Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчет по курсовому проекту**

**По курсу:** «Алгоритмы и структуры данных»

**На тему:** «Поиск кратчайшего пути в лабиринте»

Выполнил студент гр. 3530901/00003 Соколов Д.С.

(подпись)

Преподаватель Ахин М.Х.

(подпись)

“ ” 2022 г.

Санкт-Петербург

2022

**Содержание**

[1. Задание 3](#_Toc103040477)

[2. Метод решения 7](#_Toc103040478)

[3. Листинг программы 8](#_Toc103040479)

[4. Работа программы 8](#_Toc103040480)

[Вывод 17](#_Toc103040481)

# Задание

**Вариант 27.**

Лабиринт задан матрицей NxM со значениями 0 или 1 (0 – проход, 1 – стенка). Найти самый короткий путь между заданными "точками" Start (S) и Finish (F).

Изображение выглядит как стол

Автоматически созданное описание

Рис. 1 Пример визуализации лабиринта и его прохождения

Язык программирования: Kotlin.

Визуализация: GUI, Java FX.

Окна приложения:

1. Главное меню

Действия:

* Открытие окна с выбранным лабиринтом
* Открытие окна с настройками лабиринта
* Выход из приложения нажатием на крестик



Рис. 2 Главное меню

1. Конфигурация лабиринта

Действия:

* Загрузка конфигурации лабиринта из файла
* Указание размеров лабиринта и конфигурации клеток: изменение числовых значений в поле «размер лабиринта n x m». В отображаемой таблице заполнение клеток происходит путем нажатия на тип клетки (S, F, 0, 1) и указанием соответствующей ячейки таблицы кликом мышкой.
* Возврат к главному меню



Рис. 3 Конфигурация лабиринта

1. Решатель лабиринта

Действия:

* Представление лабиринта в форме таблицы
* Задание скорости прохождения лабиринта
* Запуск прохождения лабиринта
* Выделение клеток, по которым проходит решатель
* Запуск окна с результатом прохождения

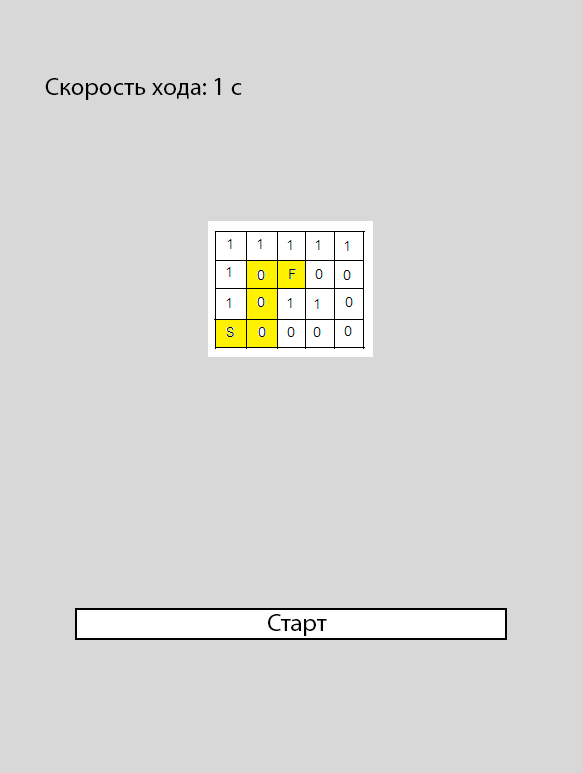


Рис. 4 Решатель лабиринта

1. Результаты прохождения

* Отображение об успешном или неуспешном прохождении лабиринта
* Предложение вернуться в главное меню

Загрузка исходных данных (конфигурации лабиринта) может осуществляться программно: изменение числовых значений в поле «размер лабиринта m x n». В отображаемой таблице заполнение клеток происходит путем нажатия на тип клетки (S, F, 0, 1) и указанием соответствующей ячейки таблицы кликом мышкой. Также загрузка конфигурации лабиринта может загружаться через конфигурационный файл. Формат файла: количество строк – высота лабиринта, количество символов в каждой строке – ширина лабиринта. Символ в строке описывает конкретное поле лабиринта и может принимать значения: S(начало лабиринта), F(конец лабиринта), 0(клетка с возможностью хода), 1(стенка лабиринта. Отсутствует возможность прохождения через данную клетку).

Изображение выглядит как текст

Автоматически созданное описание

Рис. 5 Пример конфигурационного файла

Результат прохождения лабиринта отображается в диалоговом окне: успешность прохождения, количество шагов.

