Worth: 8% **Due:** By 5:59pm on Tuesday 31 November

Remember to write the *full name* and *student number* of *every group member* prominently on your submission.

---

*Please read and understand the policy on Collaboration given on the Course Information Sheet. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes). For example, indicate clearly the **name** of every student from another group with whom you had discussions, the **title and sections** of every textbook you consulted (including the course textbook), the **source** of every web document you used (including documents from the course webpage), etc.*

*For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.*

---

1. Let $x_1, \ldots, x_n$ be n be a list of n people who have applied for a job at Google. They are interviewed in pairs: if x_i and x_j are interviewed together, one of them is chosen to be more qualified. You are given a list of outcomes of m interviews, each of the form "candidate x_i is more qualified than candidate x_j" Your task is to order the job candidates, with the best candidate first. Specifically, if x_i is more qualified than x_j (as the result of the outcome of some interview), then you must place candidate x_i ahead of x_j in the ordering.

   (a) How do you model this problem using a graph? Assume that all graphs in this problem set are implemented using adjacency list.

   (b) How do you determine whether it is possible at all to arrange all candidates in a line such that all constraints are satisfied? Explain your algorithm in clear English, and analyse its worst-case running time.

   (c) Assuming such arrangement is possible, how do you compute an actual arrangement? Explain your algorithm in clear English, and analyse its worst-case running time.

2. Let $G = (V, E)$ be a weighted undirected connected graph that contains a cycle, and let e be the maximum-weight edge among all edges in the cycle. **Prove** that there exists a minimum spanning tree of G which does NOT include e.

3. Consider a list of cities $c_1, c_2, \ldots, c_n$. Assume we have a relation R such that, for any i, j, $R(c_i, c_j)$ is 1 if cities c_i and c_j are in the same province, and 0 otherwise.

   (a) If R is stored as a table, how much space does it require?

   (b) Using a disjoint set ADT, write pseudo-code for an algorithm that puts each city in a set such that c_i and c_j are in the same set if and only if they are in the same province. (That is, you can assume that you have some implementation of the basic disjont set operations, MAKE-SET, FIND-SET, and UNION.)

   (c) When the cities are stored in the disjoint set ADT, if you are given two cities c_i and c_j, how do you check if they are in the same province?

   (d) If we use trees with the rank heuristic, what is the worst-case running time of the algorithm from (b) (Hint: the unions from your algorithm probably have a special form). Explain.

(e) If we use trees without the rank heuristic, what is the worst-case running time of the algorithm from (b). Explain. Are there more worst-case scenarios than in (d)?