**Graphs: RoboCourier**

**(this is a hard one)**

The TopCoder website offers good advice on solving computer problems, including specifically graph problems. <https://www.topcoder.com/community/data-science/data-science-tutorials/introduction-to-graphs-and-their-data-structures-section-1/>

(The Senior Computing Team participates in TopCoder Competitions.)

Here is one of TopCoder's graph problems. Write a solution. Good luck!

|  |
| --- |
| Problem Statement for **RoboCourier** |
| |  |  | | --- | --- | |  | | |  | A robotic courier needs to deliver a package through a minefield, following a safe path discovered by a robotic scout. The scout's path is not necessarily as efficient as it could be. For example, it might make unnecessary turns or loop back on itself. The courier should follow the most efficient path, taking shortcuts, but only when it can do so safely.  Each robot can make only three kinds of moves, each represented by a single letter: it can go forward 1 meter ('F'), pivot right 60 degrees ('R'), or pivot left 60 degrees ('L'). Notice that the locations on which a robot can begin or end a move form a hexagonal grid. The scout and courier begin in the same location, facing in the same direction. The courier's goal is to reach the last location visited by the scout as quickly as possible. To travel safely, the courier must always stay in the wheel tracks of the scout. That is, any forward movement made by the courier must be along a path segment traveled by the scout. Pivoting left or right is always safe. Note that the courier is permitted to follow a path segment in either the same or the opposite direction as the scout. Similarly, the courier may be facing in any direction when it reaches its final destination--it need not end up facing in same direction as the scout.  The courier requires 3 seconds to pivot left or right 60 degrees, and 4 seconds to move forward one meter. However, because of acceleration effects, the courier can move faster when it makes several consecutive forward moves. The first and last moves in any such sequence take 4 seconds each, but intermediate moves in the sequence take 2 seconds each. For example, it would take the courier 20 seconds to travel 8 meters in a straight line: 4 seconds for the first meter, 4 seconds for the last meter, and 12 seconds for the six meters in between.  For example, suppose the scout follows the path "FRRFLLFLLFRRFLF" (all quotes for clarity only). Altogether, it visits six locations, which can be drawn as  \_  /6\\_  \\_/5\\_  \\_/4\  /2\\_/  \\_/3\  /1\\_/  \\_/  To repeat, suppose the scout follows the path "FRRFLLFLLFRRFLF" (all quotes for clarity only). Altogether, it visits six locations, which can be drawn as  \_  /6\\_  \\_/5\\_  \\_/4\  /2\\_/  \\_/3\  /1\\_/  \\_/  It begins in hexagon 1, facing upward, and travels in order to hexagons 2, 3, and 4. It then returns to hexagon 2 before continuing on to hexagons 5 and 6. The courier could deliver the package in a minimum of 15 seconds, using the path "FFLF" which visits hexagons 1, 2, 5, and 6.  The scout's path will be given as a String[] **path**, rather than as a single String. For example, the scout's path above might have been given as {"FRRFLLFLL", "FRRFLF"}. Note that there is no significance to where the breaks fall between strings in **path**; it is best to think of all the strings being concatenated together. Given the **path**, you must calculate the minimum time needed for the courier to deliver its package. | |  | | | Definition | | |  | |  |  | | --- | --- | | Class: | RoboCourier | | Method: | timeToDeliver | | Parameters: | String[] | | Returns: | int | | Method signature: | int timeToDeliver(String[] path) | | (be sure your method is public) | | | |  | | | Constraints | | | - | **path** contains between 1 and 10 elements, inclusive. | | - | Each element of **path** contains between 1 and 50 characters, inclusive. | | - | Each element of **path** contains only the characters 'F', 'L', and 'R'. | |  | | |  | | | Examples | | | 0) |  | |  | |  |  | | --- | --- | | |  | | --- | | { "FRRFLLFLLFRRFLF" } | | | Returns: 15 | | |  | | --- | | The example above. | | | | 1) |  | |  | |  |  | | --- | --- | | |  | | --- | | { "RFLLF" } | | | Returns: 17 | | |  | | --- | | Even though the ending location is one meter in front of the starting location, the courier cannot simply go forward, because that would not be safe. It must follow in the tracks of the scout. | | | | 2) |  | |  | |  |  | | --- | --- | | |  | | --- | | { "FLFRRFRFRRFLLFRRF" } | | | Returns: 0 | | |  | | --- | | Scout ends up at starting location. | | |  | |  | | | 3) |  | |  | |  |  | | --- | --- | | |  | | --- | | { "FFFFFFFFFRRFFFFFFRRFFFFF",  "FLLFFFFFFLLFFFFFFRRFFFF" } | | | Returns: 44 | | |  | | --- | | The shortest path is "FFFRFFFFFFRFFFF". | | | | 4) |  | |  | |  |  | | --- | --- | | |  | | --- | | { "RFLLFLFLFRFRRFFFRFFRFFRRFLFFRLRRFFLFFLFLLFRFLFLRFF",  "RFFLFLFFRFFLLFLLFRFRFLRLFLRRFLRFLFFLFFFLFLFFRLFRLF",  "LLFLFLRLRRFLFLFRLFRF" } | | | Returns: 24 | |  | | | 5) |  | |  | |  |  | | --- | --- | | |  | | --- | | { "LLFLFRLRRLRFFLRRRRFFFLRFFRRRLLFLFLLRLRFFLFRRFFFLFL",  "RLFFRRLRLRRFFFLLLRFRLLRFFLFRLFRRFRRRFRLRLRLFFLLFLF",  "FRFLRFRRLLLRFFRRRLRFLFRRFLFFRLFLFLFRLLLLFRLLRFLLLF",  "FFLFRFRRFLLFFLLLFFRLLFLRRFRLFFFRRFFFLLRFFLRFRRRLLR",  "FFFRRLLFLLRLFRRLRLLFFFLFLRFFRLRLLFLRLFFLLFFLLFFFRR",  "LRFRRFLRRLRRLRFFFLLLLRRLRFFLFRFFRLLRFLFRRFLFLFFLFR",  "RFRRLRRFLFFFLLRFLFRRFRFLRLRLLLLFLFFFLFRLLRFRLFRLFR",  "LLFLFRLFFFFFFFRRLRLRLLRFLRLRRRRRRRRLFLFLFLRFLFRLFF",  "RLFRRLLRRRRFFFRRRLLLLRRLFFLLLLLRFFFFRFRRLRRRFFFLLF",  "FFFFLRRLRFLLRRLRLRFRRRRLFLLRFLRRFFFRFRLFFRLLFFRRLL" } | | | Returns: 169 | | |  |  | |  |  |  |  |  |  |

Here is a discussion of RoboCourier, though its code is in C++. Scroll down to the bottom of the page:

<http://www.topcoder.com/tc?module=Static&d1=match_editorials&d2=srm150>

Hello and thanks for your inquiry.

You may use any TopCoder problem that's available in our archive for educational and non-commercial purposes.  There are a few requirements:

1) Any reproduction of a TopCoder problem statement must include any trademark or copyright statement from the original statement.  Proper credit must be given to TopCoder, including a link to the TopCoder website ([www.topcoder.com](http://www.topcoder.com/)).

2) The problem statements must be used "as is" with no modifications.

Regards,

Mike Lydon

CTO

TopCoder, Inc.