

COT 5405 Midterm 1 Solutions

February 17, 2009

1 Problem 2

[10 Points] CIRCULAR SORTING

A sequence of n distinct numbers $\langle x_0, x_1, \dots, x_{(n-1)} \rangle$ is said to be circularly sorted if $x_j < x_{(j+1) \bmod n}$ for all $j \in \{0, 1, 2, \dots, n-1\} - \{i\}$ where x_i is the largest number. For example the sequence $\langle 13, 15, 18, 22, 24, 28, 33, 4, 7, 11 \rangle$ is circularly-sorted. Given a set of numbers, CIRCULAR SORTING PROBLEM asks to compute a permutation of them that is circularly-sorted. Design a comparison based algorithm for solving this problem that runs in $o(n \log n)$ time (i.e., asymptotically faster than $O(n \log n)$) or prove that no such algorithm exists.

Here, we have given two different approaches to show that no such algorithm exists. First one uses a similar technique as in showing the lower bound on sorting. We can construct a lower bound argument similar to the one for comparison-based sorting. As you remember, it was proving the lower bound by the help of decision tree. For this problem, the decision tree here has $(n-1)!$ leaves, since permutations that are just a circular shift of one another are equivalent. You have two different choices in every level of the tree. Then, the lower bound one gets is therefore $\log((n-1)!)$ which is $\Theta(n \log n)$. Hence, it is not possible to have a better algorithm.

Second approach proves the same by contradiction: Suppose there exists an algorithm which can circularly sort in $o(n \log n)$ time. Then, you could first circularly sort a list and find the minimum element in linear time and print out the sorted list. This means you have sorted your list better than $n \log n$ lower bound. However, we have already known, with comparison-based sorting, we have an $n \log n$ lower bound, which gives us a contradiction. Therefore, we cannot circularly sort a list better than $n \log n$.

Grading guidelines for this question: Graded by Hale. If you have given an algorithm saying that it can be done, unfortunately I could not give you any points. Otherwise, saying just there does not exist an algorithm worth 3 points, the rest needs a justification. Wrong justifications are also considered and given partial credits.

2 Problem 6

[10 Points] GREEDY CHOICE?

Consider a road with n potholes, which we plan to repair by repairing parts of the road. We have a special machine that can pave patches of the road only in unit length. The location of pothole is represented by a single number which is the distance from the start of the road. A single (unitlength) patch can be used to repair multiple potholes if they are pairwise at most unit length apart from each other. For example, the road with 10 potholes $\langle 3.5, 7.4, 8.2, 8.5, 9.4, 9.5, 9.6, 10.1, 14.7, 15.7 \rangle$ can be repaired with 5 patches $\{[3.0, 4.0], [6.5, 7.5], [8.0, 9.0], [9.4, 10.4], [14.7, 15.7]\}$.

(a) Design and analyze an algorithm for computing the minimum number of patches needed to repair a road with n potholes.

(b) Would your algorithm still compute an optimal solution if the start and end of the road is the same (e.g., a race track)? If yes, give a proof that still does. Otherwise, design and analyze a new algorithm for computing the optimal solution.

(a) The idea is to start with the first pothole which is located left-most, cover by a patch of unit length starting from that point and skip all the potholes covered by this patch. Repeat the same until all potholes have been covered. If we have a pothole at point x , then a patch with unit length, $u = [p, p + 1]$ covers that point x if $p \leq x < p + 1$. So, the algorithm can be given as follows:

PATCHCOVER(a list of pothole locations, n)

Sort the given set of potholes into nondecreasing order. $\langle x_1, x_2, \dots, x_n \rangle$ will denote the sorted list.

$S = \text{Empty}$ (initialize the solution)

$C_1 = [x_1, x_1 + 1]$

$S = S \cup \{C_1\}$

$RightEnd = x_1 + 1$

$j = 1$

for $i = 2$ to n **do**

 do

if $x_i > RightEnd$ **then**

 // otherwise, the point x_i is already covered

$j = j + 1$

$C_j = [x_i, x_i + 1]$

$S = S \cup \{C_j\}$

$RightEnd = x_i + 1$

end if

end for

return S

The correctness of the algorithm can be shown by the following two facts:

Lemma 1: The greedy choice property denotes, if x is the leftmost pothole in the given list of potholes. Then there is an optimal solution containing the

unit length patch $u = [x, x + 1]$.

This can be shown by considering an optimal solution S^* . Suppose that patch $u = [x, x + 1]$ is not in S^* . But, there should be another patch covering x which will denote by $u' = [p, q]$. This will imply $p \leq x$. If $p = x$, then $u' = u$. Therefore, $p < x$. We already know that there are no other potholes left of x , so we can move the patch u' such that its left point p will coincide with x . We will not be losing any points that are covered by u' . This translation will make u and u' coincide. So, the above statement is correct.

Lemma 2: This problem has an optimal substructure property, such that if x is the leftmost pothole in the given list X . Let S^* be an optimal solution for X containing the unit patch $u = [x, x + 1]$. Let X' be the list of potholes in X that are not covered by u . Then, $S' = S^* - \{u\}$ is an optimal solution for X' .

This can be shown by contradiction. Suppose S' is not an optimal solution to X' . Since S^* is a solution to X , all potholes which are not covered by u are covered by the patches in S' . So, if S' is not an optimal solution to X' , this means there exists another solution S'' , which is smaller than S' and optimal ($|S''| < |S'|$). Then, we can say $S'' \cup \{u\}$ needs to be a solution for X . We already know that $|S''| < |S'|$, therefore we can say $|S'' \cup \{u\}| < |S^*|$, which is a contradiction to our assumption that S^* is an optimal solution. This proves the optimal substructure of the problem and the correctness of the algorithm.

The analysis is straight-forward, since we examine the potholes once, which will yield $O(n)$ time. If you assume that the given list of potholes are not sorted according to location, then we also need $O(n \log n)$ time to sort the points first. So, this will make the overall complexity $O(n \log n)$.

(b) No, the above algorithm would not work. It is not considering the rollover due to the circular geometry of the race track. A counterexample would be as follows: For the list of potholes $< 0.50.64.6 >$ on 5 unit length racetrack, greedy would give 2 patches whereas the optimal is just 1. Most straight way to do this, is to apply the same algorithm for every pothole and choose the minimum of the results. Basically, we try all possible starting points to make sure we considered the rollover. This would take $O(n^2)$ time, applying the above $O(n)$ procedure to all n potholes.

Grading guidelines for this question: Graded by Hale. For the first part if you have given the correct greedy strategy, this would be 6 points. Proof of correctness is 2 points and analysis also 2 points. If you give a wrong greedy strategy which does not lead to optimal solution, again it is considered and partial credits are given. For the second part, saying that your algorithm does not work is 3 points. The explanation why it is not working is 1 point. Giving a new algorithm for finding the optimal solution is 4 points and the analysis of the algorithm worth 2 points. If you have given wrong algorithm in the first section, I have also considered this part in terms of your answer, again partial credits are given for the wrong answers.

Note: Solutions given are not the only possible solutions, there can be many different solutions.