**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

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

**ОТЧЕТ**

**по лабораторной работе №2**

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

**на тему «Решение задачи о коммивояжере с помощью метода имитации отжига»**

**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:**

Исследовать особенности решения задачи о коммивояжере с использованием алгоритма имитации отжига и сравнить его эффективность с решением задачи при помощи алгоритма ближайшего соседа, выполненного в предыдущей лабораторной работе.

# **Задачи:**

Формализация задачи о коммивояжере с использованием данного алгоритма, подготовка контрольного примера на основе взвешенного орграфа, нахождение кратчайшего гамильтонова цикла и сравнение полученного решения с результатами, полученными при применении алгоритма ближайшего соседа в предыдущей лабораторной работе.

# **Теоретическая часть:**

**Задача коммивояжера (TSP):**

Задача коммивояжера является одной из классических задач комбинаторной оптимизации. Она заключается в том, чтобы найти самый выгодный маршрут, проходящий через все города (вершины графа) ровно один раз и возвращающийся в исходный город. При этом каждая пара городов соединена ребром с определенной стоимостью (расстоянием).

**Гамильтоновы циклы:**

Гамильтонов цикл в графе - это цикл, который проходит через каждую вершину ровно один раз. В задаче коммивояжера требуется найти гамильтонов цикл с минимальной суммой весов рёбер, соединяющих вершины.

**Алгоритм имитации отжига:**

Алгоритм имитации отжига основан на физическом процессе, в ходе которого нагревается материал до высокой температуры, а затем постепенно остывает, при этом достигая более устойчивого состояния. В контексте решения задачи коммивояжера, алгоритм имитации отжига начинается с случайного маршрута и постепенно улучшает его, принимая различные изменения (перестановки городов) с определенной вероятностью. Эта вероятность уменьшается по мере уменьшения "температуры", что позволяет избежать застревания в локальных оптимумах.

**Недостатки алгоритма имитации отжига:**

Основной недостаток алгоритма имитации отжига заключается в том, что он не гарантирует нахождения глобального оптимума. Вместо этого он ищет приближенное решение, которое может быть хуже глобального оптимального решения. Кроме того, алгоритм требует настройки различных параметров, таких как начальная температура и скорость охлаждения, что может оказаться нетривиальной задачей.

# **Алгоритм метода:**

1. **Инициализация переменных и структур данных:**
   * Задать начальную температуру T, конечную температуру T\_min и коэффициент охлаждения alpha.
   * Задать начальное решение:
     + Если выбран режим "по умолчанию", то начальное решение - список вершин графа в произвольном порядке.
     + Если выбран режим "ближайший сосед", то начальное решение - гамильтонов цикл, найденный с помощью алгоритма ближайшего соседа.
2. **Основной цикл алгоритма:**
   * Пока текущая температура T больше минимальной температуры T\_min, выполнять следующие шаги:
     + Генерировать случайное соседнее решение путем случайной перестановки двух вершин в текущем решении.
     + Проверить валидность полученного соседнего решения и вычислить его длину.
     + Если новое решение короче текущего или случайное число меньше экспоненты разницы длин решений, принять это решение.
     + Уменьшить температуру T на величину, определяемую коэффициентом охлаждения alpha.
     + Обновить лучшее найденное решение, если новое решение лучше текущего.
3. **Вывод результата:**
   * Вывести гамильтонов цикл с минимальной длиной.
   * Если гамильтонов цикл не найден, вывести сообщение об этом.

# **Описание программы:**

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

*Таблица 1: функции*

| Имя функции | Тип возвращаемого значения | Описание функции |
| --- | --- | --- |
| finish | None | Закрывает приложение. |
| nearest\_neighbor\_hamiltonian\_cycle | DiGraph (NetworkX) | Находит приближенное решение задачи коммивояжера с помощью эвристического алгоритма "ближайшего соседа". Возвращает гамильтонов цикл. |
| hamiltonian\_cycle | DiGraph (NetworkX) | Находит решение задачи коммивояжера с помощью метаэвристического алгоритма Simulated Annealing. Возвращает гамильтонов цикл. |
| Gdefault | None | Создает граф по умолчанию, добавляет вершины и ребра, отображает их на холсте и в дереве. |
| edit | None | Редактирует ребро в дереве при двойном клике на нем, позволяя изменить его параметры. |
| update\_tree | None | Обновляет дерево, отображая в нем текущие ребра графа. |
| add\_node | None | Добавляет вершину на холсте по щелчку левой кнопкой мыши, обновляет дерево с вершинами графа. |
| add\_edge\_on\_right\_click | None | Добавляет ребро на холсте при щелчке правой кнопкой мыши, соединяя выбранную вершину с другой вершиной, если она находится рядом. |

# **Рекомендации для пользователя:**

**Запуск программы:**

1. Убедитесь, что на вашем компьютере установлен Python.
2. Установите необходимые библиотеки: tkinter, networkx, matplotlib, numpy, random, math. Выполните команду pip install tkinter networkx matplotlib numpy.
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:**

