**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Решение задачи о коммивояжере с помощью метода ближайшего соседа**

**Требования задания»**

**Студент гр. 22Б16-пу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Олизько С.С.**

**Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дик А.Г.**

**Санкт-Петербург**

**2023 г**

Оглавление

[**Цель работы:** 3](#_Toc153215487)

[**Задача:** 3](#_Toc153215488)

[**Теоритическая часть:** 3](#_Toc153215489)

[**Алгоритм метода:** 5](#_Toc153215490)

[**Описание программы:** 6](#_Toc153215491)

[**Рекомендации для пользователя:** 6](#_Toc153215492)

[**Контрольный пример:** 8](#_Toc153215493)

[**Анализ результатов работы алгоритма и вводных условий:** 8](#_Toc153215494)

[**Вывод:** 10](#_Toc153215495)

[**Источники:** 10](#_Toc153215496)

[**Листинг программы с детальными комментариями:** 11](#_Toc153215497)

# **Цель работы:**

Изучение и применение алгоритма ближайшего соседа для решения задачи о коммивояжере с использованием взвешенного орграфа.

# **Задача:**

Формализовать задачу о коммивояжере, применив алгоритм ближайшего соседа, и провести поиск кратчайшего гамильтонова цикла в представленном взвешенном орграфе.

# **Теоретическая часть:**

**Задача коммивояжера**

Задача коммивояжера – это классическая задача комбинаторной оптимизации, которая состоит в нахождении самого короткого маршрута, проходящего через каждый город ровно один раз и завершающегося в исходном городе. Проблема возникает в различных областях, таких как логистика, транспорт, производство и планирование.

**Гамильтоновы циклы**

Гамильтонов цикл – это путь в графе, который проходит через каждую вершину ровно один раз и возвращается в исходную вершину. Найти гамильтонов цикл в графе является NP-полной задачей, то есть, нет известного эффективного алгоритма, способного решить эту задачу для всех возможных входных данных за разумное время.

**Алгоритм ближайшего соседа**

Алгоритм ближайшего соседа – это простой эвристический алгоритм, используемый для решения задачи коммивояжера. Он начинает с произвольной вершины и на каждом шаге выбирает ближайшую вершину, которая еще не была посещена. Этот процесс продолжается до тех пор, пока все вершины не будут посещены. Хотя алгоритм прост в реализации, он не гарантирует нахождение оптимального решения и может приводить к получению подптимальных или даже неверных результатов.

**Недостатки алгоритма ближайшего соседа**

1. Локальные оптимальные решения: Алгоритм может застрять в локальных оптимумах, пропустив более выгодные пути из-за своей жадной стратегии выбора.
2. Неоптимальность: В результате жадного выбора ближайших вершин алгоритм может пропустить более короткие пути, которые становятся доступными на более поздних этапах.
3. Чувствительность к начальному выбору: Исходная вершина, с которой начинается алгоритм, может существенно влиять на получаемое решение, что делает его результаты менее надежными.

Все эти недостатки делают алгоритм ближайшего соседа менее предпочтительным для решения задачи коммивояжера в сравнении с более сложными, но более точными алгоритмами, такими как динамическое программирование или алгоритм ветвей и границ.

# **Алгоритм метода:**

1. Инициализация пустого списка hamiltonian\_cycle для хранения всех возможных гамильтоновых циклов.
2. Для каждой вершины графа:
   1. Создать временный граф hamiltonian\_cycle\_temp, содержащий только текущую вершину.
   2. Установить текущую вершину как посещенную.
3. Пока не все вершины графа не будут посещены:
   1. Найти ближайшую непосещенную вершину к текущей.
   2. Добавить эту вершину к hamiltonian\_cycle\_temp, создать ребро между текущей и следующей вершиной.
   3. Пометить найденную вершину как посещенную и обновить текущую вершину.
4. Если все вершины посещены и есть ребро от последней вершины к первой, добавить его к hamiltonian\_cycle\_temp.
5. Если в hamiltonian\_cycle\_temp есть хотя бы один гамильтонов цикл:
   1. Найти цикл с минимальной длиной среди всех найденных циклов.
   2. Вернуть минимальный цикл.
6. Если ни одного гамильтонова цикла не найдено, вернуть None.

# **Описание программы:**

Программа реализована на языке Python 3.12 с использованием следующих пакетов: tkinter, networkx и matplotlib.pyplot. В программе используются 7 функций. В таблицах описание функций программы.

*Таблица 1: функции*

| Имя функции | Тип возвращаемого значения | Описание функции |
| --- | --- | --- |
| finish() | None | Закрывает приложение при вызове. |
| nearest\_neighbor\_hamiltonian\_cycle(graph, canvas, pos) | nx.DiGraph | Находит гамильтонов цикл с использованием метода ближайшего соседа для заданного графа graph. Отрисовывает найденный цикл на холсте canvas с использованием позиций вершин pos. |
| Gdefault() | None | Инициализирует граф по умолчанию. |
| edit(event) | None | Редактирует значения в дереве при двойном щелчке мыши. |
| update\_tree() | None | Обновляет содержимое дерева с информацией о ребрах графа. |
| add\_node(event) | None | Добавляет вершину в граф при щелчке левой кнопкой мыши на холсте. |
| add\_edge\_on\_right\_click(event) | None | Добавляет ребро в граф при щелчке правой кнопкой мыши на холсте. |

# **Рекомендации для пользователя:**

**Запуск программы:**

1. Убедитесь, что на вашем компьютере установлен Python.
2. Установите необходимые библиотеки: tkinter, networkx и matplotlib. Выполните команду pip install tkinter networkx matplotlib.
3. Запустите программу для активации графического интерфейса.

