A Priority Call

CS361, Spring 2021

Contents:

1 Instructions

1. Read the instructions on How to Prepare and Submit Assignments.
2. Read the problem description below.
3. Files for this assignment may be found here or, if you are logged in to one of the CS Dept. Linux machines, in `~zeil/Assignments/cs361/graphs_emergency/`.
4. Complete the implementation of the `City` class in `city.h` and `city.cpp`.
5. Use the button below to submit your `city.h` and `city.cpp files:

   Submit this assignment

2 Problem Description

The 911 Emergency Dispatch Office for the town of Outer Lower Podunk is proud of the new equipment they have installed. GPS transceivers in each of their fire trucks, ambulances, and other emergency vehicles allow them to track the current position of each vehicle. A special map shows the closest street intersection to each vehicle and to the location of any incoming emergency calls. Careful studies have been conducted to determine the average amount of time required to travel from one intersection to another.

With this information, the 911 dispatchers hope that they can quickly determine, for any emergency call, the vehicle that is positioned to most quickly respond to that call.

Unfortunately, the money for all of this new equipment cam from the city's road maintainence budget. Upkeep of the roads has suffered, with numerous deep potholes opening up that threaten that could severely damage vehicles passing over them. Safety experts for the city have placed a strict limit on the number of potholes that a

vehicle should encounter on any given emergency run, lest an emergency vehicle be unable to respond to a call because of a broken axle or damaged wheel.

2.1 The Program

The main application program is named `dispatcher`. It receives its input from a file named in a command-line parameter, e.g.

```
./dispatcher sample.dat
```

There is also a unit test that covers some very basic cases.

There is also a program in `randGenerator.cpp` that can be used to generate random input sets of a desired size.

2.2 Input

Input to this program comes from the file named as a command line parameter.

The first line of input contains three integers:

- `s`, the integer identifier of the intersection where the emergency vehicle is currently located. $0 \leq s \leq 5000$
- `f`, the integer identifier of the intersection to which the vehicle needs to go. $0 \leq f \leq 5000$
- `p`, the maximum number of potholes the vehicle may pass over safely. $0 \leq p \leq 200$

This is followed by an arbitrary number of lines, each containing 4 numbers describing a road segment.

- `i`, the integer identifier of the intersection at one end of the road segment. $0 \leq i \leq 5000$
- `j`, the integer identifier of the intersection at the other end of the road segment. $0 \leq j \leq 5000$, $i \neq j$
- `t`, the time in minutes that a vehicle will require to travel the road segment. $0 < t \leq 240$
- `h`, the number of potholes in this road segment. $0 \leq h \leq 1$

Roads may be traveled in either direction, and `t` and `h` remain the same regardless of the direction traveled.

No two lines of input will contain the same (i, j) pair, nor will any line contain a (j, i) pair corresponding to another line's (i, j).

2.3 Output

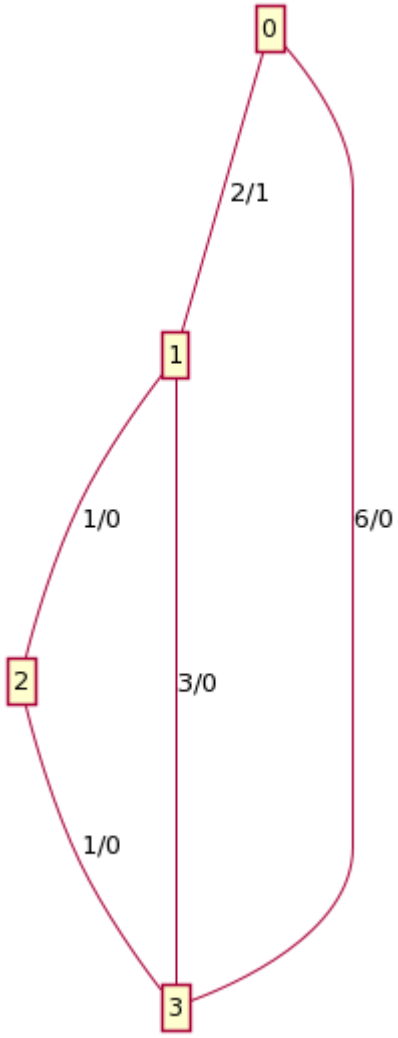
For each (emergency, vehicle) pair, the program prints a message indicating the total time reuqired for the fastest path from the vehicle to the emergency location that does not exceed the allotted maximum number of potholes.

It is possible that no such path from the vehicle to the emergency location exists, in which case a message to that effect will be printed.

2.4 Example

For the input

```
0 3 0
0 1 2 1
0 3 6 0
1 2 1 0
1 3 3 0
2 3 1 0
```



the output should be

```
Fastest time for car at 0 to the emergency at 3 is 6 minutes.
```

For the input

```
0 3 1
0 1 2 1
0 3 6 0
1 2 1 0
1 3 3 0
2 3 1 0
```

(the same graph, but allowing one pothole) the output should be

```
Fastest time for car at 0 to the emergency at 3 is 4 minutes.
```

For the input

```
0 3 0
0 1 2 1
0 3 6 1
1 2 1 0
1 3 3 0
2 3 1 0
```

(adding a pothole to the $(0, 3)$ road) the output would be

```
There is no way for the car at 0 to reach the emergency at 3
```

3 Notes

1. This problem is very similar to a classic problem described in the lecture notes, but the potholes add an additional complicating factor.

   The `graphUtils.*` files contain code for some of the graph algorithms covered in the lecture notes. You will probably not be able to use these directly, but may be able to adapt one of them to suit this problem.

2. Due to a limitation in the cross-compiler used to produce Windows binaries, only the Linux version of the compiled solution will be available for this assignment.