*Министерство образования и науки Российской федерации*

федеральное государственное бюджетное образование учреждение

высшего образования

«Курганский государственный университет»

РФ КГУ 09.03.03 КП20 910038 03

Кафедра «Программное обеспечение автоматизированных систем»

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Технология разработки Web-приложений»**

**Пояснительная записка**

Листов 30

Выполнил студент группы ИТ-20819\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Степанов Д.А. /

Проверил канд. техн. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Семахин А.М. /

Курган, 2020

# АННОТАЦИЯ

Документ содержит общие сведения о программе, ее описании, структуре и вариантах использования, а также входных и выходных данных.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc59963000)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР 4](#_Toc59963001)

[2 ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ 5](#_Toc59963002)

[3 ОПИСАНИЕ АЛГОРИТМА 6](#_Toc59963003)

[3.1 Постановка задачи об остовном дереве минимальной стоимости 7](#_Toc59963004)

[3.2. Алгоритм решения задачи 7](#_Toc59963005)

[3.3 Оценка сложности алгоритма. 8](#_Toc59963006)

[3.4 Базовое правило использования О большого 13](#_Toc59963007)

[4 ЭТАПЫ ПРОВЕДЕНИЯ ЭКСПЕРИМЕНТА 16](#_Toc59963008)

[4.1 Получение выборки статистических данных в ходе проведения эксперимента над программой. 16](#_Toc59963009)

[4.2 Определение уравнения связи в общем виде 18](#_Toc59963010)

[5 ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММНОГО КОМПЛЕКСА 25](#_Toc59963011)

[6 ОПИСАНИЕ ФУНКЦИЙ 27](#_Toc59963012)

[7 ВЫВОДЫ ПО РЕЗУЛЬТАТАМ ПРОЕКТИРОВАНИЯ 29](#_Toc59963013)

[8 СПИСОК ЛИТЕРАТУРНЫХ ПЕРВОИСТОЧНИКОВ 30](#_Toc59963014)

# 1 АНАЛИТИЧЕСКИЙ ОБЗОР

Данное приложение формализует алгоритм Прима – метод построения минимального остовного дерева взвешенного связного неориентированного графа. Визуальное приложение разработано с использованием объектно-ориентированного метода программирования. В программе обеспечены: ввод исходных данных (количество вершин, вес рёбер), поиск минимального остовного дерева, вывод данных в форме графа и таблицы.

Приложение написано на языке С++ в программной среде Microsoft Visual Studio 2019.

# 2 ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ

Исходными данными в программе являются:

1. Ввод количества вершин графа.
2. Заполнение матрицы смежности (длины рёбер графа) вручную или автоматически (генерация псевдослучайных чисел). Вводимые числа – целые.

# 3 ОПИСАНИЕ АЛГОРИТМА

На вход алгоритма Прима подаётся связный неориентированный граф, где для каждого ребра задаётся его стоимость. Из всех вершин произвольно выбирается одна и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. После этого, рассматриваются рёбра графа такие, чтобы один конец принадлежал одной из проверенных вершин, а другой – нет. Из этих рёбер выбирается ребро наименьшей стоимости. Данный алгоритм выполняется до тех пор, пока все вершины не будут проверены.

Минимальное остовное дерево (MST – minimal spanning tree) взвешенного графа G = (V, E) – остовное дерево, вес которого (сумма весов его рёбер) не превосходит вес любого остовного дерева.

Задача поиска минимального остовного дерева произвольного взвешенного графа получила множество важных применений. Остовное дерево связывает все вершины графа и из каждой вершины графа можно попасть в любую другую. В полном графе с n вершинами имеется остовных деревьев.

Остовным деревом ориентированного графа называется дерево, в котором одна из вершин графа связана со всеми остальными вершинами графа.

# 3.1 Постановка задачи об остовном дереве минимальной стоимости

1. Нахождение остовного дерева;

2. Нахождение минимального и максимального остовного дерева.

Пусть G=(V, E) – связный взвешенный неориентированный граф с n вершинами. Для графа G=(V, E) задана матрица смежности. Стоимость (вес) остовного дерева определяется как сумма стоимостей (весов) ребер.