**Использование программы:**

1. Используйте холст для создания вершин графа левой кнопкой мыши и добавления ребер правой кнопкой мыши.
2. Изменяйте значения ребер и вершин, дважды щелкнув на соответствующем элементе в таблице.
3. Нажмите кнопку "Gdefault", чтобы инициализировать граф по умолчанию.
4. Нажмите кнопку "Построить гамильтонов цикл" для поиска и отображения гамильтонова цикла.
5. Результаты будут отображены в текстовом поле и на втором холсте.

Рекомендуется убедиться в правильности ввода значений вершин и ребер перед запуском алгоритма поиска гамильтонова цикла.

**Исходный код программы доступен по ссылке ниже:**

<https://github.com/StephanOlizko?tab=repositories>

# **Контрольный пример:**

*Рис 1: пример окна программы*

# **Анализ результатов работы алгоритма и вводных условий:**

Стандартный алгоритм ближайшего соседа представляет собой жадный алгоритм, который начинает с произвольной вершины и на каждом шаге выбирает следующую вершину, ближайшую к текущей, чтобы построить гамильтонов цикл. Он продолжает этот процесс, пока не посетит все вершины. Время выполнения алгоритма составляет O(n^2), где n - количество вершин в графе.

Модификация алгоритма, рассматривающая старт из каждой вершины, имеет лучшую сходимость по сравнению со стандартным алгоритмом ближайшего соседа. В этой модификации алгоритм запускается от каждой вершины графа и выбирает наилучший гамильтонов цикл из всех полученных. Это позволяет найти оптимальное решение для данной задачи в большинстве случаев. Время выполнения модифицированного алгоритма также составляет O(n^2) на каждой итерации для каждой вершины, но общее время выполнения может быть выше из-за необходимости выполнения алгоритма от каждой вершины.

Следовательно, модификация алгоритма ближайшего соседа, учитывающая старт из каждой вершины, предпочтительна в сравнении со стандартным алгоритмом благодаря лучшей сходимости и более точным результатам.

# **Вывод:**

В ходе выполнения лабораторной работы была рассмотрена задача о коммивояжере и реализовано её решение с использованием алгоритма ближайшего соседа. Этот жадный алгоритм позволяет найти приближенное решение задачи, находя кратчайший гамильтонов цикл в графе.

Также была представлена модификация алгоритма ближайшего соседа, которая учитывает старт из каждой вершины. Это позволяет улучшить сходимость алгоритма и получить более точные результаты.

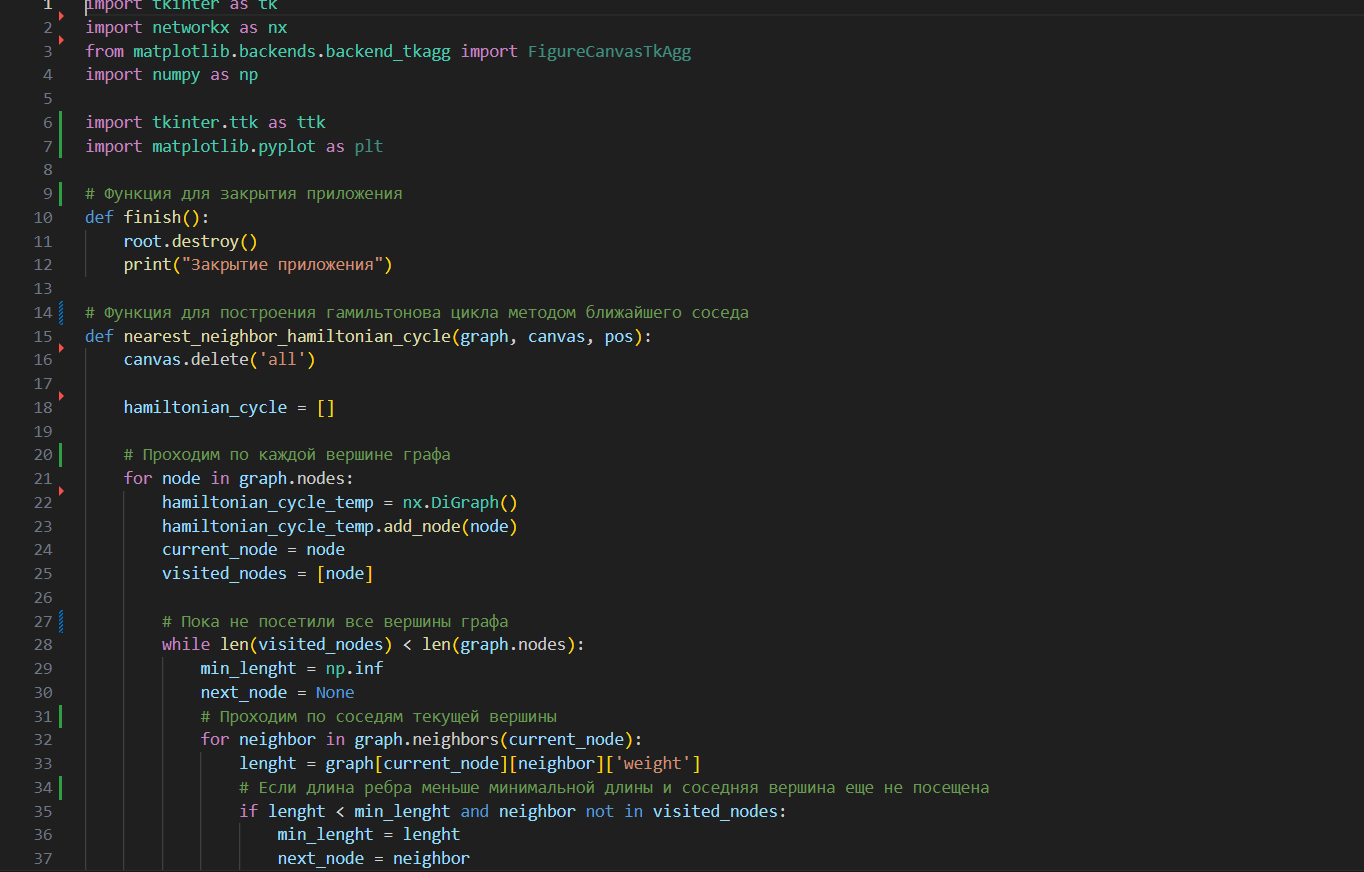
Из анализа результатов работы алгоритмов видно, что модифицированная версия сходится к более оптимальному решению, чем стандартный алгоритм. Это делает модификацию предпочтительным выбором для решения задачи о коммивояжере в большинстве случаев.

