Midterm 1 grades have been released. These are the markers' feedback for the questions. Please read these carefully *before* submitting a remark request. Also, note that remark requests will be considered only for an incorrect *application* of the marking scheme.

1a. [12] The solution is to: Find the median, see if the median is in the list, return the median if not, partition the list on median, and recurse on the smaller half. Submissions that did this were awarded full (or close to full, depending on minor errors) marks.

- Some students were familiar with the linear-time solution to this problem where we sum the values and subtract from the sum of n consecutive integers. Even though many tried to fit their solution into a divide-and-conquer-shaped hole, it was clear that this was not a divide-and-conquer approach. Such solutions received 8/12 marks (if the run-time analysis and explanation were done properly).
- There is another linear-time algorithm which fills in an indicator vector for the input and iterates through this vector looking for the entry which has not been marked. This is also not a divide-and-conquer solution and received 8/12 marks.
- Some students simply sorted the array and then went through the list looking for the missing element (albeit in a round-about, recursive manner). The trouble with such solutions is that the running time is $O(n \log n)$ and *not* $O(n)$ as required. Such solutions received 6/12 marks.
- There were also solutions which blindly divided the array in two (say into subsets of equal size) and made some recursive calls. The trouble with this is that the subsets upon which you are making the recursive calls no longer have the properties that the original set had, namely that it is a set of consecutive integers save one. Thus, it is unclear what, if any, information could be learned from these recursive calls. Such solutions received 3~4/12 marks.

1b. [8] The solution idea is to again partition on each coordinate and recurse down the subset containing fewer symbols. Submissions that did this were awarded full (or close to full) marks. The straight-forward solution which iterates over each of the k bits looking to see if there are fewer zeroes or ones in that coordinate typically received 6/8 marks. The running time here is $O(nk)$ rather than the required $O(n)$.

2. The solution is to just greedily go to the farthest station that you can reach before running out of gas.

2a. [7] Most students got this, although some students sorted the gas stations which was unnecessary and led to larger run times than needed.

[3 marks] Correct idea communicated in English

[4 marks] Greedy algorithm provided with correct pseudo code

2b. [1] Most students got this.

2c. [12] There were two main solutions: Proof by induction and proof by contradiction.

For a proof by induction, students needed to define what it means for a partial greedy solution to be promising or at least what the induction is over before jumping straight into the base case. This was something quite a few people failed to do. Furthermore, you needed to consider 3 cases: Greedy does not take, Greedy takes and Opt takes, and Greedy takes but Opt does not take. The last of which required an exchange argument.

[1 mark] Base case

[2 marks] Inductive hypothesis

[1 mark] Inductive Step

[2 marks] Greedy does not take station i

[2 marks] Greedy takes station i in Opt

[4 marks] Greedy takes station i not in Opt + Exchange

For a proof by contradiction, some students failed to pick the correct optimal solution to use. You need to pick the optimal solution that matches with greedy **the most**. This property is critical for showing a contradiction, otherwise you just turned one optimal into another one without there being any consequence. The contradiction comes from creating a new optimal solution that matched with your greedy more than the one that should have matched the most, and *not* from being able to create a new optimal solution via exchange argument.

[2 marks] Assume greedy is not optimal

[2 marks] Pick appropriate optimal solution (1 mark if you pick any, likely marks will be lost later for identifying the contradiction)

[2 marks] Mention index i in which stations differ between greedy and optimal, justify why index $i < n$

[4 marks] Exchange argument to create a new optimal solution

[2 marks] Point out the contradiction.

3. The solution is to find the maximum grade, $OPT[i, h]$, by spending h hours over the first i projects. Base case: $OPT[i, 0] = 0$. Main case, $OPT[i, h]$: fix the number of hours h' (all values between 0 and h) spent on project i, you then get a grade of $e_i(h') + OPT[i-1, h-h']$, take the maximum over all such values.

3a. [5] Some students forgot to mention how they would compute the final answer and lost 1 point.

3b. [7] Solutions without any justification lost 2 points. Some suggested algorithms were incorrect greedy algorithms or a correct brute-force (but we want an *efficient* i.e., polynomial time algorithm) and lost 5 points. Some students assumed that the estimate functions e_i's are linear. That $is\ e_i(h) = e_i(h - 1) + e_i(1)$, which is not a correct assumption and lost 2 points.

3c. [4] Solution: Use a nested for loop (for $i = 1$ to n, for $j = 0$ to H, update according to the bellman equation). It takes $O(nH^2)$ time and $O(nH)$ space. You could improve space to $O(H)$ by keeping the last row (as $OPT[i,\cdot]$ only gets updates from $OPT[i - 1,\cdot]$), but no points deducted for not optimizing memory usage. Some provided a top-down implementation (recursive function calls) and lost 2 points. Some did not realize the Bellman equation computation takes $O(h)$ time not $O(1)$, and reported $O(nH)$ time complexity instead of $O(nH^2)$ and lost 0.5 points. Another 0.5 points deducted if the space complexity was not mentioned.

3d. [4] Solution: Keep how you updated the dynamic program, traverse from $OPT[n, H]$ by going to the cell it was updated from $OPT[n - 1, H - h_n]$, then to $OPT[n - 2, H - h_n - h_{n-1}]$, and so on. There is no increase in running time and it is still $O(nH^2)$, but you will need at least $O(nH)$ memory. For incorrect or no mention of time or space complexity, 0.5 points were deducted for each.

Thanks,

Marking Team