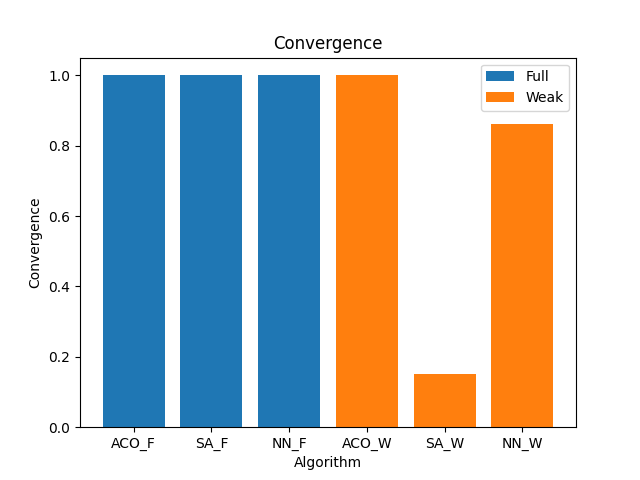
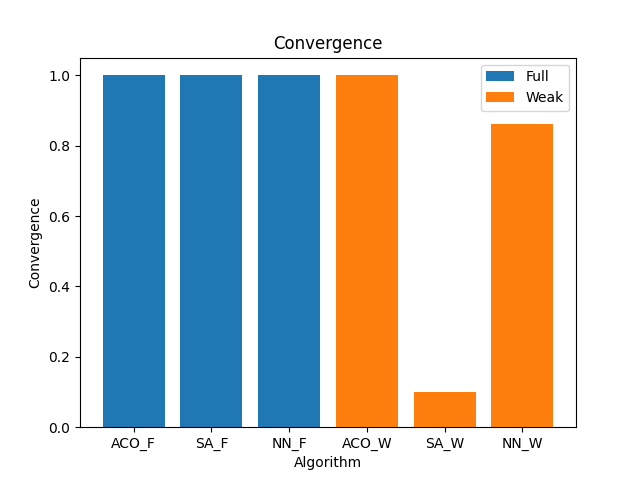
*Рис 1: пример окна программы*

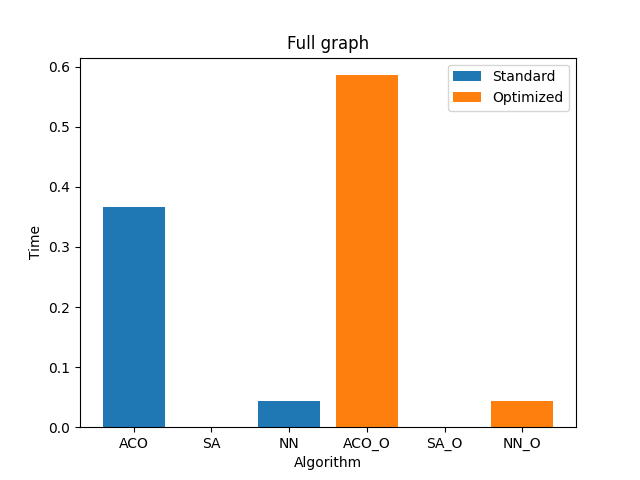
# **Анализ результатов работы алгоритма и вводных условий:**

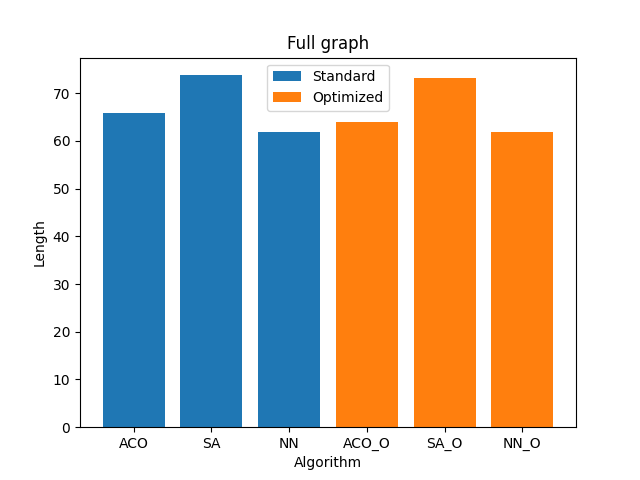
В обычном режиме работы явно видна проблема редкой сходимости алгоритма. При случайном выборе начального решения крайне маловероятно что оно окажется гамильтоновым циклом, в слабо связном графе эти проблемы только усугубляются.

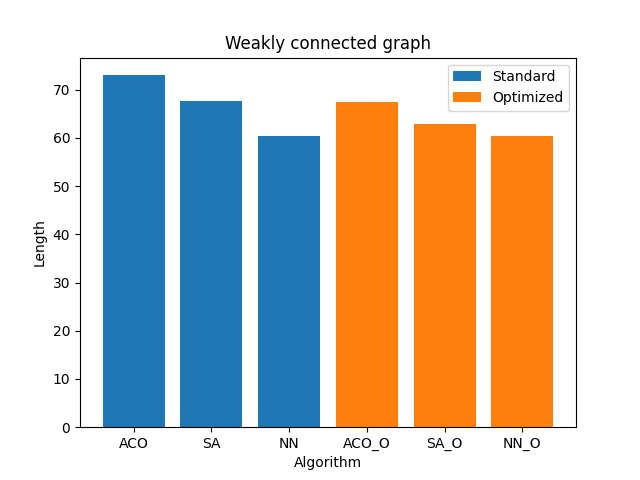
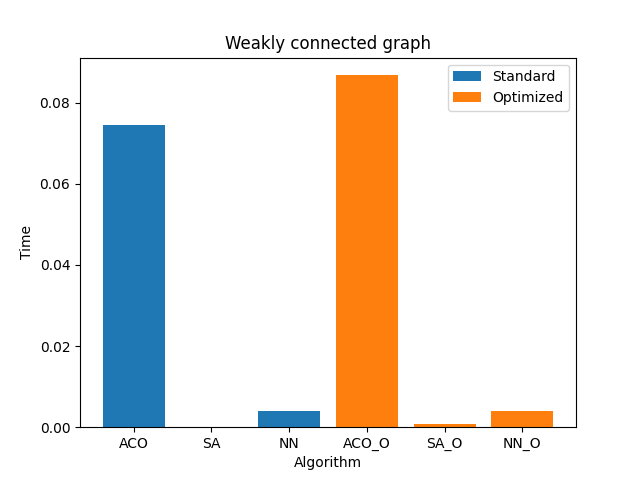
Модификация алгоритма заключается в выборе начального решения с помощью метода ближайшего соседа. В таком случае значительно возрастает вероятность сходимости, но появляется шанс “застрять” в локальном минимуме и не достичь глобального.

В сравнении с алгоритмом ближайшего соседа алгоритм отжига показывает значительное опережение по времени работы, однако явно проигрывает в проценте сходимости и немного отстает в оптимальности решения.