Таким образом, выполнение лабораторной работы позволило лучше понять принципы работы алгоритма ближайшего соседа и его модификации, а также изучить их эффективность в решении задачи о коммивояжере.

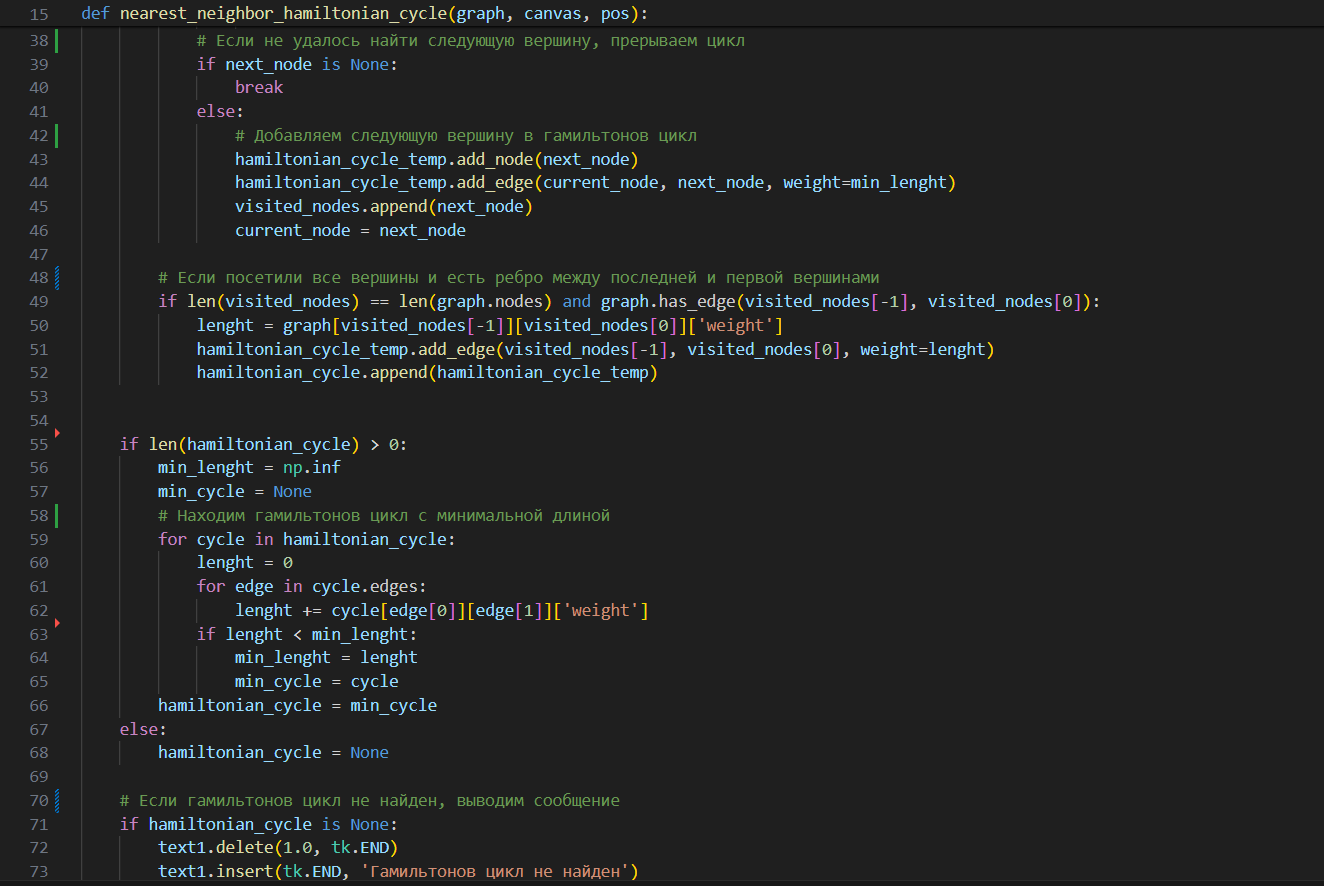
# **Источники:**

<https://docs.python.org/3/library/tkinter.html>

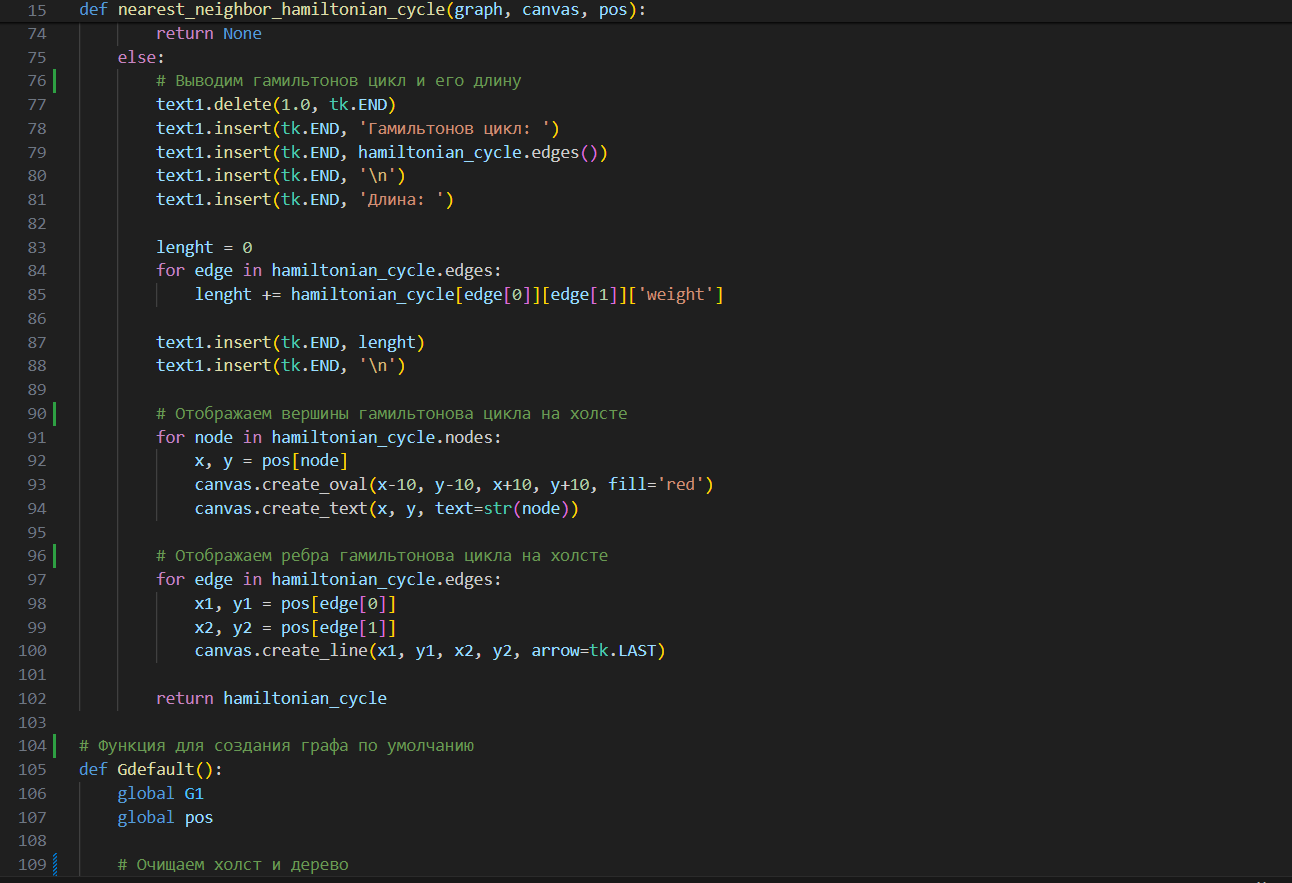
# **Листинг программы с детальными комментариями:**