Необходимо найти для графа G остовное дерево наименьшей стоимости (минимального веса).

# 3.2. Алгоритм решения задачи

Алгоритм Прима разработан в 1961 г. и включает этапы:

**Этап 1.** Выбирается вершина V. Остальные (n-1) вершин – не выбраны.

**Этап 2.** Определяются веса между выбранной вершиной V и остальными невыбранными вершинами.

**Этап 3.** Определяем вершину с наименьшим весом до нее. Фиксируем выбранные ребро и вес.

**Этап 4.** Выбранную вершину исключаем из перечня невыбранных. Число выбранных вершин уменьшаем на 1.

**Этап 5.** Этапы 1 – 4 повторяем до тех пор, пока не будут выбраны все вершины

Таблица 1 - Определение остовного дерева минимального веса методом Прима

|  |  |  |
| --- | --- | --- |
| Выбранная вершина | Невыбранная вершина | Выбранные дуга и вес |
| 1 | 2 3 4 5 6 7 8 9  7 10 | 1->2: 7 |
| 2 | 9 3 4 5 6 7 8  - 10 9 27 - - - | 2->4: 9 |
| 4 | 9 3 5 6 7 8  - 10/8 27 11 - - | 4->3: 8 |
| 3 | 9 8 5 6 7  31 - 27 11 - | 4->6: 11 |
| 6 | 9 8 5 7  31 17 27 15 | 6->7: 15 |
| 7 | 9 8 5  31 17/27 27/15 | 7->5: 15 |
| 5 | 9 8  31 17 | 6->8: 31 |
| 8 | 9  31/32 | 3->9: 31 |
| 9 | Выбраны все |  |

# 3.3 Оценка сложности алгоритма.

Под **вычислительной сложностью** (или **трудоемкостью**) алгоритма понимают количество операций, необходимых для его выполнения. Она характеризует время работы алгоритма и является некоторой функцией *Т(n),* где *n*-размер задачи.

Под **емкостной сложностью** алгоритма понимают объем памяти компьютера, требуемый для реализации алгоритма.

Для оценивания трудоемкости алгоритмов была введена специальная система обозначений – так называемая ***О*-нотация**. Эта нотация позволяет учитывать в функции *f(n)* лишь наиболее значимые элементы, отбрасывая второстепенные.

*О*-нотация при оценке функции вычислительной сложности алгоритма используется по двум основным причинам:

- чтобы не учитывать вклад малых слагаемых в математических формулах;

- чтобы классифицировать алгоритмы согласно верхней границе их общего времени выполнения.

Важность *О*-оценивания состоит в том, что оно позволяет описывать характер поведения функции *f(n)* с ростом n: насколько быстро или медленно растет эта функция.  
 *О*-оценка позволяет разбить все основные функции на ряд групп в зависимости от скорости их роста:

Постоянные функции типа *О*(1), которые с ростом *n* НЕ растут

- функции с логарифмической скоростью роста *О(log n);*

- функции с линейной скоростью роста *О(n);*

- функции с линейно–логарифмической скоростью роста *О(n\*log n);*

- функции с квадратичной скоростью роста *О(n2 );*

- функции со степенной скоростью роста при *а>2;*

- функции с показательной или экспоненциальной скоростью роста *О(2n);*

- функции с факториальной степенью роста *О(n!).*

**Виды функции сложности алгоритмов**

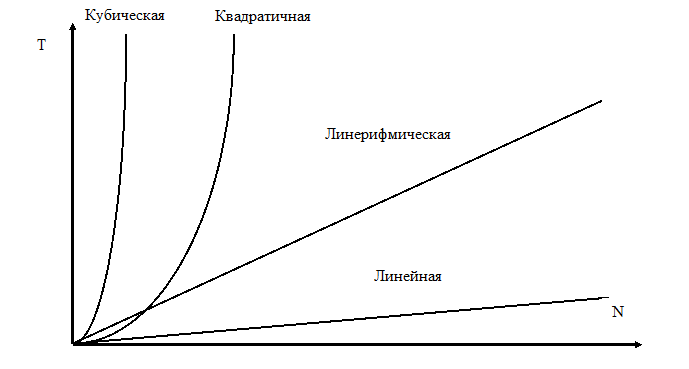


Рисунок 1 – Графики функций сложности алгоритмов

Время выполнения алгоритма T зависит от объема входных данных N

,

где - время выполнения алгоритма, мс.