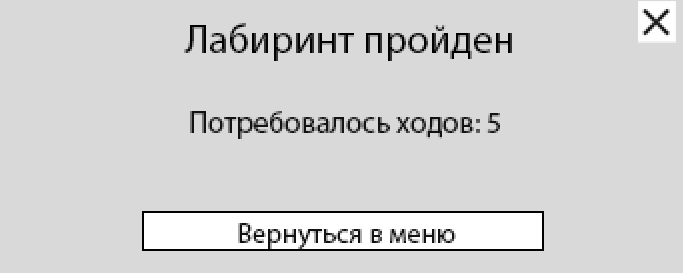


Рис. 6 Пример диалогового окна с результатом прохождения лабиринта

Обрабатываемые ошибки:

1. Неверное задание конфигурации лабиринта. При обнаружении исчезает возможность открытия окна с решателем и происходит уведомление в диалоговом окне о возникновении ошибки.
2. Неверный формат файла конфигурации лабиринта. Уведомление в диалоговом окне.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 7 Пример диалогового окна с ошибкой

Пример неверной конфигурации лабиринта: отсутствие вовсе или содержание более одного символа S и F. Содержание в файле посторонних символов (не S,F,0,1).

Исходный код программы будем хранить в репозитории по адресу: [github.com/dee-tree/maze-solver](https://github.com/dee-tree/maze-solver).

# Метод решения

Для поиска кратчайшего пути в лабиринте воспользуемся двумя различными алгоритмами:

1. Поиск в ширину (BFS). Идея алгоритма состоит в том, чтобы за каждый шаг расширяться в каждую допустимую сторону на 1 клетку. Таким образом, кратчайший путь будет достигнут ранее других допустимых решений.
2. A\*. Идея алгоритма примерно та же, что и в BFS, только теперь добавляются метрики оценки каждого хода. Каждый допустимый ход имеет «стоимость», которая высчитывается как «манхэттенское расстояние» - сумма расстояний по каждой оси от текущей клетки до финиша. Таким образом, сперва будут искаться самые «дешевые» (короткие) пути. При задании одинаковой стоимости для каждой клетки алгоритм повторяет BFS.

# Листинг программы

Исходный код программы, тесты, инструкция по сборке располагается в репозитории по адресу: [github.com/dee-tree/maze-solver](https://github.com/dee-tree/maze-solver).

# Работа программы

Проиллюстрируем работу программы и обработку ошибок.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 8 Главное меню приложения

1. Попробуем без загрузки конфигурации нажать «Solve the maze»: получаем ошибку об отсутствии инициализации конфигурации лабиринта.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 9 Ошибка об отсутствии инициализации конфигурации лабиринта

1. Попробуем загрузить конфигурацию лабиринта: в данном окне так же мы можем указать способ решения лабиринта из описанных в п.2 «метод решения».

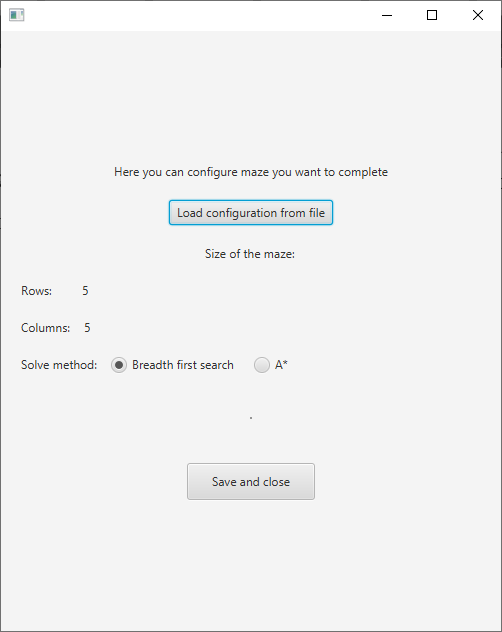


Рис. 10 Окно задания конфигурации лабиринта

1. Попробуем загрузить лабиринт некорректного размера:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 11 Файл конфигурации лабиринта неверного размера

И получим ошибку о неверно заданной конфигурации.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 12 Ошибка неверно заданной конфигурации лабиринта

1. Теперь попробуем загрузить лабиринт, содержащий некорректные символы в конфигурации и получим соответствующую ошибку.

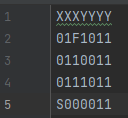


Рис. 13 Файл неверной конфигурации лабиринта

Изображение выглядит как текст

Автоматически созданное описание

Рис. 14 Ошибка неверно заданной конфигурации лабиринта

1. Зададим конфигурацию лабиринта, у которого отсутствует дорога между стартом и финишем.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 15 Конфигурация лабиринта, с отсутствующей дорогой S-F.

