Computer Science 220, S1 2023

Assignment 4 (traversal and optimisation)

See Canvas for due dates

This assignment requires you to submit programs in Python that you have written yourself to the automarker, `https://www.automarker.cs.auckland.ac.nz`. Your implementation must be from first principles and cannot use an existing library methods that might solve the problem (eg performs graph operations etc).

The automarker runs on a Linux box. Read the automarker help and FAQ for more details.

Please submit only Python source code (`.py` extensions only).

1. **Arithmetic trees** *30 marks*

   You are given an input file with multiple pairs of input lines. The first line of each pair is a tree given as a predecessor array. The second line is the value at the corresponding node. Values at leaf nodes (nodes with no children) are integers. At non-leaf nodes, the two possible values are $+$ or $*$.

   The tree represents an arithmetic expression where the value at a non-leaf node u is the sum of values at the children of u in the case of $+$, or the product of values at the children of u in the case of $*$.

   You need to calculate the value at each node and output the calculated value at the root. The tree is not constrained to be binary.

   Input format: Input consists of m pairs of lines of comma separated values, so $2m$ lines in total. The first line is each pair is a comma separated list of integers representing a tree in predecessor array format where -1 represents null.

   The second line in each pair is a comma separated list of integers and the symbols $+$ and $*$. The ith item on the list is the value or operator at the ith node in the tree.

   For example:

   ```
   -1,0,0,0,1,1
   +,*,2,3,0,7
   2,0,-1,0
   +,3,*,3
   ```

   Output format: For each pair of input lines, output a line containing the value calculated at the root of the tree.

   For the example input above, output would be:

   ```
   5
   6
   ```

2. **Optimisation** *30 Marks*

A frog needs to navigate its way from its current position to its next meal through a dangerous landscape. To do so, it leaps from boulder to boulder, but it can jump at most 1m. Your aim is to find the length of the shortest path to get the frog to its next meal using only boulders.

The landscape is a $n \times n$ square (units are cm) with boulders at exact positions given by coordinates (x, y) where $0 \leq x, y \leq n$. Boulders are scattered across the landscape. Use Euclidean distance to calculate the distance between boulders.

Input format: The input is taken from the keyboard (e.g. by sys.stdin) as multiple lines of comma separated numbers. Each line has $2p + 1$ numbers where $p \geq 2$. The first number on each line is the size of the landscape, n.

The following $2p$ numbers give locations of p boulders, so the jth boulder is at $(2j, 2j + 1)$.

The first position listed on each line is the frog's current position, the final position listed is the target boulder where the frog will find its next meal.

For example:

```
100,0,0,0,100,100,100
1000,20.892,986,602,138.97,206.2,10.44
200,25,25,10,1,50,25,140,30
```

Output format: For each line of input, output a single number to the console which is the length of the shortest path from the starting position to the next meal. Use `str.format` to give this value to 2 decimal places. Precisely, format `x` using `'{:.2f}'.format(x)`. Do not use any other rounding throughout your algorithm. If the next meal is unreachable from the origin, output -1.

For the example input above, output would be:

```
200.00
-1
115.14
```

Marking

The maximum number of submissions for each problem is fixed at 12. Each problem has three test cases associated with it worth one third of the marks for that problem. Some of the test cases will be large to test for efficiency. You get full marks if you pass all test cases.