David Rydeheard
Ian Pratt-Hartmann
Milan Mihajlovic

Comments

Q1. The first two parts of this question were related to the examinable lab on Dijkstra's algorithm in the course. Vast majority of the students who answered this question did fairly well with this bit. This can be seen from the average mark for the question (180 students attempted the question, and the average mark was 13/20, with the first two parts weighing 15/20). The last part was meant to be creative and put the students on the spot and test their ability to work out an algorithm (without the need to figure out every detail) to solve a more complex problem using the ingredients from the previous parts of the question. There were a few very good and almost complete answers, but many students completely missed the point. Quite a few candidates went completely off the track, ignoring a comprehensive hint offered in the question.

Q2. I was surprised by the poor quality of many of the answers. In summary, this question may have looked like bookwork, but required some real thought about what dynamic programming is and isn't.

a) It's floor(log n) + 1. Candidates got a mark for anything vaguelylogarithmic. In spite of the first semester exam (where almost everyone got this right), many candidates took this to be a question about C data types, and therefore got poor marks.

b) and c) Many candidates could not properly formulate a simpleconditional definition. But these parts were quite often answered correctly.

d) This part was slightly tricky: Candidates were instructed clearly to give an algorithm taking only a polynomial amount of memory in the *sizes* of the input (and were supposed to know that the *size* of C is approximately log C). So writing everything into a big matrix with W rows (or columns), where W is the maximum input weight, is not good. A recursive algorithm does not actually need all that storage, however. (Of course, you end up repeating a lot of calculations.) What you wanted was a procedure that looks superficially like the standard dynamic programming algorithm, but uses repeated recursive calls, and does not store earlier calculations in a huge matrix. Candidates who *did* give the standard dynamic programming algorithm (correctly), you only lostne point, however.

e) *This* was where the standard dynamic programming algorithm (whichis pseudopolynomial, but not polynomial) belonged. Unfortunately, many candidates, having erroneously given that algorithm for part d), then decided to give another algorithm for part e)---presumably on the reasonable assumption that two parts of a question could hardly be the same---and therefore lost all marks for this part.

f) Just a reprise of part a) really: C is exponential in the size of C. Far too few candidates got this. Again, last semester's work on complexity seems to ave been forgotten.

Q3. This question consists of two parts. In the first part the students were asked to give a step-by-step procedure of forming an AVL tree from a given set of keys. The work involved several rotations to balance the tree (both single and double). There were numerous mistakes with the rebalancing. The second part was to give a pseudo-code of a method for finding the keys in an AVL tree with a specific range. The algorithm should be of an optimal complexity and deliver the keys in ascending order. This part caused many problems. A reasonable number of students did figure out that the tree needs to be preprocessed to determine the nodes that are within the key boundaries. This can be done in an optimal complexity. Then the pruned version of a tree needs to be traversed using an inorder algorithm to get the correct key ordering - this was missed by many students. Again, there were many answers which were incomplete or completely missed the point. As a result, the average mark was below that of Q1.

Q4. Whilst some students gained high marks for this question (in the 70%-100% range), a large proportion of the students got very poor marks (below 30%) giving an average of 42% for this question. It is clear that a substantial number of students failed to engage with the lecture material in this question, through the lectures, laboratory work and in revision. This is a warning to other years: failure to engage in lectured material and proper understanding of laboratory work will mean poor marks.

(a) DFS and BFS - very few gave clear and full explanations of these traversals.
(b) Priority search - less than half appreciated what this is and expressed it well.
(c) Pseudocode for priority search - quite a few students gave reasonable answers - we looked at this in the lectures
(d) Heuristics in search - very few students answered this clearly, though it is an important aspect of CS
(e) Worked example of a tree search application. Few students got good marks on this - what is required is a form of exhaustive enumeration using a tree, and heuristics for guiding the search.