ENGINEERING TRIPOS PART IIA

Monday 28 April 2008 2.30 to 4

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

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) A simple hash table of size N has k values stored in it. You are provided with a function f that could take one of the values as stored and map it onto an integer. Using big-O or big-Theta notation as relevant explain how long it would take to discover whether there are two values, say x and y in the table such that $f(x) = f(y)$. [10%]

(b) It is well known that the ratio between the maximum and minimum possible depth for a 2-3-4-tree holding N items is close to 2. What is the corresponding ratio for a form of B-tree that where each node can have a fan-out somewhere between 256 and 512? [10%]

(c) A set of items are held in a linked list. The list is (already) sorted, and there are n items. What are the best, worst and average costs of looking up an item that is already present in the list, and what are the costs of verifying that an item is not present? [10%]

(d) Give and explain an example of an algorithm where the use of randomness provides a benefit. [10%]

(e) Explain precisely the meaning of the big-O and big-Theta notations. [10%]

(f) Construct and explain a small example sequence of allocate and free operations where best fit would lead to noticably less fragmentation than first fit. [10%]

(g) Explain what is meant by the amortised cost of operations on a data-structure. Contrast it with other ways of predicting performance, showing one case where it is useful and one where it is not. [10%]

(h) In bad cases Quicksort can be very much more costly than Heapsort. Why, then, is it still very often used? [10%]

(cont.

(i) What expectation motivates the concept of Ephemeral Garbage Collection? Describe a scenario where it will not deliver benefits. [10%]

(j) Explain how to reinstate the heap property if the one of the values in the heap is changed. What is the cost? Does it matter whether the value is increased or decreased? [10%]

(TURN OVER

SECTION B

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

2 A monitoring device needs to transmit an unending series of status messages to home base. Each message is either an X or a Y, and there is strong reason to expect both that all messages are independent and that in the long term X will occur twice as often as Y. Information is to be transmitted as a stream of bits, and data compression is to be used to reduce the amount of transmission necessary.

(a) Why would one normally expect Arthmetic Coding to be better than Huffman Coding in this situation? [10%]

(b) The design engineer responsible for this project is really keen on Huffman Coding, and so wants to use it. The engineer arranges to group the status messages into clumps of three, eg such as XYX. Huffman Coding is now applied using as its alphabet all the possible strings of 3 original symbols. Does this improve things? If so predict the extent to which it compresses data, and if not explain why. Would using groups of more than three change your analysis, so for instance would using clumps of 20 or 50 letters make sense? [40%]

(c) A second engineer looks at the problem and adapts Lempel Zif to the case where its initial alphabet consists of just the two letters X and Y. Explain how Lempel Zif works, illustrating its behaviour using the string XYXXXXXXYXYXYXXX. Explain whether it is liable to give reasonable compression. [25%]

(d) Both coding and decoding Lempel Zif are made more complicated by the need to update internal tables as you go. To remove this overhead one could try running the full process over an arbitrary sample of a list of X/Y symbols of length say 8000, then freezing the coding tables and using them as static tables permenantly built into the monitoring device. Explain roughly what collection of entries you would expect to find in the table used by Lempel Zif in such a case, and discuss whether you could propose hand designed fixed coding that would do well. [25%]

(cont.

3 A hash table is built using an array that has N slots, and each slot has a sub-table that stores all information that hashes to that location. If just 2 values are put into the table there is only one way a collision can occur, and so (if the hash functions used are good) there is a chance of $1/N$ of a collision, and putting that another way the expected number of pairwise collisions is $1/N$.

(a) If k items are inserted how many ways are there in which collisions can arise? Hence predict the expected number of collisions. [15%]

(b) Based on the result above, suggest how N must grow as a function of k so that you have a reasonable chance, say around $\frac{1}{2}$, of having no collisions at all. [25%]

(c) Explain what a Universal Hash function is, especially as it can be applied to constructing hash values for strings. [25%]

(d) A variant on the idea of hashing is set up as follows, using two levels of hash table:

(i) When a known set of k keys are to be used a master hash table is set up with a size that is close to k. Universal hashing is used, and half a dozen different sets of numbers are used with the universal hashing method, with the one of these that gives the most even distribution of keys to table entries being the one that is used;

(ii) Each entry in the table may need to store several values (when the first hash table suffered from collisions). But typically not too many. Each entry then contains a set of parameters for a second universal hash function, together with a small hash table based on this. By choice of the hash function and table size it is arranged that this second hash table does not have any collisions at all.

Discuss the performance and limitations of this scheme. Can you explain why the designer of the method allowed each sub-table to define its own universal hash function rather than just selecting a single one for use with all of them? [35%]

(TURN OVER

4 (a) Explain the structure of a Binary Search tree. If such a tree contains *n* nodes what are the best and worst cases for its height? Explain how you would look up a key to find it in such a tree. [25%]

(b) Given a tree and an item in it show how to find the key that has the next higher value. What are the best and worst-case costs involved? If you start from a key *a* and finding its successor *b* has almost worst case cost is it possible for finding the successor of *b* to be a bad case as well? [25%]

(c) Given a binary seach tree and a key that is present in it explain an efficient procedure that updates the tree so as to delete the given key from it. Indicate expected and worst-case costs for your procedure. [25%]

(d) Given a binary search tree and a key that is present in it describe a procedure that rearranges the tree so that the specified key is in the root of the updated tree. Is this always possible? Explain the costs involved. [25%]

END OF PAPER