МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ**

**ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра компьютерных образовательных технологий

Направление: нейротехнологии и программирование

Реализация волнового алгоритма для поиска минимального маршрута в неориентированном невзвешенном связном графе

Выполнил студент

Группы P3122

Шимохина Т.А.

Проверил:

Ефимчик Е.А.

Санкт-Петербург

2018 г.

Оглавление

[**Введение** 3](#_Toc517081521)

[**1 Анализ задачи** 4](#_Toc517081522)

[1.1 Описание задачи 4](#_Toc517081523)

[1.2 Волновой алгоритм 4](#_Toc517081524)

[1.3 Описание волнового алгоритма 4](#_Toc517081525)

[1.4 Расположение фронтов волны 5](#_Toc517081526)

[**2 Проектирование программного инструмента** 6](#_Toc517081527)

[**3 Реализация программного инструмента** 8](#_Toc517081528)

[**4 Тестирование** 12](#_Toc517081529)

[**Заключение** 13](#_Toc517081530)

[**Список использованных источников** 14](#_Toc517081531)

[**Приложения** 15](#_Toc517081532)

[Приложение 1 15](#_Toc517081533)

# **Введение**

Необходимо реализовать волновой алгоритм для поиска минимального маршрута в связном неориентированном невзвешенном графе. В процессе работы будут рассмотрены необходимые определения и термины дискретной математики, разработано приложение, реализующее соответствующий алгоритм и проведено тестирование.

# **1 Анализ задачи**

## 1.1 Описание задачи

Волновой алгоритм для поиска пути в неориентированном связном графе. Количество вершин графа может варьироваться в пределах [6;40]. Требуется реализовать указанный в теме алгоритм. При этом вывод итоговых и промежуточных результатов должен быть реализованы в виде SVG графики. Вершины фронтов волны нужно располагать выровненными по концентрическим окружностям, в середине которых находится вершина исхода.

## 1.2 Волновой алгоритм

Волновой алгоритм - алгоритм поиска пути, основанный на методе поиска в ширину. Волновой алгоритм в контексте поиска пути в лабиринте был формально предложен Э. Ф. Муром. Ли независимо открыл тот же алгоритм в контексте трассировки печатных плат в 1961 году. Алгоритм применяется для поиска кратчайшего пути между вершинами в связном графе[1].

## 1.3 Описание волнового алгоритма

Алгоритм предназначен для поиска кратчайшего пути от стартовой вершины к конечной вершине графа.

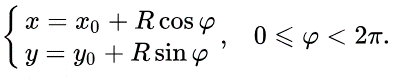
Работа алгоритма включает в себя три этапа: *инициализацию*, *распространение волны* и *восстановление пути*. Во время инициализации строится непосредственно сам граф по матрице инцидентности.

Стартовой вершине присваивается фронт волны ноль. Далее, от стартовой вершины порождается шаг в соседние, связанные с ней вершины, при этом проверяется, не отмечена ли вершина фронтом волны. Распространение фронта волны продолжается до тех пор, пока не будет достигнута конечная вершина.

Восстановление кратчайшего пути происходит в обратном направлении: при выборе вершины от финишной к стартовой на каждом шаге выбирается вершина, фронт волны которой на единицу меньше текущего. Очевидно, что таким образом находится кратчайший путь между парой заданных вершин. Путей может быть несколько. В данном случае выбирается наименьшая по номеру вершина [2].

## 1.4 Расположение фронтов волны

Окружности с общим центром, но разными радиусами, называются **концентрическими**. В центре находится стартовая вершина. Координаты зададим с помощью параметрического уравнения окружности [3]:



, где x0 = 0, y0 = 0.

R – это радиус, определим его по формуле , где count – количество вершин в данном фронте, – номер фронта волны.

φ – угол (в радианах), определяет расположение вершин на окружности. Каждый угол φ получим поворотом Δφ, начиная с нуля: Δφ =. Вершины, вошедшие в найденный путь, будут находится в крайнем правом положении и соединены ребрами.

# **2 Проектирование программного инструмента**

Приложение разрабатывалось на языке программирования C++ с использованием фреймворка Qt Creator. Был использован открытый исходный код из примеров к Qt Creator: Elastic Nodes Example [4].

