[CS 11 25.1] HOPE 3 – Trees

Cheatsheet is available here: https://oj.dcs.upd.edu.ph/cs11cheatsheet/

Submit solution [CS 11 25.1] HOPE 3

Points: 235 (partial)
Time limit: 12.0s
Memory limit: 2G

Problem type

Allowed languages
py3

Problem Statement

You are inside a Shroom Raider level!

As a refresher: the level has r rows and c columns. The rows are numbered 0 to $r-1$ from top to bottom, and the columns are numbered 0 to $c-1$ from left to right. The cell at row i and column j is denoted as (i, j).

You are currently on cell $(r-1,0)$, and you want to reach cell $(0, c-1)$. Because you don't want to get lost, you can only move either up or right.

Each cell is either empty (denoted as `.`), has a tree (denoted as `T`), or has an axe (denoted as `x`). Like with Shroom Raider, you can only hold one axe at a time, and if you use an axe on a tree, that axe can no longer be used.

How many ways are there to go from $(r-1,0)$ to $(0, c-1)$? Two ways are considered different if one of the cells you visit differs between them.

The answer may be ridiculously big, so **only return the remainder when it is divided by** $1{,}000{,}000{,}000$.

You will probably need the fact that `(a + b) % m == (a % m) + (b % m)`.

Task Details

Your task is to implement a function named `num_paths`, which should start like this:

```
def num_paths(level):
```

Here, `grid` is a `Sequence` of strings denoting the level you are on.

The function must return an integer as described in the problem statement.

Ensure that you return the remainder when the answer is divided by $1{,}000{,}000{,}000$**.**

Restrictions

Your source code must have at most $2{,}000$ bytes.

Examples

Example 1 Function Call

```
num_paths((
    "..",
    "..",
))
```

Example 1 Return Value

```
2
```

Example 2 Function Call

```
num_paths((
    "...",
    "TT.",
    ".T.",
))
```

Example 2 Return Value

```
0
```

Example 3 Function Call

```
num_paths((
    ".xT.",
))
```

Example 3 Return Value

```
1
```

Constraints

- The function `num_paths` will be called at most $70{,}000$ times.
- $1 \le r, c$
- $rc \le 350{,}000$
- The sum of rc across all calls to `num_paths` will be $\le 350{,}000$.
- Cells $(r-1,0)$ and $(0, c-1)$ will not contain a tree or an axe.

Scoring

Note: New tests may be added and all submissions may be rejudged at a later time. (All future tests will satisfy the constraints.)

- You get 40 ❤️ points if you solve all test cases where:
  - $rc \le 50$
  - The sum of rc across all calls to `num_paths` will be ≤ 100.
  - There are no trees or axes.
- You get 40 ❤️ points if you solve all test cases where:
  - $rc \le 50$
  - The sum of rc across all calls to `num_paths` will be ≤ 100.
  - There are no axes.
- You get 40 ❤️ points if you solve all test cases where:
  - There are no trees or axes.
- You get 40 ❤️ points if you solve all test cases where:
  - There are no axes.
- You get 20 🔴 points if you solve all test cases where:
  - $rc \le 50$
  - The sum of rc across all calls to `num_paths` will be ≤ 100.
- You get 55 🔴 points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.