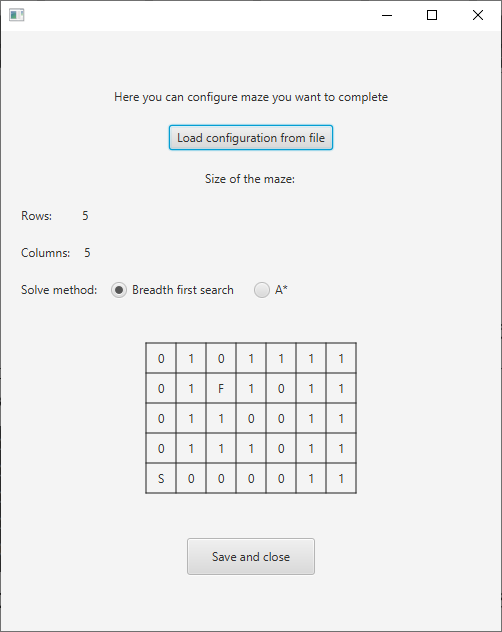


Рис. 16 Окно с заданной конфигурацией лабиринта

Заметим, что на данном этапе ошибка некорректной конфигурации не отображается. Попробуем решить такой лабиринт любым из способов.

В окне с решателем лабиринта отобразится заданная конфигурация лабиринта, возможность задать скорость решения, запустить решение и поставить его на паузу. Запустим решатель.

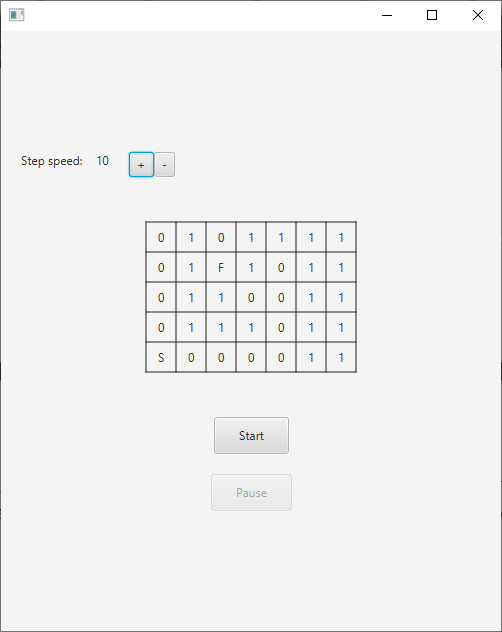


Рис. 17 Окно с решателем

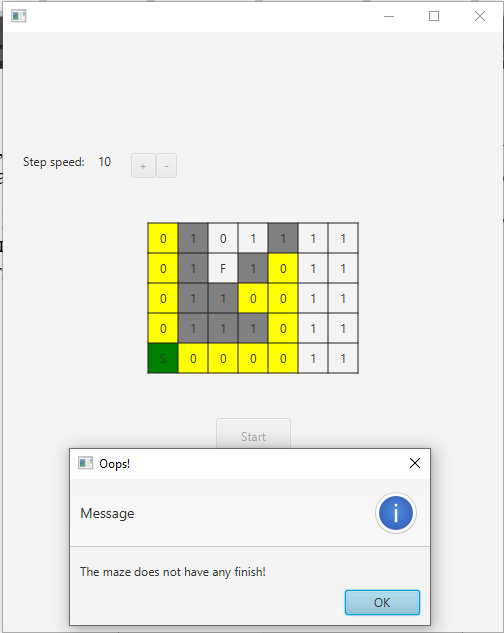


Рис. 18 Результат попытки прохождения лабиринта и сообщение об ошибке.

Решатель отметил зеленым цветом старт, желтым цветом свои пути, серым цветом встреченные на пути стенки. Заметим, что финиш находится в недоступной позиции для решателя, поэтому получаем ошибку.

1. Наконец, загрузим корректную конфигурацию лабиринта и попробуем сравнить оба способа решения.

