**Straightest path**

Attempted by: **1066**

/

Accuracy: **79%**

/

Maximum Score: **30**

/

18 Votes

Tag(s):

Algorithms, Dijkstra, Easy-Medium, Shortest-path

**PROBLEM**

**EDITORIAL**

**MY SUBMISSIONS**

**ANALYTICS**

You are playing a game on a grid of size N×MN×M. The game has the following rules:

* The grid contains cells that the player can move to. These are denoted by a period **(.)**
* The grid contains cells that the player cannot move to. These are denoted by an asterisk (\*)
* The player starts on the cell marked with a **V**.
* The player has to reach the cell marked with an **H**.

Write a program to find the path which has the minimum number of changes in direction. Print the number of times that the player needs to turn on the path

**Input format**

* First line: **N** and **M**
* Next **N** lines: **M** characters (denoting the cells of the grid)

**Output format**

Print the minimum number of times that the player needs to turn on the required path. If no path exists, print **-1**.

**Constraints**

1≤N,M≤1031≤N,M≤103

**SAMPLE INPUT**

5 4

V...

\*\*\*.

....

.\*\*\*

.H.\*

**SAMPLE OUTPUT**

4

**Explanation**

For the given sample case, Wizard will take first turn at cell (1,4)(1,4) , second turn at cell (3,4)(3,4) ,third turn at cell (3,1)(3,1), and fourth turn at cell (5,1)(5,1).

**Time Limit:**4.0 sec(s) for each input file.

**Memory Limit:**256 MB

**Source Limit:**1024 KB

**Marking Scheme:**Marks are awarded when all the testcases pass.

**Allowed Languages:**C, C++, C++14, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Julia, Kotlin, Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, R(RScript), Racket, Ruby, Rust, Scala, Swift, Visual Basic

<https://www.hackerearth.com/en-us/practice/algorithms/graphs/shortest-path-algorithms/practice-problems/algorithm/vizard-and-turns-a8c61c7e/?scroll-id=comments-183-238102&scroll-trigger=inview#c113183>

[Straightest path](https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/practice-problems/algorithm/vizard-and-turns-a8c61c7e/) / [Submission (10896022)](https://www.hackerearth.com/challenge/competitive/august-circuits-17/algorithm/vizard-and-turns-a8c61c7e/submission/10896022/) by [azukun (igor2)](https://www.hackerearth.com/@igor2)

using System;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

// (づ°ω°)づﾐe★゜・。。・゜゜・。。・゜☆゜・。。・゜゜・。。・゜

public class Solver

{

int[] dr = { 0, 1, 0, -1 };

int[] dc = { 1, 0, -1, 0 };

public void Solve()

{

int n = ReadInt();

int m = ReadInt();

var a = ReadLines(n);

int rs = 0, cs = 0, re = 0, ce = 0;

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

if (a[i][j] == 'V')

{

rs = i;

cs = j;

}

else if (a[i][j] == 'H')

{

re = i;

ce = j;

}

var q = new LinkedList<Tuple<int, int, int>>();

var d = new int[n, m, 4];

for (int i = 0; i < 4; i++)

{

d[rs, cs, i] = 1;

q.AddFirst(Tuple.Create(rs, cs, i));

}

while (q.Count > 0)

{

var t = q.First.Value;

q.RemoveFirst();

int nr = t.Item1 + dr[t.Item3];

int nc = t.Item2 + dc[t.Item3];

if (nr >= 0 && nr < n && nc >= 0 && nc < m && (d[nr, nc, t.Item3] == 0 || d[nr, nc, t.Item3] > d[t.Item1, t.Item2, t.Item3]) && a[nr][nc] != '\*')

{

d[nr, nc, t.Item3] = d[t.Item1, t.Item2, t.Item3];

q.AddFirst(Tuple.Create(nr, nc, t.Item3));

}

if (d[t.Item1, t.Item2, (t.Item3 + 1) % 4] == 0 || d[t.Item1, t.Item2, (t.Item3 + 1) % 4] > d[t.Item1, t.Item2, t.Item3] + 1)

