ENGINEERING TRIPOS PART IIA

Wednesday 25 April 2012 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.*

There are no attachments.

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

FINAL version

SECTION A

1 (a) Give one set of circumstances in which you might prefer Quicksort to Heapsort, and one in which you might prefer a simple selection sort to Mergesort. Briefly justify your answer. [10%]

(b) Lempel-Ziv style data compression involves detecting when a pair of symbols has arisen before, and then replacing them with a single symbol. What technology would be appropriate to make maintaining the table of symbol-pairs efficient? [10%]

(c) If you have a limited total amount of memory and your program will fill about 40% of it with active data, would you expect a stop-and-copy or a mark-and-sweep garbage collection strategy to be more suitable? [10%]

(d) A team of programmers have a graph and want a minimum spanning subtree. They are worried because the graph has some edges with negative weights. So they increase the weight on each edge by an amount W such that all weights become positive. They then find a spanning subtree in the adjusted graph. Will it be minimal in the original graph? [10%]

(e) Why is it generally said that sorting is an $n\log(n)$ process? Give and explain a case where sorting can be performed more rapidly than this. [10%]

(f) When you create a splay tree by adding in all the items in ascending order the "tree" ends up totally imbalanced. If n items have been added, it has depth n. You then wish to look up the item you had inserted first and find that the cost is severe. In the light of this worst case, why would anybody ever use splay trees? [10%]

(g) If algorithm A has cost $O(n^2)$ and algorithm B has cost $O(n^3)$ is it possible that algorithm B always runs faster in every practical case? Is it possible that algorithm B is always faster even in hypothetical or impractical cases? [10%]

(h) Why can you expect that Arithmetic Coding is capable of better data compression than using Huffman Codes? [10%]

(i) A team of programmers have a graph and want to find the shortest path between two particular vertices A and B. They are worried because the graph has some edges with negative weights. So they increase the weight on each edge by an amount W such that all weights become positive. They then find a shortest path in the adjusted graph. Will it be minimal in the original graph? [10%]

(j) Why is the guaranteed linear cost median finding algorithm hardly ever used? What is the worst case cost of the Quicksort-based scheme that usually (but not always) delivers linear cost? [10%]

SECTION B

2 In an experiment aimed at generating a sequence of pseudo-random integers a programmer generates a sequence x_1, x_2 ... where each value is derived just from the previous one. In other words the basis of the generator is a function f such that $x_{i+i} = f(x_i)$.

Of course once any value recurs in the sequence everything after that will be a repeat. In evaluating different functions f it is important to detect when this happens and how long the repeating loop of values is. Note that a sequence need not return to its very first value – a loop can arise later on as in the example 1,2,3,4,5,6,4,5,6,4,5,6,....

(a) A simple algorithm for detecting loops stores all values in the sequence, and when a new item x_n is generated it will compare against everything, thus detecting a cycle at the very earliest possibly stage. Using big-O notation, characterise the time and store costs of this simple algorithm in terms of when the sequence under test starts to repeat. [15%]

(b) An alternative scheme inserts each value x_i into a hash table as it is generated. It is not known how long it will be before a cycle is detected, and so initially a hash table with 256 entries is used. If the table becomes more than 80% full it is abandoned and the entire process is re-started from x_1 but using a new hash table twice the size of the previous one. What are the costs in this case? You may assume that the hash tables behave well without an excessive number of collisions. [25%]

(c) As a third attempt the programmer uses the hash table scheme as above, and if the hash table becomes too full they abandon it and create a new empty hash table of twice the size. However unlike the previous version they do not go back and re-start the sequence from x_1, but merely continue checking values x_k, x_{k+1} ... where x_k was the value under consideration when the hash table became full. Will this still even detect loops? If so how promptly and at what cost in time and memory? [25%]

(d) What are the consequences for the hash-table-based methods if unfortunately the particular sequence used interacts badly with the hash function and there are worst-case hash collisions? [10%]

(e) To avoid the need for hash functions and all that uncertainty, but building on the methods above, the programmer records x_2 and compares x_3 against it. Then they record x_4 and that is used as a basis for comparison with $x_5 \ldots x_7$. Doubling the index again, x_8 is then saved so that the sequence as far as x_{15} can be checked against it. In general at each stage a value of x_n is stored, where n is the most recent power of two passed. This can be seen as related to the idea of doubling the size of the hash table that has previously been used, but now there is no hash table at all, merely a single saved value. In what circumstances can this final scheme detect cycles? In a case when it can, the first repeat occurs after N items in the sequence and the loop is then of length M. Find a bound on how long it may take the algorithm to notice the cycle. [25%]

3 (a) Standard Red-Black trees can be viewed as an interpretation of 2-3-4 trees and they require just one extra bit in each (binary) node to mark that node as Red or Black. What is the greatest possible ratio between the height of the left and right subtrees of a Red-Black tree? What is the greatest depth of a Red-Black tree that has N items stored in it? [15%]

(b) B-trees generalise 2-3-4 trees by allowing a node to store a larger number of values. Explain the structure of a B-tree where each node can have between 4 and 8 descendants, and explain both how existing keys can be searched for and new data added. Are there any special cases that arise for either very large or very small trees? [25%]

(c) An engineer decides to design a family of binary almost-balanced trees, starting from 4-8 trees using the same ideas that led to the derivation of Red-Black trees from 2-3-4 trees. Explain the structure that could be used to represent each sort of node. Is it possible to still use just two colours or will more be required? [20%]

(d) Explain how to look up an item in the new style of tree described in part (c). How will its worst-case compare with the worst case that can arise using standard Red-Black trees? [20%]

(e) The most complicated transformation on a 2-3-4 tree is when a 4-node has to be split into a pair of 2-nodes. Correspondingly on a 4-8 tree it will be when an 8-node is split into two 4-nodes. Compare the frequency with which these operations arise when each sort of tree is built with N items. [20%]

4 A programmer wishes to implement a "best fit" storage allocation scheme. It must provide for allocating a new block of memory of requested size n, and for returning blocks once they are not needed. On allocation the smallest block of free memory large enough to hold n words must be used. When a block is returned it must be consolidated with any free block that is adjacent to it.

(a) The programmer's plan starts with the idea that each block of free or in-use memory will be preceded and also followed by extra words. Each of these will contain the length of the block plus a flag saying whether it is in use. Why might this information be useful? [10%]

(b) All free blocks of memory will be such that in addition to the header and trailer words, there is space to hold two pointers, and these will be used to arrange free blocks into a binary search tree. The ordering used in this tree will be such that if two blocks have different lengths the shorter one will be considered smaller. If two blocks have the same length then they will be ordered based on their relative positions in memory. Using this search tree, explain how to find a free block suitable either for direct use, or for use after splitting, to provide n words of memory for the user. Explain all tree operations you need in detail. [30%]

(c) Develop and present an algorithm that exploits the given structure to allow freeing of memory blocks, including the step of consolidating adjacent free blocks. Characterise its costs. Is allocation or freeing of memory more expensive or do they cost about the same? [25%]

(d) What effects would arise if instead of using a simple binary search tree, splay trees were used? [15%]

(e) Make an informal comparison between the allocation scheme considered here and one based on a Buddy system, considering both time and space costs. [20%]

END OF PAPER