N – объем входных данных. Для оценивания трудоемкости алгоритмов введена специальная система обозначений – О-нотация.

О-нотация позволяет учитывать в функции f (n) значимые элементы, отбрасывая второстепенные:

1 Кубическая функция – функция f(n), старший член которой содержит . О-нотация имеет вид .

2 Квадратическая функция – функция f(n), старший член которой содержит **.** О-нотация имеет вид .

3 Линейная функция – функция f(n), старший член которой содержит **.** О-нотация имеет вид .

4 Функция линерифмическая - функция, старший член которой равен логарифмов . О-нотация имеет вид .

Например, в функции при больших компонента превосходит остальные слагаемые. Поведение функции определяется компонентой . Остальные компоненты отбрасываются. Функция имеет оценку поведения (скорость роста значений) .

Описание О-нотаций приведено в таблице 2.

Таблица 2–Описание О-нотаций

|  |  |
| --- | --- |
| О-нотация | Описание |
| О (1) | Инструкции программы запускаются независимо от n. Время выполнения программы постоянно. (помещение в стек). Операции в программе выполняются один или несколько раз. Алгоритм независимо от размера данных требует одно и тоже время. |
| О (n) | Время выполнения программы линейно и зависит от n. Входной элемент обрабатывается линейное число раз. |
| О (n2) | Время выполнения программы является квадратичным. Алгоритмы используются для небольших n (цикл двойного уровня вложенности, сортировки выбором, вставками) |
| О (n3) | Алгоритм программы имеет кубическое время выполнения (цикл тройного уровня вложенности). Применяется для небольших задач. (умножение матриц) |
| О (logn) | Логарифмическая зависимость. С ростом n программа работает медленнее. Время характерно для программ, которые сводят большую задачу к набору меньших подзадач, уменьшая на каждом шаге размер подзадачи на постоянный коэффициент. Общее решение находится в одной из подзадач (бинарный поиск) |
| О (n\*logn) | Линерифмическая зависимость. Время выполнения программы пропорционально n\*logn. Возникает, когда алгоритм решает задачу, разбивая ее на меньшие подзадачи, решает независимо и затем объединяет решения подзадач (комбинация, сортировки быстрая, слиянием) |
| О (2n) | Экспоненциальная зависимость. прямое решение задач. (перебор и сравнение различных решений) Для комбинаторных задач нереализуемы. Сведение к прибли-женному алгоритму с приближенным значением. |

О-оценивание позволяет описывать характер поведения функции f(n) с ростом n: насколько быстро или медленно растет эта функция.

О-оценка разбивает функции сложности на группы в зависимости от скорости роста:

1. Постоянные функции , которые с ростом n не растут.
2. Функции с логарифмической скоростью роста .
3. Функции с линейной скоростью роста .
4. Функции с линейно-логарифмической скоростью роста .
5. Функции с квадратичной скоростью роста .
6. Функции со степенной скоростью роста при а>2.
7. Функции с показательной или экспоненциальной скоростью роста .
8. Функции с факториальной степенью роста .

# 3.4 Базовое правило использования О большого

Базовое правило формулируется следующим образом: время выполнения цикла не превышает времени выполнения операторов, входящих в цикл (включая операторы условия), умноженное на число итераций цикла.

Время выполнения операторов внутри группы вложенных циклов равно времени выполнения операторов, умноженному на число итераций во всех циклах. Время выполнения последовательности циклов, следующих друг за другом, равно времени выполнения доминантного цикла.

Оценим сложность алгоритма (листинг 1):

Листинг 1 – алгоритм Прима.