{

d[t.Item1, t.Item2, (t.Item3 + 1) % 4] = d[t.Item1, t.Item2, t.Item3] + 1;

q.AddLast(Tuple.Create(t.Item1, t.Item2, (t.Item3 + 1) % 4));

}

if (d[t.Item1, t.Item2, (t.Item3 + 3) % 4] == 0 || d[t.Item1, t.Item2, (t.Item3 + 3) % 4] > d[t.Item1, t.Item2, t.Item3] + 1)

{

d[t.Item1, t.Item2, (t.Item3 + 3) % 4] = d[t.Item1, t.Item2, t.Item3] + 1;

q.AddLast(Tuple.Create(t.Item1, t.Item2, (t.Item3 + 3) % 4));

}

}

int ans = int.MaxValue;

for (int i = 0; i < 4; i++)

ans = Math.Min(ans, d[re, ce, i]);

Write(ans - 1);

}

#region Main

protected static TextReader reader;

protected static TextWriter writer;

static void Main()

{

#if DEBUG

reader = new StreamReader("..\\..\\input.txt");

//reader = new StreamReader(Console.OpenStandardInput());

writer = Console.Out;

//writer = new StreamWriter("..\\..\\output.txt");

#else

reader = new StreamReader(Console.OpenStandardInput());

writer = new StreamWriter(Console.OpenStandardOutput());

//reader = new StreamReader("input.txt");

//writer = new StreamWriter("output.txt");

#endif

try

{

new Solver().Solve();

//var thread = new Thread(new Solver().Solve, 1024 \* 1024 \* 128);

//thread.Start();

//thread.Join();

}

catch (Exception ex)

{

#if DEBUG

Console.WriteLine(ex);

#else

throw;

#endif

}

reader.Close();

writer.Close();

}

#endregion

#region Read / Write

private static Queue<string> currentLineTokens = new Queue<string>();

private static string[] ReadAndSplitLine() { return reader.ReadLine().Split(new[] { ' ', '\t', }, StringSplitOptions.RemoveEmptyEntries); }

public static string ReadToken() { while (currentLineTokens.Count == 0)currentLineTokens = new Queue<string>(ReadAndSplitLine()); return currentLineTokens.Dequeue(); }

public static int ReadInt() { return int.Parse(ReadToken()); }

public static long ReadLong() { return long.Parse(ReadToken()); }

public static double ReadDouble() { return double.Parse(ReadToken(), CultureInfo.InvariantCulture); }

public static int[] ReadIntArray() { return ReadAndSplitLine().Select(int.Parse).ToArray(); }

public static long[] ReadLongArray() { return ReadAndSplitLine().Select(long.Parse).ToArray(); }

public static double[] ReadDoubleArray() { return ReadAndSplitLine().Select(s => double.Parse(s, CultureInfo.InvariantCulture)).ToArray(); }

public static int[][] ReadIntMatrix(int numberOfRows) { int[][] matrix = new int[numberOfRows][]; for (int i = 0; i < numberOfRows; i++)matrix[i] = ReadIntArray(); return matrix; }

public static int[][] ReadAndTransposeIntMatrix(int numberOfRows)

{

int[][] matrix = ReadIntMatrix(numberOfRows); int[][] ret = new int[matrix[0].Length][];

for (int i = 0; i < ret.Length; i++) { ret[i] = new int[numberOfRows]; for (int j = 0; j < numberOfRows; j++)ret[i][j] = matrix[j][i]; } return ret;

}

public static string[] ReadLines(int quantity) { string[] lines = new string[quantity]; for (int i = 0; i < quantity; i++)lines[i] = reader.ReadLine().Trim(); return lines; }

public static void WriteArray<T>(IEnumerable<T> array) { writer.WriteLine(string.Join(" ", array)); }

public static void Write(params object[] array) { WriteArray(array); }

public static void WriteLines<T>(IEnumerable<T> array) { foreach (var a in array)writer.WriteLine(a); }

private class SDictionary<TKey, TValue> : Dictionary<TKey, TValue>

{

public new TValue this[TKey key]

