Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Дисциплина: Теория графов**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_К.Данил

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Крамаренко

**Тема работы: Проверка на существования графа**

**Ход работы:**

**Задача:**Написать программу проверки существования графа по заданным вершин.

**Решение задачи:**

1. Для написании программного кода был использован язык программирования Python, доп библиотеки для изображения графа networkx, matplotlib.

2. Для существования графа были использованы следующие теоремы и леммы:

1. **Теорема:** Сумма степеней должна быть >= кол-во степеней.

2. **Лемма** о рукопожатиях

3. **Алгоритм** Хавел-Хакими для возможности построения простого и полного графа

3. Графы которые были рассмотрены в ходе работы:

1. **«Несвязанный»**

2. **«Полный граф»**

3. **«Простой граф»**

**Ход решения поставленной задачи:**

Для существования любого графа была использована теорема **1**

Теорема **1**,**2**,**3** рассмотрены на рисунке **1**

Формула нахождения ребер полного графа рассмотрена на рисунке **2**.

Алгоритм Хавел-Хакими рассмотрен на рисунке **3**.

Для построения любого из рассматриваемых графов использовался алгоритм Хавел-Хакими.

Для этого необходимо добавить метку для каждой степени вершины рис. 4

При проверки степеней вершин удаляются все нулевые вершины

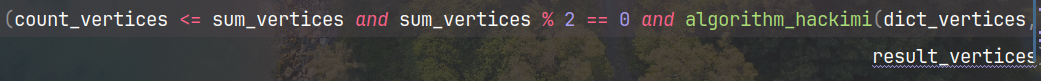
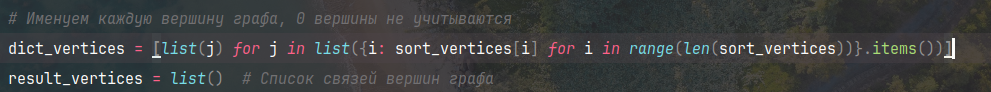
 рис. 1

 рис. 2

 рис. 3

 рис. 4

**Проверка на существования «Полного графа»**

Для проверки существования полного графа, была использована теорема 1, а сумма ребер в таком графе считалась по формуле n \* (n — 1) / 2. Если выполнялось условие из теоремы **1,2,3**, то такой граф являлся **«Полным»**, а значит он существует.

**Проверка на существования «Простого графа»**

Для проверки существования простого графа, был использован алгоритм Хавел-Хакими. Данный алгоритм рекурсивно соединяет вершины по правилу:

1. Сортируем список и проверяем нет ли там хотя бы одного отрицательного элемента, или все элементы списка равны 0. Если в списке существует отрицательный элемент, то простой граф построить нельзя, но если все элементы списка нулевые, то такой граф построить можно.

2. Удаляется последняя степени вершины в списке (она же и максимальная). Обозначим её как R.

3. Уменьшаем R степеней вершин списка на 1.

4. Делаем связь между удалённой вершиной и той у которой мы отняли 1

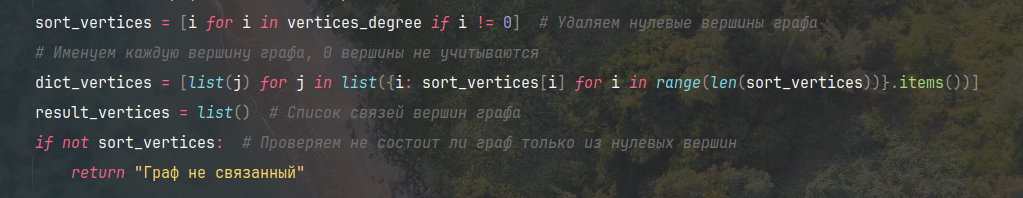
5. Записываем результат связи в список

6. Повторяем п.1, 2, 3 и так пока условие в п.1 не будет выполнено.

Если функция в котором выполняется данный алгоритм вернула False, то такой простой граф построить нельзя. В противном случае простой или полный граф существует, и его можно построить. Так же мы не учитываем 0 степени вершин которые присутствуют в нашем списке. Все пункты рассмотрены на рис.3

**Проверка на существования «Несвязанного графа»**

Если после удаления всех 0 вершин из списка степеней вершин, список степеней вершин будет пуст тогда такой граф является не связанным рис.5

 рис. 5

**Построение простого и полного графа по алгоритму Хавел-Хакими**

Для построения полного и простого графа использовался алгоритм Хавел-Хакими его принцип был рассмотрен в **Проверка на существования «Простого графа».**После того как отработала функция algorithm\_hackimi, тем самым сделая проверку о существовании простого графа и созданию связей между вершинами рис.8, создаётся класс графа G.

При помощи метода add\_edges\_from(), мы добавляем список связей в граф.

Рисуем наш по связанным вершинам при помощи метода nx.draw().

После при помощи метода show() выводим наш граф на экран

рис.6



рис. 6

**Примеры обработки степеней вершин графа:**