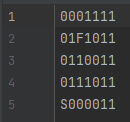


Рис. 19 Корректная конфигурация лабиринта

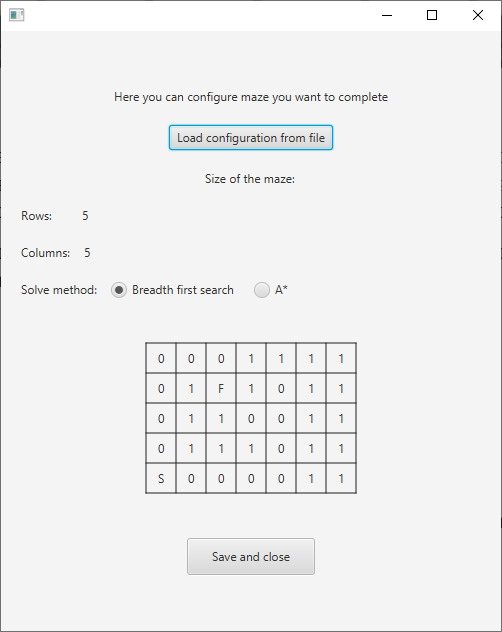


Рис. 20 Загруженная конфигурация лабиринта

Выберем поиск в ширину в качестве способа прохождения лабиринта.

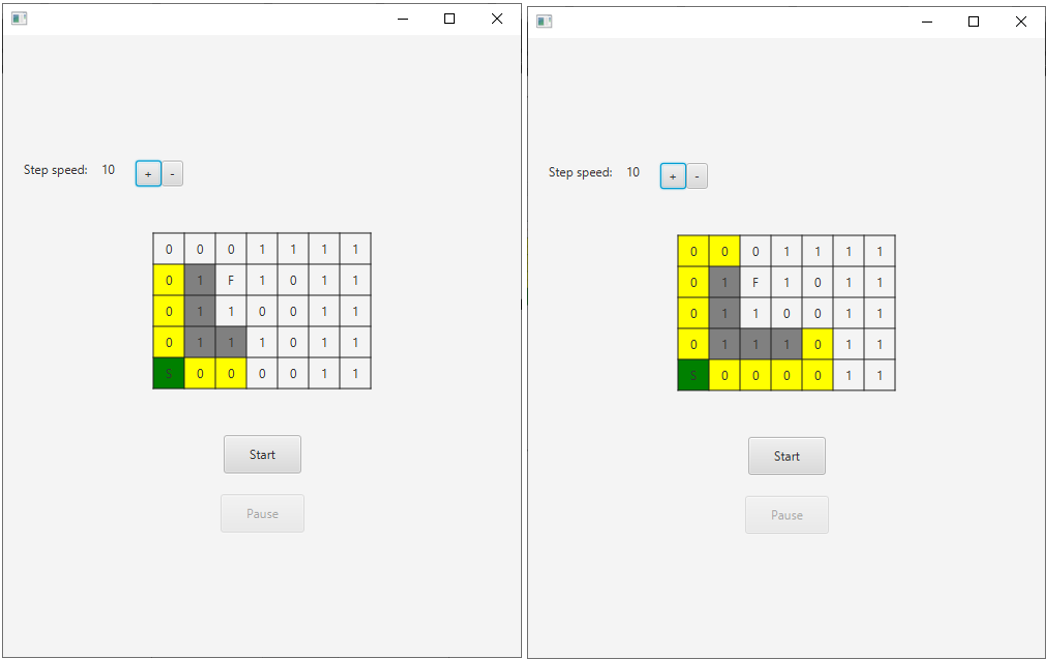


Рис. 21 Разные этапы решения лабиринта методом поиска в ширину

Заметим, что лабиринт «растет» с одинаковой скоростью во все допустимые стороны и успешно находит кратчайший путь к финишу.

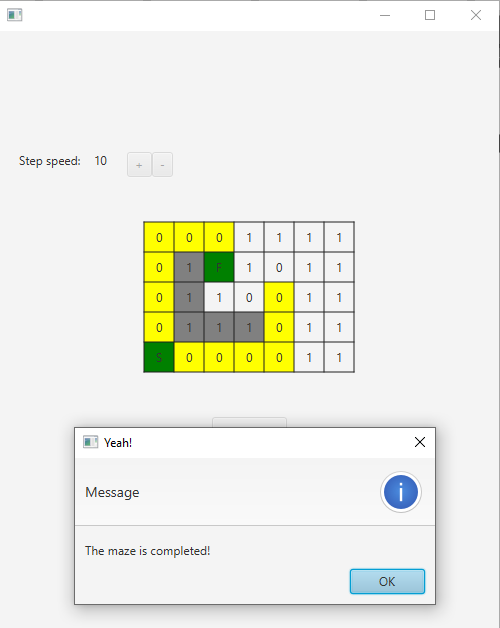


Рис. 22 Результат прохождения лабиринта

Изменим способ решения лабиринта на A\* и посмотрим на этапность решения:



Рис. 23 Выбор A\* в качестве способа решения лабиринта в конфигурации.