{

get { return ContainsKey(key) ? base[key] : default(TValue); }

set { base[key] = value; }

}

}

private static T[] Init<T>(int size) where T : new() { var ret = new T[size]; for (int i = 0; i < size; i++)ret[i] = new T(); return ret; }

#endregion

}

<https://www.hackerearth.com/submission/10955340/>

using System;

using System.Collections.Generic;

using System.Linq;

class MyClass

{

static void Main(string[] args)

{

var inputLine1 = Console.ReadLine().Trim().Split(' ').Select(x => Int32.Parse(x)).ToList();

int N = inputLine1[0];

int M = inputLine1[1];

char[,] inputArray = new char[N, M];

for (int i = 0; i < N; i++)

{

var inputLine = Console.ReadLine().Trim();

for (int j = 0; j < M; j++)

{

inputArray[i, j] = inputLine[j];

}

}

Tuple<int, int> vPosition = FindVPosition(inputArray, M, N, 'V');

Console.WriteLine(FindH(inputArray, vPosition, M, N));

}

public static Tuple<int, int> FindVPosition(char[,] inputArray, int m, int n, char C)

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

if (inputArray[i, j] == C)

return new Tuple<int, int>(i, j);

}

}

return new Tuple<int, int>(-1, -1);

}

public static int FindH(char[,] inputArray, Tuple<int, int> vPosiiton, int m, int n)

{

var pointsDic = new List<int>();

AddToDic(pointsDic, vPosiiton.Item1, vPosiiton.Item2);

bool found = false;

int noOfCount = -1;

while (pointsDic.Any())

{

noOfCount++;

var newPoints = new List<int>();

for (int i = 0; i < pointsDic.Count - 1; i = i + 2)

{

if (inputArray[pointsDic[i], pointsDic[i + 1]] == 'H')

{

found = true;

newPoints.Clear();

break;

}

GetAllPointWithOneTurn(inputArray, pointsDic[i], pointsDic[i + 1], m, n, newPoints);

}

pointsDic = new List<int>(newPoints);

}

if (!found)

noOfCount = -1;

else

noOfCount--;

return noOfCount;

}

public static void AddToDic(List<int> stack, int x, int y)

{

stack.Add(x);

stack.Add(y);

}

public static void GetAllPointWithOneTurn(char[,] charinput, int x, int y, int m, int n, List<int> newPoints)

{

// get allPoints in rightside;

for (int i = y + 1; i < m; i++)

{

if (charinput[x, i] == 'Z')

continue;

if (charinput[x, i] == '.' || charinput[x, i] == 'H')

{

AddToDic(newPoints, x, i);

if (charinput[x, i] != 'H')

charinput[x, i] = 'Z';

}

else

{

break;

}

}

// left side

for (int i = y - 1; i >= 0; i--)

{

if (charinput[x, i] == 'Z')

continue;

if (charinput[x, i] == '.' || charinput[x, i] == 'H')

{

AddToDic(newPoints, x, i);

if (charinput[x, i] != 'H')

charinput[x, i] = 'Z';

}

else

{

break;

}

}

// bottm

for (int i = x + 1; i < n; i++)

{

if (charinput[i, y] == 'Z')

continue;

if (charinput[i, y] == '.' || charinput[i, y] == 'H')

{

AddToDic(newPoints, i, y);

if (charinput[i, y] != 'H')

charinput[i, y] = 'Z';

}

else

{

break;

}

}

//top

for (int i = x - 1; i >= 0; i--)

{

if (charinput[i, y] == 'Z')

continue;

if (charinput[i, y] == '.' || charinput[i, y] == 'H')

{

AddToDic(newPoints, i, y);

if (charinput[i, y] != 'H')

charinput[i, y] = 'Z';

}

else

{

break;