Основные модули:

* Main – запуск приложения, создание главного окна.
* Mainwindow – основной модуль для обработки данных и работы алгоритма.
* Graphwidget – модуль для графического отображения графа и результатов алгоритма.
* Vertex – модуль для хранения информации о вершине и ее отрисовки на графической сцене.
* Node – модуль для графического отображения вершин графа (наследуется от Vertex, учитывает связи с другими вершинами, доступна для взаимодействия с пользователем).
* Edge – модуль для создания ребер, соединяющих вершины.

Приложение содержит две графические сцены (graphwidget): одну для отображения графа, другую для представления распространения фронтов волны в заданном виде.

Для создания графа необходимо либо загрузить данных из файла, содержащего матрицу смежности (нажатием на кнопку “Загрузить”), либо ввести количество вершин и получить случайно сгенерированный программой граф (нажатием на кнопку “Сгенерировать”). Затем двойным щелчком левой кнопки мыши на отображенной графе выбираются вершины начала и конца маршрута. Вершины помечаются красным и синим цветом соответственно. После возможен либо вывод итогового результата (кнопка “Волновой алгоритм”), либо вывод промежуточных результатов на каждом шаге (кнопка “Алгоритм по шагам”). При этом все вершины, вошедшие в найденный путь, будут помечены желтым цветом.

# **3 Реализация программного инструмента**

* Класс Main:
  + Методы:
    - Main – создание главного окна приложения (объекта класса Mainwindow).
* Класс Mainwindow:
  + Переменные:
    - vertexes – динамический массив объектов класса Node для обработки;
    - start – переменная для хранения номера выбранной вершины исхода;
    - finish – переменная для хранения номера выбранной вершины конца;
    - step – переменная для хранения длины (количества шагов) пути;
    - currentStep – переменная для хранения промежуточной длины для восстановления пути.
* Методы:
  + load\_clicked – осуществляет загрузку файла \*.txt, считывание данных из файла построчно, разбиение строки по знаку пробела, создание объектов типа node и заполнение vertexes (предварительно обнуляет массив);
  + gen\_clicked – считывает введенное пользователем число, формирует матрицу смежности и на ее основе заполняет vertexes (предварительно обнуляет массив);
  + createGraph – после заполнения vertexes одним из вышеперечисленных методов передает данные для отрисовки вершин на графической сцене;
  + check – установка значения фронта волны по умолчанию (-1), обнуление данных в графической сцене для отображения фронтов волны, поиск вершин начала и конца пути (по цвету вершины);
  + alg – запускает метод check(), проверяет, что вершины начала и конца выбраны, затем начальной вершине устанавливает значение фронта волны front равным 1. Начинается распространение фронта волны до тех пор, пока не будет отмечена финишная вершина (ее фронт волны станет равным больше 0). Тогда работа алгоритма прекращается;
  + findPath – осуществляет восстановление пути, начиная с финишной вершины до начальной, включает в путь вершину с меньшим, первым встречным номером, для которого фронт волны на 1 меньше предыдущего значения;
  + drawVertex – определим временный массив temp, куда для каждого значения фронта будем сохранять номера вершин, в него вошедших. При этом номера вершин, вошедших в найденный путь, добавляем в начало temp, так как их необходимо отрисовать первыми. Затем передаем эти номера в графическую сцену для отображения;
  + drawPath – соединяет вершины, вошедшие в путь (от конца к началу);
  + alg\_clicked – последовательно выполняет методы alg(), findPath(), drawVertex, drawPath(). Запускается при нажатии кнопки “Волновой алгоритм”;
  + byStep\_clicked – запускается при нажатии кнопки “Алгоритм по шагам”. Выполняет то же, что и предыдущий метод, при этом при каждой итерации, пока не отрисованы все фронты волны, метод “обрывается” и не проходит дальше, что создает эффект пошагового отображения работы алгоритма.
* Класс GraphWidget:
  + Переменные:
    - nodes – массив объектов класса Vertex, содержит указатели на все вершины, добавленные на сцену;
    - center – логическая переменная, необходима для проверки, добавлена ли вершина исхода;
    - R – переменная для текущего радиуса окружностей;
    - currentVertex – указатель на текущую вершину, необходим для отображения найденного пути.
  + Методы:
    - draw – получает из mainwindow номер вершины, ее фронт и количество вершин во фронте, создается объект класса Vertex, который и добавляется на сцену. Также при первом появлении фронта пересчитывается радиус R и на сцену добавляется окружность, на которой и располагаются вершины;
    - setRadius – обнуляет радиус R. Вызывается при создании нового графа;
    - setCheck – сбрасывает переменную check при создании нового графа;
    - drawPath – получает на вход номер вершины, соединяет ее с текущей и ребром и переопределяет текущую вершину;
    - setCurrentVertex – получает номер финальной вершины, ищет ее среди вершин nodes сцены и объявляет текущей.
* Класс Vertex:
  + Переменные:
    - text – переменная строкового типа, содержит номер этой вершины;
    - color – переменная для хранения цвета ячейки.
  + Методы:
    - addText – устанавливает номер вершины;
    - setColor – устанавливает цвет вершины, перерисовывает объект.
* Класс Node:
  + Переменные:
    - front – переменная типа int для хранения фронта волны вершины.
  + Методы:
    - getColor – возвращает текущий цвет вершины;
    - setFront – сохраняет фронт волны вершины;
    - getFront – возвращает текущий фронт вершины;
    - mouseDoubleClickEvent – обработчик двойного нажатия на вершину для выбора вершины конца и начала, при этом снимается выделение цветом остальных вершин.
* Класс Edge;