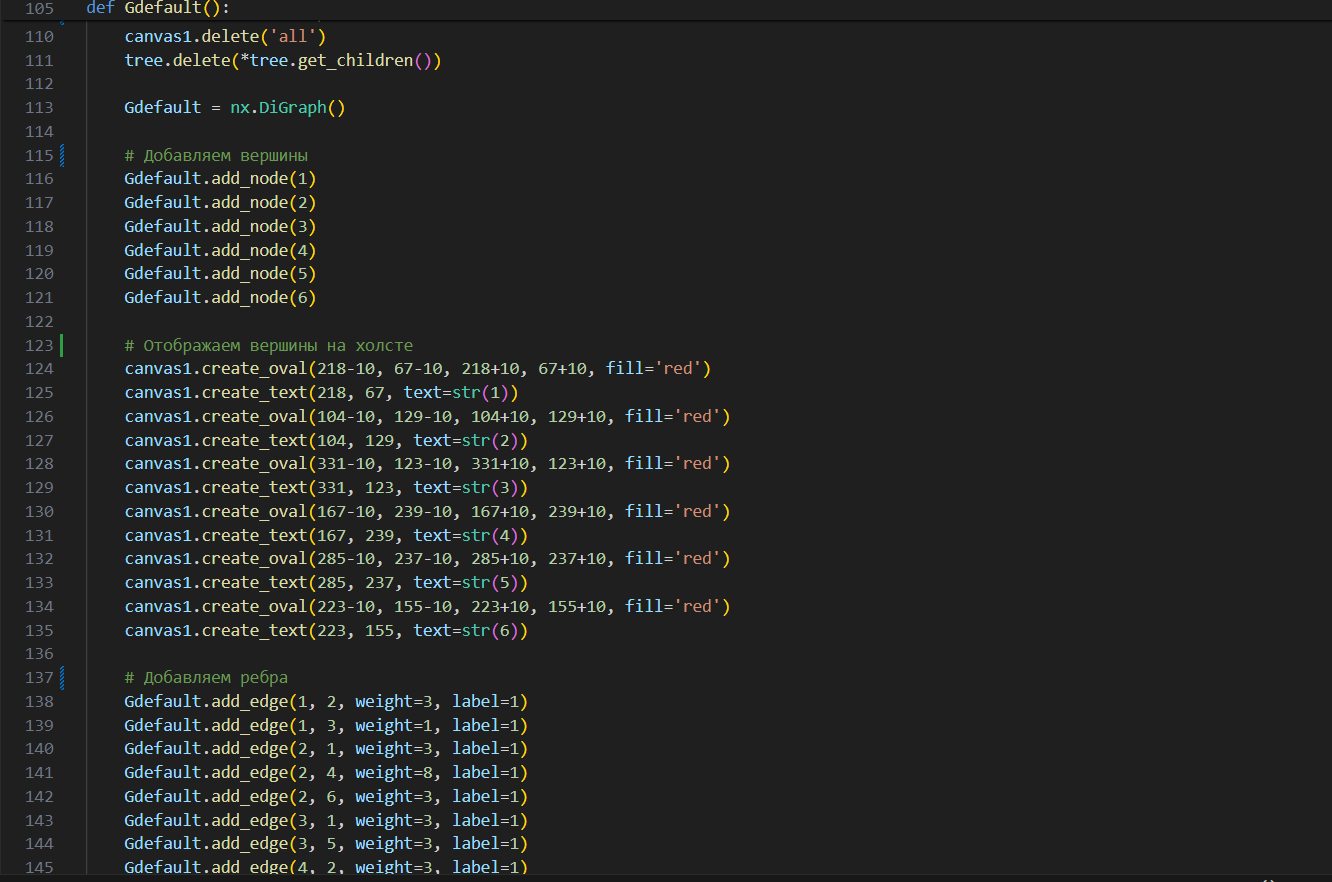
*Рис 2: листинг*



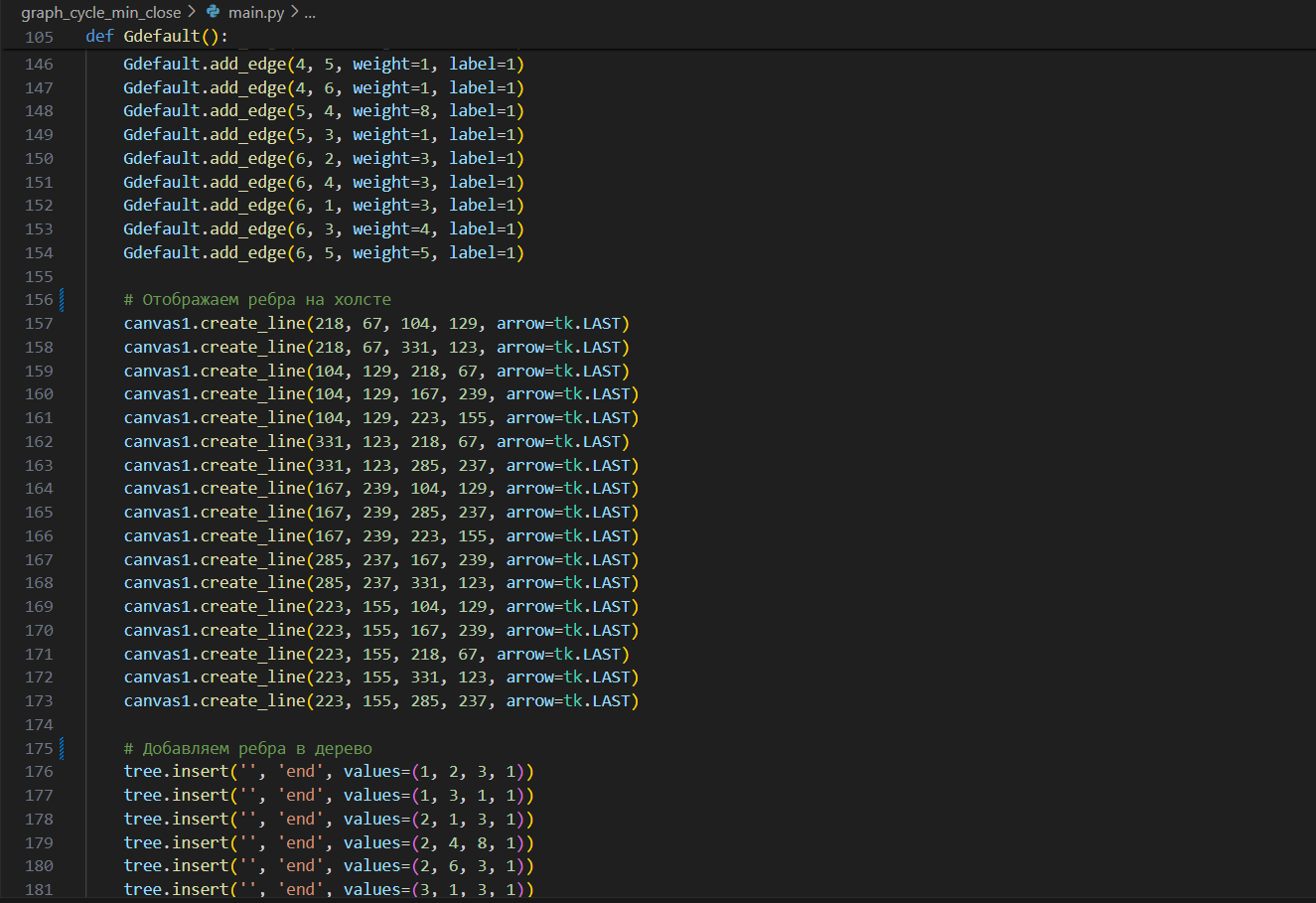