}

}

}

}

--------java----------

1. */\**
2. *\* Copyright (c) 2015, 2099, Ashok and/or its affiliates. All rights reserved.*
3. *\* ASHOK PROPRIETARY/CONFIDENTIAL. Use is subject to license terms, But you are free to use it :).*
4. *\**
5. *\*/*
6. import java.io.IOException;
7. import java.io.InputStream;
8. import java.io.PrintWriter;
9. import java.util.LinkedList;
11. */\*\**
12. *\* Problem Name: Straightest path*
13. *\* Link: https://www.hackerearth.com/challenge/competitive/august-circuits-17/algorithm/vizard-and-turns-a8c61c7e/*
14. *\* <p>*
15. *\* For complete implementation please see*
16. *\* {@link https://github.com/AshokRajpurohit/karani/tree/master/src/com/ashok/}*
17. *\**
18. *\* @author Ashok Rajpurohit (ashok1113@gmail.com)*
19. *\*/*
20. public class StraightestPath {
21. private static PrintWriter out = new PrintWriter(System.out);
22. private static InputReader in = new InputReader();
24. public static void main(String[] args) throws IOException {
25. solve();
26. in.close();
27. out.close();
28. }
30. private static void solve() throws IOException {
31. int n = in.readInt(), m = in.readInt();
32. Grid grid = new Grid(n, m);
33. grid.populate(in.readStringArray(n));
34. grid.process();
36. out.println(grid.end.state == State.PROCESSED ? grid.end.turns : -1);
37. }
39. final static class Grid {
40. final int rows, cols;
41. Cell start, end;
42. Cell[][] cells;
44. Grid(int r, int c) {
45. rows = r;
46. cols = c;
47. cells = new Cell[rows][cols];
48. }
50. void populate(String[] rows) {
51. for (int i = 0; i < this.rows; i++)
52. for (int j = 0; j < this.cols; j++)
53. cells[i][j] = new Cell(i, j, rows[i].charAt(j));
54. }
56. void process() {
57. start = getCell(CellProperty.ORIGIN);
58. end = getCell(CellProperty.DESTINATION);
60. int turns = -1;
61. LinkedList<Cell> queue = new LinkedList<>();
62. queue.addLast(start);
64. while (queue.size() > 0) {
65. int count = queue.size();
66. while (count > 0) {
67. count--;
68. Cell cell = queue.removeFirst();
69. queue.addAll(getCells(cell));
70. cell.state = State.PROCESSED;
71. cell.turns = turns;
73. if (cell == end)
74. return;
75. }
77. turns++;
78. }
79. }
81. LinkedList<Cell> getCells(Cell cell) {
82. LinkedList<Cell> nextCells = new LinkedList<>();
83. getCells(cell, Move.LEFT, nextCells);
84. getCells(cell, Move.RIGHT, nextCells);
85. getCells(cell, Move.UP, nextCells);
86. getCells(cell, Move.DOWN, nextCells);
88. for (Cell c : nextCells)
89. c.state = State.PROCESSING;
91. return nextCells;
92. }
94. void getCells(Cell cell, Move direction, LinkedList<Cell> list) {
95. if (cell.checkIfMoveRestricted(direction))
96. return;
98. int row = cell.row, col = cell.col;
99. row += direction.dr;
100. col += direction.dc;
102. while (validIndex(row, col) && cells[row][col].property != CellProperty.IMMOVABLE) {
103. Cell next = cells[row][col];
104. if (next.state == State.UNPROCESSED) {
105. list.addLast(next);
106. next.restrictMoves(direction);
107. }
109. row += direction.dr;
110. col += direction.dc;
111. }
112. }
114. private boolean validIndex(int r, int c) {
115. return r >= 0 && r < rows && c >= 0 && c < cols;
116. }
118. private Cell getCell(CellProperty property) {
119. for (Cell[] row : cells)
120. for (Cell cell : row)
121. if (cell.property == property)
122. return cell;
124. return new Cell(-1, -1, 'i');
125. }
126. }
128. final static class Cell {
129. final int row, col;
130. State state = State.UNPROCESSED; *// initial cell state.*
131. int turns = 0;
132. final CellProperty property;
133. boolean allowedHorizontal = true, allowedVertical = true;