При тестировании использовались полный граф размера 50 и разреженный граф размера 25.  
  








| Algorithm | Type | Converge full | Converge weak | Mean length, time full | Mean length, time weak |
| --- | --- | --- | --- | --- | --- |
| Ant colony optimization | Standard | 1.0 | 1.0 | 65.78, 0.3667 | 73.0, 0.0745 |
| Ant colony optimization | Optimized | 1.0 | 1.0 | 63.88, 0.5857 | 67.38, 0.0867 |
| Simulated annealing | Standard | 1.0 | 0.1 | 73.79, 0.0011 | 67.6, 0.00015 |
| Simulated annealing | Optimized | 1.0 | 0.15 | 73.27, 0.0011 | 62.93, 0.00077 |
| Nearest neighbor | Standard | 1.0 | 0.86 | 61.97, 0.0431 | 60.34, 0.0041 |

# **Вывод:**

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

Также была представлена модификация алгоритма имитации отжига которая учитывает старт из решения, заданного алгоритмом ближайшего соседа. Это позволяет улучшить сходимость алгоритма и получить более точные результаты.

Из анализа результатов работы алгоритмов видно, что модифицированная версия сходится чаще, 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 имитации отжига и его модификации, а также изучить их эффективность в решении задачи о коммивояжере.

