UG Exam Performance Feedback
Second Year
2018/2019 Semester 2

COMP26120 Algorithms and Imperative Programming

Giles Reger
Lucas Cordeiro
Milan Mihajlovic

Comments

Q1: The overall performance for this question was very good. Most of the students did the first part of the question (which was a bookwork) right. Small mistakes involved not labelling the edges as exploratory/backtrack and not getting the right data structure required for recursive implementation of the algorithm. In the second part most of the students managed to apply the algorithm to a given graph, and minor mistakes included not labelling properly all forward and backtrack edges. The final part of the question was more creative with the requirement to conclude how the algorithm from the first two parts can be applied to a problem of finding loops in a graph. A significant number of students got it right, with some interesting, but not fully correct, alternative approaches being proposed.

Q2: Most serious attempts for this question got this one right. However, various candidates were unable to select Formula Satisfiability (SAT) as a known NP-complete language in order to sketch the proof of NP-completeness. For those candidates who have correctly selected SAT, there were only a few of them who were unable to describe a reduction algorithm, which computes a function f mapping every instance x in {0,1}* of SAT to an instance f(x) of EQUIVALENCE-CHECKING, e.g., by replacing the if-then-else expression by an equivalent propositional logic expression.

Q3: This question was not attempted by many students but those who did attempt the question generally got it right. A common mistake for those who did not was to assume that q=1/2 and then simply call witness k times to get 2^k. It was necessary to find how many times to repeat witness for an arbitrary q.

Q4: There were a large number of full marks for this question. One common and unfortunate mistake was to pick the wrong initial pivot to arrive and terminate at a sub-optimal solution early. Some students forgot to give the final result after applying the algorithm correctly. Most students were able to state the initial tableaux.

Q5: Most students attempted (a) and (c) followed by (b) then (d). It should be noted that C code was not necessary here and most (not all) students realised this.

(a) A large number of students opted to give the quadratic solution (and some then failed to give the complexity and explain it). A similarly large number of students correctly identified that the input could be sorted and then processed linearly. Very few students discussed the complexity of the data structures being used.

(b) The majority of answers correctly identified this as a greedy algorithm. Some students failed to pick up easy marks associated with explaining what a greedy algorithm is. Partial credit was given if the wrong algorithmic technique was given but it was accompanied by a good explanation.

(c) This was generally answered well. The optimal solution (dynamic programming) was found by a large number of students although some forgot to state its complexity. Many students also proposed Dijkstra's algorithm but in such a case students generally failed to give specific information as to how the table could be represented as a graph, which assumptions were being made, and how the size of the input (n and m) corresponded to the number of nodes in the graph and therefore the complexity of the path-finding algorithm. A few students said that they were giving a dynamic programming solution but instead gave pseudo-code for greedy search.

(d) Simply finding the answer in this case was insufficient for full marks. Full marks were awarded to those students who dealt with the full 'generality' of the constraints that could be included and noted that this is an integer optimisation problem (hence NP-complete in general). Many students obtained 3/4 and a few did obtain full marks.