Санкт-Петербургский Национальный Исследовательский Университет Информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

**Лабораторная работа №5**

**Вариант №1**

Выполнили:

Кремпольская Е.А.

Петрова Н.Г.

Проверил:

Мусаев А.А.

Санкт-Петербург

2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc122033640)

[Задание 1 4](#_Toc122033641)

[1.1 Поиск в глубину (DFS) 4](#_Toc122033642)

[1.2 Поиск в ширину (BFS) 5](#_Toc122033643)

[Задание 2 7](#_Toc122033644)

[2.1 Лабиринт 7](#_Toc122033645)

[Заключение 8](#_Toc122033646)

[Приложение 9](#_Toc122033647)

Введение

Для становления хорошим специалистом в области программирования на языке Python необходимо знать основные алгоритмы и функционал языка.

Цель данной работы – ознакомление с алгоритмами поиска в глубину и ширину на языке программирования Python.

Данная работа представляет собой отчет о выполненных заданиях:

1. Задание 1: придумать и решить задачу для алгоритма поиска в глубину и ширину. Объяснить, почему для решения поставленных задач были выбраны именно эти алгоритмы поиска.
2. Задание 2: написать алгоритм, который найдет выход из лабиринта, если дана случайная квадратная матрица, заполненная нулями и единицами, где предполагается, что 0 – это проход, а 1 – это стена.

Задание 1

# Поиск в глубину (DFS)

Идея поиска в глубину (DFS) заключается в том, что мы двигаемся от начальной вершины по определенному пути до тех пор, пока не достигнем конца пути или искомой вершины. Если мы достигли конца пути, но он не является пунктом назначения, то мы возвращаемся назад (к точке разветвления или расхождения путей) и идем по другому маршруту.

Рассмотрим пример. Предположим, что у нас есть ориентированный граф (рисунок 1).

Изображение выглядит как часы, коллекция картинок

Автоматически созданное описание

Рисунок 1 - Ориентированный граф

Для поиска в глубину была придумана задача, где необходимо найти кратчайщее расстояние от 1 до 12 в заданном графе (рисунок 1). Алгоритм поиска в глубину подходит здесь больше, чем остальные, так как он позволяет найти все пути до конечных точек в быстрые сроки (рисунок 2). Благодаря timeit, мы сверили скорость работы поиска в глубину при заданной задаче с поиском в ширину (рисунок 3).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 - Поиск в глубину с 1 по 12

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 - Поиск в ширину с 1 по 12

# Поиск в ширину (BFS)

Вместо того, чтобы двигаться по определенному пути до конца, BFS предполагает движение вперед по одному соседу за раз. Это означает следующее: Вместо следования по пути, BFS подразумевает посещение ближайших к начальной вершине соседей за один шаг, затем посещение соседей соседей и так до тех пор, пока не будет обнаружена конечная вершина.

Рассмотрим пример. Предположим, что у нас есть ориентированный граф (рисунок 4).

Изображение выглядит как часы, коллекция картинок

Автоматически созданное описание

Рисунок 4 - Ориентированный граф

Для поиска в ширину была придумана задача, где необходимо найти кратчайщее расстояние от 1 до 3 в заданном графе (рисунок 5). Алгоритм поиска в ширину подходит здесь больше, чем остальные, так как он позволяет найти все пути до конечных точек в быстрые сроки (рисунок 6). Благодаря timeit, мы сверили скорость работы поиска в ширину при заданной задаче с поиском в глубину (рисунок 7).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 - Поиск в ширину с 1 по 3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 - Поиск в глубину с 1 по 3

**Вывод:** Может показаться, что BFS работает медленнее. Однако если внимательно присмотреться к визуализациям, можно увидеть, что они имеют одинаковое время выполнения. Очередь предполагает обработку каждой вершины перед достижением пункта назначения. Это означает, что, в худшем случае, BFS исследует все вершины и грани. Несмотря на то, что BFS может казаться медленнее, на самом деле он быстрее, поскольку при работе с большими графами обнаруживается, что DFS тратит много времени на следование по путям, которые в конечном счете оказываются ложными. BFS часто используется для нахождения кратчайшего пути между двумя вершинами.

Задание 2

# Лабиринт

Для решения задачи с поиском выхода из лабиринта, в котором 0 – это проход, а 1 – это стена, был выбран алгоритм поиска в ширину, потому что его не рекурсивную реализацию в данной программе будет проще модифицировать.

Дана случайная квадратная матрица, заполненная нулями и единицами (рисунок 7).

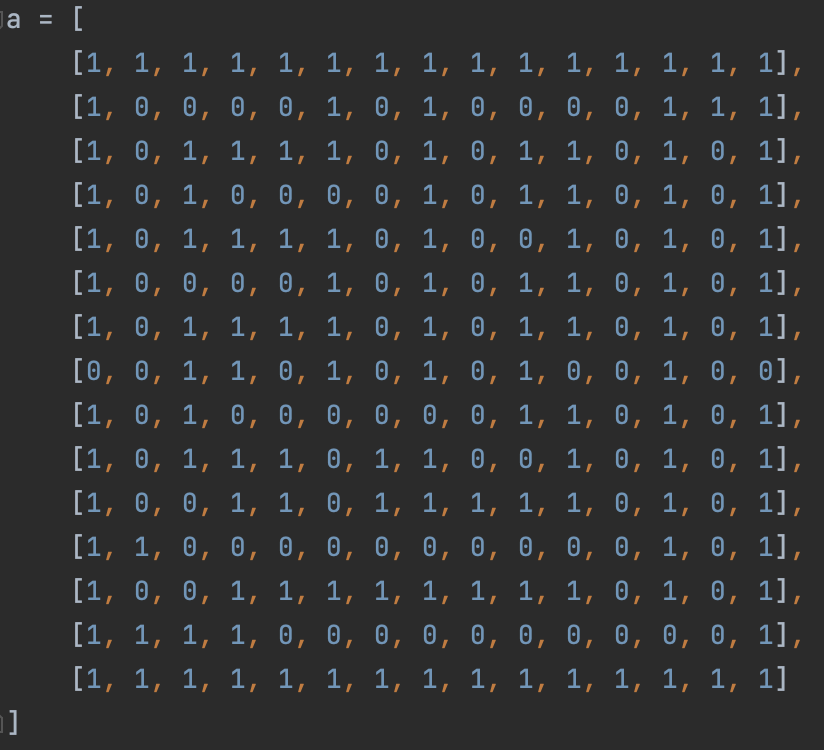


Рисунок 7 - Задача

Создается нулевая матрица, в которую будет записываться сумма всех пройденных клеток, по которой в конце будет выбран кратчайший маршрут (рисунок 8).

Изображение выглядит как текст, белый, клавиатура

Автоматически созданное описание

Рисунок 8 - Пример работы программы для заданной в условии задачи

**Вывод:** Был реализован алгоритм, который находит выход из лабиринта, с помощью поиска в ширину.

Заключение

В ходе лабораторной работы были изучены алгоритмы DFS и BFS, которые важны в теории графов. Для каждого алгоритма приведена программная реализация.

Приложение

Ссылка на полный код данной лабораторной работы:

<https://github.com/NatalyaPetrova/Algoritms>