CS 380 Algorithms[1]
Path Finder - Dijkstra's Algorithm and Kruskal's Algorithm

Date Assigned: November 1, 2013 **Points: 70**

Milestone Due Dates:

1) **November 8, 2013, 4:45pm - Class Design (30 pts)**
2) **November 20, 2013, 11:59 pm - Completed Project (30 pts)**
3) **November 22, 2013, 4:45pm - Completed Project Demo (10 pts)**

For this project you will build a DarkGDK application to demonstrate Dijkstra's algorithm and Kruskal's algorithm. You will first need to import the project I give you into Subversion. There are a number of configuration options you must ensure are setup correctly to use C++ I/O functions (<< and >>) with Dark GDK.

DarkGDK and the DarkGDK setup instructions as well as the starter project are available in **\\Turing\Students\CS380-01 Public**.

You will be given a small amount of starter code but will be responsible for designing the vast majority of the classes for this project. Remember, 30% of your grade is style and design. You are free to reuse code previously produced for this class (you may need to update some of this existing code to add more functionality). Further, you are free (and encouraged) to use any code from the STL.

1Thanks to Doug Ryan for his help with the Dark GDK portion of this project.
This project was based on the PathFinder project from http://www.stanford.edu/class/cs106x

The Starter Project
The starter project will display an image on the screen and overlay a set of nodes (with labels) and edges on that image. Two buttons, Dijkstra and Kruskal, will also be displayed. In order to display the nodes and edges, a container object (or two) must be provided to the drawRoads() and drawCities() methods. This/These container object(s) must implement the necessary interface for each method to operate properly.

The Project

You will be provided a set of nodes (cities) and edges (roads) in a data file. You will need to read the nodes and edges into a graph data structure and implement the appropriate interface that will allow the starter project to display the nodes and edges over an image file.

Once displayed, the user must be able to select two cities and press the Dijkstra button on the user interface. You need to run Dijkstra's algorithm to determine the shortest path between those two cities. Highlight that path by changing the color of the edges.

If the user presses the Kruskal button, you must run Kruskal's algorithm to build the minimum spanning tree for the graph. Highlight the edges in the MST.

The Visual Studio Solution contains a number of files: **READ THIS CODE.**

- **NodeContainer.h - complete code for the abstract NodeContainer interface.**
- **EdgeContainer.h - complete code for the abstract EdgeContainer interface.**
- **RGBColor.h / RGBColor.cpp - complete code for the concrete RGBColor clas**
- Node.h - partial code defining the required interface to interact with the starter project.
- Edge.h - partial code defining the required interface to interact with the starter project.
- main.cpp - code that will display a map and buttons on the screen. You will need to write code to handle button clicks and node clicks.

You can change whatever you like in these files.

You should be able to reuse the heap to provide the priority queue necessary for Dijkstra. You should be able to reuse some sorting algorithm to provide the sorting for Kruskal's algorithm. You may need to update heap or sort to add some functionality.

Milestone 1: Design Before: Nov 8, 2013, 4:45pm
You must meet with me and provide to me a class diagram. This must include each class involved in the project as well as inheritance structure and the major methods necessary for each class. The class diagram can be hand drawn, ASCII art, PowerPoint clip art, Google Drawings, or colored construction paper.

Feel free to meet with me multiple times to review your design. You are not to share your design with other students. I want this design to be your own work.

November 8 is the deadline, but I recommend you start your design early! Once you meet with me and get your design approved you are free to write your code.

Milestone 2: Implement the project. Nov 22, 2013, 11:59pm
Milestone 3: Demo the project to Chadd. Nov 22, 2013, 4:45pm

Bonus:
In your Dijkstra and Kruskal implementations to display the step by step operation of the algorithm so we can see each edge being considered.

What to Submit

I will pull your code out of Subversion.

You must print out, in color, double-sided, and stapled, all of the source code in the project with the following exceptions. If you reuse code from a previous project and did not change that code, do not print that code.

Name your project: CS380Pathfinder_PUNetID.

Notes:

DarkGDK with C++ input/output

http://forum.thegamecreators.com/?m=forum_view&t=187925&b=22

"you will need to go to menu -> project -> <projectname> properties -> configuration properties -> C/C++ -> code generation and set runtime library to /MT (or /MTd)"

Also, you may need to:
Configuration Properties | Linker |Input |Ignore All Default Libraries | NO
Ignore Specific Default Libraries| <blank>

64 vs 32 bit Windows

DarkGDK is installed in C:\Program Files (x86)\The Game Creators\Dark GDK on 64 bit Windows machines (as in the CS Lab) and in C:\Program Files\The Game Creators\Dark GDK on 32 bit Windows machines (possibly your home computer). This can cause problems when you move a DarkGDK project from a 64- to 32-bit computer (or vice versa). You need to change Properties | Configuration Properties | VC++ Directories | {Include Directories and Library Directories} to point to the correct installation of Dark GDK.

Hints:
std::set std::map std::vector std::numeric_limits multiple inheritance

The pseudocode for Dijkstra and Kruskal in the textbook is difficult to follow. You will need to fully understand the pseudocode *and surrounding tex*t before implementing the algorithm.

Be sure to use MEM_DEBUG to check for memory leaks. It appears the DarkGDK has contains some memory leaks. You should only be concerned about leaks that list a file and line number that is inside your code. Be sure to include mem_debug.h in every file you produce that you want to check for memory leaks.

► An example executable is posted in \\Turing\Students\CS380-01 Public! This example has the bonus done for Kruskal.