МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра алгебры, геометрии и дискретной математики**

Направление подготовки «Прикладная математика и информатика»

**ОТЧЕТ**

по учебной практике

на тему:

**«Изучение алгоритма поиска в ширину и задачи нахождения кратчайшего пути»**

**Выполнил:** студент группы 381703-1

**\_\_\_\_\_\_\_\_\_\_\_\_\_** Шахнов А. В.

Подпись

**Научный руководитель:**

Доцент кафедры АГДМ, кандидат физико-математических наук

**\_\_\_\_\_\_\_\_\_\_\_\_\_** Сорочан С.В.

Подпись

Нижний Новгород

2021

**Содержание**

1. **Введение**
2. **Постановка учебно-практической задачи**
3. **Руководство пользователя**
4. **Наблюдения и выводы**

# Введение

Графы являются существенным элементом математических моделей в самых разнообразных областях науки и практики. Они помогают наглядно представить взаимоотношения между объектами или событиями в сложных системах. Многие алгоритмические задачи дискретной математики могут быть сформулированы как задачи, так или иначе связанные с графами, например, задачи, в которых требуется выяснить какие-либо особенности устройства графа, или найти в графе часть, удовлетворяющую некоторым требованиям, или построить граф с заданными свойствами.

# Постановка учебно-практической задачи

# Основные понятия и определения

*Обыкновенным графом* называется пара G = (V, E), где V – конечное множество, E – множество неупорядоченных пар различных элементов из V. Элементы множества V называются *вершинами графа*, элементы множества E – его *ребрами*.

Если ребра имеют направления – граф называют *ориентированным*. Если между любой парой вершин существует как минимум один путь – граф называют *связным*. Разные рёбра могут иметь одинаковые начала и концы. В таком случае говорят, что графе допускаются *кратные рёбра*, а граф называют *мультиграфом*. Ребро, соединяющее вершину с самой собой называют *петлёй*. Если такие ребра не допускаются – говорят, что рассматривается граф без петель.

Также есть несколько разных *способов задания* графов.

Пусть G - граф с n вершинами, причем VG = {1,2…n}. Построим квадратную матрицу A порядка n, в которой элемент Aij, стоящий на пересечении строки с номером i и столбца с номером j, определяется следующим образом: Aij = 1, если между i и j вершинами существует ребро, и Aij = 0, если ребро отсутствует.

Матрица A называется *матрицей смежности* графа. Матрицу смежности можно построить и для ориентированного графа, и для неориентированного, и для графа с петлями. Для обыкновенного графа она обладает двумя особенностями: из-за отсутствия петель на главной диагонали стоят нули, а так как граф неориентированный, то матрица симметрична относительно главной диагонали. Обратно, каждой квадратной матрице порядка n, составленной из нулей и единиц и обладающей двумя указанными свойствами, соответствует обыкновенный граф с множеством вершин {1,2…n}.

Другая матрица, ассоциированная с графом - это *матрица инцидентности.* Для ее построения занумеруем вершины графа числами от 1 до n, а ребра - числами от 1 до m. Матрица инцидентности I имеет n строк и m столбцов, а ее элемент Iij равен 1, если вершина с номером i инцидентна ребру с номером j, в противном случае он равен нулю.

В практической работе используются связные неориентированные графы без петель.

# Поиск в ширину Поиск в ширину – (англ. *breadth-first search*, BFS) – один из методов обхода графа.

Идея поиска в ширину состоит в том, чтобы посещать вершины в порядке их удаленности от некоторой заранее выбранной или указанной стартовой вершины a. Иначе говоря, сначала посещается сама вершина a, затем все вершины, смежные с a, то есть находящиеся от нее на расстоянии 1, затем вершины, находящиеся от a на расстоянии 2, и т.д.

Основная особенность поиска в ширину, отличающая его от других способов обхода графов, состоит в том, что в качестве активной вершины выбирается та из открытых, которая была посещена раньше других. Именно этим обеспечивается главное свойство поиска в ширину: чем ближе вершина к старту, тем раньше она будет посещена. Для реализации такого правила выбора активной вершины удобно использовать для хранения множества открытых вершин очередь – когда новая вершина становится открытой, она добавляется в конец очереди, а активная выбирается в ее начале.

**Поиск кратчайшего пути**Задача о кратчайшем пути является одной из важнейших задач теории графов. Значимость данной задачи определяется ее различными практическими применениями. Например, в GPS-навигаторах осуществляется поиск кратчайшего пути между двумя перекрёстками. В качестве вершин выступают перекрёстки, а дороги являются рёбрами, которые лежат между ними. Если сумма длин дорог между перекрёстками минимальна, тогда найденный путь самый короткий. Задача поиска кратчайшего пути на графе может быть определена для неориентированного, ориентированного или смешанного графа. В моей практической работе рассмотрена задача для неориентированного графа. Для смешанного и ориентированного графа дополнительно должны учитываться направления ребер.

