*Муниципальное общеобразовательное бюджетное учреждение*

*«Средняя общеобразовательная школа №3»*

Научно-практическая конференция учащихся

«Гагаринские чтения»

Исследование графов и выбор оптимального алгоритма для нахождения кратчайшего пути.

Выполнили ученики 9б класса

Шенягин Даниил и Субботин Андрей

Руководитель – учитель информатики

Краев Николай Владимирович

Муром – 2018

Содержание…

## **Актуальность исследования:**

В настоящий момент многие системы представляют собой сеть, которую удобно предствить в виде абстракционной модели - графа. Поэтому легче все работать с этой моделью и использовать алгоритмы на ней.

И необходимо знать и уметь перебирать элементы этой модели, а так же находить крачайший путь до любого элемента.

Эта тема все больше становится популярной в современной математике.

## **Цель работы:**

Выяснить какой алгоритм для нахождения крачайшего пути в графе более оптимальный.

Критерии оптимальности:

1. Лучшее время работы алгоритма
2. Универсальность алгоритма
3. Простота работы

## **Задачи:**

* Изучить графы и их виды.
* Рассмотреть алгоритмы.
  + Для нахождения кратчайшего путь в графе.
  + Для нахождения компоненты связанности.
* Разработать приложение
  + Генерация графа
  + Визуализации графа
  + Применение алгоритмов на графе
  + Вывод практического времени работы алгоритмов
* Сравнить и проанализировать
  + время работы алгоритмов
  + универсальность работы алгоритмов
  + простоту работы алгоритмов
* Выбрать наилучший алгоритм

# **Введение**

Когда мы готовились к ОГЭ по информатике, столкнулись с задачами для решения которых, используются графы. И у нас возникли вопросы: что такое графы, где они используются, и где их можно применять. Нам стало интересно более подробно изучить данный вопрос.

Как оказалось, эта тема очень объемная и мы решил изучить более углубленно вопрос о нахождении крачайшего пути из точки А в точку Б, а так же затронуть нахождение компоненты связанности в графе.

# **История изобретения графов**

Графы – абстракная модель, способ предствления взаимодействия объектов в системе. В математике существует целый раздел – теория графов, который изучает графы, их свойства и применение.

Графом в математике называется конечная совокупность точек, именуемых вершинами; некоторые из них соединены друг с другом линиями, называемых ребрами графа.

При взгляде на географическую карту сразу бросается в глаза сеть железных и автомобильных дорог. Это типичный граф: кружочки обозначают станции или города - вершины графа, а соединяющие их пути - ребра.

Если вам, к примеру, нужно найти крачайший путь из точки А в точку Б, может возникнуть проблема: чем больше промежуточных точек, тем больше вариантов маршрутов будет получаться и одному человеку просчитать крачайший путь будет очень сложно и долго.

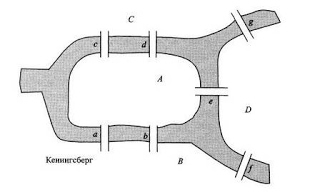
Родоначальником теории графов принято считать математика Леонарда Эйлера (1707-1783).

Рисунок 1

*Задача о Кенигсбергских мостах.* На рис. 1 представлен схематический план центральной части города Кенигсберг (ныне Калининград), включающий два берега реки Перголя, два острова в ней и семь соединяющих мостов. Задача состоит в том, чтобы обойти все четыре части суши, пройдя по каждому мосту один раз, и вернуться в исходную точку. Эта задача была решена (показано, что решение не существует) Эйлером в 1736 году.

# **Алгоритмы для графов**

В этой работе мы решили затронуть алгоритмы нахождения кратчайшего пути и компоненты связанности в графе. При поиске мы нашли 7 алгоритмов связанных с графами:

1. Алгоритм обхода в глубину
2. Алгоритм обхода в ширину
3. Алгоритм Дейкстры
4. Алгоритм Флойда — Уоршелла
5. Алгоритм Джонсона
6. Алгоритм Ли(волновой алгоритм)
7. Алгоритм Беллмана — Форда

Два первых алгоритма находят компоненту связанности в графе.