*Рис 3: листинг*



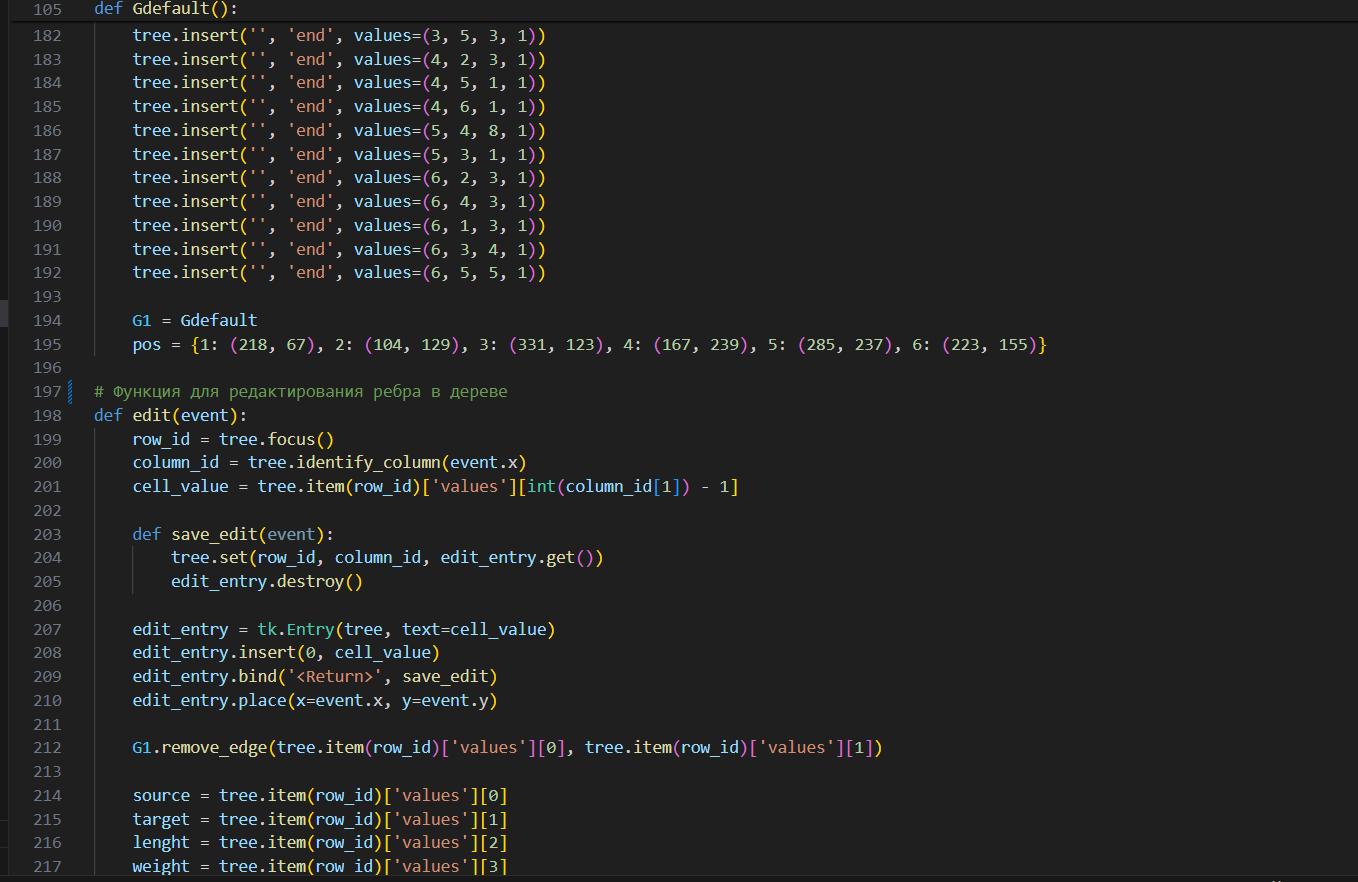
*Рис 4: листинг*



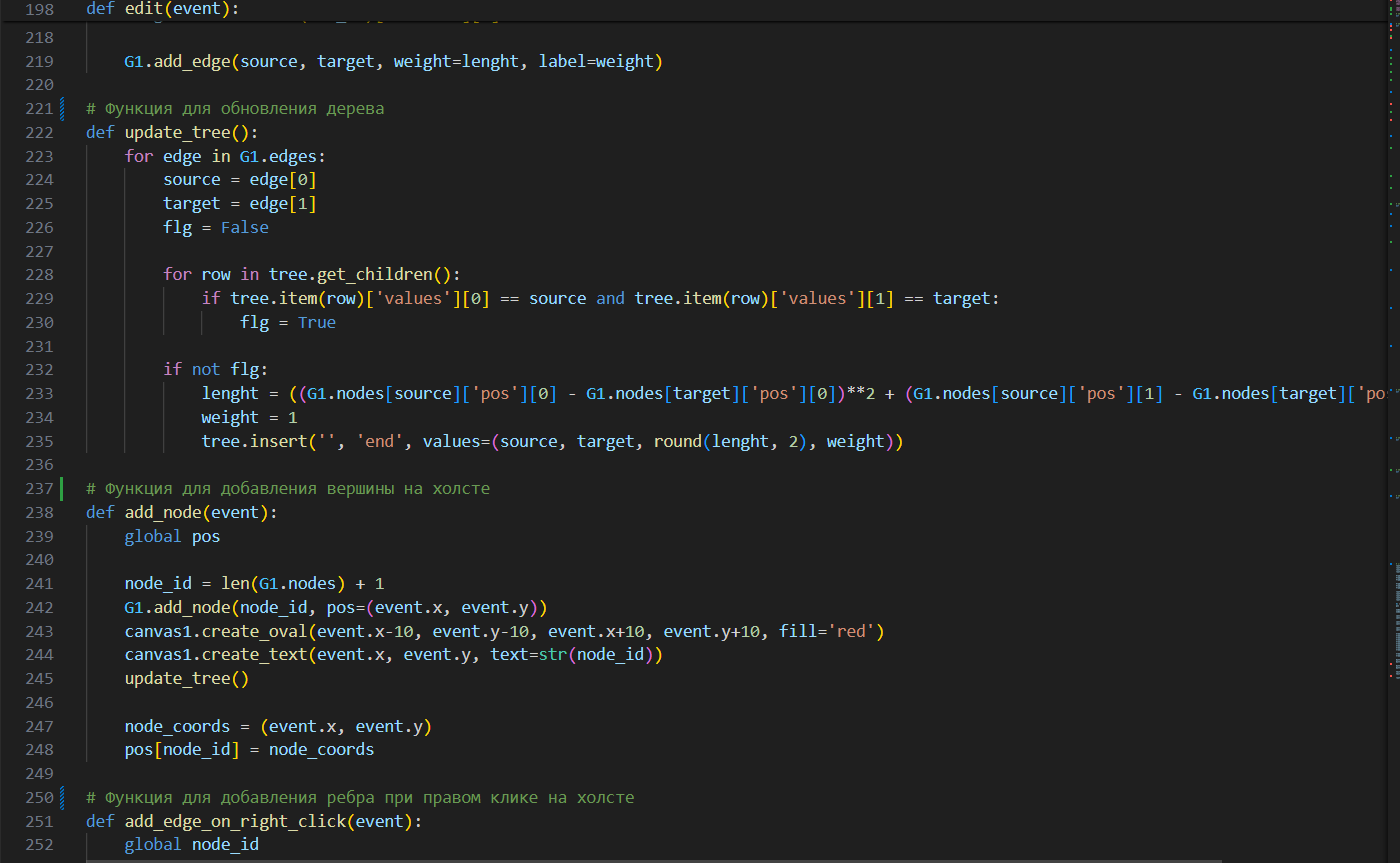