**Задачи:**

1. Написать программу, которая будет выполнять обход графа методом поиска в ширину для двудольных графов и расщепляемых графов
2. Добавить в программу функцию, которая будет находить кратчайший путь между заданными вершинами
3. Определить трудоемкость алгоритма поиска в ширину на определенном классе графов, путём нахождения среднего времени поиска на графах со 100,200,300,…,1000,2000,2500 вершинами. Отобразить результат на графике

# Руководство пользователя

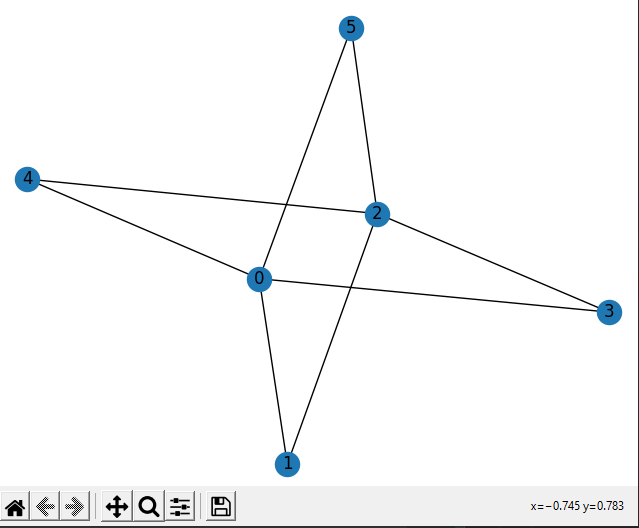
Для начала работы программы для класса двудольных графов, необходимо запустить проект bi\_graph.py

Программа генерирует по 10 двудольных графов на каждое число вершин из списка data\_n, который представлен в следующем виде:

data\_n = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 2500]

Для каждого из сгенерированных графов производится обход методом поиска в ширину и измеряется время, потраченное на поиск. Затем формируется список, элементами которого являются значения времени, затраченного на поиск в ширину. После формирования данного списка вычисляется сумма всех его элементов (т.е. время, затраченное на поиск в ширину на этих 10 графах), а затем вычисляется среднее время поиска в ширину.   
Из всех средних значений формируется новый список, который в дальнейшем будет использован для построения графиков с результатами.

Чтобы убедиться, что программа действительно строит двудольный граф, нарисуем сгенерированный граф для 6-ти вершин:



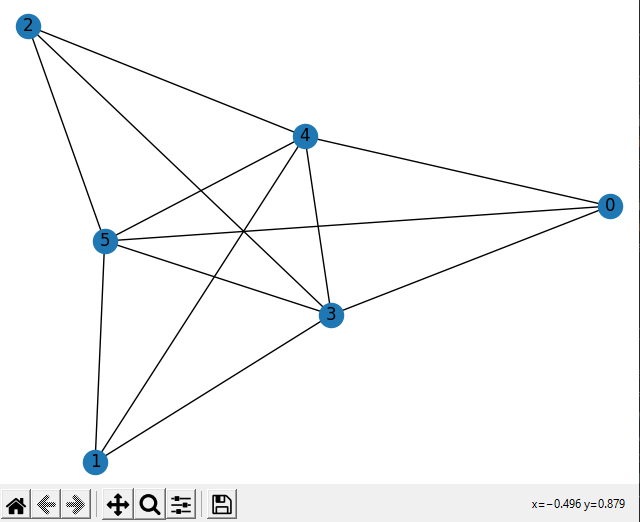
Помимо поиска в ширину, в программе реализован поиск кратчайшего пути от заданной вершины до любой другой. По умолчанию поиск ведётся от вершины с номером «0» до всех остальных вершин в графе. После выполнения поиска пользователю выводится список, в который включены маленькие списки с путями от нулевой вершины до остальных, а также эксцентриситет вершины, от которой ведётся поиск.

Пример:  


На картинке показаны кратчайшие пути от вершины 0 до вершины 3, 5, 7 и 9.

Для начала работы программы для класса расщепляемых графов графов, необходимо запустить проект split\_graph.py

Алгоритм работы программы абсолютно идентичен, за исключением того, что на каждое число вершин из списка data\_n генерируется по 10 расщепляемых графов.

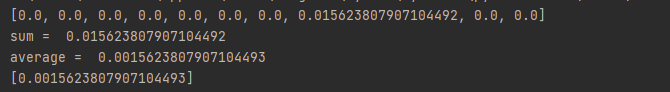
Чтобы убедиться, что программа действительно строит расщепляемый граф, нарисуем сгенерированный граф для 6-ти вершин:  


Вершины 0, 2 и 1 являются независимым множеством, а вершины 1, 3, 4, 5 – кликой (все вершины попарно смежны между собой).

