CSC263 Assignment 5

Akhil Gupta

December 2015

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

   Answer: This problem can be modelled using a directed graph. We have n people who applied for a job, therefore, our graph has N vertices. Since we have m interviews and each interview is in pairs, so our graph has m edges. Each interview represents an edge between two people. Using the adjacency list implementation, each candidate stores a list of other candidates he/she has been interviewed with.

   (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.

   Answer: In order to arrange all candidates in a line, we have to make sure that our graph does not contain any cycles i.e. acyclic graph. If our graph contains cycles, there is no possible way of arranging all the candidates in a line such that all constraints are satisfied because there is no way of knowing which candidate is better than the other, hence we reach a deadlock. In order to determine if our graph does not contain any cycle(s), we perform a Depth-First Search on our graph. If the DFS yields a back-edge, then our graph contains a cycle(s). Runtime analysis: Constructing the graph takes $\Theta(n)$ time and performing the DFS takes $\Theta(|N| + |m|)$ where N is the number of candidates and m is number of interviews performed (Week 9 lecture notes), therefore the worst-case running time is $\Theta(|N| + |m|)$

   (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.

   Answer: In order to compute an actual arrangement, we first create a graph using the conditions in part a). For example: if the outcome of an interview is "candidate A is more qualified than candidate B", then we draw a edge from A to B (arrow pointing to B). Then we perform a topological sort of the candidates in the graph. A topological sort ensures that the job candidates are ordered with the best candidate first as it will generally have more out-edges than the others. To perform a topological sort, the algorithm first performs a DFS to compute the finishing time of each node. As each node is finished, it inserts it to the front of a linked list. At the end, we just return the linked list i.e. the result is a list of nodes sorted in order of decreasing finishing times. Runtime analysis: Constructing the graph takes $\Theta(n)$ time, performing the DFS takes $\Theta(|N| + |m|)$ (from part b)) and finally inserting into a linked list takes $\Theta(1)$ constant time but since we have n candidates, it will take $\Theta(n)$ time. Therefore, the worst-case running time is $\Theta(|N| + |m|)$.

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.

   Answer: We solve this problem by performing a Proof by Contradiction.
   Suppose there exists a minimum spanning tree T of G which includes edge e in the cycle C. Let W be the graph of T without e. Since T was a tree, removing edge e results in a graph with 2 connected components W_1 and W_2. Since C is a connected cycle, there must be another edge $e' \neq e$ in C with endpoints in different connected components, which shows that e' is not in W. Also, since adding e' connects two components of the forest W, $W + e'$ is a tree. The weight of tree $W + e' \leq$ weight of tree T. This gives us a contradiction since T is the MST, so the heaviest edge in any cycle is not included in the minimum spanning tree of that graph.

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?

   Answer: If R is stored as a table, it is similar to an adjacency matrix representation of a graph. There are i rows and j columns, and since there are n cities in total, every pair of cities in the table must have an entry. Therefore, the space required is $\Theta(n^2)$.

   (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.)

   Answer: Refer to Algorithm 1 (Pseudocode)

   (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?

   Answer: The cities 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.

   Answer: From our algorithm, the line UNION(c_j, c_i), we see that the second argument is always a tree of size 1 because of the line above MAKE-SET(c_i). Since all trees have height 1, the union takes $O(1)$ constant time. Therefore, the worst-case running time of our algorithm is $\Theta(n^2)$.

   (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)?

   Answer: Regardless of whether we use trees with the rank heuristic or without, because our unions from the algorithm have a special form, it does not make a difference to the worst-case running time of our algorithm. Therefore, the worst-case running time of our algorithm is still $\Theta(n^2)$. Hence, there are no more worst-case scenarioes than in (d).

Algorithm 1 Pseudocode

```
for $i$ from 1 to $n$ do
  MAKE-SET($c_i$)
  for $j$ from 1 to $i - 1$ do
    if $R(c_j, c_i) == 1$ then
      UNION($c_j, c_i$)
      break loop
    end if
  end for
end for
```