1. **k-friendliness in CS 161.** (4 points)

   Suppose you represent the friendships of students in `CS 161` as an undirected graph $G = (V, E)$ (i.e. there exists an edge between vertices A and B if these students are friends). For any integer k, we say that a group of students, $S \subset V$, is "k-friendly" if every student in S has at least k friends in S.

   Describe an efficient algorithm which, given the graph G and an integer $k > 0$, finds the largest set $S \subset V$ that is k-friendly; if no such set exists, then the algorithm should say so. Also, justify the correctness of your algorithm and its runtime.

   [**We are expecting a description or pseudocode of your algorithm as well as a brief justification of its correctness and runtime.**]

2. **Transportation networks.** (5 points)

   Given a set of n cities, we would like to build a transportation system such that there exists some path from any city to any other city. There are two ways to travel: by driving or by flying. Initially all of the cities are disconnected. It costs $c_{i,j}$ to build a road between city i and city j. It costs a_i to build an airport in city i. For any two cities i and j, we can fly directly from i to j if there is an airport in both cities.

   Describe an $O(m \log(n))$-time algorithm that accepts as input the costs $c_{i,j}$ and a_i, and returns a list of roads and airports to build that minimizes the cost of connecting the cities. Here, "connecting the cities" means that there should be some way to get from any city to any other. Also, justify the correctness of your algorithm and its runtime.

   [**We are expecting a description or pseudocode of your algorithm as well as a brief justification of its correctness and runtime. You may (and, hint, you may wish to) call any algorithm we have seen in class.**]

3. **Placing receivers.** (9 points)

   Suppose there are n transmitters fixed in place along a linear track. The i'th transmitter has communication range $[a_i, b_i]$, for $a_i \leq b_i$. That is, any receiver placed within the range $[a_i, b_i]$ can receive signals from the i'th transmitter. Assume that the transmitters are sorted by the right endpoint of their communication range: that is, if $i < j$, then $b_i \leq b_j$.

   We want to pick a set of points on the track to place receivers such that we can receive signals from every transmitter while minimizing the number of receivers necessary. That is, we want to find a minimum set of points S on the line such that for every transmitter i, there is some receiver $s \in S$ such that $a_i \leq s \leq b_i$.

   In this problem, we will design an algorithm that finds the minimum set S of receiver locations in expected time $O(n)$, and prove that it is correct.

Consider the greedy algorithm which works as follows: we place receivers one at a time. At each step, suppose that i^* is the smallest i so that transmitter i cannot be heard by any receiver placed so far, and place a receiver at b_{i^*}. Continue placing receivers in this way until all the transmitters can be heard.

(a) (2 points) Based on this English description, write pseudocode to implement the algorithm in time $O(n)$. Assume the input to your algorithm is two arrays, a, and b, which contain the values of a_i and b_i in the order described above.

[We are expecting detailed pseudocode and a brief justification of its runtime.]

(b) (7 points) Prove that this algorithm is optimal, following the outline below (don't worry about proving feasibility). We recommend using a greedy exchange argument by induction on i.

i. (2 points) State your inductive hypothesis.

ii. (1 point) Prove the base case.

iii. (3 points) Prove the inductive step.

iv. (1 point) Finish the argument. That is, once the induction argument is complete, show that this implies that the algorithm is correct.

[We are expecting: for i, a statement of an inductive hypothesis. For ii, iii, iv, we are expecting a formal proof, including a statement of what it is you are proving.]

4. **Properties of MSTs.** (7 points)

In all of the following questions, assume that G is an undirected, connected graph on n nodes with **distinct**, positive edge weights. Also assume that G is simple (no self-loops and no multi-edges) in all parts.

(a) (1 point) Consider the edges of G sorted in increasing order of weight and place the first $n-1$ edges into a set S. That is, S is the set of $n-1$ edges with the smallest weights. Prove or disprove: S is a minimum spanning tree in G (assume $n \geq 2$)

[We are expecting either a formal proof or a counterexample.]

(b) (2 points) Prove or disprove: There is a unique minimum spanning tree in G.

[We are expecting either a formal proof or a counterexample.]

(c) (4 points) Let T be a minimum spanning tree of G. Let e be some edge in G (which may or may not belong to T). Let us obtain a new graph G' from G by preserving everything the same except that we decrease the weight of e (but assume that the weights of all edges in G' are still distinct). Design an algorithm that can find a minimum spanning tree T' of G' in time $O(n)$, given G, T, e and its newly decreased weight w. Prove the correctness and analyze the runtime.

[We are expecting an algorithm (description of one is fine), a proof of correctness, and a runtime analysis.]

5. **Min Gradespan.** (4 points)

   An analog of the following problem arises when you are trying to assign jobs to servers; while the greedy algorithm will not be optimal, it comes pretty close.

   Suppose we have n exams to grade, and there are k Course Assistants (CAs) who will be doing the grading. Suppose can tell the exact amount of time it will take to grade a given exam (from glancing at the handwriting and text density), and all the CAs would take this exact amount of time to grade that exam. I want to divide the set of n exams among the CAs so as to minimize the maximum amount of time it will take any of the CAs to grade (i.e. we will all wait until the last CA is done grading their pile, and I want to minimize the total time from when we all start grading, until this last CA is done.)

   Suppose I start with the stack of exams, and want to divide the exams into the k stacks by stepping through the pile of exams just once—namely, when I look at the ith exam, I will immediately know how long it takes to grade, and then assign it to one of the k CA piles. Suppose I do this by keeping track of the total grading time of each stack, and assigning the next exam to the stack that currently has the shortest grading time.

   (a) (1 point) Describe a concrete instance of this problem involving $n = 3$ exams and $k = 2$ CAs that illustrates that this algorithm will not always result in the optimal allocation. Specifically, give a list of the three grading times such that the greedy algorithm will result in one of the two CAs spending longer than necessary, and describe both the optimal allocation of the exams to the two piles, and the allocation that the greedy algorithm will choose.

   [We are expecting an example. You can be as specific with the example as you want.]

   (b) (3 points) Prove that the allocation given by the greedy algorithm will be suboptimal by *at most* a factor of 2. Namely, if there exists an allocation of the n exams into k piles such that all CAs can finish grading within time t_{opt}, prove that in the allocation chosen by the greedy algorithm, the maximum time will be at most $2 \cdot t_{opt}$. [Hint: what can you say about the amount of grading time that the longest-working CA would have had, if the last exam did not exist?]

   [We are expecting a formal proof.]

   (c) (0 points, Food for thought) How much can you improve this factor-of-two ratio of greedy and opt (i.e. this factor of 2 "competitive ratio" of the greedy algorithm) if you sort the exams in decreasing order of grading time before running the greedy algorithm?

   [We are not expecting anything.]

6. **How is the course so far?** (0.5 points extra credit)

   Please complete the poll at `https://goo.gl/forms/2tipSmXawmrBV3is1`.