Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский химико-технологический университет имени Д.И. Менделеева»
Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Выполнил студент группы.....КС-36 Жижин Алексей Павлович

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/APZhizhin_36_2SEM

Приняли: .....Пысин Максим Дмитриевич
.....Краснов Дмитрий Олегович

Дата сдачи: 21.03.2022

Оглавление

Описание задачи.

Написать и проанализировать методы поиска в глубину и в ширину.

Описание метода/модели.

Поиск в глубину (англ. Depth-first search, DFS) — один из методов обхода графа. Стратегия поиска в глубину, как и следует из названия, состоит в том, чтобы идти «вглубь» графа, насколько это возможно. Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины рёбра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.

Поиск в ширину-один из методов обхода графа. Стратегия поиска в ширину, как и следует из названия, состоит в том, чтобы идти «вширь» графа, насколько это возможно. Т.е смотря все вершины, которые идут от текущей вершины, а потом уже последующие. Алгоритм поиска работает со всеми вершинами-соседями нашей текущей вершины. Если одна из соседок-нужный нам элемент, то работа заканчивается, если нет, то проверяем всех соседей наших соседей

Алгоритмическая сложность наших методов зависит от количества вершин и в какой структуре данных они представлены. Если наш метод работает с матрицей смежности, то сложность алгоритма считается O(n^2). Если со списком смежности, то сложность алгоритма считается O(n). Это объясняется тем, что для того, чтобы узнать с какими вершинами наша вершина соединяется, нам нужно пройти весь массив в двойном цикле(так как матрица) и не факт, что все n будут соединяться, а при списке, нам нужно пройти массив вершин, который будет хранить в себе конкретное количество соединений или не хранить совсем.

Я провел тесты и зафиксировал результаты работы алгоритмов. На Рис.1 представлены результаты в виде таблица, а на Рис.2 в виде графика. Глядя на эти результаты можно уверенно сказать, что метод обхода в ширину работает эффективнее, так как он выиграл 8 тестов из 10. 1 из тестов показал, что оба метода искали элемент, которого не оказалось среди наших вершин. При это результат работы- обыкновенный перебора всех элементов- показал одинаковую скорость работы программы. Что говорит о том, что быстрота нахождения элемента в первую очередь зависит от его расположения внутри графа.

Количество вершин	Обход в глубину	Обход в ширину
689	9	6
1027	7	4
2108	20	20
3053	25	28
4996	73	21
5313	104	76
6718	143	47
7892	235	95
8534	223	56
9428	226	245

Рис.1 Результат работы методов

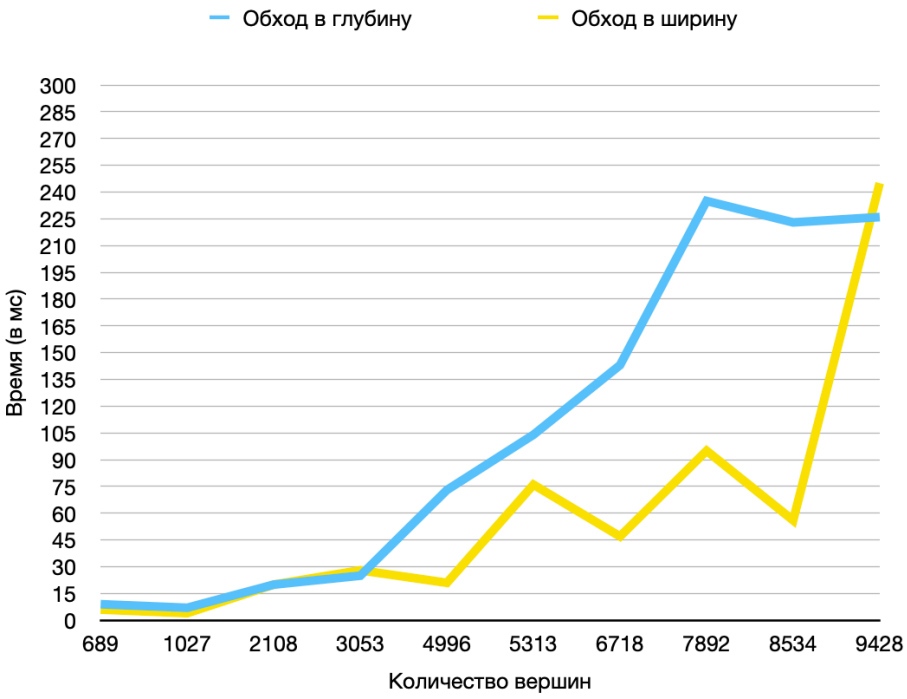

Рис. 2 Результат работы методов

Заключение.

Проводя тесты, я убедился в том, что метод обхода в ширину все-таки эффективнее, но у него есть свои минусы. Первый минус- непредсказуемое расположение элемента в графе, например, если искомый элемент будет левым листом, то метод обхода в глубину найдет его быстрее, так как он на это настроен, но с другой стороны, если элемент будет находится где-то в середине нашего графа, то

метод поиска в ширину отработает быстрее. К тому же главный минус метода обхода в глубину- рекурсивный вызов функции. Как нам известно многие языки программирования не следят за очисткой памяти и ответственность за это ложиться на разработчика, поэтому нам необходимо вручную очищать память. Я бы предпочел использовать метод обхода в ширину, так как считаю, что он эффективнее ищет элементы и занимает меньше памяти, но все-таки нужно исходить из задачи и от графа с которым работаем.