Данный документ является документацией к программному обеспечению TEST. В нем описывается интерфейс программы, используемый алгоритм и формат сохраняемых данных. Так же рассматриваются основные функции, используемые в разработанном ПО.

1. Условия задачи

Имеется квадратный лабиринт, состоящий из клеток (). Для каждой клетки задана стоимость (). Некоторые клетки помечены как красные, таких процентов от всего числа клеток в лабиринте (1). Лабиринт можно представить в виде матрицы, где в левом нижнем углу клетка , а в правом верхнем клетка , персонаж может двигаться по этому лабиринту только вправо и вверх. Персонаж должен пройти путь с наибольшей стоимостью из в при условии, что в красные клетки можно заходить не больше раз (). Стоимость пути есть сумма стоимости всех клеток, которые персонаж посетил. Если такой путь не найден, то стоимость пути равна 0.

1. Принцип работы разработанной программы

Входными данными программы являются 2 строки с названием файлов (in.txt, out.txt). При корректном указании аргументов? программа запустится и создаст лабиринт, основываясь на данных из файла in.txt. Пользовательские поля будут заполнены соответствующими данными. После чтения входного файла программа автоматически запустит алгоритм поиска оптимального пути и выведет результат – графическое представление найденного пути и его стоимость. Стоимость так же будет записана в выходной файл out.txt. Для генерации лабиринта с другими параметрами пользователю нужно нажать кнопку «Создать лабиринт». После этого лабиринт будет перестроен в соответствии с указанными параметрами. При нажатии кнопки «Найти путь» запустится алгоритм поиска пути и будет выведен результат его работы. Алгоритм можно пересоздавать неограниченное количество раз при условии корректно указанных данных.

1. Пользовательский интерфейс

Интерфейс разработанного программного обеспечения представлен на рисунке 1.



Его можно разделить на две части – взаимодействие с пользователем и графическое представление «лабиринта».

Левая часть графического представления ПО отвечает за ввод пользовательских данных, создание лабиринта по введенным данным, запуск алгоритма и вывод результата. Пользовательскими данными являются:

* Размер лабиринта с ограничением от 20 до 30;
* Процент красных клеток относительно общего количества клеток;
* Количество возможных посещений красных клеток с ограничением от 3 до 5.

В графе «Результат» выводится общая стоимость пути, рассчитанная алгоритмом после его выполнения. Если путь не найден, то стоимость пути равняется нулю.

Кнопка «Создать лабиринт» позволяет сгенерировать лабиринт по указанным данным со случайной «стоимостью» каждой клетки и случайным расположением красных клеток.

Справа располагается графическое представление лабиринта. Лабиринт может быть прочитан из файла входных данных или сгенерирован пользователем по введенным параметрам. Каждая клетка лабиринта имеет свою «стоимость», которая указана на ней. Цвета клеток лабиринта означают:

* Черный – обычная клетка;
* Красный – красная клетка. Такие клетки можно посетить ограниченное количество раз;
* Зеленый – путь, полученный в результате работы алгоритма.

1. Работа алгоритма

Алгоритм представляет из себя рекурсивный алгоритм поиска оптимального пути. Путь строится по принципу динамического программирования, запоминая ранее просчитанные состояния для ускорения вычислений. После нахождения оптимального пути восстанавливается реальный путь исходя из таблицы мемоизации и выбираются оптимальные шаги.

1. Функции и архитектура ПО

Программный код разбит на несколько классов для соблюдения принципов ООП и удобства использования. Ниже представлены основные классы и их роль в проекте:

* MainWindowViewModel – основной класс отвечающий за взаимодействие с пользователем;
* Cell – класс клетки;
* FileHelper – класс, отвечающий за работу (чтение и запись) с файлами;
* PathFinder – класс, отвечающий за работу алгоритма (поиск пути, восстановление пути);
* IntToBrushConverter – класс конвертации значения переменной в цвет клетки;
* App – родительский класс графического приложения

Приложение выстроено по принципу MVVM, благодаря чему логика представлений (графика) отделена от бизнес-логики (вычисления). Основной переменной класса MainWindowViewModel является переменная RP. Она является двумерной динамической коллекцией клеток Cell и служит в качестве дискретного рабочего поля – лабиринта. Функции, используемые в классе MainWindowViewModel:

* FindOptimalPath() – проверяет входные данные, запускает алгоритм поиска пути и измеряет его время работы, а так же вызывает другие функции;
* GenerateMaze() – создает случайный лабиринт по заданным параметрам;
* DrawPath(List<(int, int)> path) – визуализирует найденный путь.

Функции класса PathFinder:

* Solve() – ищет оптимальный путь и восстанавливает его;
* FindMaxPathCost(int row, int column, int remainingRedVisits) – ищет путь с максимальной стоимостью;
* ReconstructPath() – восстанавливает путь с помощью мемоизации.

Функции класса FileHelper:

* ReadInputData(string file) – читает данные из входного файла;
* ReadCostMatrix(IEnumerable<string> lines, ObservableCollection<ObservableCollection<Cell>> RP) – читает стоимость клеток из файла;
* ReadRedMatrix(string[] lines, int RPSize, ObservableCollection<ObservableCollection<Cell>> RP) – читает положение красных клеток из файла;
* WriteOutputData(List<(int, int)> path, int RPSize, float PercentRed,

int CountRedVisit, int Result, double duration, string output) – записывает данные в выходные файлы.

1. Формат сохраняемых данных

Результатом выполнения алгоритма является графическое отображение построенного пути, а так же 2 файла с результатами работы – out.txt и result.txt. Файл out.txt содержит только стоимость найденного пути в виде целого числа. Файл result.txt содержит информацию о размере лабиринта, проценте красных клеток, количестве возможных посещений красных клеток, найденный путь в виде последовательности точек, его стоимость и время, затраченное на выполнение алгоритма. Пример выходного файла result.txt представлен на следующем рисунке.

