Отчёт

Использовать предоставленный шаблон лабиринта, где 0 представляет

свободное пространство, а 1 - стены лабиринта. "Начальная точка лабиринта задается

координатами (1, 3), а конечная точка - (8, 1)."

2. Разработать алгоритм для нахождения кратчайшего пути от начальной точки

до конечной. Можно использовать алгоритм поиска в ширину (BFS) или поиска в глубину

(DFS) для решения этой задачи.

3. На основе алгоритма изобразить блок-схему. Блок-схема должна включать

основные шаги алгоритма, такие как инициализация, условия проверки, циклы

исследования и возврат результата. Каждый блок должен быть четко обозначен и

понятен."

4. Составить отчёт о выполненной работе:

* Код программы с комментариями.
* Скриншот результата выполнения программы.
* Описание используемого алгоритма и его выбор.
* Блок-схема

lab = [  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 0, 0, 0, 1, 0, 0, 0, 0, 1],  
 [1, 0, 1, 0, 1, 0, 1, 1, 0, 1],  
 [1, 0, 1, 0, 0, 0, 1, 0, 0, 1],  
 [1, 0, 1, 1, 1, 0, 1, 0, 1, 1],  
 [1, 0, 0, 0, 0, 0, 1, 0, 0, 1],  
 [1, 1, 1, 1, 1, 0, 1, 0, 1, 1],  
 [1, 0, 0, 0, 1, 0, 0, 0, 1, 1],  
 [1, 0, 1, 0, 0, 0, 1, 0, 0, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]  
]  
  
  
def BFS(start, end, lab):  
 current = start  
 visited = [[1, 3]]  
 stack =[]  
  
 while end not in visited:  
  
 for i in range(current[0] - 1, current[0] + 2):  
 for j in range(current[1] - 1, current[1] + 2):  
 if lab[i][j] == 0:  
 if ([i,j] not in stack) and ([i,j] not in visited):  
 stack.append([i,j])  
  
  
  
 current = stack[0]  
 visited.append(stack[0])  
 stack.remove(stack[0])  
 print(current)  
  
BFS([1,3],[8,1], lab)

Результат выполнения:

[1, 2]

[2, 3]

[1, 1]

[2, 1]

[3, 3]

[3, 4]

[3, 1]

[2, 5]

[3, 5]

[4, 5]

[4, 1]

[1, 5]

[1, 6]

[5, 4]

[5, 5]

[5, 1]

[5, 2]

[1, 7]

[5, 3]

[6, 5]

[1, 8]

[2, 8]

[7, 5]

[7, 6]

[3, 7]

[3, 8]

[8, 4]

[8, 5]

[6, 7]

[7, 7]

[8, 7]

[4, 7]

[7, 3]

[8, 3]

[5, 7]

[5, 8]

[8, 8]

[7, 2]

[7, 1]

[8, 1]

