**Tutorial: Exploring Dijkstra’s Shortest Path Algorithm**

You are presented with the following graph which shows a set of nodes, each presented as a box, connected together by traffic paths represented as lines with arrows. Each traffic path is uni-directional and connects two boxes; the path has an associated cost which is represented by a number on the right-hand side of the path.

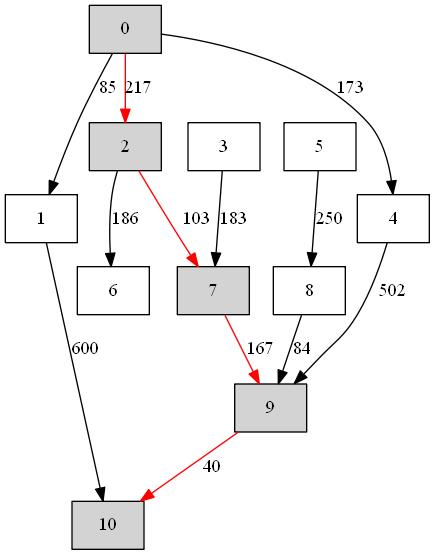


Figure 1 A unidirectional graph showing the least cost path from node 0 to node 10.

**The shortest path from node 0 to 10**

Figure 1 shows the shortest path from node 0 to node 10. This has been calculated using an implementation of Djikstra’s shortest path (least cost path) algorithm. You are given the code for this in several Java files but some key code has been removed.

Your task in this tutorial is restore the code to full working order and to test its execution with the graph in Figure 1. Show that the path marked in red is calculated by your code as the shortest path between nodes 0 and 10.

**The code and a test**

The code is composed of the following files

Vertex.java

Edge.java

Graph.java

DijkstraAlgorithm.java

TestDijkstraAlgorithm.java

You should use this tutorial to practice using the CMD window calling the Java compiler and runtime directly. You may use a GUI if you prefer and known how to link in multiple classes.

You could create a directory such as C:\temp\dijkstra on the system and place all the files there. Then open the CMD window and type

cd c:\temp\dijkstra

For each <file> execute the javac compiler. For example

javac Vertex.java

Some of the code needs to be modified before it will compile properly.

The execution of the test takes place from the file

TestDijkstraAlgorithm.java

Study the contents of this file carefully and write notes for yourself on what is taking place.

When you have all files compiling successfully then execute the test by invoking the Java run-time on the appropriate class file

java TestDijkstraAlgorithm

Compare the results to that shown in Figure 1.

**Extending the functionality**

Now extend the code in TestDijkstraAlgorithm.java to explore other paths. Confirm that the code is working correctly by looking at the figure and working out manually the cheapest path. Can you add a timing analysis to determine how long it takes to run the algorithm?

**[ End Document ]**