# **4 Тестирование**

Было проведено 10 тестов с различными входными данными, включая возможные критические случаи. Все входные данные и результаты работы программы приведены в Приложении 1.

# **Заключение**

В процессе работы над курсовой работой была рассмотрена необходимая теория графов и алгоритмов, создан программный инструмент, реализующий волновой алгоритм и представляющий конечные и промежуточные результаты в виде графики SVG в соответствии с вариантом и проведено тестирование.

# **Список использованных источников**

1. Алгоритм Ли (Волновой алгоритм) / <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9B%D0%B8> (09.06.2018)
2. Курс «Методы и алгоритмы теории графов» / Лисицына Л.С. / <https://courses.openedu.ru/courses/course-v1:ITMOUniversity+AGRAPH+fall_2017/info> (09.06.2018)
3. Пискунов Н. С. Дифференциальное и интегральное исчисления для втузов, т.1: Учебное пособие для втузов. — 13-е изд.— М.: Наука. Главная редакция физико-математической литературы, 1985. — 432 с.
4. Elastic Nodes Example / <http://doc.qt.io/Qt-5/qtwidgets-graphicsview-elasticnodes-example.html> (09.06.2018)

# **Приложения**

## Приложение 1

Тест 1. Входные данные:

0 1 0 1 0 1

1 0 1 0 1 0

0 1 0 1 0 1

1 0 1 0 1 0

0 1 0 1 0 1

1 0 1 0 1 0

Дан граф на 6 вершин (рис. 1), результат работы алгоритма представлен на рис.2.

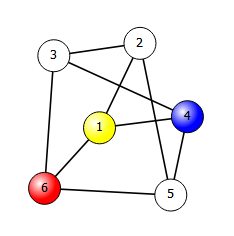
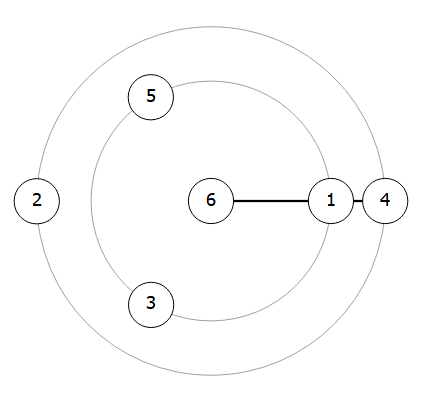


Рисунок 2 – Результат для 6 вершин

Рисунок 1 – Граф на 6 вершин

Тест 2. Входные данные:

0 1 0 0 0 0 0

1 0 1 0 0 0 0

0 1 0 1 0 0 0

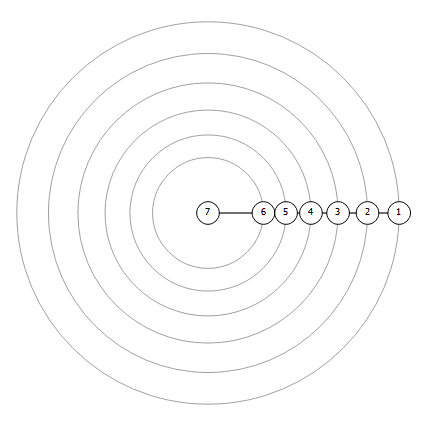
0 0 1 0 1 0 0

0 0 0 1 0 1 0

0 0 0 0 1 0 1

0 0 0 0 0 1 0

Дан граф на 7 вершин (рис. 3), результат работы алгоритма представлен на рис.4.



