Департамент образования администрации г. Братска

Муниципальное бюджетное общеобразовательное учреждение

«Гимназия № 1 имени А.А. Иноземцева»

**Теория графов. Алгоритм поиска в ширину. Алгоритм Дейкстры.**

Работа ученика 10 Б класса Бородина К. А.

Руководитель проекта

Буряк Нелли Зуфаровна

Работа допущена к защите « 17 » апреля 2021г.

Подпись руководителя проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_( Буряк Н.З. )

Братск

2021 г.

Оглавление

[Глава 1 4](#_Toc69575322)

[1.1 Теория графов 4](#_Toc69575323)

[1.2 Виды графов 5](#_Toc69575324)

[1.3 Поиск в ширину 6](#_Toc69575325)

[1.4 Алгоритм Дейкстры 6](#_Toc69575326)

[Глава 2 8](#_Toc69575327)

[2.1 Реализация графов в коде 8](#_Toc69575328)

[2.2 Алгоритм поиска в ширину 8](#_Toc69575329)

[2.3 Алгоритм Дейкстры 9](#_Toc69575330)

[2.4 Создание программы для нахождения кратчайшего маршрута, между городами 10](#_Toc69575331)

[2.4.1 Получение данных 10](#_Toc69575332)

[2.4.2 Создание графа 10](#_Toc69575333)

[2.4.3 Нахождение кратчайшего пути 11](#_Toc69575334)

[2.4.4 Получение кратчайшего маршрута 12](#_Toc69575335)

[Заключение 13](#_Toc69575336)

[Список использованных источников 14](#_Toc69575337)

[Приложение 15](#_Toc69575338)

**Введение**

В нашей жизни мы постоянно сталкиваемся с задачами на графы. Например, нам нужно посетить несколько магазинов, и мы знаем, где они располагаются, поэтому мы решаем ехать в тот, что находится ближе, и тем самым строим оптимальный маршрут. Но если таких магазинов становится во много раз больше и мы уже не можем так легко построить наш маршрут, тогда нам может помочь компьютер. Вписав названия магазинов и расстояние между ними, компьютерная программа очень быстро проложит маршрут. Какие компьютер использует для этого алгоритмы и как их создать на языке Python? Именно на эти вопросы я буду искать ответ и создам программу, которая будет находить кратчайшее расстояние между городами, если до необходимого города нет прямого маршрута, при заданном движении автобусов.

**Цель работы:** создание программы на Python, просчитывающей маршрут с пересадками для перемещения из города A в город B за минимальное время при заданном расписании движения автобусов между городами.

**Задачи:**

1. Узнать о алгоритме поиска в ширину и его применениях.
2. Узнать о алгоритме Дейкстры и его применениях.
3. Реализовать алгоритм поиска в ширину в коде.
4. Реализовать алгоритм Дейкстры в коде.
5. Создать программу на Python, просчитывающей маршрут с пересадками для перемещения из города A в город B за минимальное время при заданном расписании движения автобусов между городами

**Конечный продукт:** программа на Python.

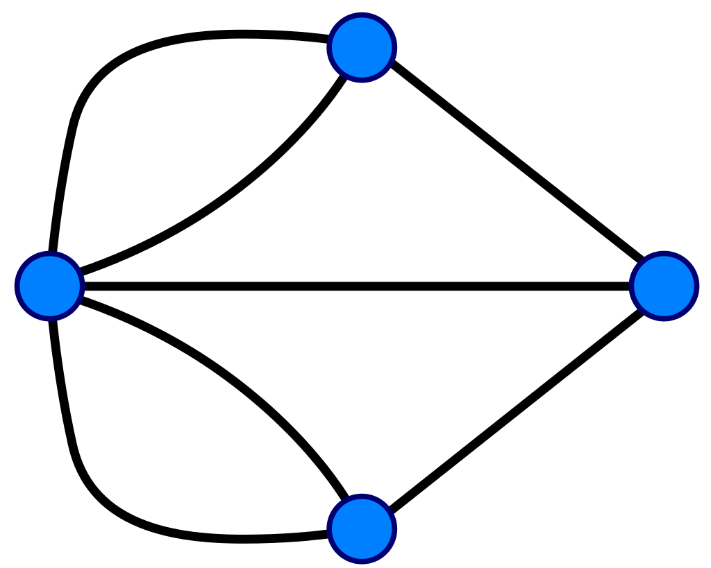
# Глава 1

## 1.1 Теория графов

Граф – множество вершин, соединенных ребрами.

