# ECE 275 Assignment 5

**DUE DATE:** Wednesday, December 10, 11:59PM

In this assignment, you will create a C++ application named weighs that searches for the optimal route to travel to a location based on the shortest, and longest, potential travel times along a route.

The assignment name for this assignment is: **weighs**

## Command-line Arguments

Your program must be capable of utilizing command-line arguments as follows.

weighs start end streetMapGraphFile

* start is the id of the starting waypoint in the graph (i.e., your current location).
* end is the id of the ending waypoint in the graph (i.e., your destination).
* streetMapGraphFile is the name of the input graph file using the format described below.

As usual, if the command line argument is incorrectly specified your program should print a usage statement and exit.

## Graph Input File

The graph input file consists of two sections: WAYPOINTS and STREETS, which will be defined in that order (i.e., the WAYPOINTS section must come before the STREETS section.

The WAYPOINTS section lists all nodes in the graph, and may list nodes that are potential stopping points (i.e., a store on the south side of an eastbound route). Each waypoint has a unique integer id, and a (potentially not unique[[1]](#footnote-1)) waypointName. The waypointName can (and usually will) have spaces and could also consist of other kinds of descriptions for waypoints for navigating complex intersections (e.g., Campbell and Speedway northbound right turn lane is a potential name for a waypoint.

The STREETS section lists all streets in the map. Every street has four properties startId, endId, minTravelTime, and maxTravelTime. The properties startId and endId represent the two waypoints linked by this street (in that direction). The properties minTravelTime and maxTravelTime represent the range of travel time (in minutes) that this street could take. The graph input file will specify a directed graph structure using the following format:

WAYPOINTS:

id waypointName

STREETS:

startId endId minTravelTime maxTravelTime

All properties will be separated by whitespace. Following is a sample streetMapGraph file:

WAYPOINTS:

1 Campbell and Speedway

2 Campbell and Grant

3 Campbell and Fort Lowell

STREETS:

1 2 1.6 4

2 3 3.5 8

2 1 1 2.1

3 2 3 4.3

Clearly in this example file travel in either direction may not take the same amount of time. You have probably noticed that this is the case in your previous travels—road networks are asymmetric. Newer routing tools dynamically change the weights of edges during travel (or predict what the weights will be during travel so as to avoid rush hour in certain parts of town). These kinds of things are outside the scope of this course, but may give you an idea for just how sophisticated graph-based techniques can be in solving problems.

Your program should ensure that the input file is correctly specified. Waypoints and streets must be unique, which leads to the following set of potential errors[[2]](#footnote-2):

* Duplicate waypoint id
* More than one street between any two waypoints in the same direction
* Street source or destination waypoint id does not exist
* The values passed in for start and end do not match a waypoint id
* Max travel time is strictly less than min travel time for an edge
* Min or max travel time is invalid (less than 0, or not a number)

When errors like these are found, the program cannot execute so it should exit after printing an appropriate error message.

## Travel Path Requirements

The weighs program should search the provided graph to find travel path types according to the following requirements:

* Shortest path – the first path returned should be the shortest possible path in terms of time taken.
* Most reliable path – the second path returned should have a time difference between its maximum and minimum times that is at least as small as that of the shortest path, and should not have a minimum time that is longer than the shortest path’s maximum time; *paths with a minimum time longer than the max time of the shortest path should not be considered*.
* Each returned path must start, and end, as specified in the waypoints passed in on the command line.

If there is a tie for one of these paths, it doesn’t matter which path you choose.[[3]](#footnote-3)

## Printing out the Path

The weighs program should report each path using the following format:

Shortest path: startId to endId (minTime to maxTime)\n  
 startWp\n  
 wp1\n  
 wp2\n  
 ...\n  
 wpn\n  
 endWp\n  
\n  
Most reliable path: startId to endId (minTime to maxTime)\n  
 startWp\n  
 wp1\n  
 wp2\n  
 ...\n  
 wpn\n  
 endWp\n  
\n

startId and endId are the id values for the start and end waypoints (as passed in on the command line). For startWp and endWp, as well as wp1, wp2, ..., wpn these are the *names* of the waypoints taken along the way.

If no paths are found to go from the two waypoints (i.e., “You can’t get there from here”), then you should print only this statement:

No routes found.

## Extra Credit (10%)

Extra credit will be awarded for a program that also plots a return trip back to the starting waypoint[[4]](#footnote-4). To invoke your program to plot the return path, the following command line should be used instead[[5]](#footnote-5):

weighs start end –roundtrip streetMapGraphFile

Note that if the additional command line is not used, then the normal invocation will be taken. When you print out the results using the –roundtrip command, the output should be as follows:

Shortest roundtrip path: startId to endId (minTime to maxTime)\n  
 startWp\n  
 wp1\n  
 wp2\n  
 ...\n  
 wpn\n  
 endWp\n  
 wp1a\n  
 wp1b\n  
 ...\n  
 wp1n\n  
\n  
Most reliable roundtrip path: startId to endId (minTime to maxTime)\n  
 startWp\n  
 wp1\n  
 wp2\n  
 ...\n  
 wpn\n  
 endWp\n  
 wp1a\n  
 wp1b\n  
 ...\n  
 wp1n\n  
\n

To be considered for the extra credit, you must include a statement in your README.txt. In addition, you should name your submission archive using the format NetID\_weighs\_ec.tgz.

1. Previously waypoint names were mentioned to be unique, but you should not depend on this. [↑](#footnote-ref-1)
2. Adjusted to a list to make for easier checkoff, and also stated in a consistent way. [↑](#footnote-ref-2)
3. I made this easier by taking out the possibility of duplicate paths. [↑](#footnote-ref-3)
4. In the most reliable case, the trips to/from the destination can be considered as different paths when deciding whether a path should be eliminated based on the max travel time of the shortest path. [↑](#footnote-ref-4)
5. Revised to indicate that start/end are the same as for the default case. [↑](#footnote-ref-5)