135. Cell(int i, int j, char ch) {
136. row = i;
137. col = j;
138. property = CellProperty.getCellProperty(ch);
139. }
141. void restrictMoves(Move move) {
142. if (move.isHorizontal())
143. allowedHorizontal = false;
144. else
145. allowedVertical = false;
146. }
148. boolean checkIfMoveRestricted(Move move) {
149. return move.isHorizontal() ? !allowedHorizontal : !allowedVertical;
150. }
151. }
153. enum State {
154. UNPROCESSED, PROCESSING, PROCESSED;
155. }
157. enum CellProperty {
158. MOVABLE('.'), IMMOVABLE('\*'), ORIGIN('V'), DESTINATION('H');
159. char property;
161. CellProperty(char ch) {
162. property = ch;
163. }
165. static CellProperty getCellProperty(char ch) {
166. switch (ch) {
167. case '.':
168. return MOVABLE;
169. case '\*':
170. return IMMOVABLE;
171. case 'V':
172. return ORIGIN;
173. default:
174. return DESTINATION;
175. }
176. }
177. }
179. enum Move {
180. LEFT(0, -1), RIGHT(0, 1), UP(-1, 0), DOWN(1, 0), LU(LEFT, UP), LD(LEFT, DOWN), RU(RIGHT, UP), RD(RIGHT, DOWN),
181. INVALID(0, 0);
182. int dr, dc;
183. Move reverse;
185. static {
186. LEFT.reverse = RIGHT;
187. RIGHT.reverse = LEFT;
188. UP.reverse = DOWN;
189. DOWN.reverse = UP;
190. }
192. Move(int r, int c) {
193. dr = r;
194. dc = c;
195. }
197. Move(Move a, Move b) {
198. dr = a.dr + b.dr;
199. dc = a.dc + b.dc;
200. }
202. boolean isHorizontal() {
203. return this == LEFT || this == RIGHT;
204. }
205. }
207. final static class InputReader {
208. InputStream in;
209. protected byte[] buffer = new byte[8192];
210. protected int offset = 0;
211. protected int bufferSize = 0;
213. public InputReader() {
214. in = System.in;
215. }
217. public void close() throws IOException {
218. in.close();
219. }
221. public int readInt() throws IOException {
222. int number = 0;
223. int s = 1;
224. if (offset == bufferSize) {
225. offset = 0;
226. bufferSize = in.read(buffer);
227. }
228. if (bufferSize == -1)
229. throw new IOException("No new bytes");
230. for (; buffer[offset] < 0x30 || buffer[offset] == '-'; ++offset) {
231. if (buffer[offset] == '-')
232. s = -1;
233. if (offset == bufferSize - 1) {
234. offset = -1;
235. bufferSize = in.read(buffer);
236. }
237. }
238. for (; offset < bufferSize && buffer[offset] > 0x2f; ++offset) {
239. number = (number << 3) + (number << 1) + buffer[offset] - 0x30;
240. if (offset == bufferSize - 1) {
241. offset = -1;
242. bufferSize = in.read(buffer);
243. }
244. }
245. ++offset;
246. return number \* s;
247. }
249. public String read() throws IOException {
250. StringBuilder sb = new StringBuilder();
251. if (offset == bufferSize) {
252. offset = 0;
253. bufferSize = in.read(buffer);
254. }
256. if (bufferSize == -1 || bufferSize == 0)
257. throw new IOException("No new bytes");
259. for (;
260. buffer[offset] == ' ' || buffer[offset] == '\t' || buffer[offset] ==
261. '\n' || buffer[offset] == '\r'; ++offset) {
262. if (offset == bufferSize - 1) {
263. offset = -1;
264. bufferSize = in.read(buffer);
265. }
266. }
267. for (; offset < bufferSize; ++offset) {
268. if (buffer[offset] == ' ' || buffer[offset] == '\t' ||
269. buffer[offset] == '\n' || buffer[offset] == '\r')
270. break;
271. if (Character.isValidCodePoint(buffer[offset])) {
272. sb.appendCodePoint(buffer[offset]);
273. }
274. if (offset == bufferSize - 1) {
275. offset = -1;
276. bufferSize = in.read(buffer);
277. }
278. }
279. return sb.toString();
280. }
282. public String[] readStringArray(int size) throws IOException {
283. String[] res = new String[size];
284. for (int i = 0; i < size; i++)
285. res[i] = read();
287. return res;
288. }
289. }
290. }