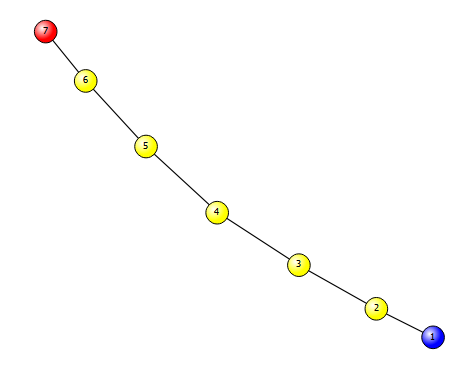


Рисунок 3 – Граф на 7 вершин

Рисунок 4 - Результат для 7 вершин

Тест 3. Входные данные:

0 1 0 0 0 0 0 1 1

1 0 1 0 0 0 0 0 0

0 1 0 1 0 0 0 1 1

0 0 1 0 1 0 0 0 1

0 0 0 1 0 1 0 1 0

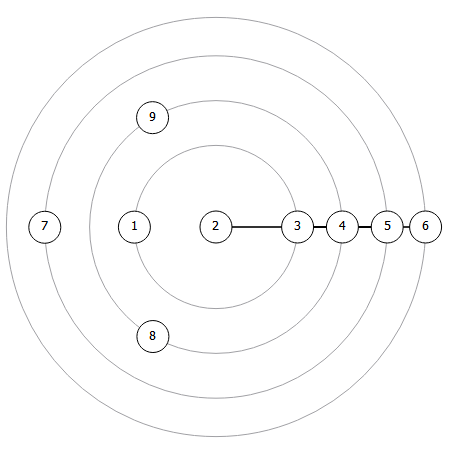
0 0 0 0 1 0 1 0 0

0 0 0 0 0 1 0 1 1

1 0 1 0 1 0 1 0 1

1 0 1 1 0 0 1 1 0

Дан граф на 9 вершин (рис. 5), результат работы алгоритма представлен на рис.6.



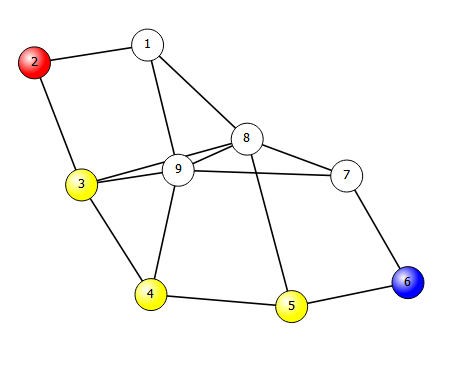


Рисунок 5 - Граф на 9 вершин

Рисунок 6 - Результат для 9 вершин

Тест 4. Входные данные:

0 1 0 0 0 0 0 1 1 0 0 1 1 0 1

1 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 0 0 1 1 1 1 1 1 1 0

0 0 1 0 1 0 0 0 1 0 1 0 1 0 1

0 0 0 1 0 1 0 1 0 1 1 1 0 0 0

0 0 0 0 1 0 1 0 0 0 1 1 0 1 1

0 0 0 0 0 1 0 1 1 1 1 0 0 1 1

1 0 1 0 1 0 1 0 1 0 0 0 1 1 1

1 0 1 1 0 0 1 1 0 1 1 1 0 0 0

0 0 1 0 1 0 1 0 1 0 1 0 0 0 0

0 0 1 1 1 1 1 0 1 1 0 0 0 0 1

1 0 1 0 1 1 0 0 1 0 0 0 1 1 1

1 0 1 1 0 0 0 1 0 0 0 1 0 0 0

0 0 1 0 0 1 1 1 0 0 0 1 0 0 0

1 0 0 1 0 1 1 1 0 0 1 1 0 0 0

Дан граф на 15 вершин (рис. 7), результат работы алгоритма представлен на рис.8.

