CSE 2320 Lab Assignment 4

Due December 6, 2017

Goals:

1. Understanding of hashing.
2. Understanding of graphs.
3. Understanding of Krukal's algorithm.

Requirements:

1. Design, code, and test a C program to determine a minimum spanning *forest* for a weighted, directed graph.

   a. Input is to be read from standard input (like the first three assignments):

      1. The first line is two integer values: `n`, the number of vertices, and `m`, the number of edges.
      2. The remaining `m` lines will each contain three values defining an edge: a tail name (string of no more than 25 characters), a head name (another string), and an integer value for the weight of the edge.

   b. While reading the input, new vertex names should be stored in a double hash table along with consecutively assigned vertex numbers. In addition, a table of vertex names to allow printing the name for a vertex number will be needed. If the input does not include `n` different vertex names, give a warning and continue. If the input has more than `n` different names, give a disparaging message and stop.

   c. Sort the edges by ascending weights using any technique, including those in the standard library.

   d. Initialize the disjoint-subset representation (Notes 15).

   e. Perform Kruskal's algorithm (Notes 15). You may simply overwrite the rejected edges with edges that are kept, but avoid a $\Theta\left(n^2\right)$ implementation. You may modify the code (`kruskal.c`) on the webpage

   f. Print the trees in the forest. If the input graph is connected, there will be only one tree in the final result. Otherwise, the edges should be sorted so that the edges in each tree are output together and labeled as such. Additional `find()`s will be useful. The edges in each tree should be printed in ascending order. Be sure to output edges using the vertex names, not numbers.

2. Submit your C program on Blackboard by 5:00 p.m. on Wednesday, December 6.

Getting Started:

1. Your double hash table should have the smallest prime size to assure a load factor no larger than 50%. You will need code for a simple function to compute this value.

2. You do not need an adjacency matrix or an adjacency list representation. A table of structs will be convenient.