Теория графов – раздел дискретной математики, изучающий свойства графов, с помощью которых можно решать различные задачи, имеющие формулировку: «как за наименьшее количество времени добраться из пункта А в пункт B», помогают решить алгоритмы, описываемые в теории графов.

Появился данный раздел дискретной математики благодаря Леонарду Эйлеру в 1736 для решения задачи о семи кёнигсбергских мостах. Суть данной задачи заключается в нахождении пути, по которому можно пройти все семь мостов, не проходя по ним дважды. Вот так выглядит граф этой задачи:



В ходе рассуждений Эйлер вывел первые правила графов:

* Число нечётных вершин (вершин, к которым ведёт нечётное число рёбер) графа должно быть чётно. Не может существовать граф, который имел бы нечётное число нечётных вершин.
* Если все вершины графа чётные, то можно начертить этот граф без отрыва карандаша от бумаги, при этом можно начинать с любой вершины графа и завершить его в той же вершине.
* Если ровно две вершины графа нечётные, то можно начертить этот граф без отрыва карандаша от бумаги, при этом нужно начинать с одной из нечётных вершин и завершить его в другой нечётной вершине.
* Граф с более чем двумя нечётными вершинами невозможно начертить одним росчерком.

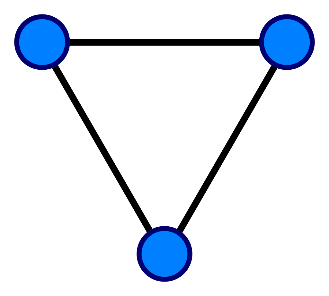
Граф кёнигсбергских мостов имел четыре нечётные вершины, следовательно невозможно пройти по всем мостам, не проходя ни по одному из них дважды.

На данный момент теория графов имеет примените в множестве отраслей: проектировка дорог, протягивание ЛЭП между городами, нахождения кратчайшего пути в магазин в навигаторах и т.д.

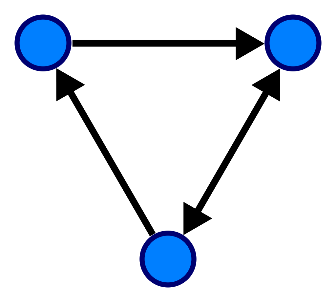
## 1.2 Виды графов

Все графы можно примерно разделить на 4 группы:

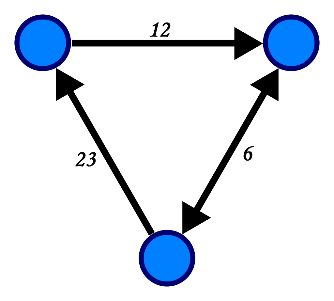
* Неорграф (неориентированный граф) – граф, имеющий неупорядоченные вершины, при которых если узлы соединены ребрами, то можно перемещаться между вершинами в обе стороны.



* Орграф (ориентированный граф) – граф, имеющий упорядоченные вершины, из которых можно перейти в другой узел только по заданному направлению ребер.



* Невзвешенный граф – граф, не имеющий параметров ребер, таких как длина, время прохождения и т.д. Может являться как орграфом, так и неорграфом.
* Взвешенный граф – граф имеющий параметр ребер, обозначающих их вес – величину, требуемую при прохождении из одного узла в другой по заданному ребру. Может являться как орграфом, так и неорграфом.



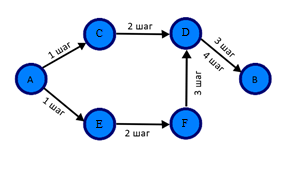
## 1.3 Поиск в ширину

Данный алгоритм помогает найти ответ на 2 вопроса:

* Существует ли путь от узла A к узлу B?
* Как выглядит кратчайший путь от узла A к узлу B?

Поиск в ширину помогает решать задачи с невзвешенными орграфами и неорграфами. Компьютер, решая задачу, использует данный алгоритм из заданного узла проходит к узлам соседям (1 шаг). Дальше из узлов соседей мы перемещаемся к соседям данного узла (2 шаг), пока не дойдем до нужного узла (3, 4 и т.д. шагов). Как только компьютер за определённое количество шагов достигает необходимого узла, алгоритм прерывается и выводится кратчайший путь, если он существует.