**Наблюдения и выводы**

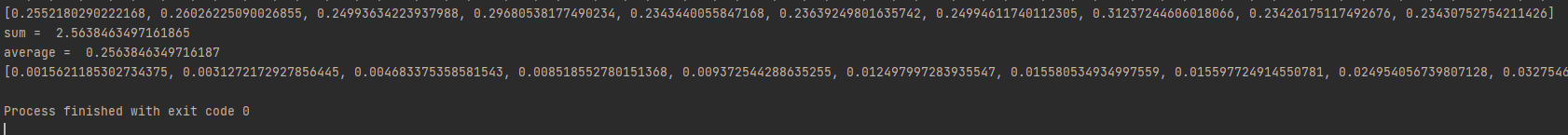
В данной работе были была разработана программа для генерации двудольных и расщепляемых графов, а затем проведения экспериментов на этих классах графов.   
В рамках учебной практики был проведён следующий эксперимент: замерялось среднее время работы 10 поисков в ширину на двудольных/расщепляемых графах из 100,200,300…1000,2000,2500 вершин, и строился график зависимости времени от числа вершин.

Пример вывода программы при проведении поиска в ширину для 10-ти двудольных графов со 100 вершинами:

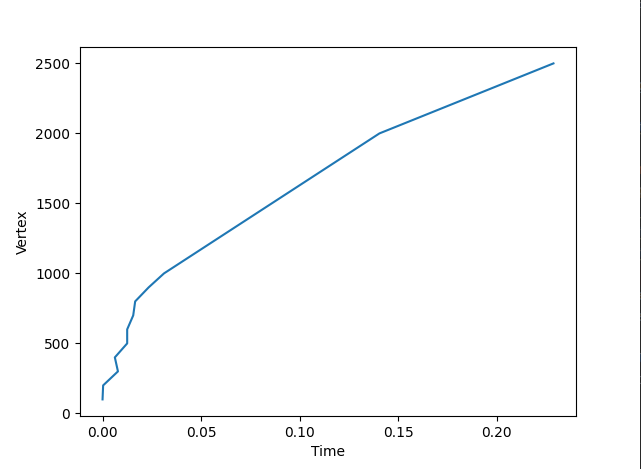


Интересное наблюдение – при относительно малом количестве вершин в графе поиск в ширину может занимать 0.0 секунд, так как время, затраченное на него считается незначительным (особенность языка Python)

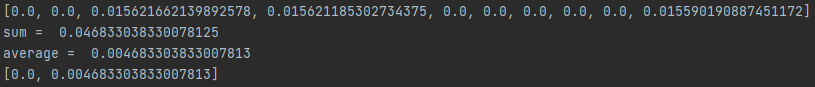
Пример вывода программы при проведении поиска в ширину для 10-ти двудольных графов с 2500 вершинами:



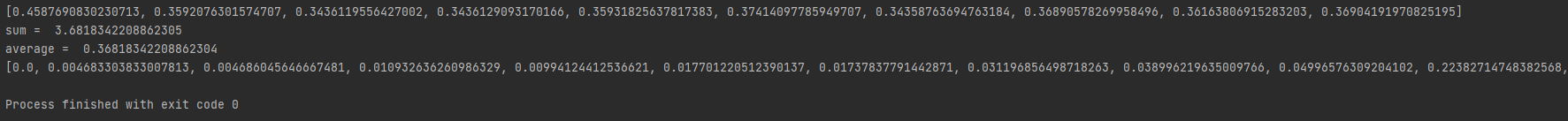
Результат проведения экспериментов на двудольных графах:



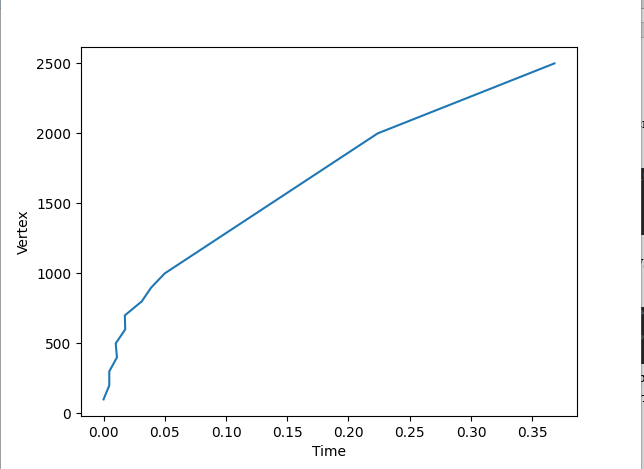
Пример вывода программы при проведении поиска в ширину для 10-ти расщепляемых графов с 200 вершинами:



Пример вывода программы при проведении поиска в ширину для 10-ти расщепляемых графов с 2500 вершинами:

  
Интересное наблюдение – в зависимости от того, в какую часть расщепляемого графа (независимое множество или клика) попала нулевая вершина, её эксцентриситет будет равен либо 1 (если вершина попала в клику), либо 2 (если вершина попала в независимое множество)

Результат проведения экспериментов на расщепляемых графах:



Результат исследования: Посмотрев на графики, можно увидеть, что поиск в ширину на расщепляемых графах работает немного медленнее, чем на двудольных графах.