УДК 004.421.2

Пусный Д.О., бакалавр направления подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем»

НИУ «БелГУ», Россия, г. Белгород

Pusyy D.O., bachelor of training direction 02.03.03 «Information Systems software and administration»

NRU "BelSU" Russia, Belgorod

РЕАЛИЗАЦИЯ АЛГОРИТМА ПОИСКА ВЫХОДА ИЗ ЛАБИРИНТА
IMPLEMENTATION OF EXIT SEARCH ALGORITHM FROM MAZE

Аннотация: *в данной статье рассмотрены основные алгоритмы, позволяющие находить кратчайший путь разными способами. Проведено сравнение алгоритмов, и выбран самый оптимальный по заданным параметрам. Алгоритм приведён в блок-схеме с результатами его работы.*

Abstract: *This article discusses the basic algorithms that allow you to find the shortest path in different ways. Comparison of algorithms is carried out, and the most optimal according to the given parameters is selected. The algorithm is shown in the flowchart with the results of it's work.*

Ключевые слова*: алгоритм, лабиринт, путь, граф, волна.*

Keywords: *algorithm, maze, way, graph, wave.*

Задача нахождения кратчайшего пути из одной точки в другую решается достаточно давно. Она используется в логистике для уменьшения затрат на транспортировку грузов, что способствует значительному увеличению прибыли организации. Но и для обычного человека является довольно полезной, так как позволяет сэкономить время на передвижение. Поэтому можно сказать, что нахождение оптимального пути до сих пор является актуальным.

На текущий момент известно несколько основных алгоритмов поиска пути.

1. **Правило "одной руки"** - суть которого состоит в движении по лабиринту, касаясь правой или левой рукой его стены. Недостатком является необходимость пройти долгий путь, заходя во все тупики, и невозможность пройти все маршруты или найти выход, если есть отдельно стоящие стены.

2. **Алгоритм Люка-Тремо** - выйдя из любой точки лабиринта, надо сделать отметку на его стене (крест) и двигаться в произвольном направлении до тупика или перекрестка; в первом случае вернуться назад, поставить второй крест, свидетельствующий, что путь пройден дважды - туда и назад, и идти в направлении, не пройденном ни разу, или пройденном один раз; во втором - идти по произвольному направлению, отмечая каждый перекресток на входе и на выходе одним крестом; если на перекресте один крест уже имеется, то следует идти новым путем, если нет - то пройденным путем, отметив его вторым крестом.

Но данные варианты не удовлетворяют одному важному условию. Они не позволяют найти оптимальный путь до выхода из лабиринта. Возможно, проблема состоит в том, что нужно преобразовать лабиринт во что-то другое, с чем можно работать. Когда хотят найти кратчайшее расстояние между двумя точками, используют графы. В роли вершин графа выступают контрольные точки, а его рёбра отображают расстояние между ними. С таким подходом к представлению лабиринта можно работать.

3. **Поиск в ширину** - выполняет исследование равномерно во всех направлениях. Ключевая идея алгоритма заключается в том, что мы отслеживаем состояние расширяющегося кольца, которое называется границей. Повторяем эти шаги, пока граница не окажется пустой: выбираем и удаляем точку из границы; помечаем точку как посещённую,

чтобы знать, что не нужно обрабатывать её повторно; расширяем границу, глядя на её соседей. Всех соседей, которых мы ещё не видели, добавляем к границе. В этом цикле заключается вся сущность алгоритмов поиска по графу.