Для данного графа кратчайший путь будет равен 3 шагам, путь: A –> C –> D –> B.



Применением данного алгоритма можно назвать целое множество, поэтому приведу три примера:

* Создать алгоритм для игры в шашки, чтобы победить за наименьшее количество ходов.
* Проверка правописания, преобразуя ошибочно написанное слово в правильное за наименьшее количество изменений.
* Как переместится из города A в город B за наименьшее количество пересадок.

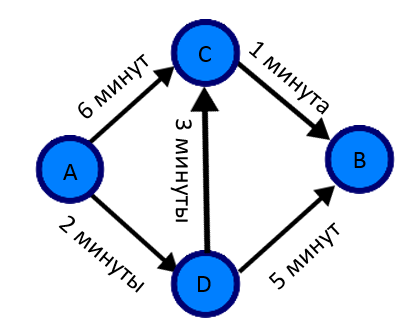
## 1.4 Алгоритм Дейкстры

Данный алгоритм является необходимым, когда нам нужно найти кратчайший путь в взвешенных графах. Используя в данных задачах алгоритм поиска в ширину будет некорректно, ведь при расчёте кратчайшего пути не используется вес (стоимость) ребер, поэтому данный алгоритм не будет давать правильный ответ. Для нахождения кратчайшего пути в задачах, которые имеют вес ребер, был создан алгоритм Дейкстры.

Алгоритм Дейкстры состоит из 4 пунктов:

1. Взять начальный узел и записать стоимость его соседей, перейти к следующему узлу.
2. Записать стоимость соседей данного узла, обновить стоимость соседей, если она оказалась меньше данной.
3. Повторять, пока это не будет сделано для всех узлов в графа.
4. Вычислить итоговый путь.

Рассмотрим данный алгоритм на примере. Мы находимся в городе A нам нужно попасть в город B, но прямой дороги в город B нет, только через города D и C.



Мы находимся в вершине A, записываем стоимости для узлов C – 6, D – 2. Переходим к следующему узлу C, записываем стоимость для узла D – 6 + 1 = 7. Переходим к узлу B, из него мы не можем попасть в другие узлы, но у нас еще остался узел D, который мы не посещали. Переходим в узел D, через узел D расстояние до B 5 + 2 = 7, стоимость узла B, не меняется, но через узел D, можно попасть в узел C, при этом стоимость данного узла уменьшится с 6 до 2 + 3 = 5, так как стоимость для узла C поменялась, обновляется и расстояние до узла B, вместо 6 + 1 = 7, оно будет 5 + 1 = 6. Все узлы пройдены, алгоритм прекращает работу, результат 6 минут – кратчайшее время за которое можно добраться из вершины A в вершину B.

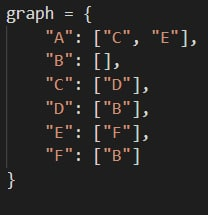
Для визуального представления работы алгоритма предлагаю ознакомится с приложением 3.

# Глава 2

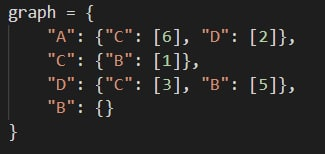
## 2.1 Реализация графов в коде

Графы почти во всех языках программирования, реализуются в виде словарей, где ключом является узел, значением являются соседние узлы, в которые можно попасть из данного узла, в взвешенных графах, так же указывается вес ребра ведущего к этому узлу.

Граф для п. 1.3:

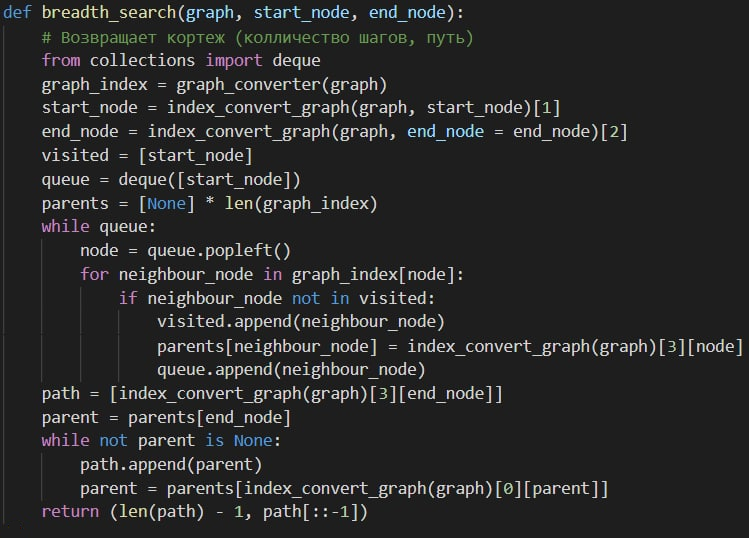


Граф для п 1.4:



## 2.2 Алгоритм поиска в ширину

Для реализации данного алгоритма в коде создадим функцию «breadth\_search», которая будет находить наименьшее количество шагов необходимое для достижении конечного узла, если это возможно.



На вход мы получаем граф, начальный узел и узел до которого необходимо добраться. Данный алгоритм обходит каждую из вершин и находит для нее кратчайшее расстояние и выдает кратчайшее расстояние до необходимой вершины.

Чтобы проверить правильность работы данного алгоритма, возьмем граф предложенный в главе 1.3 и сверим результаты.

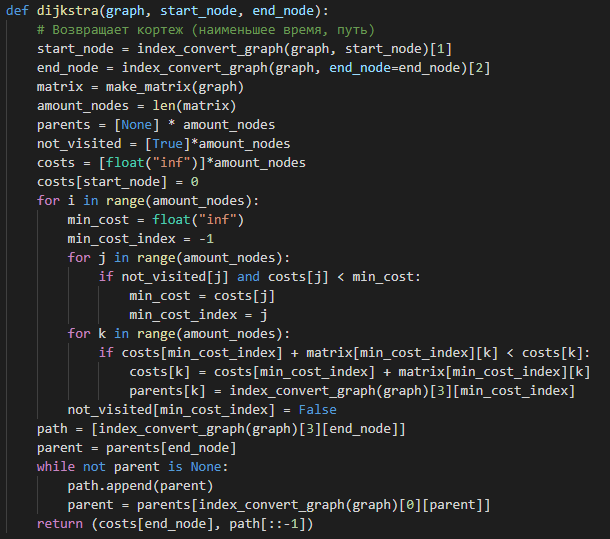


Так как результаты совпали, алгоритм корректно работает.

## 2.3 Алгоритм Дейкстры

Для реализации данного алгоритма создадим функцию «dijkstra», которая будет находить наименьшее время за которое можно попасть в конечный узел, если это является возможным.

На вход мы получаем граф, начальный узел и узел до которого необходимо добраться. Алгоритм обходит каждую из вершин графа и составляет стоимость до каждого из них и проверят есть ли более короткий путь, через другие вершины, если такой находится, обновляется стоимость для данного узла. Когда алгоритм проходит все вершины, выдается наименьший результат, до необходимой нам вершины.



Чтобы проверить правильность работы данного алгоритма, возьмем граф предложенный в главе 1.4 и сверим результаты.

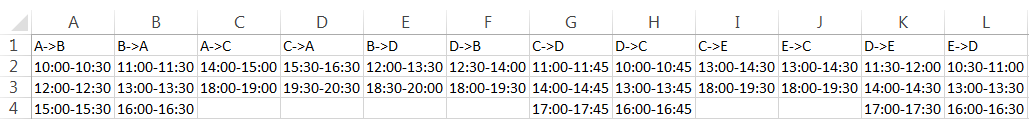


Результаты, совпали, поэтому можно утверждать, что данный алгоритм успешно работает.

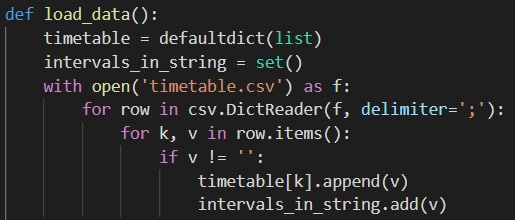
## 2.4 Создание программы для нахождения кратчайшего маршрута, между городами

### 2.4.1 Получение данных

Программа будет получать данные из таблицы с расширением «csv», которую также можно создавать и открывать в приложении «Excel», указав соответствующее расширение при сохранении файла.



