Дмитрий Хамутский

Школа бэкенд-разработки 2022 (лето)

6 июн 2022, 11:15:19
старт: 5 июн 2022, 17:36:08
финиш: 5 июн 2022, 22:36:08
длительность: 05:00:00
начало: 21 фев 2022, 22:47:45

D. Ориентация в лабиринте

Ограничение времени	2 секунды
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Януш Воронов решил провести очередную выставку своих работ. В качестве места проведения он выбрал один из этажей заброшенного здания прямоугольной формы $N \times M$ метров.
У Януша есть схема этажа в виде $N \times M$ клеток (каждая клетка задаёт пространство площадью 1×1 метров), где «#» обозначает кусок стены, а «`.`» — пространство, доступное для перемещения посетителей.

Также на карте ровно одна клетка обозначена как «`S`» — участок, из которого посетители начнут осмотр выставки.

Гарантируется, что планировка этажа удовлетворяет следующим условиям:

- Все клетки в первых и последних строках / столбцах схемы являются стенами.
- От стартовой клетки можно добраться до любой пустой клетки, перемещаясь только вверх / вниз / влево / вправо.
- Между любой парой пустых клеток на схеме существует ровно один путь, возможно проходящий через стартовую клетку.

Януш хочет добиться идеального впечатления от осмотра выставки, поэтому хочет учесть направление, в котором посетитель впервые зайдет в каждую свободную клетку. Гарантируется, что такое направление определяется однозначно.

Помогите Янушу и выведите для каждой клетки направление, в котором посетитель впервые зайдет в эту клетку при осмотре выставки.

Формат ввода

В первой строке даны два целых числа N и M ($3 \le N, M \le 500$) — количество строк и столбцов на схеме этажа.
В следующих N строках расположено по M символов из множества ($\texttt{\#}$, $\texttt{.}$, $\texttt{S}$).

Гарантируется, что

- Все клетки в первых и последних строках / столбцах схемы равны `#`.
- На схеме расположена ровно одна стартовая клетка `S`.
- От стартовой клетки можно добраться до любой пустой клетки, перемещаясь только вверх / вниз / влево / вправо.
- Между любой парой пустых клеток на схеме существует ровно один путь, возможно проходящий через стартовую клетку.

Формат вывода

Выведите N строк по M символов в каждой — схему этажа, где каждая пустая клетка `.` заменена на направление первого захода в эту клетку.
Занумеруем все строки от 1 до N сверху вниз, все столбцы — от 1 до M слева направо. В таком случае пустая клетка (r, c) должна содержать:

- `L` — если в клетку (r, c) зашли из клетки $(r, c + 1)$;
- `R` — если в клетку (r, c) зашли из клетки $(r, c - 1)$;
- `U` — если в клетку (r, c) зашли из клетки $(r + 1, c)$;
- `D` — если в клетку (r, c) зашли из клетки $(r - 1, c)$.

Пример 1

Ввод **Вывод**

Ввод	Вывод
5 8 ######## #......# #.#S#.## ##...### ########	######## #LLURRR# #D#S#D## ##LDR### ########

Пример 2

Ввод	Вывод
3 3 ### #S# ###	### #S# ###

Примечания

Рассмотрим первый тестовый пример.
Стартовой является клетка $(3, 4)$.

Из стартовой клетки посетитель может попасть в клетку $(2, 4)$, сделав шаг вверх, или в клетку $(4, 4)$, пройдя вниз.

Из клетки $(2, 4)$ можно пойти налево в клетки $(2, 3)$ и $(2, 2)$; из клетки $(2, 2)$ можно дойти до клетки $(3, 2)$, пройдя вниз.

Также из клетки $(2, 4)$ можно пройти направо в клетки $(2, 5)$, $(2, 6)$ и $(2, 7)$; из клетки $(2, 6)$ можно попасть в клетку $(3, 6)$, пройдя вниз.

Из клетки $(4, 4)$ можно пройти всего в две клетки — налево в $(4, 3)$ и направо в $(4, 5)$.

Во втором тестовом примере посетитель никуда не может пройти из стартовой клетки.

Язык Python 3.7 (PyPy 7.3.3)

Набрать здесь | Отправить файл

```
from collections import deque


def find_start(array, n, m):
    position = None
    for row in range(n):
        for column in range(m):
            if array[row][column] == 'S':
                position = (row, column)
                break
        if position != None:
            break
    return position

def way_back(x, y):
    if x == 1 and y == 0:
        return 'R'
    elif x == -1 and y == 0:
        return 'L'
    elif x == 0 and y == -1:
        return 'U'
    else:
        return 'D'

def paint_map(array, n, m):
    start_position = find_start(maze, n, m)
    my_deque = deque()
    my_deque.append(start_position)
    while my_deque:
        current_position = my_deque.popleft()
        for delta_x, delta_y in (
                (1, 0),
                (-1, 0),
                (0, -1),
                (0, 1),
        ):
            new_x = current_position[0] + delta_x
```

Отправить

Предыдущая | Следующая