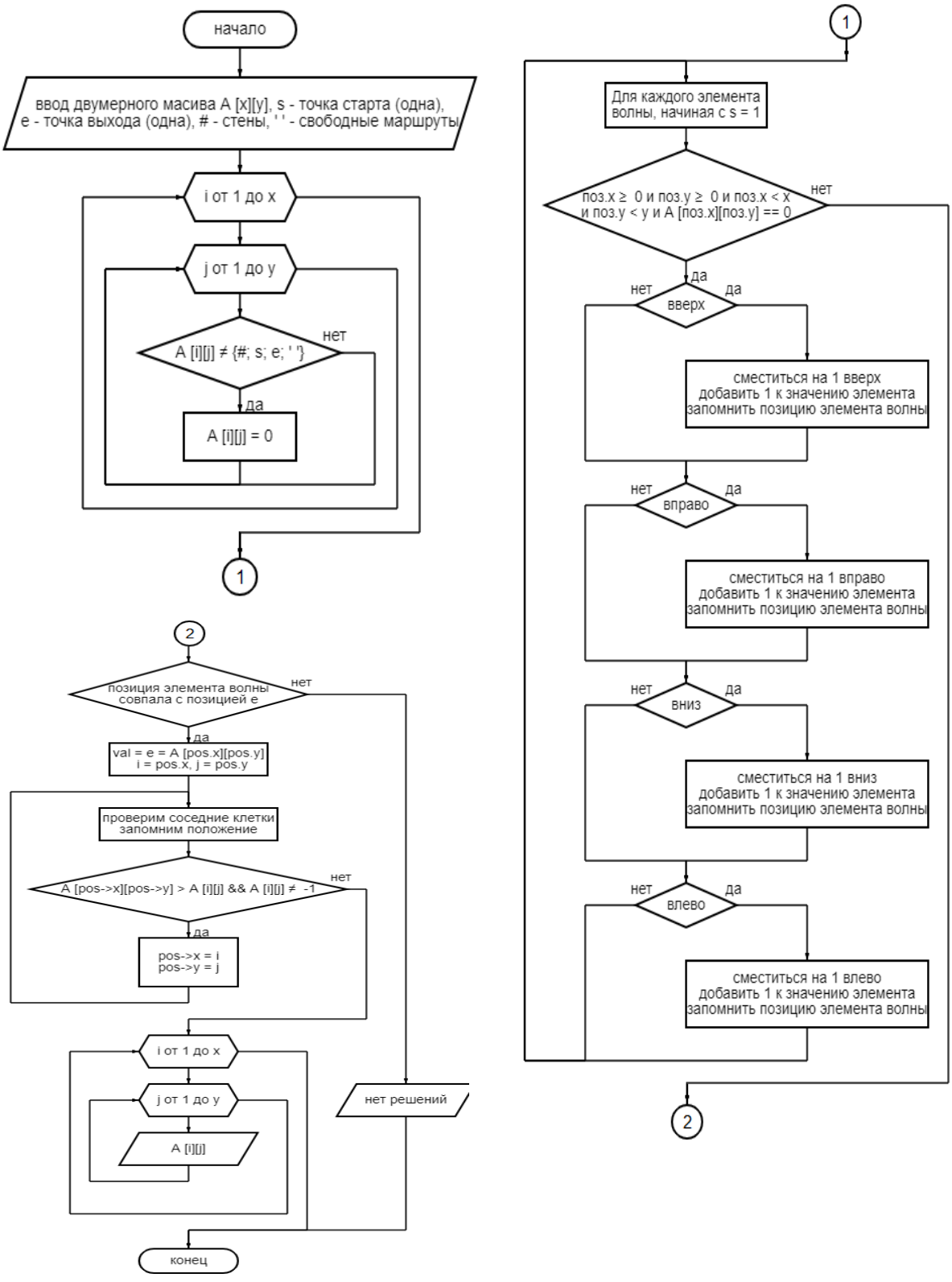
4. **Алгоритм Дейкстры** (также называемый поиском с равномерной стоимостью) - позволяет нам задавать приоритеты исследования путей. Вместо равномерного исследования всех возможных путей он отдаёт предпочтение путям с низкой стоимостью. Мы можем задать уменьшенные затраты, чтобы алгоритм двигался по дорогам, повышенную стоимость, чтобы избегать лесов и врагов, и многое другое. Когда стоимость движения может быть разной, мы используем его вместо поиска в ширину.

5. **A*** — это модификация алгоритма Дейкстры, оптимизированная для единственной конечной точки. Алгоритм Дейкстры может находить пути ко всем точкам, A* находит путь к одной точке. Он отдаёт приоритет путям, которые ведут ближе к цели. Алгоритм A* использует и подлинное расстояние от начала, и оцененное расстояние до цели.

Обобщив изложенный материал, можно сделать вывод, что поиск в ширину, алгоритм Дейкстры и A* могут найти кратчайший маршрут до выхода из лабиринта. А так как стоимость перемещения одинаковая и скорость нахождения пути нам не важна, то поиск в ширину будет самым простым в реализации.

Процесс решения задачи реализации алгоритма поиска выхода из лабиринта представлен в данной блок-схеме.

Алгоритм проходит волнами по всем свободным путям в лабиринте и нумерует все свободные элементы лабиринта, присваивая номер волны, в которой он находится.



Затем алгоритм проверяет наличие выхода и идёт дальше, если он есть.

Далее алгоритм двигается из точки выхода к соседним ячейкам с наименьшим значением и оставляет за собой след, тем самым строя кратчайший путь.

Данный алгоритм был реализован на языке программирования C++. Результаты его работы представлены на рисунке 1. Лабиринт задаётся «#», где выступает в роли стены. В левом верхнем углу находится точка старта. Буква «e» обозначает выход. А кратчайший маршрут показывается точками.

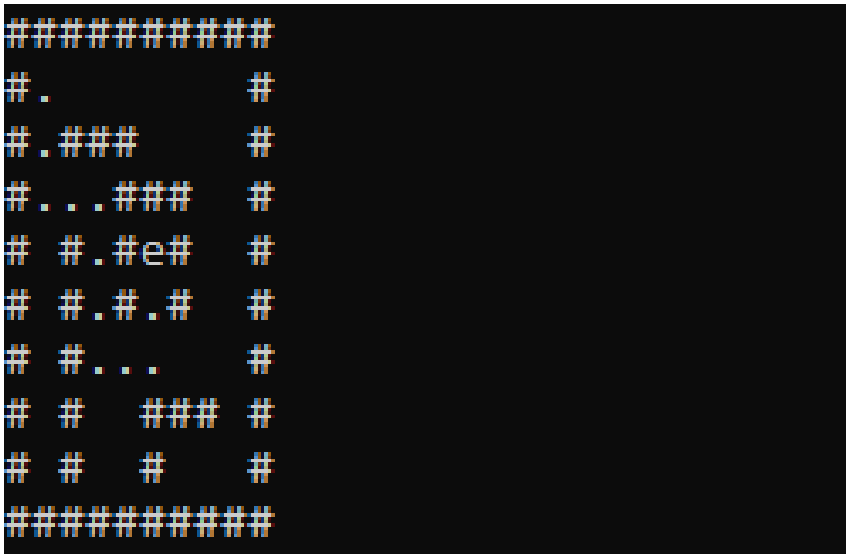


Рисунок 1 Консоль с выведенным лабиринтом и кратчайшим путём

Список литературы

1. Бережной, В.В. Дискретная математика / В.В. Бережной, А.В. Шапошников; Министерство образования и науки РФ, Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – Ставрополь: СКФУ, 2016. – 199 с. : ил.

2. Окулов, С. М. Основы программирования /Окулов С. М. М.: БИНОМ. Лаборатория знаний, 2012. 340С.