*// Предикат для функции min\_element.*

*struct mypred {*

*bool operator()(const pair<int, int> f\_p, const pair<int, int> s\_p) { return f\_p.second < s\_p.second;*

*}*

*};*

*// Алгоритм Прима.*

*int algorithm\_prima(vector<vector<int>> vec\_name, int number\_of\_vertex, int beg\_vertex) {*

*int new\_vertex = 0;*

*vector<int> checked\_vertex, unchecked\_vertex;*

*vector<pair<int, int>> ways\_arr;*

*pair<int, int> min\_vertex;*

*// Выбирается вершина.*

*checked\_vertex.push\_back(beg\_vertex - 1);*

*// Остальные number\_of\_vertex - 1 вершин - невыбраны.*

*for (int i = 0; i < number\_of\_vertex; i++)*

*if (i != beg\_vertex - 1) unchecked\_vertex.push\_back(i);*

*// Алгоритм Прима.*

*for (int i = beg\_vertex - 1; i < number\_of\_vertex; ) {*

*// Определение веса между выбранной вершиной и остальными невыбранными.*

*for (int g = 0; g < number\_of\_vertex; g++) {*

*if (checked\_vertex.size() > 1) {*

*auto f = find(checked\_vertex.begin(), checked\_vertex.end(), g);*

*if (f != checked\_vertex.end()) continue;*

*}*

*if (vec\_name[i][g] != 0 && vec\_name[i][g] != -1)*

*ways\_arr.emplace\_back(g, vec\_name[i][g]);*

*}*

*// Определяем вершину с наименьшим весом до неё. Фиксируем выбранное ребро и вес.*

*min\_vertex = \*min\_element(ways\_arr.begin(), ways\_arr.end(), mypred());*

*// Выбранную вершину исключаем из перечня невыбранных.*

*if (unchecked\_vertex.size() != 0) {*

*auto it = find(unchecked\_vertex.begin(), unchecked\_vertex.end(), min\_vertex.first);*

*unchecked\_vertex.erase(it);*

*ways\_arr.erase(min\_element(ways\_arr.begin(), ways\_arr.end(), mypred()));*

*for (auto del\_it = ways\_arr.begin(); del\_it != ways\_arr.end(); ) {*

*if ((\*del\_it).first == min\_vertex.first) del\_it = ways\_arr.erase(del\_it);*

*else del\_it++;*

*}*

*}*

*checked\_vertex.push\_back(min\_vertex.first);*

*i = min\_vertex.first;*

*new\_vertex += min\_vertex.second;*

*// Если проверены все вершины, то функция завершает свою работу.*

*if (checked\_vertex.size() == number\_of\_vertex) break;*

*}*

*return new\_vertex;*

*}*

Время работы алгоритма напрямую зависит от способа хранения графа и способа хранения вершин, не входящих в дерево. В данном случае, и граф, и вершины, хранятся в векторе из Стандартной Библиотеки Шаблонов (STL). Также в алгоритме присутствует 2 вложенных цикла for. Поэтому, время работы алгоритма Прима будет равняться Можно сделать вывод, что данный алгоритм нуждается в доработке.

# 4 ЭТАПЫ ПРОВЕДЕНИЯ ЭКСПЕРИМЕНТА

## 4.1 Получение выборки статистических данных в ходе проведения эксперимента над программой.

Для алгоритма Прима при оценивании коэффициентов уравнения связи, устанавливающего математическую зависимость, используется классическая регрессионная модель множественной корреляции.

Таблица 3 – Результаты эксперимента (мс)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер наблюдения i | Количество вершин, | Количество рёбер, | Время выполнения программы, мс, |  |  |  |  |  |
| 1 | 15 | 60 | 16 | 225 | 3600 | 900 | 240 | 960 |
| 2 | 20 | 120 | 15 | 400 | 14400 | 2400 | 300 | 1800 |
| 3 | 30 | 235 | 15 | 900 | 55225 | 7050 | 450 | 3525 |
| 4 | 35 | 350 | 32 | 1225 | 122500 | 12250 | 1120 | 11200 |
| 5 | 50 | 675 | 47 | 2500 | 455625 | 33750 | 2350 | 31725 |
| Сумма | 150 | 1440 | 125 | 5250 | 651350 | 56350 | 4460 | 49210 |

В ходе проведения экспериментальной части курсовой работы получены исходные статистические данные (таблица 3).

Для расчёта коэффициентов найдём средние квадратические отклонения:

(3.1)

(3.2)

(3.3)

Где – среднее значение , – среднее значение , – среднее значение , – среднее значение , – среднее значение , – среднее значение .