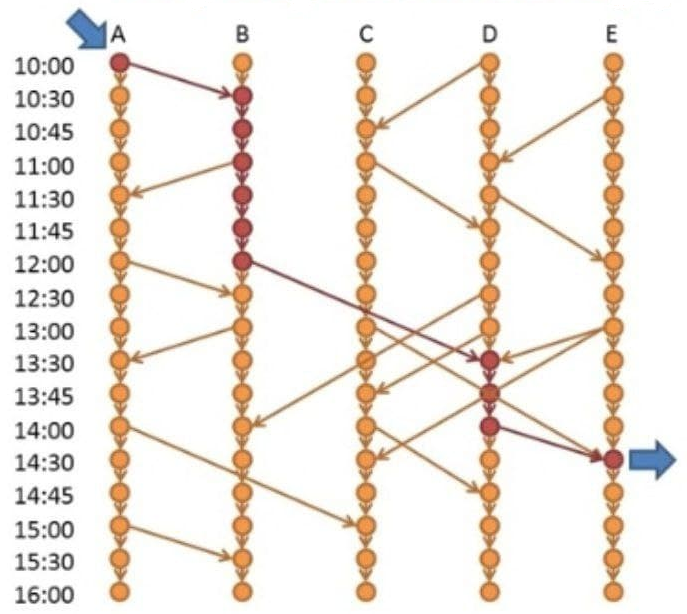
Создадим функцию «load\_data», которая обработает и загрузит данные из таблицы в программу. Данная функция обрабатывает данные и заносит их в словарь, где ключом являются города между которыми мы перемещаемся, а значениями промежутки времени.



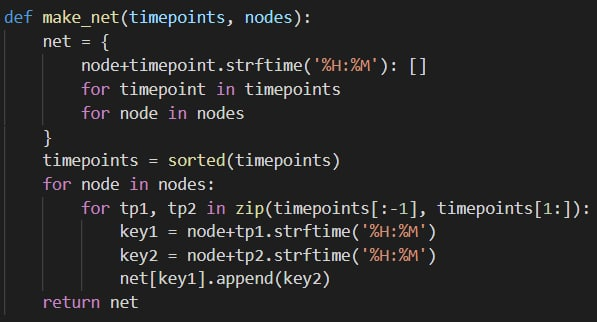
### 2.4.2 Создание графа

#### 2.4.2.1 Создание модели графа

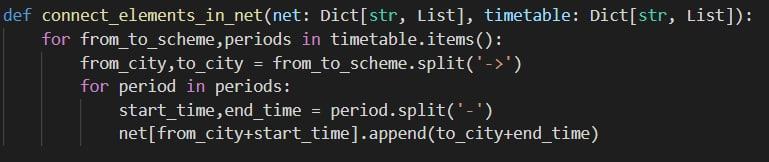
Перед началом решения задач на графы, необходимо построить граф, чтобы можно было представить данную задачу в виде программного кода.

Так будет выглядеть граф для нашей задачи.Кратчайшим путем из пункта A в пункт E, в данном графе будет являться: A 10:00 –> B 12:00–> D 14:00 –> E 14:30. 

#### 2.4.2.2 Реализация графа в коде

Для реализации данного графа в коде, создадим функцию «make\_net», которая связывает определённый город с временным интервалом и на выходе мы получаем граф, полученный выше. 

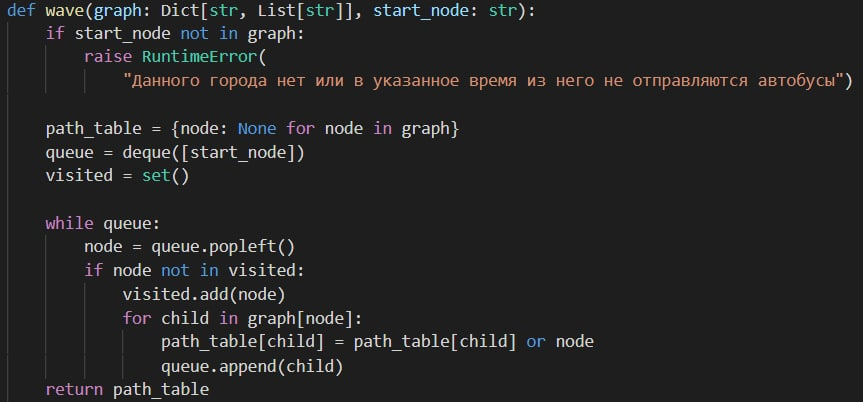
#### 2.4.2.3 Связывание узлов графа

Для обозначения связей между каждым узлом графа, создадим функцию «connect\_elements\_in\_net», которая будет связывать узлы графа по дынным таблицы с расписаниями автобусов. Например из города А в 10:00 мы можем попасть в город B в 10:30, из города B в 11:00 в город А в 11:30 и т.д. 

