Strongest weighted chain of people connecting two people in a social network

Huimin Sun | Hao Ding |Joe Wong

https://github.com/bamboovir/TeamProject\_New

ABSTRACT

When given a graph, there are often times when computing the shortest distance between two nodes is necessary. In particular, social networks use this type of information extensively to broadcast information or to find new possible connections between two people. Sometimes, the type of relationship will determine whether or not information is passed. For example, if Erin is graduating and posts graduation pictures, parents and friends of Erin would be interested in seeing the pictures but perhaps an acquaintance of Erin who has only met or merely heard of her may not care. This pattern continues to degrade as we move further and further down the relationship tree. We address this issue using the classic Djikstra’s Algorithm on a weighted directed graph.

ALGORITHM PSEUDOCODE

DIJKSTRA(Graph, src, dest)

FOREACH (Vertex v in Graph)

dist[v] = INFINITE //Initialize distances of every node to infinite

finished[v] = FALSE //Initialize visited array to false

prev[v] = null //Initially the parent node will not be known

Insert[v] into Queue //Initially the Priority Queue contains every vertex

dist[src] = 0 //Initialize source node’s distance to zero

WHILE(Queue is not empty)

u = EXTRACT-MIN(Queue)

finished[u] = TRUE

IF (u is our dest) //We stop here because no other path is shorter

PRINT(path from src to dest)

RETURN dist[u]

FOREACH (Vertex v adjacent to u)

minDistance = dist[u] + weight(u,v) //Temp variable

IF (dist[v] > minDistance)

THEN dist[v] = minDistance

RETURN NULL //No link exists between src and dest

INPUT FORMAT

Format of the ‘relation’ data file: Each line is a relation. Each relation's format (case-insensitive) is: [relation string], [relation distance number], [whether relation is bidirectional(optional)]   
\* This file should be generated before loading the program  
\* Note: distance should be greater than 0   
  
Here is an example of the ‘relation’ data file we’ve provided for testing purposes:

parent, 3, true (means A is B's parent, father or mother)   
friend, 5, true   
classmate, 10, true   
neighbor, 14, true   
schoolmate, 18, true   
know, 20 , NULL(means A knows B, but B doesn't know A)   
know, 20, false (means A knows B, but B doesn't know A)   
knoweachother, 20, true (means A knows B, and B knows A too)

And the ‘test’ document has the following relations:

Alden | Blaine | classmate   
Alden | Cary | friend   
Blaine | Forrest | met   
Cary | Forrest | neighbor   
Cary | Graham | familiar   
David | Forrest | friend   
Forrest | Graham | parent   
Forrest | Herman | schoolmate   
Elbert | Herman | knoweachother   
Elbert | Graham | know

Alden (Our src node)  
Elbert (Our dest node)

Our program also has the option of keying in the relations manually. The program will automatically create a new node when encountering a name it does not recognize. If the format is not recognized, the program has a buffer that will allow a few errors, however, upon excessive error detection it will exit.

HOW TO RUN

A makefile has been included in the project folder: **TeamProject/teamproject/ConsoleApplication3**

To run with manual input simply type “./teamproject” into the command line.

To run with input.txt simply type “./teamproject < input.txt” into the command line.

PROGRAM OUTPUT

After entering a source name and destination name, the program will calculate the shortest path and output the path to the console.

VIEWING GRAPH OUPUT

Our program uses GraphViz to create an image file that depicts each node and edge in the graph as a human readable chart. The file will be named output.dot located in the same directory as the makefile. To view this file, you will need to install:

Graphviz 2.38(gvedit.exe) or above

A sample image file “output.png” has been provided to show the format of the output file.