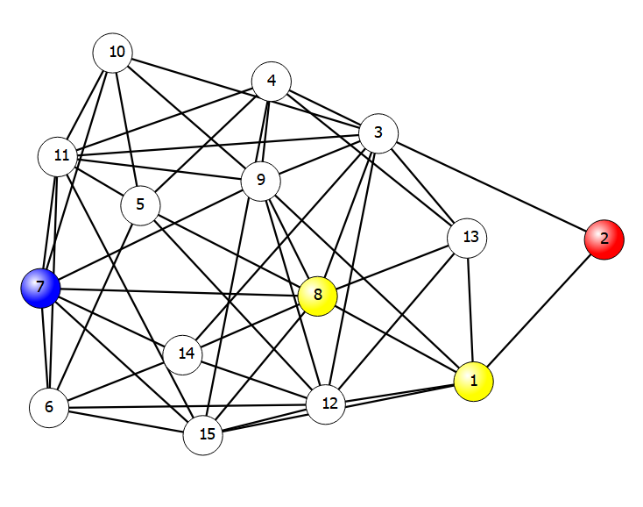
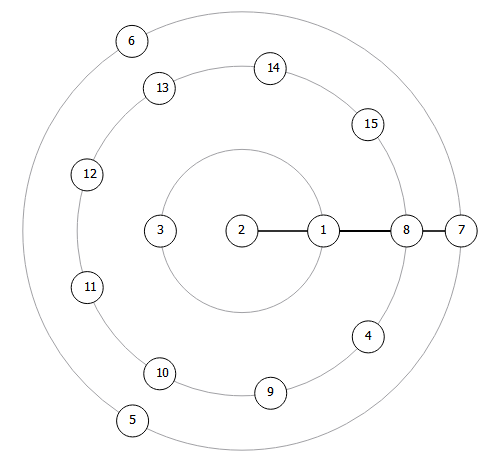


Рисунок 8 - Результат для 15 вершин

Рисунок 7 - Граф на 15 вершин

Тест 5. Входные данные:

0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 0

0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 1 0 1 1 0

1 0 0 1 1 0 0 1 0 0 1 0 0 0 0 1 1 0 1 0

0 1 1 0 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0

0 1 1 1 0 0 0 1 1 0 1 0 1 1 0 0 0 1 1 0

0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1 0

0 0 0 1 0 1 0 0 0 1 0 0 1 1 0 1 0 1 1 0

0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 1 1 1 1

1 1 0 0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1

0 0 0 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1 1

0 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1

1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0

1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 0

1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1

0 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0

1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0

0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1

1 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 1

1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0

0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 0

Дан граф на 20 вершин (рис. 9), результат работы алгоритма представлен на рис.10.

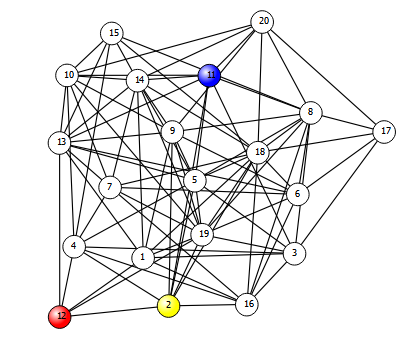
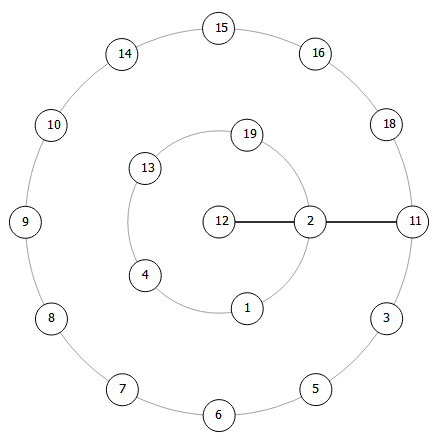
 

Рисунок 9 - Граф на 20 вершин

Рисунок 10 - Результат для 20 вершин

Тест 6. Входные данные:

0 1 1 1 0 1 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1

1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 1 1

1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0

1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 1 0 1 1 1

0 1 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 1

1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1

1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 1 0 1 1 0 1 1

0 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0

1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0 0

0 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1 1 1

1 0 1 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0

1 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 1 1 0 1

0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1

0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1

1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 0 1 1 0 0 1

1 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0

1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0

0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0

0 0 1 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 0 1 1 1 1 0 0

0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0

0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 1 1 0 1 1 1 1

0 1 0 0 0 0 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 0 0 0 1

0 1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 1 1 1 0 0 1 1

1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0

1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0

Дан граф на 25 вершин (рис. 11), результат работы алгоритма представлен на рис.12.

