CS:3330 Homework 8, Spring 2018

Due in class on Tue, April 17

---

1. Let X be a set of n intervals on the real line. A proper coloring of X assigns a color to each interval, so that any two overlapping intervals are assigned different colors. Describe and analyze an efficient algorithm to compute the *minimum* number of colors needed to properly color X. Assume that your input consists of two arrays $L[1..n]$ and $R[1..n]$, where $L[i]$ and $R[i]$ are the left and right endpoints of the ith interval.

   The motivation for this problem is similar to the motivation for the *Interval Scheduling* problem we studied in class. Suppose that we are given a bunch of events (e.g., courses) we need to schedule, then finding the fewest number of rooms we can schedule all of these events in, is the *Interval Coloring* problem described here.

   Here is a brief description of a greedy algorithm, you should consider. Let $\mathcal{P} = \{1, 2, 3, \ldots\}$ be the set of "colors" we want to use for the interevals. Consider the intervals one-by-one in left-to-right order of start times and to each interval, assign the smallest color from $\mathcal{P}$ that is "available."

   You'll have to think pretty carefully about how to prove the correctness of this algorithm and how to present your proof. Your proof will be graded for correctness as well as clarity.

2. Problem 5.5

3. **Traveling Salesman Problem (TSP)** Given a set P of points (e.g., a bunch of cities) and pairwise "distances" between these points, the *Traveling Salesman Problem (TSP)* seeks to find a shortest tour that starts from a point and visits every point exactly once and returns to the starting point. TSP is computationally very difficult; the fastest algorithms for this problem run in exponential time and there are reasons to believe that faster algorithms do not exist for TSP. A special case of TSP is the *metric TSP* problem in which distances satisfy the triangle inequality, i.e., for any three points p_1, p_2, p_3, $distance(p_1, p_3) \leq distance(p_1, p_2) + distance(p_2, p_3)$. Even metric TSP is computationally difficult, but there are good approximation algorithms for metric TSP. In this homework you are required to implement a simple 2-approximation algorithm for metric TSP that is based on computing an MST.

   The data set We have posted a data file called `miles.dat`, which contains basic geographic data on 128 cities in North America. This data set was obtained from Donald Knuth's Stanford Graphbase. The file `miles.dat` contains, for each city, the following pieces of information: (i) name of the city, (ii) state of the city, (iii) latitude and longitude of the city, (iv) population of the city, and (v) distances to all other cities This is old data - from a 1949 Rand McNally mileage guide, so the population numbers are definitely out of date. The highway mileage may also be out of date, but the data serves our purposes quite nicely. Note that the data file specifies for each city, the distances to all cities before it in the file; this is sufficient to get pairwise distance between any pair of cities. To understand how the distances are specified take a look at the following first few lines of the data file. This data is telling us that Worcester, MA is 2964 miles from Yakima, WA, 1520 miles from Yankton, SD, and 604 miles from Youngstown, OH.

```
Youngstown, OH[4110,8065]115436
Yankton, SD[4288,9739]12011
966
Yakima, WA[4660,12051]49826
1513 2410
Worcester, MA[4227,7180]161799
2964 1520 604
```

Also note that latitudes and longitudes are specified without decimal points or N, S, E, W indicators: for example Youngstown, OH is at longitude 41.09972 N and latitude 80.64972 W and this is specified (approximately) as "[4110,8065]."

Your first coding task is to read from this data file and store the data is appropriate data structures.

MST algorithm Your next coding task is to implement Kruskal's MST algorithm. Your implementations should be faithful implementation of the algorithm discussed in class and should have asymptotic running time of $O(m \log n)$ on graphs with n vertices and m edges. For Kruskal's algorithm you will need to implement the Disjoint Set Union Find data structure using union by rank.

MST to a Traveling Salesman tour It is easy to see how to transform an MST to a Traveling Salesman tour whose total length is at most twice the length of an optimal tour. This would be a 2-approximation algorithm. Here is a fairly clear explanation of both the algorithm and the proof of why it yields a 2-approximation:

`http://www.cs.tufts.edu/ cowen/advanced/2002/adv-lect3.pdf`

Your final coding task is to implement an algorithm that takes as input an MST and returns a 2-approximate Traveling salesman tour.

What to submit (a) Output the 127 edges in the computed MST and the weight of the computed MST. (b) After transforming the computed MST into a Traveling Salesman Tour, output the tour and its weight.