При приближении получаем, что

Тогда уравнение связи имеет вид: .

Парный коэффициент корреляции между и рассчитывается по формуле

(3.12)

Парный коэффициент корреляции между и рассчитывается по формуле

(3.13)

Парный коэффициент корреляции между и рассчитывается по формуле

(3.14)

Найдем приблизительные коэффициенты

Коэффициент корреляции и детерминации считаем по следующим формулам:

,

- совокупный коэффициент детерминации.

Поскольку коэффициент корреляции находится в промежутке (0.8;1], то следует полагать, что связь сильная.

## 4.2 Определение уравнения связи в общем виде

Классическая регрессионная модель множественной корреляции (КРММК) устанавливает связь между эндогенной переменной и двумя и более экзогенными переменными. Уравнение связи имеет вид

(3.4)

где - эндогенная переменная;

- экзогенная переменная, ;

- свободный коэффициент;

- выборочные коэффициенты;

- количество экзогенных переменных;

- объем выборки, .

КРММК используется в задачах определения минимального остовного дерева графа, критического пути. Эндогенная переменная – время выполнения программного алгоритма решения задачи, экзогенные переменные -количество вершин графа, - количество дуг.

Экспериментальное исследование программного алгоритма решения задачи включает этапы:

**Этап 1.** *Получение выборки статистических данных в ходе проведения эксперимента на ЭВМ.*

Количество наблюдений m=10 … 15. Результаты проведения эксперимента представляются в таблице 4.

Таблица 4 - Исходные статистические данные:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер наблюдения i | Время выполнения программы, мс, | Количество вершин, | Количество дуг, |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |
| … | … | … | … | … | … | **…** | … | … |
| m |  |  |  |  |  |  |  |  |
| Сумма |  |  |  |  |  |  |  |  |

**Этап 2.** *Определение уравнения связи в общем виде.*

По результатам проведения эксперимента на ЭВМ строятся и анализируются точечные графики зависимости от , **:** и

Анализ точечных графиков позволяет сделать вывод о линейной тенденции. Уравнение связи в общем виде имеет вид

(3.5)

где - время выполнения программного алгоритма;

- количество вершин графа;

- количество дуг графа;

- свободный коэффициент;

, - выборочные коэффициенты.

**Этап 3.** *Оценивание коэффициентов уравнения связи.*

Оценивание коэффициентов , и производится МНК по шагам.

**Шаг 1.** Подставим в математическое описание МНК вместо правую часть уравнения связи в общем виде . Математическое описание МНК будет иметь вид

(3.6)

**Шаг 2.** Определим СНУ для оценивания коэффициентов , и уравнения связи.

(3.7)

(3.8)

(3.9)

Выполним преобразования: сократим на (-2) левую и правую части, раскроем скобки, сгруппируем слагаемые, заменим аддитивную операцию на мультипликативную операцию . СНУ имеет вид

(3.10)

СНУ решается методом Гаусса.

**Этап 4.** *Анализ уравнения связи и расчет доверительного интервала.*

Анализ уравнения связи включает расчет коэффициента множественной корреляции, совокупного коэффициента детерминации, частных коэффициентов корреляции, частных коэффициентов детерминации, частных коэффициентов эластичности и частных -коэффициентов.

Коэффициент множественной корреляциирассчитывается по формуле

(3.11)

где - парный коэффициент корреляции между и ;

- парный коэффициент корреляции между и ;

- парный коэффициент корреляции между и ;

Коэффициент корреляции . Если , то связь отсутствует, - связь слабая, - связь умеренная, - связь сильная.

Парный коэффициент корреляции между и рассчитывается по формуле

(3.12)

Парный коэффициент корреляции между и рассчитывается по формуле

(3.13)

Парный коэффициент корреляции между и рассчитывается по формуле

(3.14)

- совокупный коэффициент детерминации.

Частные коэффициенты корреляции позволяют провести анализ тесноты связи между эндогенной переменной и одной из экзогенных переменных при неизменных других.

Частный коэффициент корреляции между и при неизменной рассчитывается по формуле

(3.15)

Частный коэффициент корреляции между и при неизменной рассчитывается по формуле

(3.16)

, - частные коэффициенты детерминации.