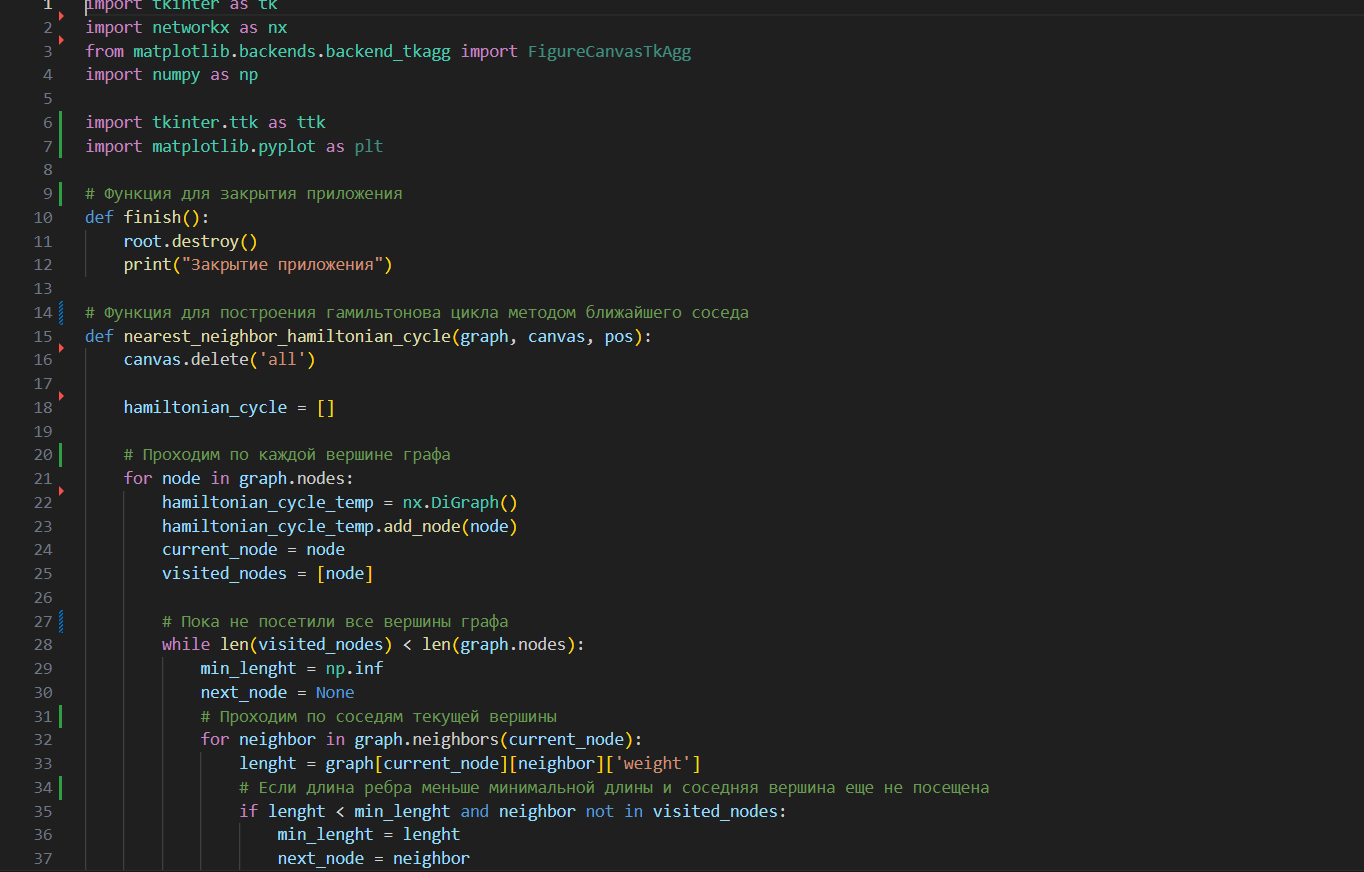
# **Источники:**

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

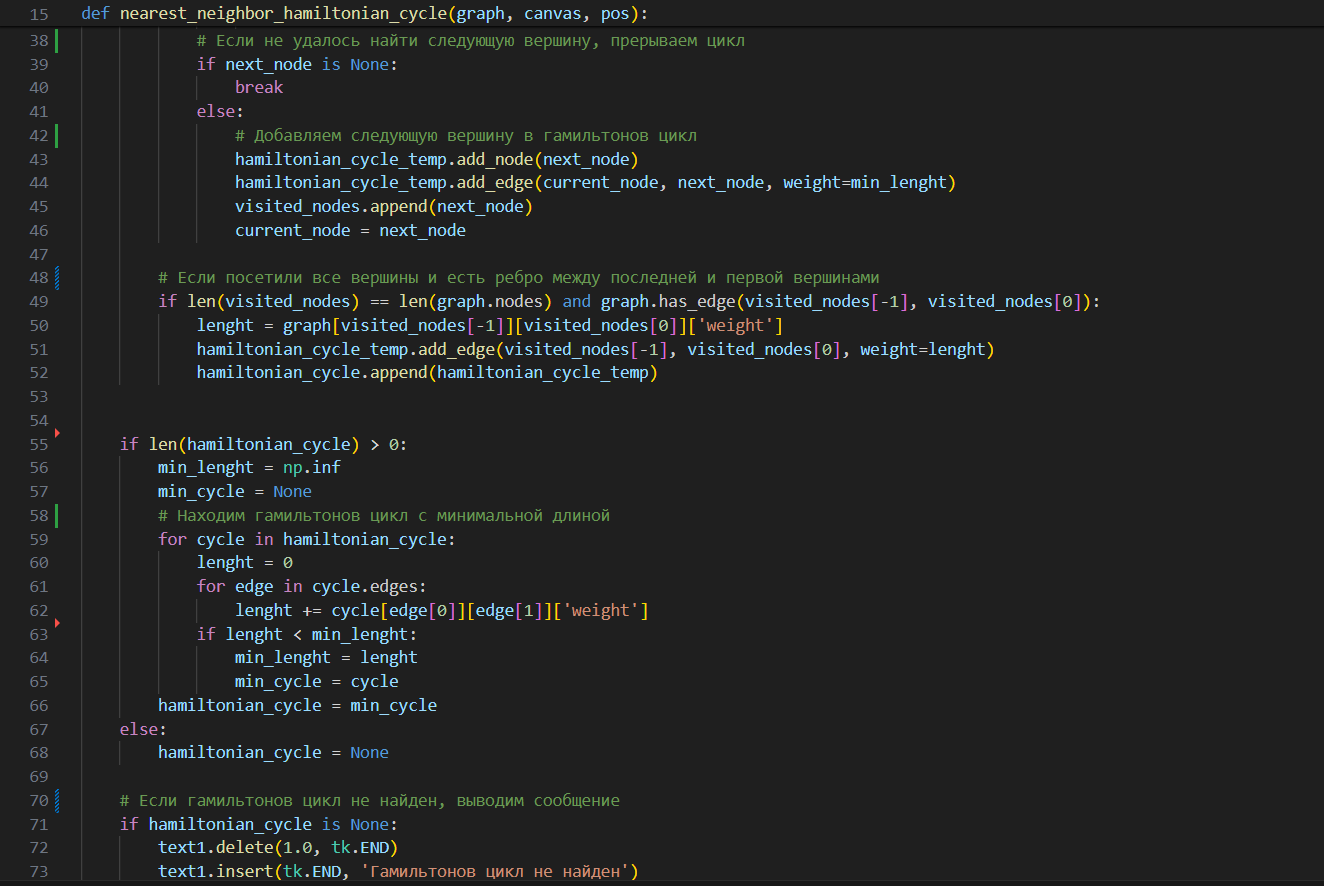
