This assignment must be submitted to GradeScope by November 14 at 3:00 PM.

1. I am teaching a class that has \( n \) teaching assistants, each of whom hold office hours outside, at JavaCity. The \( i \)th TA holds office hours starting at time \( S_i \) and ending at time \( E_i \). Each TA holds exactly one office hour (which may last any amount of time, not necessarily one hour, and different TAs may have different interval lengths, which may overlap with one another). Attendance has been low because students, being Computer Science majors, are afraid of the sun, so the department announces that all TAs who do not have a student visit them this week will be fired\(^1\). Because all of our teaching assistants are awesome\(^2\), we want to ensure that none get fired. However, due to student fears of the sun, we want to minimize the total number of visits made to TAs.

For purposes of this problem, assume each TA’s office hour time is one continuous interval with no breaks, and that a student visiting the sundeck at time \( t \) counts as visiting all TAs whose office hours interval contains \( t \). Also assume that student visits are “instantaneous,” in the sense that the amount of time a student stays in the sun is negligible – formally, each student visits the deck at a single point in time. This also helps to avoid exposing a student to the sun unnecessarily.

You propose the following: first, we sort all of the intervals by end time. We send a student to attend office hours at the moment immediately prior to the end of the first ending interval. We remove from our input all TAs who overlap with this time, and if the remaining set is non-empty, we repeat.

Prove that the greedy algorithm in the previous statement minimizes the number of students we need to send to visit TAs. You may be as informal as the proof from lecture on October 31, or more formal if you desire.

2. We’re asked to help the captain of the UCI’s tennis team to arrange a series of matches against Long Beach State’s team\(^3\). Both teams have \( n \) players; the tennis rating (a positive number, where a higher number can be interpreted to mean a better player) of the \( i \)th member of UCI’s team is \( a_i \) and the tennis rating for the \( k \)th member of LBSU’s team is \( s_k \). We would like to set up a competition in which each person plays one match against a player from the opposite school. Because we get to select who plays against whom, our goal is to make sure that in as many matches as possible, the UCI player has a higher tennis rating than their opponent. Give an efficient greedy algorithm for this problem and prove that it is correct (maximizes the number of matches in which our team has the player with the higher tennis rating). Note that “sort both rosters and pair up the top players, the second-best, etc” is not the correct answer to this problem.

For CompSci 260P this quarter, for the proof, it is sufficient to demonstrate that the repair of an arbitrary inversion in any alternate ordering, compared to yours, produces an ordering that is no worse. You may use the fact from lecture that any permutation that differs from yours has a consecutive pair of elements that are inverted relative to yours, however for this particular problem, the proof may not need the fact that they are adjacent.

\(^1\)This part is not actually true and is fiction for the problem.
\(^2\)Worth noting: this part is true for CompSci 260P this quarter.
\(^3\)Not really.
3. The **Steiner Tree** problem is as follows. Given an undirected graph \( G = (V, E) \) with nonnegative edge costs and whose vertices are partitioned into two sets, \( R \) and \( S \), find a tree \( T \subseteq G \) such that for every \( v \in R \), \( v \in T \) with total cost at most \( C \). That is, the tree that contains every vertex in \( R \) (and possibly some in \( S \)) with a total edge cost of at most \( C \).

This is similar to a Minimum Spanning Tree with some vertices not being required to be part of the output graph.

(a) The difficulty in the Steiner tree problem is determining which set \( S' \subseteq S \) should be included in \( T \). Give a polynomial time algorithm to find the Steiner tree of minimum cost for the case where the optimal \( S' \) is also provided.

(b) This problem, when \( S' \) is not given is \( \mathcal{NP} \)-complete. If we want to solve this problem anyway, we will need to abandon correctness and get an efficient, approximate solution. We are solving the problem of finding a minimum-cost Steiner tree from a complete graph (a simple graph with \( n \) vertices and all \( \binom{n}{2} \) possible edges). We are not given the optimal \( S' \subseteq S \). You may assume that the edge costs obey the triangle inequality if you would like to do so, although that isn’t strictly necessary to obtain the correct answer. Suppose we compute a minimum spanning tree on \( R \), treating \( S' = \emptyset \) (which might not be true in the optimal solution). Show that this approximation algorithm has cost at most twice the optimal.

4. Suppose we are given a graph \( G = (V, E) \) and want to color the vertices from amongst \( k \) distinct colors. Give a polynomial-time algorithm that guarantees that the number of monochromatic edges is at most \( |E|/k \). Explain why your algorithm makes this guarantee and why it is polynomial time.

*Hint: For any arbitrary coloring, how many monochromatic edges do you have? If you have more than the required amount, can you adjust the solution?*

*Hint 2: If you are struggling with this question, try to solve it with \( k = 2 \) and see if your solution generalizes.*