### 2.4.3 Нахождение кратчайшего пути

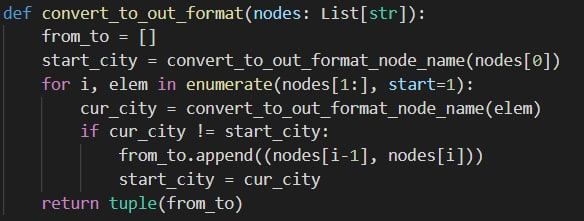
Чтобы просчитать кратчайший путь в данной задаче, воспользуемся уже изученным нами алгоритмом поиска в ширину. Для нахождения кратчайшего пути, нам необходимо получить все возможные пути в нужный нам город и сравнить длины полученных маршрутов. Путь, оказавшийся наименьшим, будет является ответом к нашей задачи.

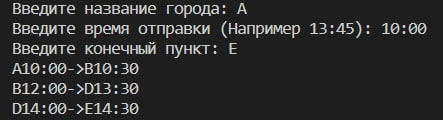
Создадим функцию «wave» которая будет проходить по всем узлам графа и найдет необходимый наименьший маршрут в нужный нам город.



### 2.4.4 Получение кратчайшего маршрута

Осталось обработать полученные данные и вывести результат. Для этого создадим функцию «convert\_to\_out\_format», которая и переработает данные в приемлемый вид.



 Необходимо проверить правильность работы программы, для этого сверимся с данными п.2.3.2.1.

Программа успешно нашла кратчайший путь, поэтому ей можно пользоваться при необходимости нахождения пути между городами, если до нужного города нет прямых рейсов автобусов.

# Заключение

В данной проектной работе собрана часть информации из теории графов, которая широко употребляется в нашей жизни. Применение теории графов можно найти во многих областях: химия – составление сложных реакций, математика – решение задач на нахождение кратчайших путей, информатика – определение кратчайшего маршрута передачи данных через интернет и т.д.

Я рассмотрел два алгоритма из теории графов, помогающих находить кратчайшие пути в графах между вершинами и реализовал их на языке программирования Python.

Продуктом данной работы является программа, просчитывающая кратчайший маршрут между городами, при заданном расписании автобусов между ними, если между городами нет прямого рейса. Для создания данной программы я воспользовался знаниями полученными в ходе данной проектной работы.

# Список использованных источников

1. Электронная энциклопедия «Википедия». Теория графов [Электронный ресурс] режим доступа: [https://ru.wikipedia.org/wiki/Теория\_графов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)
2. Обучающий ресурс «Python Scripts». Алгоритм Дейкстры [Электронный ресурс] режим доступа: <https://python-scripts.com/dijkstras-algorithm>
3. Электронная энциклопедия «Вики Учебник». Алгоритм поиска в ширину [Электронный ресурс] режим доступа: [https://ru.wikibooks.org/wiki/Реализация\_алгоритмов/Поиск\_в\_ширину](https://ru.wikibooks.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%B2_%D1%88%D0%B8%D1%80%D0%B8%D0%BD%D1%83)
4. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. [Текст]/ Бхаргва А. – СПб.: Питер, 2020. -288 с.
5. Изучаем Python. Программирование игр, визуализация данных, веб-приложения 2-е изд. [Текст]/ Мэтиз Э. – СПб.: Питер 2019. – 496 с.

# Приложение

*Приложение 1.*

Созданные программы в ходе проекта: <https://github.com/Mizkus/graph_theory>

*Приложение 2.*

Работа поиска в ширину на примере гиф-анимации: <https://upload.wikimedia.org/wikipedia/commons/5/57/Dijkstra_Animation.gif>

*Приложение 3.*

Исполнение алгоритма Дейкстры на примере гиф-анимации: <https://upload.wikimedia.org/wikipedia/commons/thumb/5/5d/Breadth-First-Search-Algorithm.gif/440px-Breadth-First-Search-Algorithm.gif>