Компонента связности графа (или просто компонента графа ) — максимальный (по включению) связный подграф графа . Другими словами, это подграф , порождённый множеством вершин, в котором для любой пары вершин в графесуществует -цепь и для любой пары вершин , не существует -цепи.

Мы не стали брать алгоритм Ли или волновой алгоритм, т.к. он предназначен для нах. кр. пути в лабиринте. Алгоритм Джонсона был сложен в реализации, и мы не стали брать его в свой кругозор исследования и еще этому поспособствовало нехватка времени

Итак, мы взяли для исследования 3 самых популярных алгоритмов для нахождения кратчайшего пути в графе:  
  
1. Алгоритм Дейкстры  
2. Алгоритм Флойда - Уоршелла  
3. Алгоритм Форда – Беллмана

Важно учесть, что побочно с ними будут работь алгоритмы обхода графа

1. Алгоритм обхода в глубину
2. Алгоритм обхода в ширину

# История разработки web-приложения

Приложение было написано на платформе React JS с применение таких технологий как HTML5, CSS3, JavaScript([ECMAScript 6](https://www.google.ru/search?newwindow=1&rlz=1C1KPLB_enRU680RU680&q=ECMAScript+6&spell=1&sa=X&ved=0ahUKEwiy3oq68qDZAhWD3iwKHfhLDvUQkeECCCUoAA)). Приложение писалось около полутора месяцев(по показателю github).

В двух словах: виуализация представлена на HTML5 -> tag <svg></svg>, ход дизайнерской идеи был реализован на CSS3 (Less или Sass), контроллером служила сеть компонентов, связанные между собой, работающая на React JS (Twitter).

Приложение визуалть состоит из двух частей: само поля для графа и консоль-панель для контроллинга, которая состоит из:

1. Изменение характеристик графа (цвет, радиус вершин)
2. Поле модов
   1. Удаление вершин
   2. Обозначение вершин
3. Вывод матрицы данного графа
4. Последняя панель
   1. История действий
   2. Генерация графа (последовательно)
   3. Вывод результата работы алгоритма на графе

По написанию приложения стоит сказать, что мы столкнулись со следующими проблемами или трудностями:

1. Реализация алгоритмов на языке JS (Coffee Script)
2. Нам пришлось впитать много теории для разработки приложения
3. …

# Основная часть нашей работы

Исследование заключается в генерации графов и пропускание их через алгоритмы. Далее получение результатов, сравнение их, распределение и соррировка. Затем выявление наилучшего алгоритма, который показывает наилучший результат в 3 характеристиках.

Этап 1. Генерация графа  
Принимается 2 аргумента (кол-во вершин и кол-во ребер), притом, что граф будет не ориентированным и с положительным весом ребер.  
Мы создавали 13 графов, где начальное кол-во вершин – 10, а конечное – 70, с каждым разом кол-во вершин увеличивалось на 5. Кол-во ребер будет максимальным() – мы это задали формулой  
a0=3,

* i=3,
* an+1=an+i
* i++(i с каждым разом увеличивается на единицу)

Этап 2. Пропуск графа через алгоритмы

1. Выбор вершины
2. Выбор алгоритма
3. Запуск алгоритма

Данные действия будут выполняться для каждой вершины(кол-во = N). При этом параллельно этим действиям, будет считаться практическое время алгоритма для одной вершины и среднее время для всех вершин графа. Кол-во тестингов = кол-ву вершин.

Этап 3. - Получение результатов и запись их в таблицу  
Из 2 этапа видно, что для каждого графа и для каждого алгоритма существует свое время работы, отличающееся от других.  
Дальше мы записывали данные в таблицу.

Этап 4. - Расчет теоретического времени  
Для каждого алгоритма существует теоретическое время работы, называемое временной сложностью алгоритма(дописать про big O…).

Этап 5. - Сравнение показателей  
Теоретическое время изменяется по закону квадратной функции и кубической функции(Для нахождения кратчайшего пути в графе).

# Вывод

Из таблицы мы видим, что алгоритм Флойда-Уоршела самый быстрый, но он очень трудный в реализации. Алгоритм Дейкстры очень простой в реализации, но он очень долгий. Поэтому самый оптимальный алгоритм это алгоритм Беллмана-Форда он средний по времени выполнения и прост в реализации.