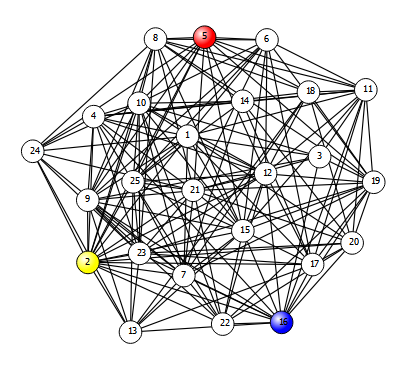
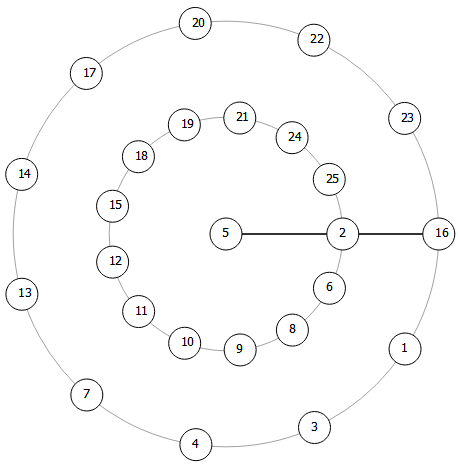


Рисунок 11 - Граф на 25 вершин

Рисунок 12 - Результат для 25 вершин

Тест 7. Входные данные:

0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 1 1 1

1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0

0 1 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 1 0

0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 0

1 0 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 1 1

0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1

0 1 1 0 0 0 0 1 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 0 1 1 1 0

0 1 0 0 0 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 1 1 0 0

1 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 0 0 0

1 1 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 1

1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 1 0 0

1 0 1 0 0 0 1 0 1 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1

0 0 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 0 1

1 1 1 1 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0 0 1 0 0 1

0 1 1 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0

1 1 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0

1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0

0 1 1 0 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0

0 1 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0

1 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0

0 0 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 1 1

0 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0

0 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1

1 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0 1 0

0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0

1 0 1 0 0 0 0 1 0 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1

0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0

1 0 1 0 1 0 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 0 0 1 1 0 0 0

1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 0 1 0 0 0 1

1 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0

Дан граф на 30 вершин (рис. 13), результат работы алгоритма представлен на рис.14.

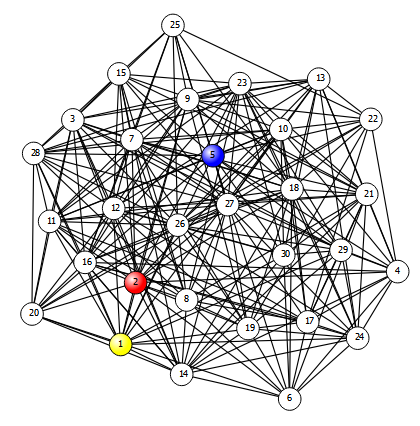
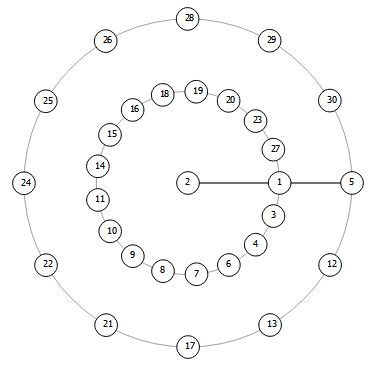


Рисунок 14 - Результат для 30 вершин

Рисунок 13 - Граф на 30 вершин

Тест 8. Входные данные:

0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0

0 0 0 1 1 1 0 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0

0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 1

0 1 0 0 0 0 1 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0

0 1 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0

0 1 1 0 1 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 1 0 1 1 0 1 0 0

1 0 1 1 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1

1 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 1 0 0 0

0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 1 1 0 1 0 0 0 0 0 1 1 1 0 1 1 0

1 0 0 1 1 0 0 0 1 0 0 1 1 0 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0

1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 1 1

1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 0 1 0 1

1 1 0 1 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 1 1

0 0 1 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0

0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0

1 0 0 0 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1

1 1 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 0

1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 1

0 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0

0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 0 0 0

0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0

1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1

1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 1 1

1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 0 1 0 0

0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0

0 1 1 0 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0 0

0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1

0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1

1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0 1 1 1

1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 0 1 1 0 1 1

0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1

1 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0

0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0 1 0 0 1 0

1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 1 0 0 1 0 0

0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 0 0 0

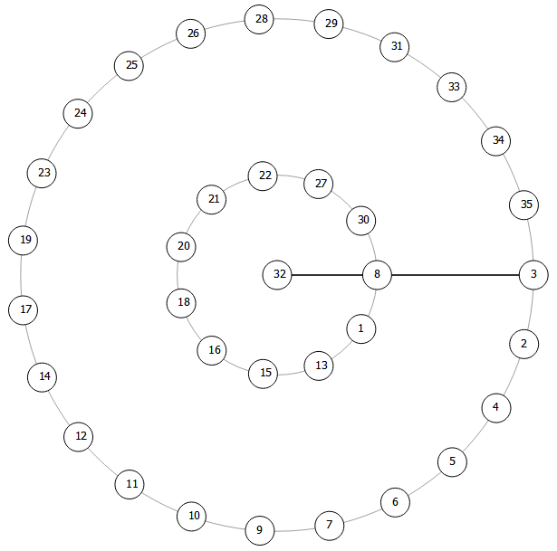
Дан граф на 35 вершин (рис. 15), результат работы алгоритма представлен на рис.16.

Рисунок 14 - Результат для 35 вершин

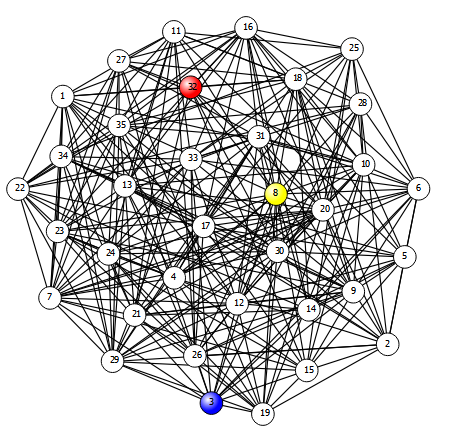


Рисунок 13 - Граф на 35 вершин

Тест 9. Входные данные:

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

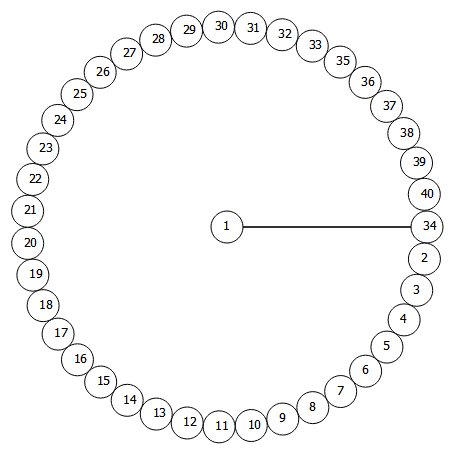
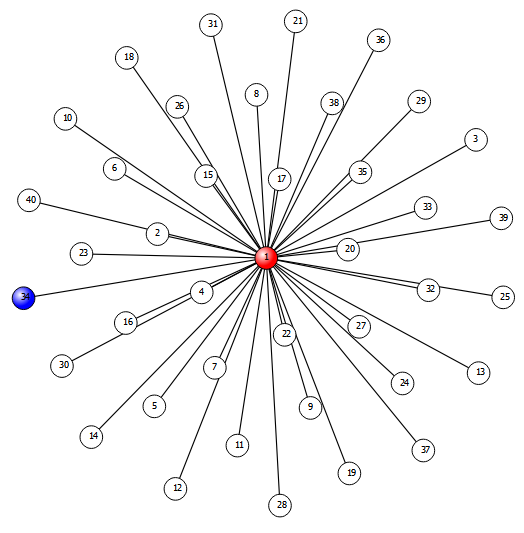
Дан граф на 40 вершин (рис. 17), результат работы алгоритма представлен на рис.18.

