Intern Mobile Meetup 2024

5 мар 2024, 23:04:22
старт: 5 мар 2024, 21:04:22
финиш: 5 мар 2024, 23:04:22
длительность: 02:00:00
начало: 20 фев 2024, 04:01:00
конец: 7 мар 2024, 03:59:00

3. Бомбермэн

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Черный космонавт похитил возлюбленную Бомбермэна, упрятав ее далеко-далеко за реками, горами и лесами. Задача Бомбермэна пройти весь путь и спасти любимую, сражаясь при этом с кучей врагов, используя для борьбы с ними только бомбы.
Каждый уровень в игре представляет собой прямоугольное поле из n строк и m стобцов. Каждая клетка поля может быть пустой (обозначается как "."), стеной (обозначается как "W") или неподвижным противником (обозначается как "B").

Бомбермен можен разместить бомбу на пустой клетке. Бомба уничтожает противников, которые находятся в той же строке или столбце до ближайшей от бомбы стены или края поля. Т.е. противник будет уничтожен, если от клетки с бомбой можно дойти до него, двигаясь влево, вправо, вверх или вниз и на этом пути не будет клетки со стеной.

Нумерация игрового поля начинается с левого верхнего угла, клетки нумеруются с единицы.

Помогите Бомбермену определить, в какой клетке нужно разместить бомбу, чтобы уничтожить как можно больше противников.

Формат ввода

В первой строке задаются числа n и m $(1 \leq n, m \leq 1000)$ — размеры поля.
В следующих n строках описывается уровень игры. Каждая строка состоит из m символов ".", "W" и "B".

Формат вывода

Выведите номер строки и столбца пустой клетки, в которой нужно разместить бомбу. В случае, если ответов несколько — выведите любой из них.

Пример 1

Ввод	Вывод
3 3 WBW B.B WBW	2 2

Пример 2

Ввод	Вывод
3 4	1 1

Пример 3

Ввод	Вывод
1 10 .BBWBB.BBW	1 7

Язык Kotlin 1.9.21 (JRE 21)

Набрать здесь | Отправить файл

```kotlin
fun main() {
    val (ySize, xSize) = readln().split(" ").map(String::toInt)
    // val cells = (1..ySize).flatMap { readln().asSequence() }
    val input = generateSequence(::readLine)
    val cells = input.toList().flatMap { it.asSequence() }

    var bestX = 0
    var bestY = 0
    var maxKills = -1
    cells.forEachIndexed { index, c ->
        if (c == '.') {
            val x = index % xSize
            val y = index / xSize
            var kills = 0
            for (i in x + 1..<xSize) when (cells[i + xSize * y]) {
                'B' -> kills++
                'W' -> break
            }
            for (i in x - 1 downTo 0) when (cells[i + xSize * y]) {
                'B' -> kills++
                'W' -> break
            }
            for (i in y + 1..<ySize) when (cells[x + xSize * i]) {
                'B' -> kills++
                'W' -> break
            }
            for (i in y - 1 downTo 0) when (cells[x + xSize * i]) {
                'B' -> kills++
                'W' -> break
            }
            if (kills > maxKills) {
                maxKills = kills
                bestX = x
                bestY = y
            }
        }
    }
    println("${bestY + 1} ${bestX + 1}")
```

Отправить

Предыдущая