Assignment Objective: Demonstrate the course skills while implementing Dijkstra’s algorithm for determining the length of shortest paths by extending the **dGraph** ADT. Dijkstra works only on acyclic graphs. So, when testing, be sure to use an acyclic graph.

Requirements:

* Copy the p12 graph.cpp to thf.cpp.
* Copy the p12 graph.h to thf.h.
* Do not change the class name for the graph ADT.
* Add the provided public graph::dumpAllPairsShortestPath() function to your thf.cpp and the appropriate prototype of that function to your thf.h.
* Within the **thf** files, add the following additional methods and variables:
  + Public method:
    - bool dijkstra(int sLabel, int dLabel, int &distance); // this will return true if sLabel and dLabel are both valid labels of real nodes. It will also return the distance from node sLabel to node dLabel. If either label is not a valid node, distance will be set to -1.
  + Private variable:
    - int \*lambda; // This will be an array of size n that will hold distances used in performing Dijkstra
    - int \*set; // This will be an array of size n to track which set, X or Y, that a given node is in.
    - int const INFINITE = 1000000;
    - int const X = 0; // Used with “set”, this will mark a node in set X
    - int const Y = 1; // Used with “set”, this will mark a node in set Y
  + Private method:
    - void dijkstra(int s); // will populate lambda with the distances from node s to all other nodes; note that s is an index, not a label.
    - bool minLambdaY(int &minV); // If at least one node is in Y, this will return true with minV set to the index of a node in Y that has minimum value of all nodes in Y. If no node is in Y, minLambdaY returns false with minV set to -1.
* You must not use any other data structure, whether built-in in C++ or otherwise.
* An example output is provided.
* Note, the Dijkstra functionality should work for both undirected and directed graphs.
* Run your program to get the output file for your submission. A THFmain.cpp will be provided about 24 hours before the due time of the assignment.
  + Create an appropriate makefile and compile your program as follows:

make thf

* + Run your program as follows:

thf < THFinput.txt > THFoutput.txt

* Deliverables:
  + By the due date/time, put your files into D2L, not in a ZIP file.
    - A queue.h file for your queue ADT
    - A queue.cpp file for your queue ADT implementation
    - A thf.h file for your updated graph ADT
    - A thf.cpp file for your updated graph ADT implementation
    - A THFoutput.txt file containing the output
    - DO NOT CHANGE THE NAMES OF THE FILES
    - DO NOT put an IDE project into D2L
  + Turned into class: a hardcopy of the files above, except for the THFoutput.txt.