CS526 O2

Project Assignment

Nguyen Dao Vu

There are 2 Java files included in my project:

1. project.java – This is the main code for my project where the 2 algorithms are being used
2. ddNode.java – I defined a ddNode class to store a pair of value. I use this class to store corresponding node and a distance value.

In my final project, I decided to use the following data structure:

* Matrix: This is used to store the adjacency matrix of the graph input text file.
* Linked Stack:
  + shortest path : will be used to store the shortest path taken during graph traversal
  + sequence path: will be used to store all nodes on path taken during graph traversal
  + Stacks will also be used to store all adjacent nodes at each node
* Hash Map:
  + Two maps were created to match Alphabetical name of the nodes to their corresponding numeric indexes and vice versa.
  + A map was created to store the nodes and their direct distance from the text file
  + Two maps of stacks were created, one for each algorithm, where the key is the node name and the value is a stack of adjacent nodes to that key node. Each stack stores the ddNode objects where the stack is arranged in such a way where the top of the stack contains the lowest distance value and the bottom of the stack stores the largest distance value.
* Arrays:
  + Arrays will be used to store alphabetical names of nodes to be printed out.

Pseudocode:

Read all processes from the input file and store them in an appropriate data structure

Input an adjacency matrix with V nodes where edge is w(n,v) between node n and v

Input a direct distance map where node: dd(n) where dd(n) is the direct distance from node n to Z

Match each node to numeric value from 0 to V-1 in a map

//sort stack

Create method sortStack to take in a stack as input and return a sorted stack:

initiate a temp stack

While the input stack is not empty do:

temporary node = pop node from input

while temporary stack != empty & temp stack’s top node < temporary node do:

push the top node of the temp stack into the input node

push temporary node into the temp stack

return the temp stack

//both algorithms:

for i=0 to V do:

for j=0 to V do:

if an edge ( i, j) exists do:

push node j and its distance = dd(j) into stack1

push node j and its distance = dd(j) + w(i, j) into stack2

sort stack 1 //sort adjacent nodes by distance of stack 1

sort stack 2 //sort adjacent nodes by distance of stack 2

Add to a map stackOnStack1 where node i is matched with stack 1 //support algorithm 1

Add to a map stackOnStack2 where node i is matched with stack 2 //support algorithm 2

//Both algorithms use the same pseudocode as follows. Algorithm 1 will use the map //stackOnStack1 and Algorithm 2 will use the map stackOnStack2

initiate position at the start Node

add position to the shortest path

add position to the sequence path

while position != node Z do:

temp Node = pop adjacent Node to position with the smallest distance from stackOnStack

add temp node to shortest path

add temp node to sequence path

position = top node on the shortest path

if node at position has 1 or less adjacent node do:

add the adjacent node to position to sequence path

pop node out of shortest path

position = top node of shortest path