https://matplotlib.org/stable/index.html

https://numpy.org/doc/

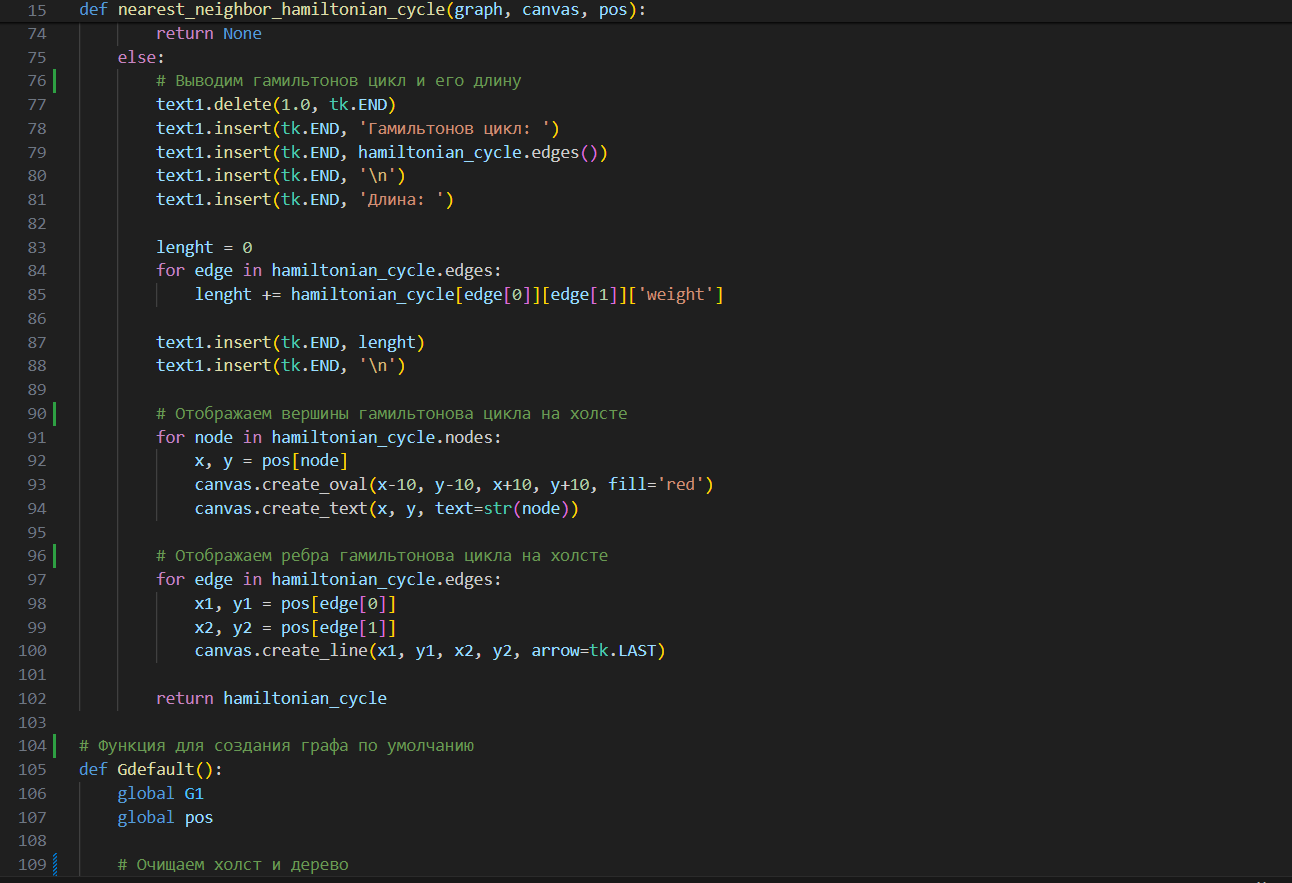
# **Листинг программы с детальными комментариями:**



