Challenge 03: Shortest path

September 23, 2019

The challenge

You are in charge of route planning for a brand new navigation software called Freerunner. The software, aimed at providing pedestrians directions, represents the world on a rectangular grid.

When directions from a source to a destination are requested, Freerunner computes the directions as a series of LEFT turns, RIGHT turns, an amount of time the user should walk forward, and an ARRIVAL event.

Freerunner makes some assumptions about the world by abstracting physical land into unit blocks. It assumes that it takes one (1) time unit to travel forward one (1) block.

For example, a series of instructions `10R1L3R0A` indicates that a user should travel straight for 10 blocks, make a 90 degree right turn, travel forward 1 block, make a 90 degree left turn, travel forward 3 blocks, then make a 90 degree right turn to arrive at their destination.

Furthermore, Freerunner is an app for parkour enthusiasts who want to get from one place to another as quickly as possible. It aims to generate the shortest paths, even traversing through building and other obstacles, because users will find a creative way around them.

Unfortunately, due to a bug in the mapping software, the open-world — no obstacle is impassable — model is often violated, and sometimes the directions it computes are not very efficient!! It will often have people walking for far longer than necessary!

The product managers at Freerunner anticipate a huge market of scuba parkourers in the coming year and *insist* that feature development for underwater navigation needs to be released tomorrow! There is no time to fix the bug! Instead your team has been given one working hour to create a work-around.

Your task is to take a list of instructions generated by the buggy Freerunner server and create a new list of instructions which minimize walking time.

Additional constraints

- You *must* write your solution using Python 3 code. Yes, even if you don't know it. See the syntax primer.
- You *cannot* use the Internet! Moderators will be spying on you to make sure you don't cheat!
- Every 20 minutes there will be a rotation within your group. Make sure your teammates can pick up where you left off!

Illustrated example

Imagine that you live in a city bound by the following 10 block by 10 block grid. You begin in the place marked *S*, and want to arrive at the place marked *E*.

```
+-+-+-+-+-+-+-+-+-##E
+-+-+-+-+-+-+-+-###-+
+-+-+-+-+-+-+-###-+-+
+-+-+-+-+-+-###-+-+-+
+-+-+-+-+-###-+-+-+-+
+-+-+-+-###-+-+-+-+-+
+-+-+-###-+-+-+-+-+-+
+-+-###-+-+-+-+-+-+-+
+-###-+-+-+-+-+-+-+-+
S##-+-+-+-+-+-+-+-+-+
```

The shortest route is depicted in the grid by `#` signs.

Freerunner will unfortunately generate instructions that might create a path like this:

```
+-+-+-+-+-+-+-+-+-+-E
+-+-+-+-+-+-+-###-+-#
+-+-+-+-+-+-+-#-#####
+-+-+-+-+-+-+-#-+-+-+
#########-+-+-#-+-+-+
#-+-+-+-#-+-+-#-+-+-+
#####-+-#######-+-+-+
+-+-#-+-+-+-+-+-+-+-+
+-+-#-+-+-+-+-+-+-+-+
S####-+-+-+-+-+-+-+-+
```

Such a path might avoid buildings, etc. but it's absolutely no good for the parkour enthusiasts that Freerunner is targeting!

Input format

The input to your program will be a sequence of repeated letters and numbers. There will always be at least one number before the next letter.

For example, the following are valid navigation instructions:

1. `0R10L12R10R10R10R3L1A`
2. `10A`
3. `0A`
4. `4L2R0L0R0L0R0L10A`

While the following are not valid instructions:

1. `A`
2. `10RR3LA`
3. `10AA`

etc.

Output format

The output format is the same as the input format.

You can validate the correctness of your program if, when an output from your program is fed as the input, the same path is produced as the output.

The most trivial case of this would be

$$0A \rightarrow program \rightarrow 0A$$

Running the program

Ensure that your python program is executable and accepts the following options.

```
./pathfix.py INPUT [OUTPUT]
```

Where INPUT is the path to an input text file, and OUTPUT is an optional path to the output file.

If no output file is supplied, then your program should print the results to the console (`stdout`).

In other words, your program should be able to run like this:

```
./pathfix.py input.txt output.txt
```

or like this:

```
./pathfix.py input.txt
```