--- Day 18: RAM Run ---

You and The Historians look a lot more pixelated than you remember. You're inside a computer at the North Pole!

Just as you're about to check out your surroundings, a program runs up to you. "This region of memory isn't safe! The User misunderstood what a pushdown automaton is and their algorithm is pushing whole **bytes** down on top of us! Run!"

The algorithm is fast - it's going to cause a byte to fall into your memory space once every nanosecond! Fortunately, you're **faster**, and by quickly scanning the algorithm, you create a **list of which bytes will fall** (your puzzle input) in the order they'll land in your memory space.

Your memory space is a two-dimensional grid with coordinates that range from `0` to `70` both horizontally and vertically. However, for the sake of example, suppose you're on a smaller grid with coordinates that range from `0` to `6` and the following list of incoming byte positions:

```
5,4
4,2
4,5
3,0
2,1
6,3
2,4
1,5
0,6
3,3
2,6
5,1
1,2
5,5
2,5
6,5
1,4
0,4
6,4
1,1
6,1
1,0
0,5
1,6
2,0
```

Each byte position is given as an `X,Y` coordinate, where `X` is the distance from the left edge of your memory space and `Y` is the distance from the top edge of your memory space.

You and The Historians are currently in the top left corner of the memory space (at `0,0`) and need to reach the exit in the bottom right corner (at `70,70` in your memory space, but at `6,6` in this example). You'll need to simulate the falling bytes to plan out where it will be safe to run; for now, simulate just the first few bytes falling into your memory space.

As bytes fall into your memory space, they make that coordinate **corrupted**. Corrupted memory coordinates cannot be entered by you or The Historians, so you'll need to plan your route carefully. You also cannot leave the boundaries of the memory space; your only hope is to reach the exit.

In the above example, if you were to draw the memory space after the first `12` bytes have fallen (using `.` for safe and `#` for corrupted), it would look like this:

```
...#...
..#..#.
....#..
...#..#
..#..#.
.#..#..
#.#....
```

You can take steps up, down, left, or right. After just 12 bytes have corrupted locations in your memory space, the shortest path from the top left corner to the exit would take **22** steps. Here (marked with `O`) is one such path:

```
OO.#OOO
.O#OO#O
.OOO#OO
...#OO#
..#OO#.
.#.O#..
#.#OOOO
```

Simulate the first kilobyte (`1024` bytes) falling onto your memory space. Afterward, **what is the minimum number of steps needed to reach the exit?**

To begin, get your puzzle input.

Answer: [Submit]

You can also [Share] this puzzle.