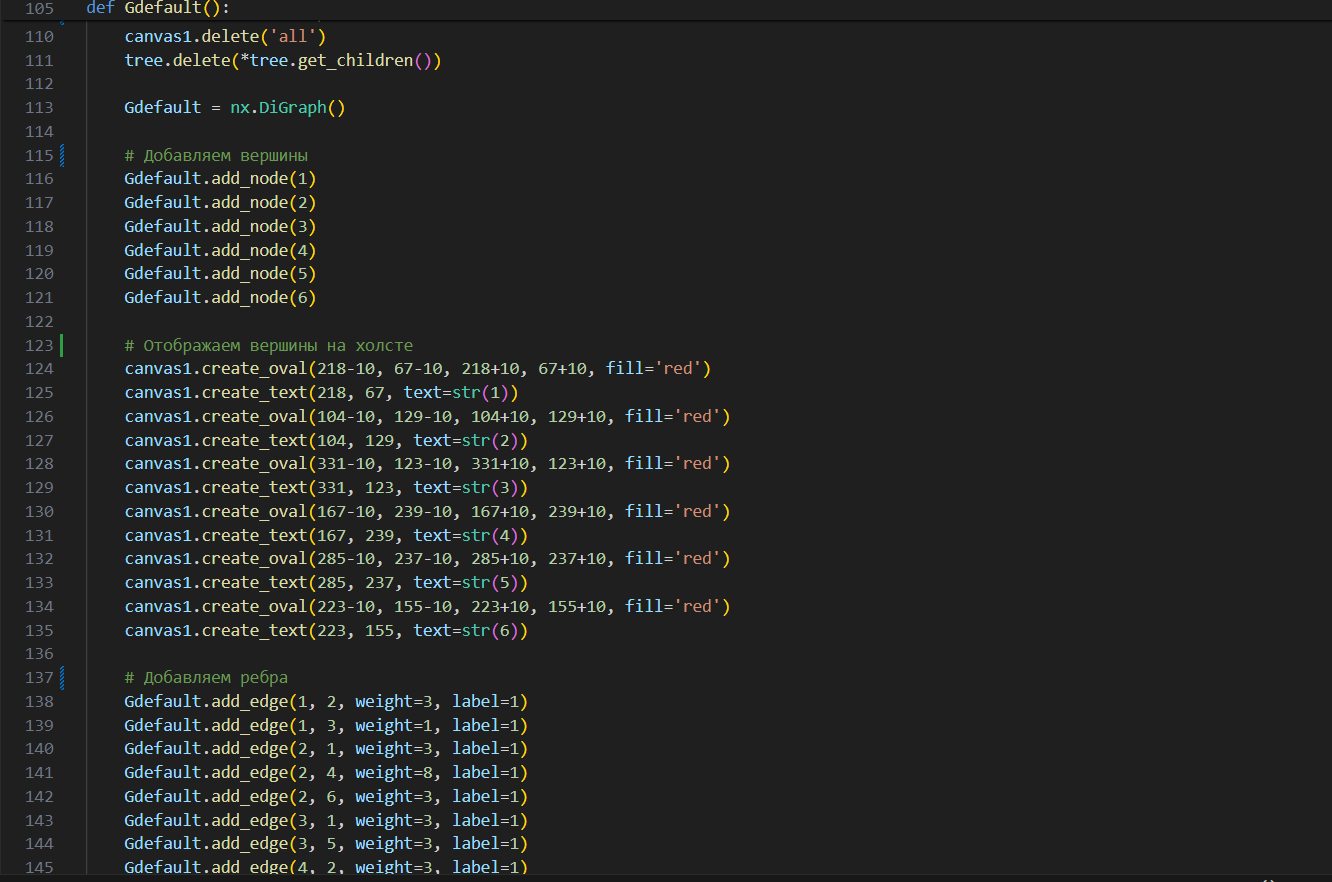
*Рис 2: листинг*



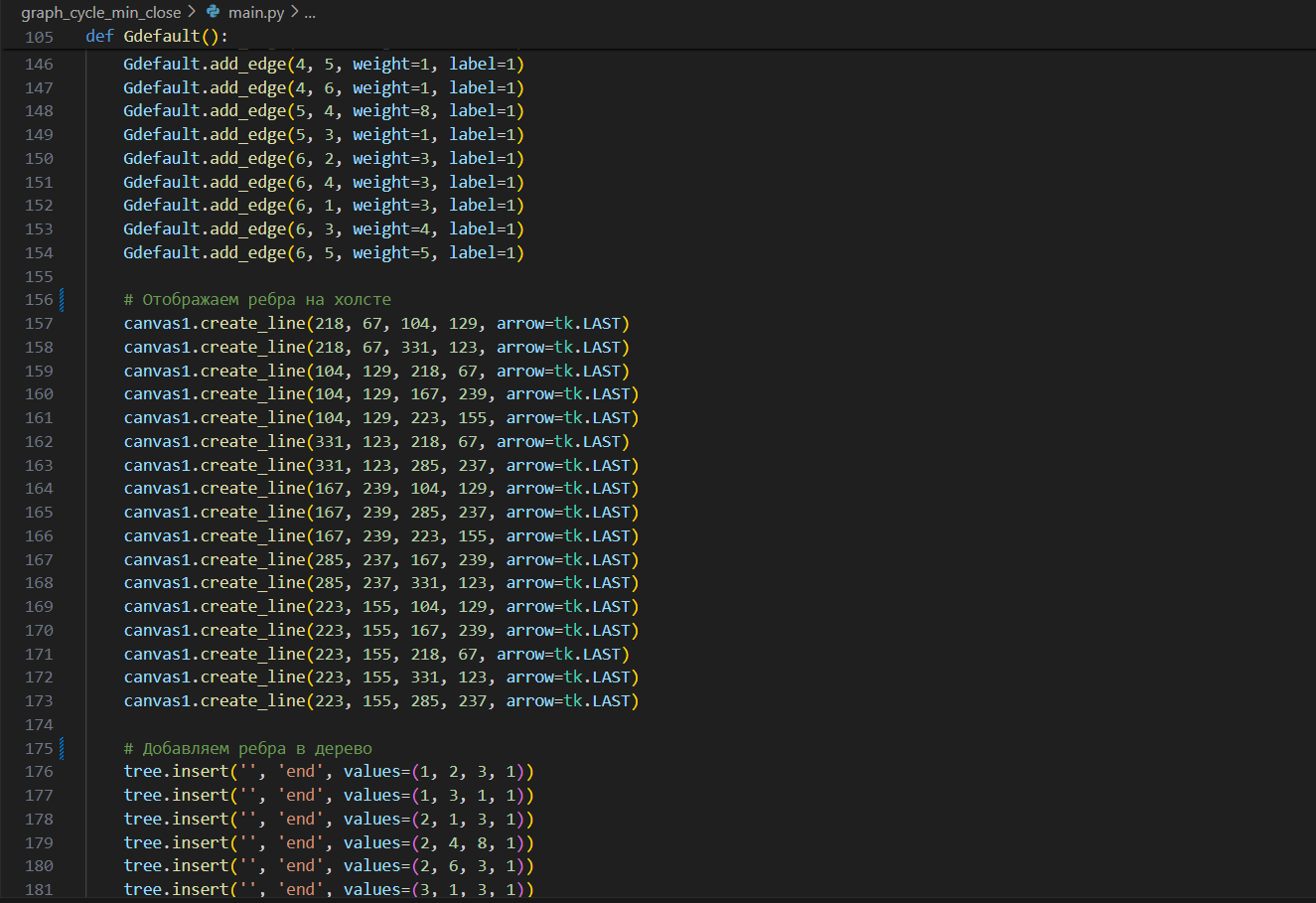
*Рис 3: листинг*