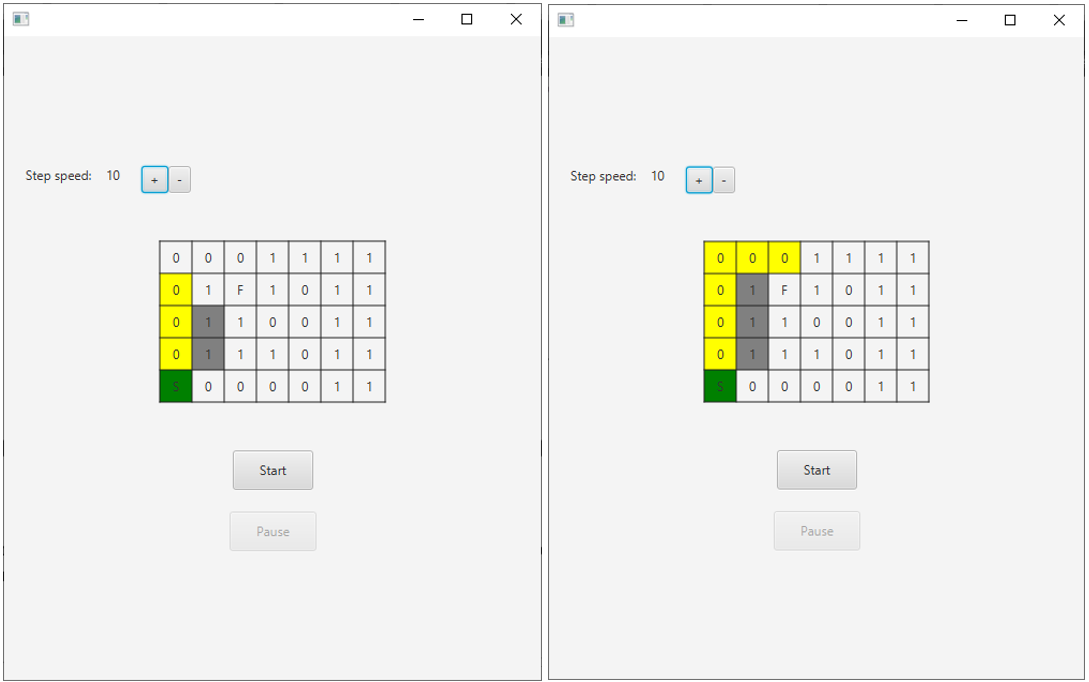


Рис. 24 Этапы решения лабиринта с помощью A\*

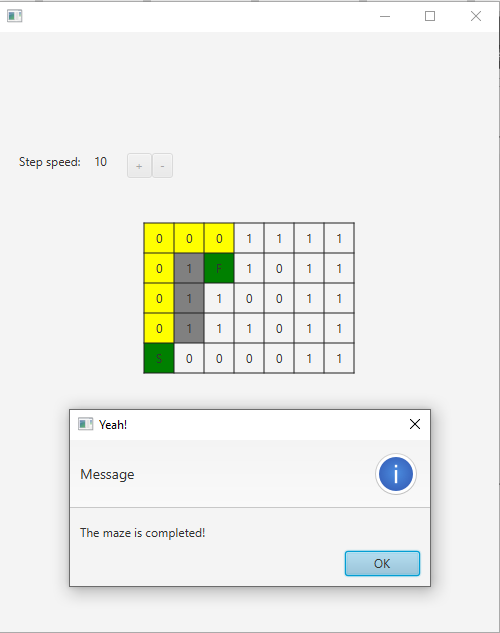


Рис. 25 Результат решения лабиринта с помощью A\*

Увеличим лабиринт и попробуем решить его с помощью двух методов.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 26 Конфигурация большого лабиринта

Изображение выглядит как стол

Автоматически созданное описание

Рис. 27 Результат прохождения большого лабиринта поиском в ширину.

Изображение выглядит как текст, шкафчик, снимок экрана

Автоматически созданное описание

Рис. 28 Результат прохождения большого лабиринта с помощью A\*.

Лабиринт решен двумя способами. С помощью A\* искались наиболее дешевые пути, за счет этого некоторые невыгодные места, стенки, ведущие в обратное направление решатель не затронул.

# Вывод

Таким образом, в ходе выполнения курсовой работы был реализован решатель лабиринта на JavaFX с возможностью задания конфигурации лабиринта через текстовый файл и с функцией обработок ошибок.