Prepare the Bunnies' Escape

==========

You're awfully close to destroying the LAMBCHOP doomsday device and freeing Commander Lambda's bunny prisoners, but once they're free of the prison blocks, the bunnies are going to need to escape Lambda's space station via the escape pods as quickly as possible. Unfortunately, the halls of the space station are a maze of corridors and dead ends that will be a deathtrap for the escaping bunnies. Fortunately, Commander Lambda has put you in charge of a remodeling project that will give you the opportunity to make things a little easier for the bunnies. Unfortunately (again), you can't just remove all obstacles between the bunnies and the escape pods - at most you can remove one wall per escape pod path, both to maintain structural integrity of the station and to avoid arousing Commander Lambda's suspicions.

You have maps of parts of the space station, each starting at a prison exit and ending at the door to an escape pod. The map is represented as a matrix of 0s and 1s, where 0s are passable space and 1s are impassable walls. The door out of the prison is at the top left (0,0) and the door into an escape pod is at the bottom right (w-1,h-1).

Write a function solution(map) that generates the length of the shortest path from the prison door to the escape pod, where you are allowed to remove one wall as part of your remodeling plans. The path length is the total number of nodes you pass through, counting both the entrance and exit nodes. The starting and ending positions are always passable (0). The map will always be solvable, though you may or may not need to remove a wall. The height and width of the map can be from 2 to 20. Moves can only be made in cardinal directions; no diagonal moves are allowed.

Languages

=========

To provide a Python solution, edit solution.py

To provide a Java solution, edit Solution.java

Test cases

==========

Your code should pass the following test cases.

Note that it may also be run against hidden test cases not shown here.

-- Python cases --

Input:

solution.solution([[0, 1, 1, 0], [0, 0, 0, 1], [1, 1, 0, 0], [1, 1, 1, 0]])

Output:

7

Input:

solution.solution([[0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 0], [0, 0, 0, 0, 0, 0], [0, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1], [0, 0, 0, 0, 0, 0]])

Output:

11

-- Java cases --

Input:

Solution.solution({{0, 1, 1, 0}, {0, 0, 0, 1}, {1, 1, 0, 0}, {1, 1, 1, 0}})

Output:

7

Input:

Solution.solution({{0, 0, 0, 0, 0, 0}, {1, 1, 1, 1, 1, 0}, {0, 0, 0, 0, 0, 0}, {0, 1, 1, 1, 1, 1}, {0, 1, 1, 1, 1, 1}, {0, 0, 0, 0, 0, 0}})

Output:

11

Use verify [file] to test your solution and see how it does. When you are finished editing your code, use submit [file] to submit your answer. If your solution passes the test cases, it will be removed from your home folder.