Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №9

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему: «Поиск расстояний в графе»

Выполнил:

студент группы 23ВВВ4

Соснин Глеб

Проверил:

доцент, Юрова О. В.

Пенза, 2024

**Цель** – научиться реализовывать алгоритм поиска расстояний в графе.

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа G. Выведите матрицу на экран.

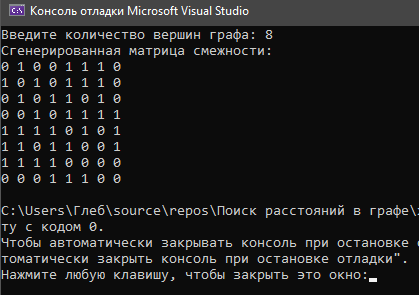


Рисунок 1 – Сгенерированная матрица смежности

1. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс queue из стандартной библиотеки С++.

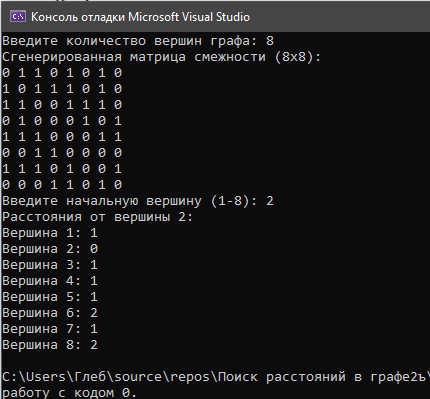


Рисунок 2 – Поиск расстояний в графе

Функция *generateGraph*: генерирует случайный граф в виде матрицы смежности. n — количество вершин графа. Двумерный вектор (матрица смежности) размером n×n, где каждая ячейка *adjacencyMatrix[i][j]* равна 1, если между вершинами i и j есть ребро, и 0 — в противном случае. Инициализирует матрицу смежности нулями. Заполняет верхнюю половину матрицы случайными значениями (0 или 1). Синхронизирует нижнюю половину, чтобы граф был неориентированным.

Функция *findDistances*: реализует поиск минимальных расстояний от заданной начальной вершины до всех остальных вершин графа с использованием алгоритма BFS (обход в ширину). Инициализирует массив расстояний значением INT\_MAX (неизвестное расстояние). Устанавливает расстояние до начальной вершины равным 0. Использует очередь *std::queue* для обработки текущей вершины и её соседей. Для каждой смежной вершины, которая ещё не обработана, обновляет её расстояние и добавляет в очередь. По завершении выводит расстояния или сообщение о недостижимости вершин.

3.\* Реализуйте процедуру поиска расстояний для графа, представленного списками смежности.

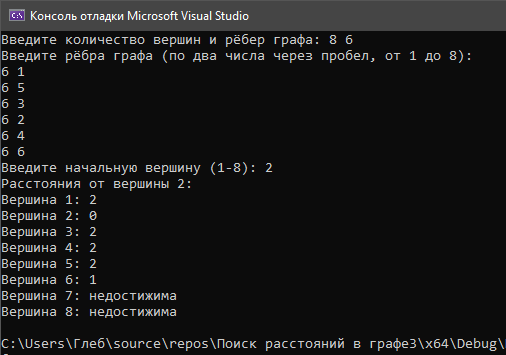


Рисунок 3 – Поиск расстояний для графа, представленного списками смежности

Функция программы *findDistances* (поиск расстояний с использованием обхода в ширину). Эта функция выполняет алгоритм поиска кратчайших расстояний от заданной начальной вершины до всех остальных вершин графа.

После завершения обхода в ширину функция выводит расстояния до всех вершин от начальной вершины. Если расстояние до вершины осталось *INT\_MAX*, то она считается недостижимой.

**Задание 2\***

1. Реализуйте процедуру поиска расстояний на основе обхода в глубину.

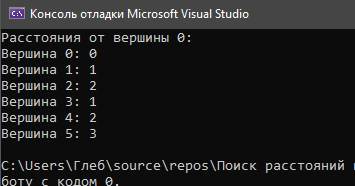


Рисунок 4 – Поиск расстояний на основе обхода в глубину

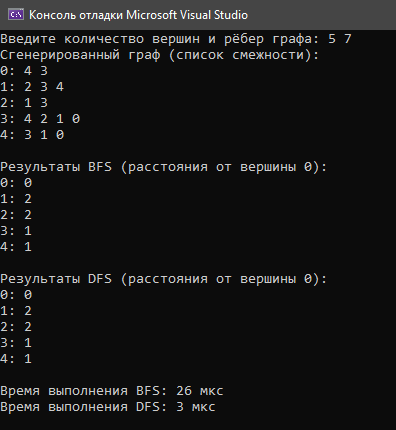
Функция программы *dfsMatrix* (поиск расстояний с использованием обхода в глубину). Эта функция реализует алгоритм поиска расстояний от начальной вершины в графе, представленном матрицей смежности, используя метод обхода в глубину.

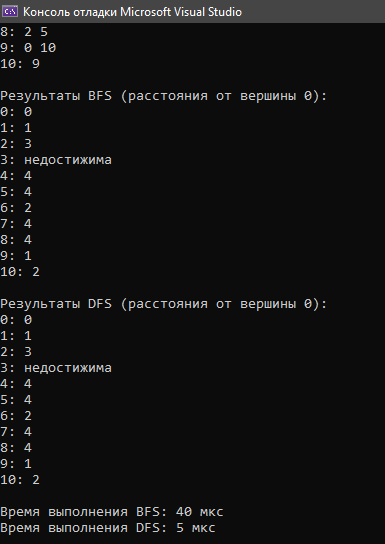
Граф представлен матрицей смежности. Для каждой вершины хранится информация о том, с какими другими вершинами она соединена (1 — соединена, 0 — не соединена).

Алгоритм *dfsMatrix* использует рекурсию для обхода графа и вычисления расстояний от начальной вершины до всех остальных.

После завершения поиска расстояний программа выводит расстояние от начальной вершины до каждой другой вершины. Если вершина недостижима, её расстояние будет равно "бесконечность".

1. Оцените время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.





Рисунки 5, 6 – Время работы алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков

На основе представленных выводов программы и полученных данных о времени выполнения для алгоритмов BFS и DFS, можно сделать следующие оценки:

1. BFS использует очередь для обработки вершин и посещает каждую вершину и каждое ребро ровно один раз. Для графа с 5 вершинами и 7 рёбрами выполнение заняло 26 мкс. Для графа с 10 вершинами и 10 рёбрами выполнение заняло 40 мкс.
2. DFS использует рекурсию или стек для посещения всех достижимых вершин. Для графа с 5 вершинами и 7 рёбрами выполнение заняло 3 мкс. Для графа с 10 вершинами и 10 рёбрами выполнение заняло 5 мкс.
3. DFS работает быстрее в ваших тестах из-за меньшего числа операций с памятью, так как рекурсия (или использование стека) минимально нагружает систему. Однако разница может нивелироваться или измениться при большом числе вершин и рёбер. BFS может потребовать больше памяти для хранения очереди, особенно если ширина графа (количество смежных вершин) значительная.

**Вывод:** в ходе выполнения лабораторной работы научились реализовывать алгоритм поиска расстояний в графе.