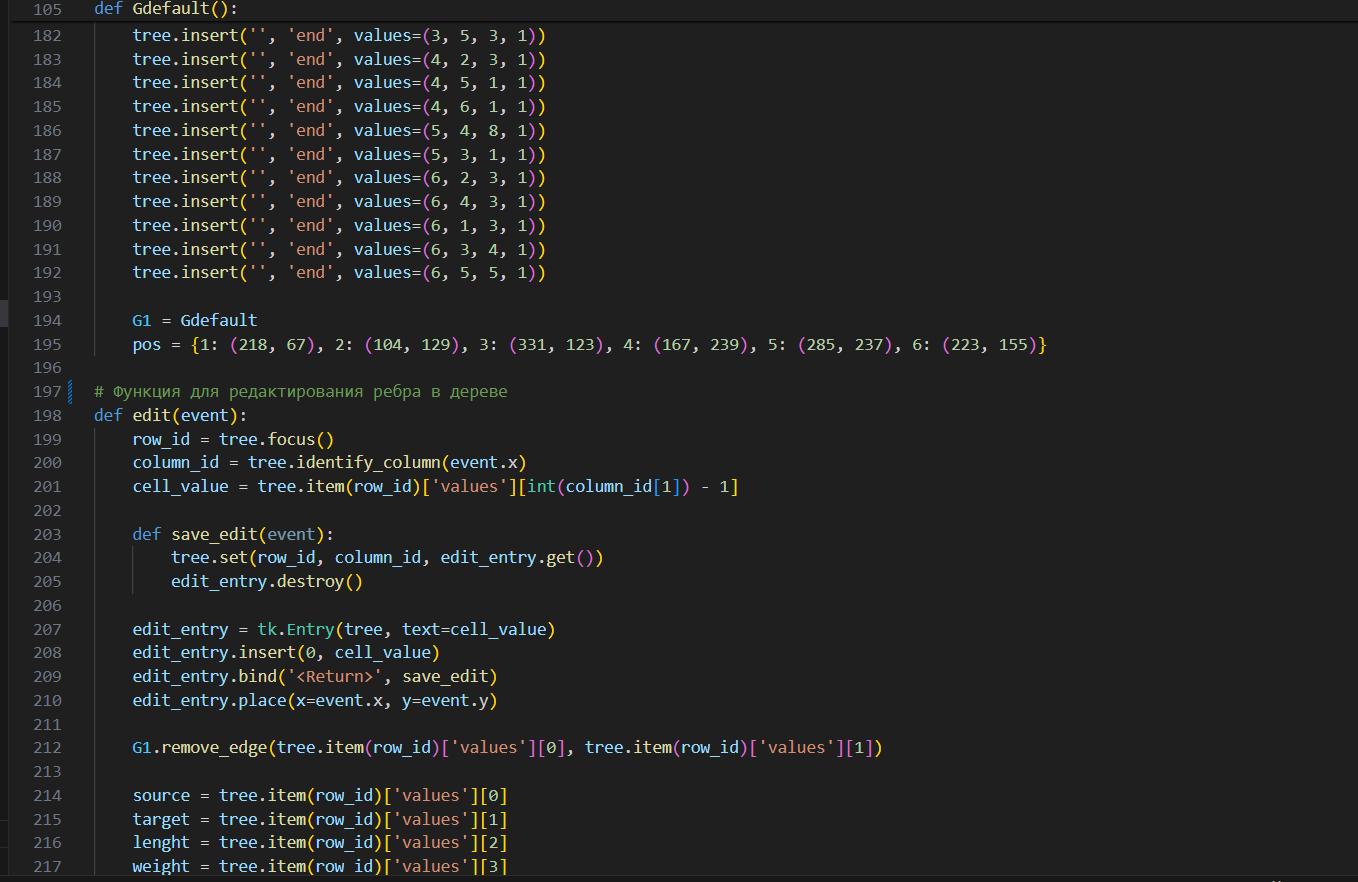
*Рис 4: листинг*



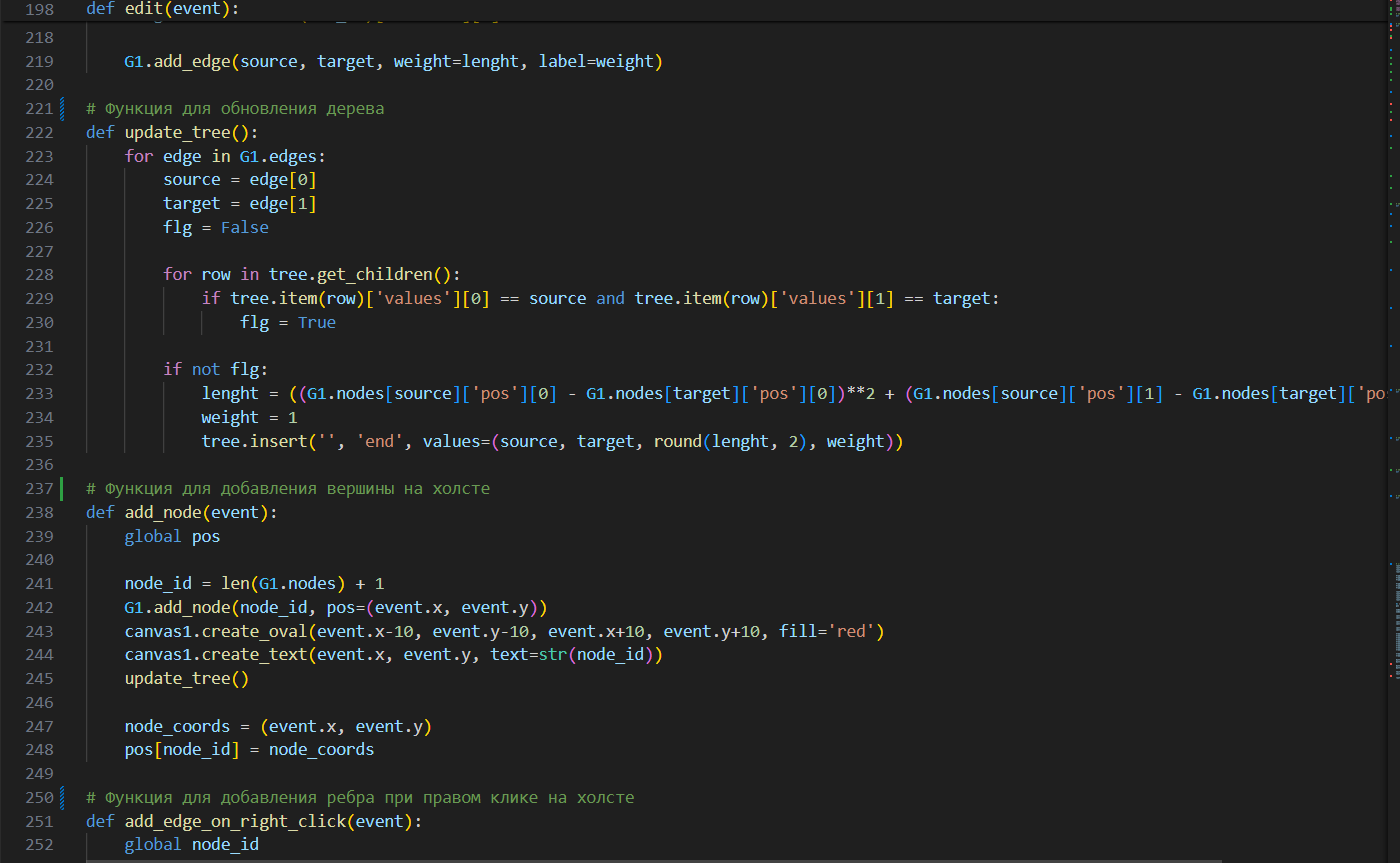
*Рис 5: листинг*



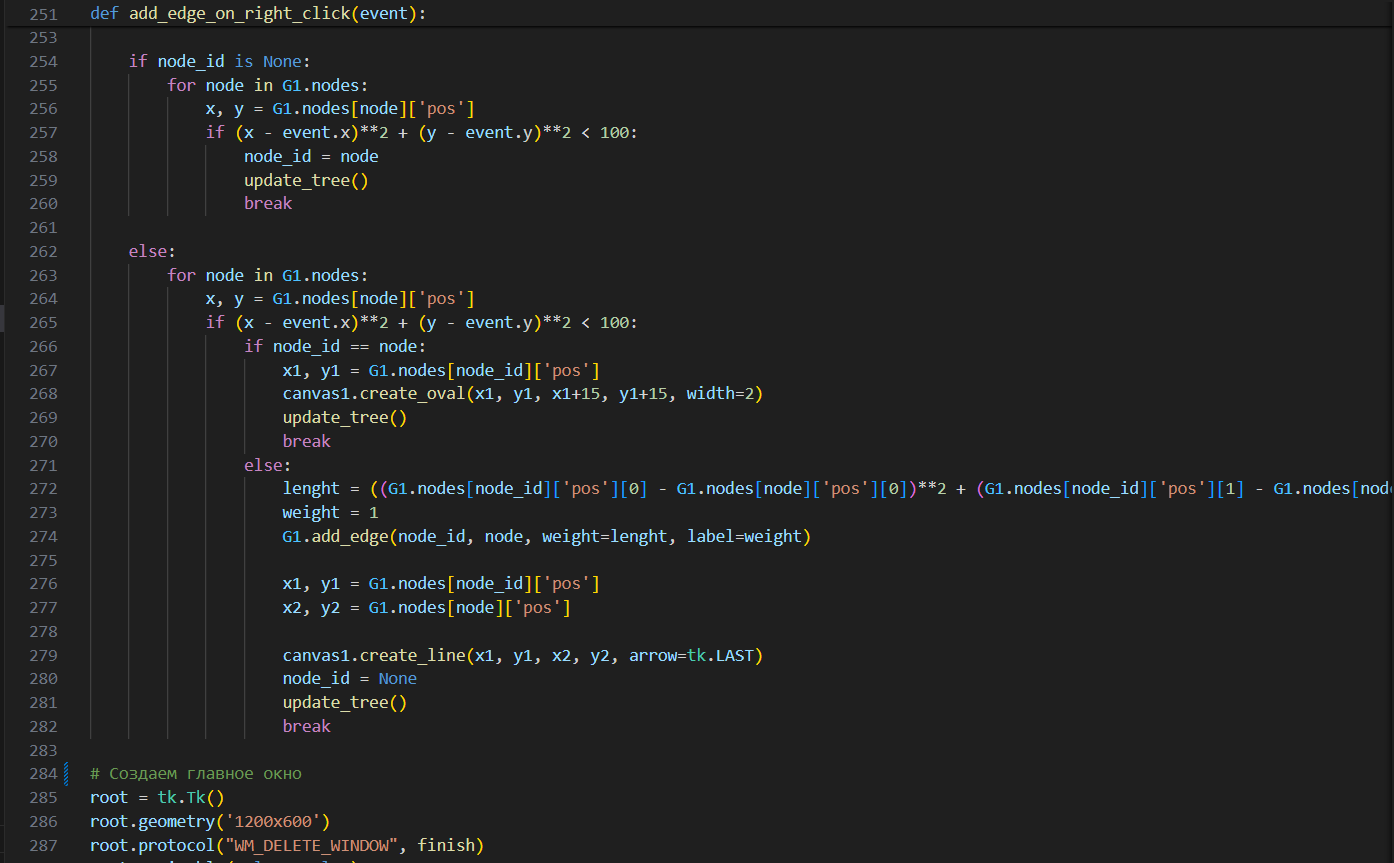
