Algorithms: CSE 202 — Take-home Final

Maximum points: 44 + 8.5 make-up points

Due December 14th, 2017. 10:00 PM. Submit the solutions via GradeScope. No late submissions.

Instructions:

1. There are six problems in the test. *Solve exactly five problems.* All problems carry equal weight.
2. You are only allowed to consult the following textbook, your own course (Fall 2017 CSE 202) lecture notes and the course website. You should not consult any other source including the Web. You are also not allowed to consult with any other person.
   (a) Algorithm Design by Kleinberg and Tardos
3. This test is designed to give you an opportunity to demonstrate *your* understanding of the principles of algorithm design, analysis and application. Consequently, you are required to work by yourself in answering the problems. Group work or consultation with others is not allowed.

   You can check with the instructor or the TAs for any clarification during the office hours. Please do not send any email or questions over Piazza regarding the final examination.
4. Each problem requires that you achieve an algorithm with certain complexity (for example, polynomial time, linear time, etc.). To the extent the problem permits latitude in terms of complexity (for example, polynomial time provides considerable latitude), obtain the most efficient algorithm.
5. Guidelines for writing a solution:
   (a) Write your solutions neatly. Proofread them before you submit.
   (b) Write a *high-level description* of your algorithm. State the main ideas and strategies used in developing your algorithm. Include all important details.
   (c) Describe *briefly* how you would implement your algorithm providing important implementation details. You may sparingly use pseudo code to describe the details of your implementation if you believe the code will bring more clarity.

   We are not expecting you to write down the obvious details of your implementation. Describe only the unusual aspects of your implementation.
   (d) If you are using a modified version of a well-known algorithm, show clearly what your modifications are. If you are using a well-known algorithm without any modifications, it is enough to specify how you plan to apply the algorithm.
   (e) Provide a detailed *proof of correctness.*
   (f) Analyze the *time complexity* of your algorithm.
   (g) Quality of the high-level description of your algorithm, its correctness and efficiency are the prime determinants of your grade.
6. Well-written solutions will be rewarded. Characteristics of good solutions: correct solution, logical coherence, an appropriate balance between completeness and succinctness, and an orderly presentation of ideas. Excessively long solutions will be penalized. We do not expect you to include information not relevant to the solution.

7. If you include two different solutions to the same problem, we are not obligated to consider both the solutions.

Problem 1: Sequence matching

Let T and P be respectively two sequences $t_1, \cdots, t_n$ and $p_1, \cdots, p_k$ of characters such that $k \leq n$. The characters range over a finite alphabet Σ. With each position of T, we associate a nonnegative value. Let $v(i)$ denote the value for the i-th position in T for $1 \leq i \leq n$. Find a matching subsequence of T for the pattern P that maximizes the sum of values. That is, find the sequence of indices $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ such that for all $1 \leq j \leq k$, we have $t_{i_j} = p_j$ and $\sum_{j=1}^{k} v(i_j)$ is maximized. Design an efficient algorithm for this problem, prove its correctness and analyze its complexity.

To deserve any credit, you must present a polynomial-time algorithm along with a complete solution. You must strive for as efficient an algorithm as possible.

Problem 2: Covering points with gain

Describe an efficient algorithm that, given a set $\{x_1, x_2, \ldots, x_n\}$ of points on the real line where the point x_i has a gain $w_i \geq 0$ and a set of closed intervals $[s_i, f_i]$ for $1 \leq i \leq d$ where the cost of the i-th interval is $c_i > 0$, select a set of closed intervals so that the sum of the gains of the points covered by the selected intervals less the sum of the costs of the selected intervals is maximized. Design an efficient algorithm, prove its correctness and determine its complexity.
Note: You can assume that the gains and costs are nonnegative integers.

Problem 3: Round-robin tournament

Assume that a round-robin tournament is played among n players. That is, each player plays once against all $n-1$ other players. There are no draws and the results of all games are given in a matrix A. Entry $A(i,j)$ is 1 if player P_i beat player P_j and 0 otherwise. It is not possible in general to sort the players, since P_i may beat P_j , P_j may beat P_k, and P_k may beat P_i. In other words, the results are not necessarily transitive. We are interested in a weak sorting as follows. Design an algorithm to arrange the players in an order $P_{\sigma(1)}, P_{\sigma(2)}, \ldots, P_{\sigma(n)}$ such that $P_{\sigma(1)}$ beat $P_{\sigma(2)}$, $P_{\sigma(2)}$ beat $P_{\sigma(3)}$, $\ldots$, $P_{\sigma(n-1)}$ beat $P_{\sigma(n)}$ where σ is a permutation (one-one and onto) function on the integers $\{1, 2, \ldots, n\}$. In other words, $A(\sigma(1), \sigma(2)) = A(\sigma(2), \sigma(3)) = \cdots = A(\sigma(n-1), \sigma(n)) = 1$. The worst-case running time of the algorithm should be $O(n \log_2 n)$. Any entry in the matrix can be accessed in constant time.

Design an $O(n \log_2 n)$ algorithm and prove its correctness.

Problem 4: Cellular network

Consider the problem of selecting nodes for a cellular network. Any number of nodes can be chosen from a finite set V of potential sites. We are also given an undirected graph $G = (V, E)$ over the set of sites where an edge $ij \in E$ connects sites i and j. We know the benefit $b_i \geq 0$ of selecting a node at site i. However, if sites i and j are selected as nodes and if there is an edge $ij \in E$ between them, then the benefit is offset by c_{ij}, which is the cost of interference between the two nodes. Both the benefits and costs are non-negative integers. Find an efficient algorithm to determine the subset of sites as the nodes for the cellular network such that the sum of the node benefits less the interference costs is as large as possible.

Design an efficient polynomial-time algorithm.

Provide a high-level description of your algorithm, prove its correctness, and analyze its time complexity.

Problem 5: Center selection with producers and consumers

Consider the following variation of the center selection problem. You are given a set V of n vertices and a distance function $d(.,.)$ that satisfies the non-negativity, symmetry and triangle inequality properties. However, the vertices are partitioned into two categories, producers and consumers. Let $P \subseteq V$ denote the set of vertices corresponding to the producers and $Q = V - P$ denote the set of vertices corresponding to the consumers. You are also given an integer $k \geq 1$. The objective is to select a set C of k producers so that $\max_{q \in Q} d(q, C)$ is minimized, that is, the maximum distance from a supplier to a consumer is minimized. Provide an efficient 3-approximation algorithm for the problem. Prove its correctness, bound its approximation ratio and determine its complexity.

For a set of points R and a point s, we define $d(s, R) = \min_{r \in R} d(s, r)$. You can assume that the distance between two vertices can be computed in constant time.

Problem 6: Approximation algorithm

Consider the following optimization problem: Given n integers $c_1, c_2, \ldots, c_n$, find a partition of $\{1, 2, \ldots, n\}$ into two subsets S_1 and S_2 that minimizes the quantity $\max(\sum_{i \in S_1} c_i, \sum_{i \in S_2} c_i)$. Consider the following heuristic for some fixed integer k:

1. Choose the k largest c_i's.
2. Find the optimal partition of these k integers
3. Complete this into a partition of $\{1, 2, \ldots, n\}$ by considering each of the remaining c_i's and adding it to the partition which at the time has the smallest sum.

Prove that this algorithm with $O(2^k + n)$ time complexity outputs a sum that is within $(k + 3)/(k + 1)$ fraction of the optimal output.