Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет *Информатика и системы управления (ИУ)*

Кафедра *"Информационная безопасность" ИУ-8*

Отчет по домашнему заданию

Алгоритмы и структуры данных

Цыденов Илья Андреевич
Группа ИУ8-53

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Постановка задачи.

Постройте алгоритм генерации лабиринтов с нужным уровнем проходимости (лабиринт гарантированно должен быть проходим).

1.2. Описание известной задачи

Исходя условия задачи принято решение в качестве известной задачи использовать задачу обхода графа в глубину.

По условию существует задача построения лабиринта. В первоначальном виде задача не годится для реализации на языке программирования. Поэтому необходимо свести эту задачу к уже известной задаче , которую можно будет реализовать на языке программирования.

На вход первоначальной задачи подаются клетки лабиринта (пусть лабиринт будет прямоугольный). Они могут быть стенами и проходами. У каждой клетки есть координата, чтобы клетка имела свое место на поле лабиринта. Теперь поставим каждой клетке-проходу в соответствие вершину графа. Граф должен отображать двумерную координатную плоскость заданного прямоугольника, поэтому вершины необходимо расположить в виде прямоугольника с ровными рядами вершин и каждая вершина будет связана со своими соседями ребрами. На выходе первоначальной задачи получается лабиринт в любом виде , в соответствие ему ставится выход полученный в реализации задачи при помощи обхода графа в глубину, в котором получившееся дерево и есть лабиринт и между всеми не связанными вершинами соответственно находятся стены.

1.3. Описание основных подходов к решению известной задачи

Алгоритм поиска в глубину

Шаг 1. Всем вершинам графа присваивается значение не посещенная.

Выбирается первая вершина и помечается как посещенная.

Шаг 2. Для последней помеченной как посещенная вершины выбирается смежная вершина, являющаяся первой помеченной как не посещенная, и ей присваивается значение посещенная. Если таких вершин нет, то берется предыдущая помеченная вершина.

Шаг 3. Повторить шаг 2 до тех пор, пока все вершины не будут помечены как посещенные

Данный алгоритм имеет сложность O(|E| + |V|).

1.4. Описание метода для решения задачи

1. Сделать начальную клетку текущей и отметить ее как посещенную.
2. Пока есть непосещенные клетки
    1. Если текущая клетка имеет непосещенных «соседей»
        1. Поместить текущую клетку в стек
        2. Выбрать случайную клетку из соседних
        3. Поместить в стек ребро между текущей клеткой и выбранной
        4. Сделать выбранную клетку текущей и отметьте ее как посещенную.
    2. Иначе если стек не пуст
        1. Изъять клетку из стека
        2. Сделать ее текущей

1.5. Описание используемых структур данных

Так как метод решения задачи предполагает наличие стека , то эта структура данных необходима. В то же время для вывода полученных результатов в файл была бы удобна такая структура как очередь , так как в файл предполагается выводить списки вершин одного направления.

Очередь - это структура данных, добавление и удаление элементов в которой происходит путём операций push и pop соответственно. Притом первым из очереди удаляется элемент, который был помещен туда первым,

то есть в очереди реализуется принцип «первым вошел — первым вышел». У очереди имеется голова и хвост . Когда элемент ставится в очередь, он занимает место в её хвосте. Из очереди всегда выводится элемент, который находится в ее голове. Очередь поддерживает следующие операции:

- empty — проверка очереди на наличие в ней элементов,
- push (запись в очередь) — операция вставки нового элемента,
- pop (снятие с очереди) — операция удаления нового элемента,
- size — операция получения количества элементов в очереди.

Стек - структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека. Притом первым из стека удаляется элемент, который был помещен туда последним, то есть в стеке реализуется стратегия «последним вошел — первым вышел». Описание операций стека:

- empty — проверка стека на наличие в нем элементов,
- push (запись в стек) — операция вставки нового элемента,
- pop (снятие со стека) — операция удаления нового элемента.