*Рис 5: листинг*



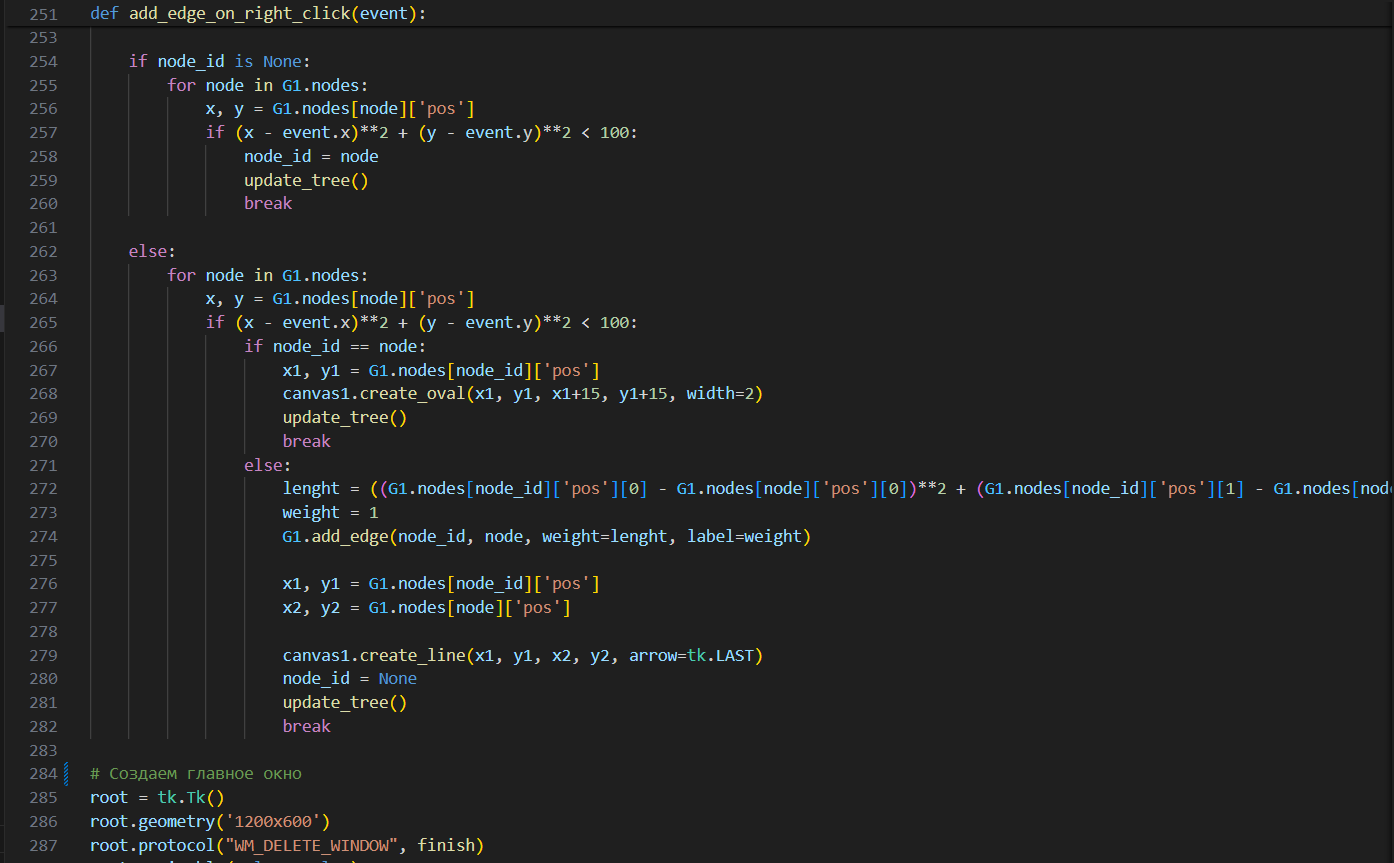
*Рис 6: листинг*



