Portals

There is a cake placed inside a labyrinth and you want to eat it. You have acquired a portal gun from Aperture Science™, and you will use it to reach the cake in the shortest possible time. You have a map of the labyrinth, which is a grid with `H` rows and `W` columns. Each grid cell contains one of the following characters:

`#` — walled off square

`.` — open square.

> Some indentation would be nice. Perhaps define a new environment?

You may only walk on the open squares. Additionally, the rectangular area depicted on the map is surrounded by walls from the outside with no open spaces.

The portal gun can at any time fire a portal in one of the four directions *up*, *left*, *down* and *right*. When you fire a portal in a direction, it will fly in that direction until it reaches the first wall. When this happens, a portal will be spawned on the side of that wall that faces you.

At most two portals can exist at any time. If two portals are already placed in the labyrinth, then one of them (selected by you) will be removed immediately upon using the portal gun again. Firing a portal at a wall where there is already a portal placed will replace that portal (there may be at most one portal per side of wall). Notice that there may be multiple portals placed on different sides of same wall.

> Did I get this note right?

When two portals are placed on the map, you can use them to teleport yourself. When standing next to one of the portals, you can walk into it and end up at the square next to the other portal. Doing this takes as much time as moving between two adjacent squares.

You may assume that firing portals does not take time and moving between two squares (or teleporting through portals) takes one unit of time.

Task. Given the map of the labyrinth together with your starting location and the location of the cake, calculate how much time you need to reach the cake.

Implementation. Write a function `least_time(H, W, M, X, Y, U, V)` that takes the following parameters:

- `H` — the number of rows in the map
- `W` — the number of columns in the map

- `M` — a two–dimensional array that describes the map. Each entry `M`$[i][j]$ ($0 \leq i \leq$ `H` $- 1$, $0 \leq j \leq$ `W` $- 1$) is the cell in the i–th row and j–th column of the map. It is

  `.` if the corresponding place is open

  `#` if there is a wall.

- `X` — row of your starting location
- `Y` — column of your starting location
- `U` — row of the cake's location
- `V` — column of the cake's location

Your starting location and location of the cake are positions on the map (that is, $0 \leq$ `X`, `U` $\leq$ `H` $- 1$, $0 \leq$ `Y`, `V` $\leq$ `W` $- 1$.)

It is guaranteed that you can reach the cake from your starting location. In particular, locations (`X`, `Y`) and (`U`, `V`) are open spaces (that is, `M`[`X`][`Y`] = `M`[`U`][`V`] = .)

Can $(X, Y) = (U, V)$?

Your function has to return the least time needed to reach the cake from the starting position.

Example.

Let us consider the example where

```
H=3
W=4          . # . .
X=0    M =   . # . #
Y=3          . . . .
U=2          . . . .
V=0
```

Perhaps add a diagram showing the wall–boundary surrounding the labyrinth as well as positions of the cake and the player.

Here you have to reach the top–right corner from the bottom–left one. One quickest sequence of moves is as follows:

- move two squares right
- shoot one portal up, and one portal down
- move through the bottom portal — you will appear at the location $row = 0, column = 2$

- move one square right and reach the cake.

This takes time 4 and no quicker way is possible. Therefore your function should return `4`.

Subtasks.

Subtask 1 (25 points)

$0 \leq \texttt{H} \leq 2$

$0 \leq \texttt{W} \leq 2$

Subtask 2 (25 points)

$0 \leq \texttt{H} \leq 10$

$0 \leq \texttt{W} \leq 10$

Subtask 3 (25 points)

$0 \leq \texttt{H} \leq 100$

$0 \leq \texttt{W} \leq 100$

Subtask 4 (25 points)

$0 \leq \texttt{H} \leq 1000$

$0 \leq \texttt{W} \leq 1000$

Looks ugly but I'm confident this layout can be very nice with sufficient tweeking.

Details.

Is this a good title?

Time limit: $1s$.

Memory limit: $256MB$.

Also, boring stuff about graders that I don't have yet.