Due on March 5th, 2018
55 points total

General Directions: If you are asked to provide an algorithm, you should clearly define each step of the procedure, establish its correctness, and then analyze its overall running time (for this assignment, this means arguing why your algorithm achieves the target running time specified by the question). There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice.

All the answers must be typed, preferably using LaTeX. If you are unfamiliar with LaTeX, you are strongly encouraged to learn it. However, answers typed in other text processing software and properly converted to a pdf file will also be accepted. Before submitting the pdf file, please make sure that it can be opened using any standard pdf reader (such as Acrobat Reader) and your entire answer is readable. **Handwritten answers or pdf files that cannot be opened will not be graded and will not receive any credit.**

Finally, please read the detailed collaboration policy given on the course website. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (10 points)
You are planning a second party (remember the first party?) for spring break. You have a total of n friends, each of whom has given you two lists with a subset of these n friends of yours. Each friend will attend your party if and only if everyone in their first list and no one in their second list is invited to your party. You quickly realize that you must throw multiple parties if you want to invite every friend. What is the most efficient algorithm you can come up with to decide if you can throw a set of parties such that each of your n friends attends a party?

Problem 2 (20 points)
You have been called to mediate in an "algorithmic dispute" between Alice Algorithmix and Bob Bitfiddler! Alice and Bob have been tasked by a courier company to find routes in Durham such that each destination is reached using the shortest route starting at their delivery center. The company wants to save overhead; therefore, they require that the roads traversed in all the routes should be acyclic (call this a shortest path tree). Having learnt minimum spanning tree (MST) algorithms from you, Alice has found an MST on the Durham roadmap (think of roads as edges and intersections/destinations as vertices) and claims that for every graph, there always exists an MST that is also a shortest path tree. One the other hand, Bob claims that it is possible to have an MST and a shortest path tree that are completely edge-disjoint. (Assume all roads are bi-directional, i.e., the graph is undirected. Also, assume that all edge lengths are positive and distinct.)

(a) (10 points) Is Alice's claim correct? Give a proof for your answer.

(b) (10 points) Is Bob's claim correct? Give a proof for your answer.

Problem 3 (25 points)
Show that the following algorithm for minimum spanning trees is correct: Initialize F to the empty set. In each round of the algorithm, for each connected component S of F, add the minimum weight edge in the cut $(S, V \setminus S)$ to F. Terminate when F contains a single connected component, and output F.

(Hint: First, show that F remains acyclic throughout the algorithm. Then, use the generic property of minimum spanning trees proved in class to show that the algorithm computes a minimum spanning tree.)