1. Let $x_1, \dots, x_n$ 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.

   Solution: We model this problem using a directed graph with n vertices and m edges, where each vertex is a child, and edge (u, v) represents "u is more qualified than v".

   (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.
   Solution: An arrangement is possible if and only if the graph does NOT contain a cycle, which we can determine by running a DFS and detecting the existence of a back edge. The worst-case runtime is $\mathcal{O}(n+m)$.

   (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.
   Solution: Computing an actual arrangement is a topological sort problem. We perform a DFS and order the vertices according to their finish time (read Textbook Chapter 22.4 for details of the algorithm). The worst-case runtime is $\mathcal{O}(n+m)$.

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.

   Solution: Denote the cycle as $C = (V_C, E_C)$, and assume that there exists an MST that includes e (otherwise, if such an MST does not exist, the proposition is proven automatically). Let $T = (V, E_T)$. Then there must exist an edge $e' \in E_C$ that is not in E_T, because T is acyclic (by definition of a spanning tree). (Note: if C is the only cycle in G then any edge in E_C and not E_T can be picked as e'; if C is not the only cycle in G, then we need to pick such an e' that it does not create another cycle in T, i.e., pick the e' that keeps T' connected.) Then since e is the maximum-weight edge in C, we have $w(e') \leqslant w(e)$. Therefore if we remove edge e from T and replace it with e' then we get a new subgraph $T' = T - \{e\} \cup \{e'\}$, which satisfies the following properties:

   - T' is a **tree**, since we don't change the number of edges in the tree (a connected undirected graph with n vertices is a tree if and only if it has exactly $n-1$ edges, adding an edge would create a cycle, deleting an edge would disconnect it into a forest).
   - T' is a **spanning** tree, since the cycle C has only one edge e missing, which still covers all vertices in C, other vertices of G are covered by the unmodified part of T.
   - T' is a **minimum** spanning tree, because the total weight of T' is $w(T') = w(T) - w(e) + w(e') \leqslant w(T)$ because $w(e') \leqslant w(e)$. Since T is an MST, T' must be an MST.

   Hence we have an MST T' of G which does not include edge e, as desired.

3. Consider a list of cities $c_1, c_2, \dots, c_n$. 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?

Solution:

R must have an entry for every pair of cities. There are $\Theta(n^2)$ of these.

(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.)

Solution:

```
For i <- 1 to n do
  MAKE-SET(c_i)
  For j <- 1 to i-1 do
    If R(c_j, c_i) then
      UNION(c_j, c_i)
      BreakLoop
    EndIf
  EndFor
EndFor
```

(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?

Solution:

c_i and c_j are in the same province if and only if $FIND-SET(c_i) = FIND-SET(c_j)$.

(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.

Solution:

Whenever we do a union in the algorithm from part (b), the second argument is a tree of size 1. Therefore, all trees have height 1, so each union takes time $O(1)$. The worst-case running time is then $\Theta(n^2)$.

(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)?

Solution:

Because of the special case of the unions, union-by-rank does not make a difference for our algorithm. Hence, everything is the same as in part (e).