CS 161 **Problem Set 7**
Spring 2017 **Due:** June 2, 2017, 3pm

Please answer each of the following problems. Refer to the course webpage for the **collaboration policy,** as well as for **helpful advice** for how to write up your solutions.

Note: For all problems, if you include pseudocode in your solution, please also include a brief English description of what the pseudocode does.

1. **True or False? (3 points)** Decide whether the following statement is true or false. If it is true, give a short justification. If it is false, give a counterexample.

   > Let $G = (V, E)$ be an arbitrary connected weighted undirected graph with no self-loops (that is, no edges from a vertex to itself), so that the weight of an edge $e \in E$ is $w(e)$, and all the weights are distinct. Let $e^* \in E$ be the cheapest edge: that is, the edge so that $w(e^*) = \min_{e \in E} w(e)$. Then there is a minimum spanning tree T of G that contains e^*.

   [We are expecting: your true/false answer, and either a justification (a few sentences) or a counterexample (a graph).]

2. **Greedy algorithms don't always work. (3 points)** Let $G = (V, E)$ be an undirected unweighted graph. Say that a vertex v can "see" an edge if v is an endpoint of that edge. Say that a set $S \subseteq V$ is "all-seeing" if every edge $e \in E$ is seen by at least one vertex in S. In this problem, we will try to greedily construct the smallest all-seeing set possible.

   Consider the following greedy algorithm to find an all-seeing subset:

   ```
   findAllSeeingSubset(G):
     S = {}
     while G contains edges:
       choose an edge e = (u,v) in G
       S.add(u)
       S.add(v)
       remove u and all of its adjacent edges from G
       remove v and all of its adjacent edges from G
     return S
   ```

   (a) (1 pt) Prove that `findAllSeeingSubset(G)` always returns an all-seeing subset.
   [We are expecting: A short but rigorous proof (a few sentences).]

   (b) (2 pts) Give an example of a graph on which `findAllSeeingSubset(G)` does not return a smallest all-seeing subset, for at least one way of choosing edges.
   [We are expecting: your example, and a brief justification that it is an example of this.]

   (c) (0 pts) **[BONUS: this question is NOT REQUIRED but it might be fun to think about. It will not affect your grade.]**

   i. Can you find a greedy algorithm that does find a smallest all-seeing subset?

   ii. `findAllSeeingSubset` has a very nice property. Let $OPT(G)$ denote the size of the smallest all-seeing subset in G. Prove that for all graphs G,

   $$\frac{|\texttt{findAllSeeingSubset}(G)|}{OPT(G)} \leq 2.$$

   [We are expecting: Nothing, this problem is optional.]

3. **Transportation Networks. (5 points)** Given a set of n cities, we would like to build a transportation system such that there is 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. Give an efficient algorithm for determining which roads and airports to build to minimize 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.

   You algorithm should take as input the costs $c_{i,j}$ and a_i, and return a list of roads and airports to build. It should run in time $O(n^2 \log(n))$.

   [**We are expecting: pseudocode and a short informal explanation of why it is correct. You may (and, hint, you may wish to) call any algorithm we have seen in class.**]

4. **Placing receivers. (8 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 i, there is some $s \in S$ such that $a_i \leq s \leq b_i$.

   In this problem, we will design an algorithm that finds the minimum set S 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 $O(n)$.

   [We are expecting: detailed pseudocode, and an informal justification of the running time.]

   (b) (6 points) Prove that this algorithm is correct, following the outline below. We will use induction on t, where the inductive hypothesis should be something like the following: after we have processed t transmitters and have a set S of receivers, there is an optimal set S^* of receiver locations so that $S \subseteq S^*$.

   i. (1 points) Formalize the inductive hypothesis that is described above. (The details of how you state it may depend on how you've written your pseudocode in part (a); if the statement above makes sense with the pseudocode you have written and the rest of your argument, you may just repeat it).

   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.]

5. **Well-connected subgraphs. (10 points)** Given an undirected, unweighted graph $G = (V, E)$ and a set of vertices $S \subseteq V$, we say that the **subgraph of G induced by** S is the graph $G' = (S, E')$, where the vertex set is S, and the edge set is

$$E' = \{(u, v) \in E : u \in S, v \in S\}.$$

That is, the induced subgraph is formed by keeping all the vertices in S, and keeping all the edges with both endpoints in S.

We seek an algorithm to solve the following problem:

- **Input:** A graph $G = (V, E)$ and an integer k
- **Output:** A set $S \subseteq V$ so that every vertex in the subgraph $G' = (S, E')$ induced by S has degree at least k in G', so that S is as large as possible given that it has this property, if such a set exists. If no such set exists, return "`Does not exist.`"

(a) (4 pts) Give pseudocode for an efficient greedy algorithm that solves the problem above. Your algorithm should run in expected time $O((m + n)^2)$ if $m = |E|$ and $n = |V|$. (Note: It is possible to come up with an algorithm that runs faster than this, but anything no worse than this running time is fine for full credit). You may assume that G is stored using the adjacency-list representation.

[We are expecting: pseudocode, and a short informal analysis of the runtime.]

(b) (6 pts) Prove formally that your algorithm is correct.

[We are expecting: a formal proof. If you use induction, make sure you explicitly state what your inductive hypothesis is, and make sure that you explicitly conclude that your algorithm is correct.]