Рисунок 18 - Результат для 40 вершин

Рисунок 17 - Граф на 40 вершин

Тест 10. Входные данные:

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

Дан граф на 40 вершин (рис. 19), результат работы алгоритма представлен на рис.20.

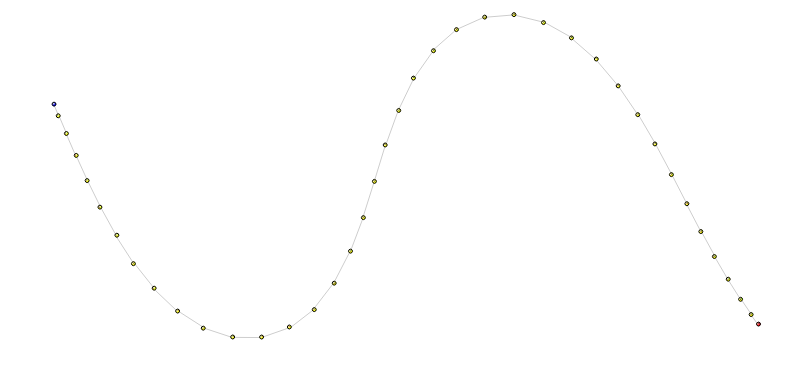


Рисунок 19 - Граф на 40 вершин. Критический случай

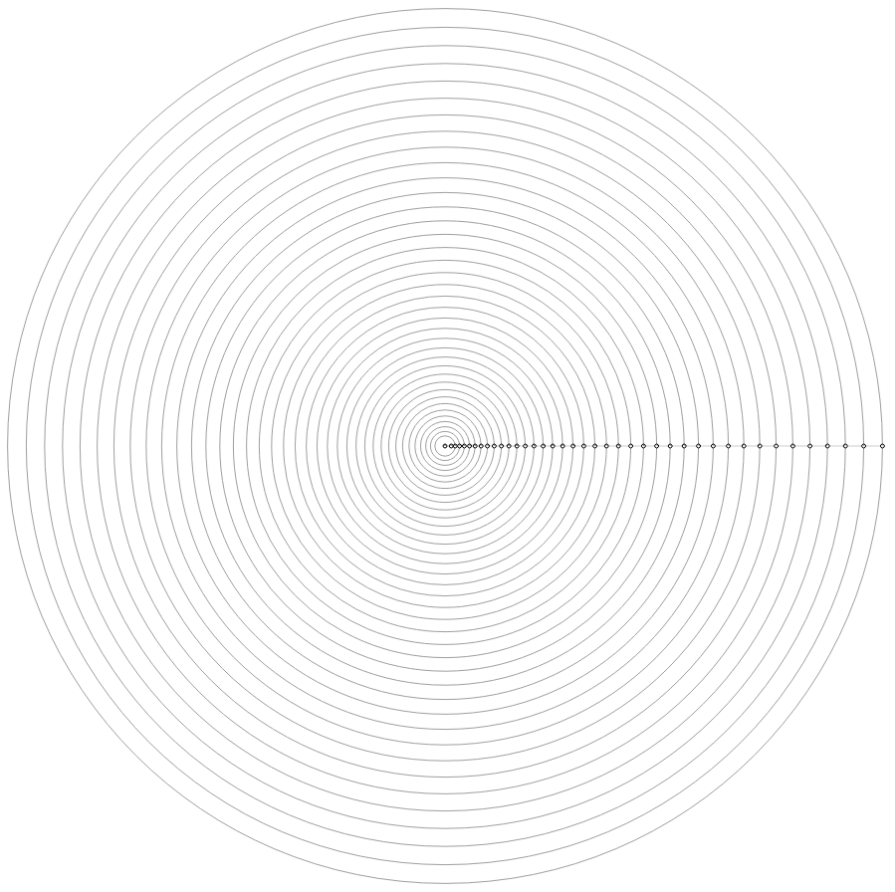


Рисунок 20 - Результат для 6 вершин. Критический случай