*Рис 6: листинг*



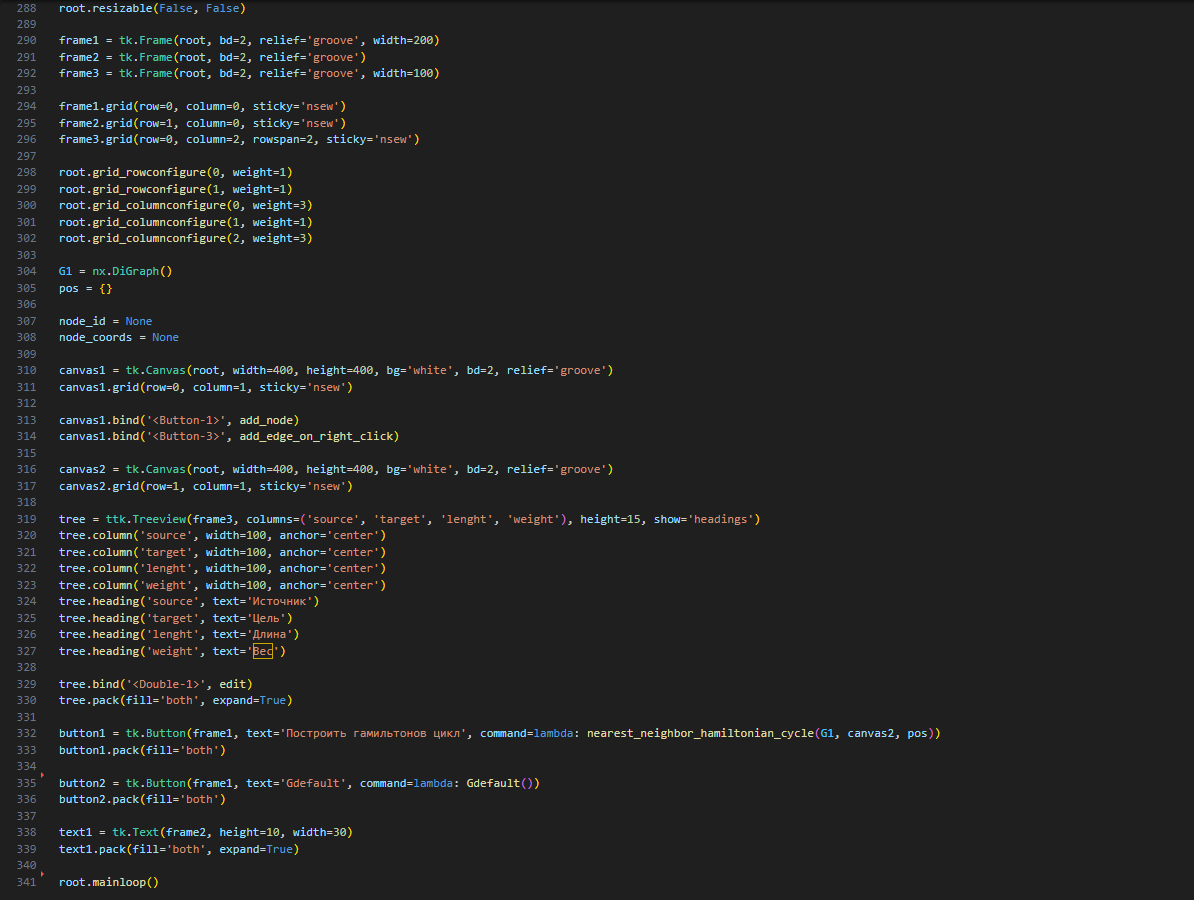
*Рис 7: листинг*



*Рис 8: листинг*



*Рис 9: листинг*



*Рис 10: листинг*