Project 2 CSc 402 Kutztown

Purpose: Data Structures with and without the STL; Graphs & Paths; Analysis
Points: Program: 25 points Paper: Course homework portion (10% of grade)
Due: Program: TBD, using the turnin script. Late submissions will not be accepted.
Paper: TBD, a docx or odt file on D2L in the appropriately named dropbox on D2L. No pdfs.
Description: Prim's method for computing a least-cost spanning tree for a graph strongly resembles Dijkstra's least cost path algorithm. There are literally hundreds of implementations of these algorithms on the web, and we are going to take advantage of that by asking you to find two implementations of Prim's algorithm, adapt them for use for this project, and write a paper about them.

Find on the internet (or write on your own) two implementations of Prim's algorithm that each use an adjacency list version of a graph **G=(V,E)**. One must use STL data structures, while the other may not use the STL at all. Both must be adapted to input files formatted with the number of vertices and edges on the first line, and with each subsequent line (the number of remaining lines is equal to the number of edges) containing the two vertices at the end of an edge, followed by the edge's cost. The output is to be the spanning tree as a list of edges, ordered by cost, as well as the spanning tree's total cost.

Apply the algorithm from a vertex input from the user. Output the actual clock time the algorithm required, in microseconds, measured from the time input from the file into the graph object ended until the time the algorithm completed its work, but before output began. This will be explained further, below.

Notes:

- The implementations can be in a different programming language, with the instructor's approval. You will be responsible for finding and clearly documenting how the timing is carried out.
- Your program must be runnable either from the command-line or interactively. If run from the command line, the argument is the file name. Submissions that can't work with an input file won't be graded.
- The user must be able to enter multiple vertices as the starting point by looping until the user enters a negative value
- An example data file will be provided in the project directory an on the web.
- You are required to have a priority queue implemented as a binary heap in both implementations.
  - An STL *priority_queue* object can be constructed with a comparator. You may need this to assure proper handling of vertices and costs in the queue. Be sure (read the STL documentation) that your priority queue implementation and operations (the STL has specific heap operations in Section 5.3) is *log n*.
- Be sure to give proper attribution whenever appropriate in your code files. Failure to properly attribute is a serious issue, both legal and ethical.
- A graph can have cycles. This should be inconsequential.
- It is possible a vertex could be unreachable. In that case, it won't be part of the spanning tree.
- To get microsecond precision in C++ requires using C++ 11. There is an example named *Microseconds* in the instructor's CSc237/Examples/Microseconds directory on the web and acad. It requires use of the *chrono* library and a switch in the makefile.

Paper: You will write a term paper to explore the efficiency of Prim's algorithm in practice. Using graphs with 5 and 10 vertices, explore the density (average in-degree and out-degree, depends on number of edges) where the algorithm slows down (if it does) for each implementation. You must consider the algorithm and provide reasoning for testing methodology.

The paper is to contain an abstract of maximum 100 words that describes your tools (the programs) and what you found. The introduction will describe the data structures used in detail in each implementation. It will also state what you hope/expect to determine. The following section describes the experiments you ran to determine the algorithm's veracity, i.e. where it slowed down in each implementation, or even if when fully connected that it didn't. In the following section describe the results. Include a chart or two (or more) showing the comparative running times of the algorithms as average in/out-degree and number of vertices varies. Lastly, provide your conclusions and, if applicable, a bibliography. Conclusions might be best compared to the claimed Big-Oh efficiency of Prim's algorithm.

There is no length requirement on this paper, but chintzy or lame work will be penalized heavily.

Turnin:
Project: Two files, named primSTL.cpp and primDS.cpp. Provide a makefile where the *make* command creates executables named *stl* and *ds*, respectively. Also submit a file named **readme.txt** (no caps) that includes the link to your Doxygen website, which must have a main page (a good place for a readme). You may also describe design decisions and anything else someone running your program should know before your paper comes out.
Paper: In the dropbox set up on D2L.