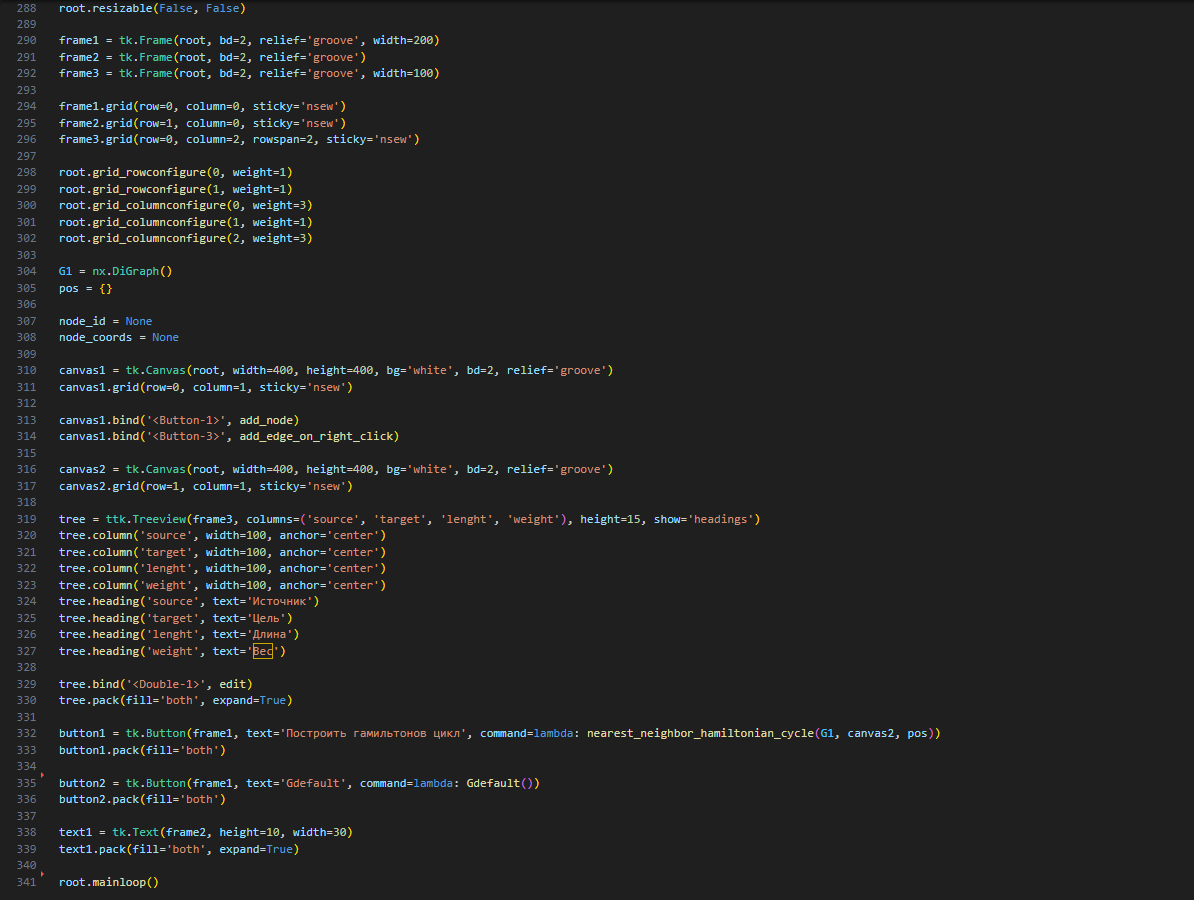
*Рис 7: листинг*



*Рис 8: листинг*



*Рис 9: листинг*



*Рис 10: листинг*