

Национальный исследовательский университет ИТМО

Кафедра программных систем

**Практическая работа 3**

Выполнил: Шебут

Денис Айссаевич

Группа № K3221

Проверил: Иванов С. Е.

Санкт-Петербург

2020

Практическая работа №3

АЛГОРИТМЫ НА ГРАФАХ

В работе будем использовать класс Matrix, созданный при выполнении практических работ 1 и 2. Необходимо дополнить данный класс, чтобы организовать работу с прямоугольными матрицами смежности и инцидентности.

Также создадим класс Graph, который будет объединять все данные о графе и содержать методы взаимодействия с ним. В дополнительном классе AlgoHelper будут реализованы требуемые алгоритмы на графах.

1. Дополним класс Matrix, отредактируем некоторые методы, добавим конструктор:

Изображение выглядит как черный, оранжевый, держит, красный

Автоматически созданное описаниеИзображение выглядит как монитор, экран, стол, черный

Автоматически созданное описание

Рисунок 1 – поля класса Matrix, кол-во строк и столбцов в матрице

Рисунок 2 – перегруженный конструктор класса Matrix

1. Создадим класс Graph:
   1. Объявим поля класса

**Изображение выглядит как снимок экрана, монитор, экран, телефон

Автоматически созданное описание**

Рисунок 3 – поля класса Graph

* 1. Теперь перейдем к методам

Изображение выглядит как снимок экрана, сидит, телефон, стол

Автоматически созданное описание

Рисунок 4 – конструкторы класса Graph

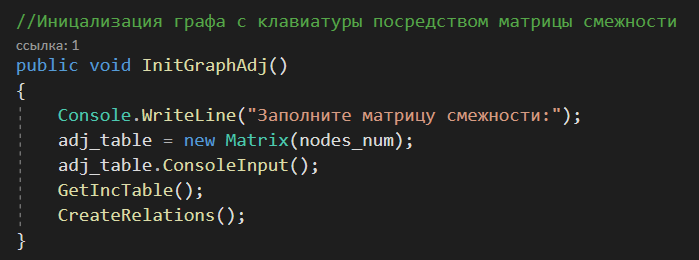


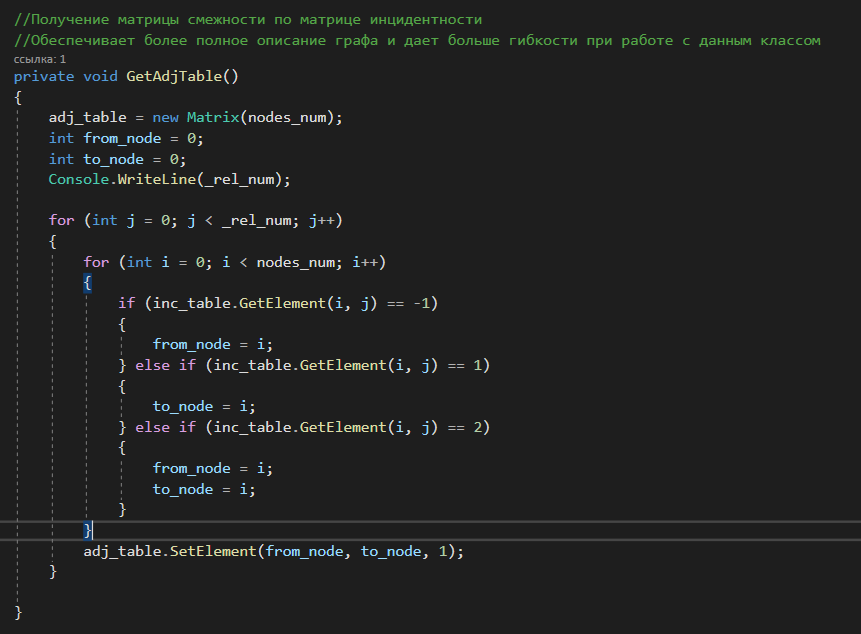
Рисунок 5 – инициализация графа посредством матрицы смежности. Использован метод ConsoleInput класа Matrix из прошлых практических работ

Изображение выглядит как экран, ноутбук, черный, комната

Автоматически созданное описание

Рисунок 6 – инициализация графа посредством матрицы инцидентности

Методы GetAdjTable() и GetIncTable() строят матрицу смежности и инцидентности соответственно.

Рисунок 7 – метод GetAdjTable()

Изображение выглядит как снимок экрана, телефон

Автоматически созданное описание

Рисунок 8 – метод GetIncTable()

Метод CreateRelations() (рис. 5-6) инициализирует словарь relations, который потом будет удобно использовать в алгоритмах как представление всего графа.

Изображение выглядит как компьютер

Автоматически созданное описание

Рисунок 9 – метод CreateRelations()

Для последующей реализации алгоритма Дейкстры создадим метод InitWeights()

Изображение выглядит как снимок экрана, компьютер

Автоматически созданное описаниеРисунок 10 –

Метод InitWeights()

* 1. Дополнительно переопределим метод ToString() для вывода актуальной информации о графе на экран

Изображение выглядит как монитор, стол, черный, экран

Автоматически созданное описание

Рисунок 11 – переопределенный метод ToString()

1. Теперь, когда у нас есть удобные инструменты для организации работы с графами, перейдем к реализации алгоритмов DFS, BFS, Дейкстры. Создадим статический класс AlgoHelper с методами, реализующими вышеуказанные алгоритмы:

Изображение выглядит как счетчик, темный, мяч, комната

Автоматически созданное описание

Рисунок 12 – класс AlgoHelper

* 1. Изображение выглядит как телефон

     Автоматически созданное описаниеРассмотрим алгоритм DFS:

Рисунок 13 – метод DFS() инициализирующий рекурсивный поиск в глубину DFSVisit()

Изображение выглядит как снимок экрана, телефон

Автоматически созданное описаниеОсновная работа алгоритма происходит в методе DFSVisit()

Рисунок 14 – метод DFSVisit(). Происходит рекурсивный запуск метода из каждого узла графа. Ведется учет посещенных вершин в массиве marker[]

* 1. Изображение выглядит как снимок экрана, компьютер

     Автоматически созданное описаниеТеперь перейдем к алгоритму BFS:

Рисунок 15 – метод BFS(). «Видимые» из начальной вершины непосещенные узлы откладываются в очередь еще не посещенных вершин. При этом обновляются данные о расстоянии (кол-во ребер) до вершины. Так до тех пор, пока в очереди есть узлы

* 1. Алгоритм Дейкстры поиска кратчайших путей на графе:

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 16 – метод Dijkstra(), в котором происходит инициализация алгоритма и вывод итогового результата работы

Изображение выглядит как снимок экрана, монитор, сидит, черный

Автоматически созданное описание

Рисунок 17 – метод DijkstraVisit(), в котором происходит основная работа алгоритма. Находясь в данной вершине, обновляем длину пути для всех связанных вершин если новый путь короче старого. Затем, выбираем самую близкую доступную вершину, текущую помечаем как посещенную, перемещаемся в выбранную вершину

**Вывод:** в ходе выполнения данной практической работы был реализован класс Graph обеспечивающий не только удобные инструменты работы с графами, но и являющийся целостное представление данных. На основе данного класса были реализованы базовые алгоритмы на графах: DFS, BFS, алгоритм Дейкстры.