Частные коэффициенты эластичности рассчитывают по формулам

(3.17)

где - среднее арифметическое значение объема выборки .

(3.18)

где - среднее арифметическое значение объема выборки .

Частные -коэффициенты рассчитываются по формулам

(3.19)

где - выборочный коэффициент;

- средняя квадратическая ошибка (отклонение) объема выборки ;

- средняя квадратическая ошибка (отклонение) объема выборки .

(3.20)

где - средняя квадратическая ошибка (отклонение) объема выборки ;

Средняя квадратическая ошибка объема выборки рассчитывается по формуле

(3.21)

Средняя квадратическая ошибка объема выборки рассчитывается по формуле

(3.22)

# 5 ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММНОГО КОМПЛЕКСА

На рисунке 2 и 3 представлены скриншоты программы.

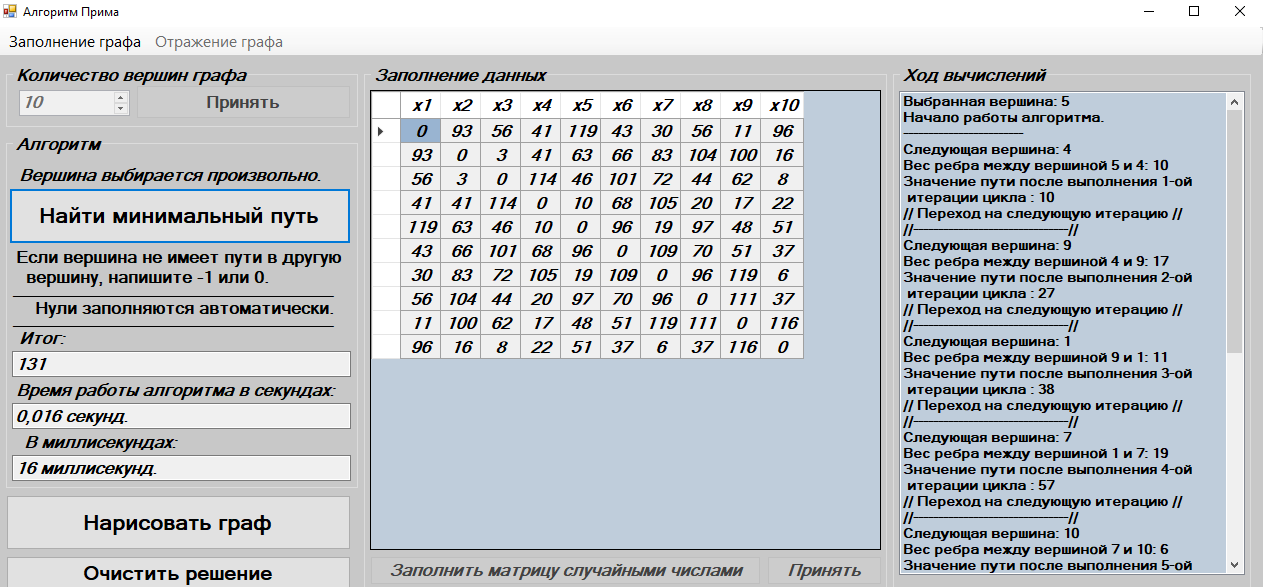


Рисунок 2 – Интерфейс программы и результат работы алгоритма

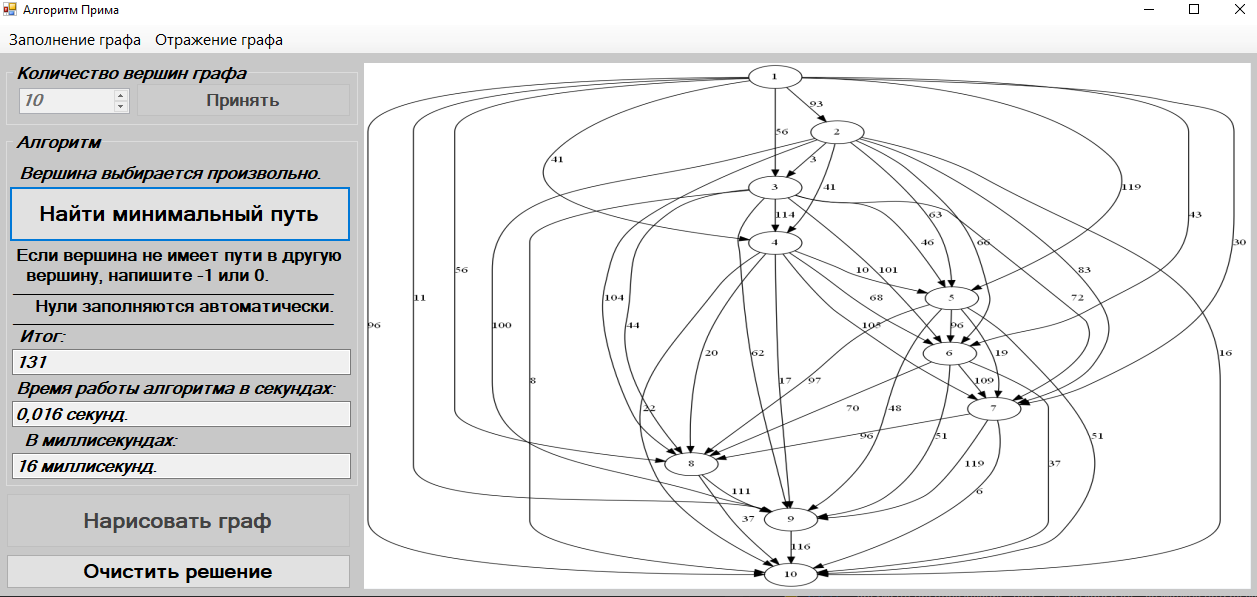


Рисунок 3 – Интерфейс программы и результат вывода графа на экран

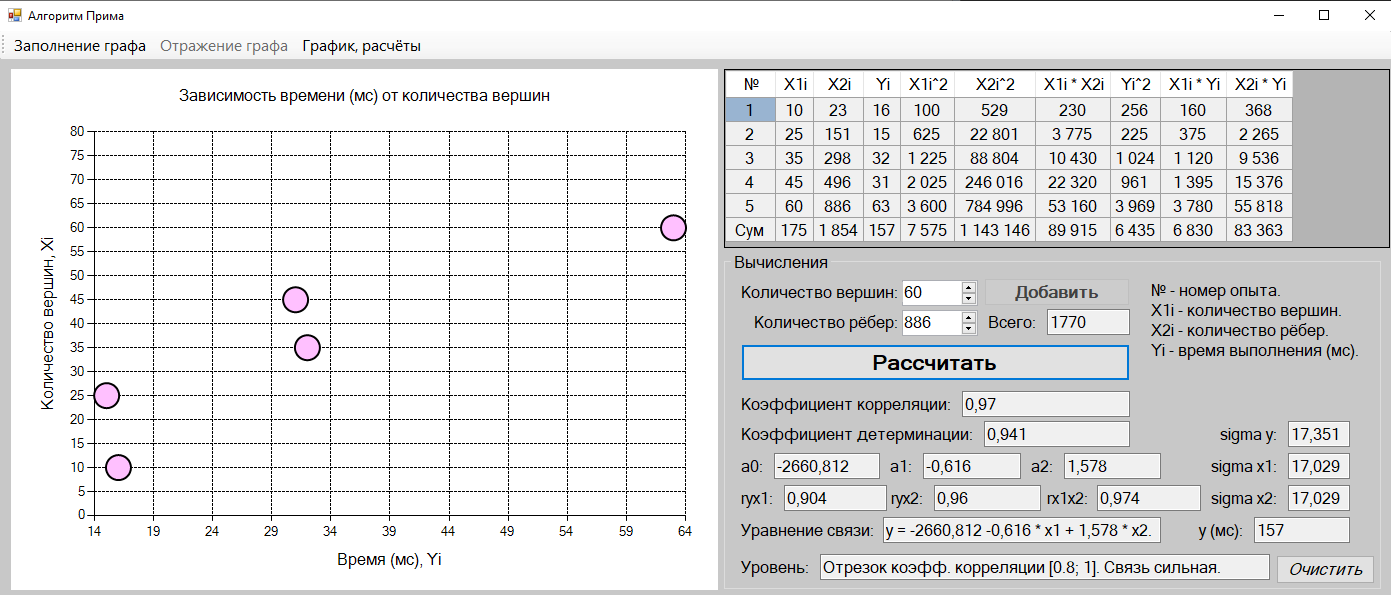


Рисунок 4 – Интерфейс вкладки «График, расчёты».

