CSE 3318 Lab Assignment 4

Due November 20

Goals:

1. Understanding of hashing.

2. Understanding of graphs.

3. Understanding of strongly-connected components (Notes 13).

Requirements:

1. Design, code, and test a C program to determine strongly connected components for a directed graph. You may modify `http://ranger.uta.edu/~weems/NOTES3318/dfsSCC.c`

   a. Input is to be read from standard input (like the first four 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 two values defining an edge: a tail name (string of no more than 25 characters) and a head name (another string).

   b. While reading the input, the vertex numbers for new vertex names should be stored in a double hash table. Vertex numbers are assigned consecutively.

      In addition to the double hash table, a non-hash table of vertex names will be needed. This is needed when printing your results for the end user and when searching the double hash table.

      If the input does not have exactly `n` different names, give a disparaging message and stop.

      The assigned vertex numbers are used to build compressed adjacency lists (Notes 13).

   c. Perform Kosaraju's SCC algorithm (Notes 13). The elements of each SCC must be output using the vertex *names*, not numbers.

   d. The following outputs are required:

      1. The size of your double hash table.
      2. The final table of vertex names (in order of first appearance in the input file).
      3. The final double hash table. If an entry is storing *never-used* (-1), no other output is needed. Otherwise, you must provide the vertex name (only stored in the non-hash table), h1, h2, and the prefix of the probe sequence that will be used to find the key.
      4. The total number of probes that will be used if a search occurs for each vertex name.
      5. The vertices in each strongly connected component produced by the second depth-first search.

2. Submit your C program on Canvas by 5:00 pm on Thursday, November 20. One of the comment lines should include the compilation command used on OMEGA.

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. Since you have not been given the details for computing h1 and h2, it is very unlikely that your double hash table will be identical to the sample outputs.

3. The SCCs for a given graph are unique, but the output order could vary.