integers that are roughly the same size what is the cost (in big-O notation) of adding them? [10%]

Start with an integer 1 represented as above (i.e. as a list) perform a long sequence of increment operations. Find a formula for the average cost of the increments. [20%]

A special variant on a tree data-structure has the following properties:

(a) There are trees that can hold 1, 2, 4, 8, $\ldots 2^n$ items, but no other size.

(b) In any tree the root element will hold the smallest key.

(c) It is possible, in O(1) time, to take two trees each of size 2^i and combine them to form a tree holding 2^{i+1} items.

If you have k items to store you can represent k in binary and thus store your data in a list of these special trees. Explain how to add a single item to a list, preserving the property that it never contains two trees of the same size. Give the amortised cost of your method. [25%]

Show how to merge a list containing m keys and another with n to make a single list of size $m+n$. What is the cost? [25%]

Suppose that you can take one of the special trees containing 2^n keys and in O(1) cost you can remove the smallest key (i.e. the one at the top). You have $2^n - 1$ keys left and these will be returned to you as a list of trees. Given this operation, what algorithm can be used and at what cost if you start off with a list of trees containing k keys in all and you want to remove the smallest key? [20%]

3 Why is it normally said that the minimal cost of sorting n items is proportional to $n\log(n)$? [10%]

Give *two* special cases where you could propose an algorithm that would sort in linear time supposing your particular condition applied, Explain why these examples do not undermine the usual $n\log(n)$ prediction. [20%]

Now suppose you have an ideal parallel computer that can perform several comparisons and associated bits of data movement at once. But if the selection of what data is to be compared depends on the result of another comparison then the two comparisons must happen one after the other rather than at the same time.

Describe Heapsort and analyse its performance if adapted to exploit parallel operations. You may assume that administration and synchronisation of the various parallel operations is not a problem, and that having multiple operations all accessing the array that holds the data does not lead to any slow-down. What performance can be achieved and about how many operations must happen at once? How does your result relate to the "minimum cost of sorting"? [45%]

An alternative parallel sort scheme, potentially suitable for use in dedicated hardware, is to bring in each value to be sorted on its own wire. The basic hardware unit has two wires that go in (say a and b), and it transmits the two input values on a pair of output wires (A and B) but now with their positions swapped if necessary so that the larger value is sent on output A.

If the items to be sorted are $(v_1, v_2, v_3, \ldots)$ then a first row of comparison units processes the pairs (v_1, v_2), (v_3, v_4) and so on and a second row follows up on (v_2, v_3), $(v_4, v_5), \ldots$

If rows like this are merely stacked to get a grid of hardware that guarantees to fully sort its input how many comparators are needed in all? [10%]

Explain how ideas from Shell's software sorting method might improve on this scheme by both reducing the amount of hardware needed and shortening the delay between when data is initially presented and the final sorted data becomes available. [15%]

4 Define the height of a tree as the maximum distance from its root to a leaf. Define the balance of a tree as the difference between the height of its left and right sub-trees. Thus a tree where the tree itself and all its subtrees all have a balance of zero is a perfect binary tree.

If a tree where the balance is always zero has height h how many items are there in the tree? [5%]

Suppose that every position in a tree has a balance that is -1, and that its height is h. Find a recurrence formula that gives the number of items, $N(h)$, in the tree. [10%]

For large h this recurrence has a solution that grows as $N(h) \approx \lambda^h$ for some value of λ. Substitute that approximation into the recurrence and hence find the value of λ. [10%]

The course material on splay-trees included the following simple rotation that can be applied to a binary tree:

```
      Y              X
     / \            / \
    X   c   ==>    a   Y
   / \                / \
  a   b              b   c
```

Suppose that the node Y starts off so it has a balance of 2 (ie its left sub-tree is 2 taller than its right sub-tree) but all other nodes (including X) have a balance of $+1$, 0 or -1. Find the balance of the tree after the rotation for the three possible values of the balance at X. Identify cases when this does or does not improve the overall balance of the tree, allowing for the possibility that node Y might not be the root of the whole tree. [15%]

In any case where the balance would remain worse than 1 check the effect of performing a rotation on the tree that starts at X before doing the one at Y and document what can be achieved. [30%]

You have a tree that starts off with every node having balance $+1$, 0 or -1. Show that on adding a new item to it you can use rotations to restore its balance to the same range. How many rotations might you need to make? [20%]

Does this scheme lead to better or worse worst-case imbalance then Red-Black trees? [10%]

END OF PAPER

Version: final