UNIVERSITY OF BRISTOL

January 2018

FACULTY OF ENGINEERING

Examination for the Degrees of
Bachelor of Engineering
Master of Engineering

COMS10001
PROGRAMMING AND ALGORITHMS II

TIME ALLOWED:
2 Hours.

This paper contains *four* questions.
Each question is worth *30 marks*.
***Three* answers will be used for assessment.**
THIS IS A SAMPLE PAPER
The maximum for this paper is 90 marks.

<u>Other instructions</u>

1. **Calculators must have the Engineering Faculty seal of approval.**

TURN OVER ONLY WHEN TOLD TO START WRITING

Q1: This question is about algorithmic complexity and merge sort.

(a) Use merge sort to sort (12,5,65,4,9,76,1,5) describing all the steps.

(10 marks)

(b) What is the big-O algorithmic complexity of merge sort.

(10 marks)

(c) Explain how the big-O complexity of merge sort and quicksort differ.

(10 marks)

Solution: The lists get broken down to individual list, now 12 and 9 are merged to give (9,12), 4 and 65 to give (4,65), similarly (9,76) and (1,5). Now, crucially, these lists can be merged pair-wise by a sort of zipping mechanism where the lowest of the two elements available, one for each list, is added to the new list, so since 4 is less than 5 we get (4), then 5 is less than 65 so we get (4,5), next 12 is less than 65 and finally 65 is the only remaining element so we get (4,5,12,65), the other pair gives (1,5,9,76) and finally zipping together these two give (1,4,5,5,9,12,65,76). The algorithm is $O(n \log n)$, this can be done, either by quoting the master theorem or noting each level is n operations and there are roughly $\log n$ layers. Finally, quicksort is not $n \log n$ for its worst case, though it is easy to see that it is for almost every case. quicksort has $O(1)$ storage, merge sort as presented needs $O(n)$.

Q2: This question is about algorithmic complexity and recursion relations.

(a) An algorithm is $O(n^2)$, a second algorithm is $O(n)$; a new, third algorithm involves running the first algorithm and then the second. What is its complexity? Why?

(10 marks)

(b) An algorithm is $O(n \log n)$, a second algorithm uses this algorithm inside a loop which if performed $3n + 2$ times. What is its complexity? Why?

(10 marks)

(c) Find big-Theta for the recursion relation

$$T(n) = 3T(n-1) + 1$$

with $T(1) = 4$.

(10 marks)

Solution: Adding just gives you the larger of the two, for the same reason $n^2 + n$ is $O(n^2)$; in the second it is $O(n^2 \log n)$ by counting. The recursive question can be done ansatz, try $A3^n + B$ to get $A + B = A + 3B + 1$ so $B = -1/2$, substituting the initial condition gives $4 = 3A - 1/2$ so $3A = 9/2$ or $A = 3/2$.

Q3: This question is about heap sort.

(a) Define a binary heap. When might a heap be useful?

(5 marks)

(b) Heapify $(3, 5, 1, 6, 23, 2, 24, 14)$ so that the largest element is on top.

(10 marks)

(c) Remove the largest element from the heap and re-heapify.

(10 marks)

Solution: A binary heap is a binary tree in which all layers but the lowerest are full and every child is no bigger than its parent. To heapify add to the next available slot and bubble up until the heap property is satisfied. To delete the root replace it with the last added node and bubble down by swapping with the larger child until the heap property is restored.

Q4: This question is about minimization.

(a) What problem with gradient flow is solved by conjugate gradients?

(5 marks)

(b) Why might it not be possible to use gradient flow?

(5 marks)

(c) Describe the downhill simplex algorithm detailing the different moves and when they are employed?

(15 marks)

(d) When might simulated annealling or the genetic algorithm be used instead of downhill simplex?

Solution: Conjugage gradient solves the problem of zigzagging up and down the side of steep valleys. It is not possible to use gradient flow if the gradient is not known. The downhill simplex tries to move the worst node by reflecting it through all the other nodes, if the new point is the best it tries to extend the new point the same distance again in the new direction. If the new point is worse it shrinks in the worst point, if that doesn't work it shrinks all points towards the best point. Gradient flow finds local minima, the two heuristic algorithms find the global minimum, eventually, if you're lucky.

END OF EXAM PAPER