EC330 Applied Algorithms and Data Structures for Engineers
Spring 2020

Homework 8

Out: April 23, 2020
Due: April 30, 2020

This homework only has a written part. It is due at 11:59 pm on April 30. You should submit your written solution on Gradescope. Your solution should be a single PDF file containing either typeset answers or scanned copies of hand-written answers. Make sure you write your answers clearly.

1. **Maybe Minimum Spanning Tree? [20 pt]**
   Consider the following procedure for finding a minimum spanning tree for an undirected and connected graph G.

   ```
   MAYBE-MST(G):
       while there is a cycle c in G:
             remove the maximum weight edge in c
   ```

   a. Does this algorithm always result in a minimum spanning tree of G? If not, provide a *small* graph as a counterexample. **[5 pt]**
   b. Provide an efficient algorithm for *implementing* `MAYBE-MST`. You can write your algorithm in pseudo code similar to the ones I gave in recent lectures, or write actual C++ code. Analyze the running time of the algorithm (if the worst-case running time is different from the best-case, analyze both). **[15 pt]**

2. **Prim's Algorithm [10 pt]**
   Analyze the worst-case running time of an implementation of Prim's algorithm using unordered linked-list (as the data structure for *key(v)*, the *minimum weight of any edge connecting v to a vertex in the tree*). Give your answer in **Θ**. Justify your answer (and state any assumptions you make).

3. **Single-Source Shortest Path [20 pt]**
   Explain how to modify Bellman-Ford's algorithm to efficiently count the *number* of shortest paths (of equal weight) from a source vertex to every other vertex in the graph. You can write the modified algorithm in pseudo code similar to the ones I gave in recent lectures, or write actual C++ code.

4. **All Paths from Source to Destination [20 pt]**
   Given a directed, acyclic graph *G* with *n* nodes, a source node *s* and a destination node *t* in *G*, describe an algorithm in pseudo code that can find all possible paths from *s* to *t*.

What is the running time of your algorithm? Give your answer in **O** (hint: how many paths can there be in the worst case from *s* to *t*?).

5. **Extra Edge [15 pt]**
   Consider an arbitrary tree *T* of *n* nodes with *n-1* (undirected) edges. I have added one extra edge to *T* so that it forms a cycle somewhere. Can you come up with an efficient algorithm that can remove an edge (not necessarily the one that I added) from *T* so that the resulting graph is again a tree of *n* nodes? What is the running time of your algorithm? Give your answer in **O**. Justify your answer (and state any assumptions you make). Can you do better than **O**(n^2)?

6. **Social Distancing [15 pt]**
   Consider a $n \times n$ grid-world where the value of cell (*i, j*) is 1 if there is a person standing in that cell and 0 otherwise. Suppose we use *Manhattan distance* to measure the distance between any two cells (*i, j*) and (*k, l*), i.e. the distance between (*i, j*) and (*k, l*) is $|i - k| + |j - l|$. Give a social distancing requirement *sd* (a positive integer), describe an algorithm that outputs all persons (in terms of cell coordinates) that violate the social distancing requirement, i.e. the shortest distance from another person in the grid is smaller than *sd*. You should start by describing how you construct a graph model for this problem, and then describe the graph algorithm for solving this problem. What is the running time of your algorithm? Give your answer in **O**. Justify your answer (and state any assumptions you make).