# 6 ОПИСАНИЕ ФУНКЦИЙ

Описания функций приведены в таблице 4.

Таблица 5 – Описания функций

|  |  |
| --- | --- |
| **Описание метода** | **Функция метода** |
| Int algorithm\_prima() | Алгоритм нахождения остовного дерева наименьшего веса |
| Void btnaccept\_Click() | Запоминает число вершин, заполняет таблицу смежности нулями. |
| Void datamatrix\_CellValueChanged() | Ручное заполнение таблицы смежности |
| Void btnrandom\_Click() | Автоматическое заполнение таблицы смежности псевдослучайными целыми числами |
| Void btnrunprima\_Click | Запуск работы алгоритма, вызов функции algorithm\_prima() |
| Void graphpaint\_Click() | Вывод графического представления графа |
| Void btnclear\_Click() | Очистка переменных, таблицы и т.д. |
| Void btncleargraph\_Click() | Очистка данных вкладки «График, расчёты». |
| Void btncomputinggraph\_Click() | Расчеты значений корреляции, детерминации, коэффициентов а1, а0 и т.д. |
| Void toolgraph\_Click() | Открывает вкладку с графом. |
| Void toolhidepicbox\_Click() | Скрывает вкладку с графом. |
| Void toolbtncomputing\_Click() | Открывает вкладку «График, расчёты». |
| btnaddvertex\_Click() | Добавляет количество вершин для таблицы во вкладке «График, расчёты». |

# 7 ВЫВОДЫ ПО РЕЗУЛЬТАТАМ ПРОЕКТИРОВАНИЯ

В ходе курсового проектирования разработано визуальное объектно-ориентированное приложение C++, реализующее алгоритм Прима. Проведен анализ оценки сложности алгоритма. Проведен эксперимент, устанавливающий время работы алгоритма в зависимости от количества заданных вершин.

Использование данного программного обеспечения позволяет крайне быстро и правильно рассчитать наименьший путь для графа, при этом посетив все вершины.

В результате курсового проектирования повышены теоретические знания и приобретены практические навыки в разработке приложений.

# 8 СПИСОК ЛИТЕРАТУРНЫХ ПЕРВОИСТОЧНИКОВ

1. Сэджвик Р. Фундаментальные алгоритмы на С++. – М.: «DiaSoft», 2001. – 688 с. - Части 1-5.

2. Анашкина Н.В. Технологии и методы программирования: учеб. пособие для студ. Учреждений высш. проф. Образования / Н.В. Анашкина, Н.Н. Петухова, В.Ю. Смольянинов. – М.: Издательский центр «Академия», 2012. – 384 с.

3. Хусаинов Б.С. Структуры и алгоритмы обработки данных. Примеры на языке Си (+CD): Учеб. пособие. – Финансы и статистика, 2004. – 464 с.

4. Таха Хемди А. Введение в исследование операций, 7-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. -912 с.

5. Гагарина Л.Г. Алгоритмы и структуры данных: учеб. пособие/Л.Г. Гагарина, В.Д. Колдаев. – М.: Финансы и статистика, ИНФРА-М, 2009 – 304 с.

6. Романенко Т.А. Структуры и алгоритмы обработки данных. ЭУМК. – Новосибирск, Издательство НГТУ - 2012. [электронный ресурс http://edu.nstu.ru/courses/saod/file\_structure.htm]

7. Уайс М. А. Организация структур данных и решение задач на С++/Уайс М.А. пер. с англ. – М.: ЭКОМ Паблишерз, 2008. – 896 с.

8. Дик Д.И. Требования к оформлению текстовой документации курсовых и дипломных проектов (работ): Методические указания для студентов направлений (специальностей) 230000 (230105), 090000 (090105). – Курган: Из-во КГУ, 2008. - 40 с.