МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Лабораторная работа №5**

**По дисциплине «Введение в цифровую культуру и программирование»**

**Работа с графами**

Выполнил студент группы: №М3110

Юрченко Владислав Витальевич

Проверил:

Хлопотов Максим Валерьевич

***САНКТ-ПЕТЕРБУРГ***

***2019***

**Ответы на вопросы:**

**1 ВОПРОС**

Всего ребер: 2419

**2 ВОПРОС**

Всего изолятов: 10

Сами изоляты: 9, 242, 268, 355, 380, 401, 525, 667, 751, 905,

**3 ВОПРОС**

Вершина с самой большой степенью - 155. Степень связности - 17.

**4 ВОПРОС**

Диаметр компоненты связности: 8

**5 ВОПРОС**

Найти путь от 807 до 216

Длина: 6

Путь: 807 -> 560 -> 99 -> 72 -> 193 -> 216

**6 ВОПРОС**

Найти путь от 463 до 908

Длина: 7

Путь: 463 -> 266 -> 563 -> 903 -> 796 -> 315 -> 908

**7 ВОПРОС**

Найти путь от 817 до 37

Длина: 7

Путь: 817 -> 869 -> 154 -> 139 -> 335 -> 399 -> 37

---УДАЛЕНИЕ---

**8 ВОПРОС**

Всего ребер: 2197

**9 ВОПРОС**

Всего изолятов: 29

Сами изоляты: 9, 221, 242, 268, 340, 355, 380, 401, 408, 425, 442, 525, 595, 629, 646, 663, 667, 680, 751, 799, 816, 833, 867, 901, 905, 918, 935, 952, 969,

**10 ВОПРОС**

Граф - 889. Степень связности - 12.

**11 ВОПРОС**

Диаметр компоненты связности: 10

**12 ВОПРОС**

Найти путь от 807 до 216

Длина: 7

Путь: 807 -> 411 -> 185 -> 158 -> 44 -> 114 -> 216

**13 ВОПРОС**

Найти путь от 463 до 908

Длина: 8

Путь: 463 -> 38 -> 588 -> 938 -> 76 -> 269 -> 315 -> 908

**14 ВОПРОС**

Найти путь от 817 до 37

Длина: 7

Путь: 817 -> 869 -> 154 -> 139 -> 335 -> 399 -> 37

**Код:**

import networkx as nx  
graph\_edges = open('graphedges61.txt')  
  
  
def find\_de\_way(graph, a, b):  
 print('Найти путь от ' + str(a) + ' до ' + str(b))  
 path = nx.shortest\_path(graph, a, b)  
 print('Длина: ' + str(len(path)))  
 print('Путь: ' + str(path[0]), end='')  
 for i in range(1, len(path)):  
 print(' -> ', end='')  
 print(str(path[i]), end='')  
 print()  
  
  
print('-----Подготовка-----')  
original\_graph = list()  
set\_graph = set()  
dic\_graph = dict()  
g = nx.Graph()  
  
for line in graph\_edges:  
 original\_graph.append(list(map(int, line.split())))  
  
for node in original\_graph:  
 set\_graph.add(node[0])  
 set\_graph.add(node[1])  
  
 for i in range(2):  
 if node[i] not in dic\_graph:  
 dic\_graph[node[i]] = 1  
 else:  
 dic\_graph[node[i]] += 1  
  
 g.add\_edge(node[0], node[1])  
  
print('Подготовились')  
print('-----Подготовка-----\n')  
  
print('\n1 ВОПРОС')  
print('Всего ребер: ' + str(len(original\_graph)))  
  
print('\n2 ВОПРОС')  
isolated = sorted(list({i for i in range(1000)} - set\_graph))  
print('Всего изолятов: ' + str(len(isolated)))  
print('Сами изоляты: ', end='')  
for isolate in isolated:  
 print(isolate, end=', ')  
print()  
  
print('\n3 ВОПРОС')  
maximum = 0  
t\_node = 'Нет такого графа'  
t\_value = 'Никакая'  
for i in set\_graph:  
 if dic\_graph.get(i) > maximum:  
 t\_node = i  
 t\_value = dic\_graph[i]  
 maximum = t\_value  
  
print('Вершина с самой большой степенью - ' + str(t\_node) + '. Степень связности - ' + str(t\_value) + '.')  
  
print('\n4 ВОПРОС')  
all\_subgraphs = []  
maximum = -1  
for component in list(nx.connected\_components(g)):  
 temp = g.subgraph(node for node in component)  
 all\_subgraphs.append(temp) # Все компоненты нашего графа  
for component in all\_subgraphs:  
 print('...')  
 t = nx.diameter(component)  
 if t > maximum:  
 maximum = t  
print('Диаметр компоненты связности: ' + str(maximum))  
  
print('\n5 ВОПРОС')  
find\_de\_way(g, 807, 216)  
  
print('\n6 ВОПРОС')  
find\_de\_way(g, 463, 908)  
  
print('\n7 ВОПРОС')  
find\_de\_way(g, 817, 37)  
  
print('\n---УДАЛЕНИЕ---')  
for node in original\_graph:  
 for i in range(2):  
 if (node[i] == 193 or node[i] == 903 or node[i] == 72 or node[i] == 266 or  
 node[i] == 139 or node[i] == 154 or node[i] == 155 or node[i] == 796 or  
 node[i] % 17 == 0):  
 original\_graph.remove(node)  
 break  
  
print('\n-----Подготовка-----')  
set\_graph = set()  
dic\_graph = dict()  
g = nx.Graph()  
  
for line in graph\_edges:  
 original\_graph.append(list(map(int, line.split())))  
  
for node in original\_graph:  
 set\_graph.add(node[0])  
 set\_graph.add(node[1])  
  
 for i in range(2):  
 if node[i] not in dic\_graph:  
 dic\_graph[node[i]] = 1  
 else:  
 dic\_graph[node[i]] += 1  
  
 g.add\_edge(node[0], node[1])  
  
print('Подготовились')  
print('-----Подготовка-----\n')  
  
print('\n8 ВОПРОС')  
print('Всего ребер: ' + str(len(original\_graph)))  
  
print('\n9 ВОПРОС')  
isolated = sorted(list({i for i in range(1000)} - set\_graph))  
print('Всего изолятов: ' + str(len(isolated)))  
print('Сами изоляты: ', end='')  
for isolate in isolated:  
 print(isolate, end=', ')  
print()  
  
print('\n10 ВОПРОС')  
maximum = 0  
t\_node = 'Нет такого графа'  
t\_value = 'Никакая'  
for i in set\_graph:  
 if dic\_graph.get(i) > maximum:  
 t\_node = i  
 t\_value = dic\_graph[i]  
 maximum = t\_value  
  
print('Граф - ' + str(t\_node) + '. Степень связности - ' + str(t\_value) + '.')  
  
print('\n11 ВОПРОС')  
all\_subgraphs = []  
maximum = -1  
for component in list(nx.connected\_components(g)):  
 temp = g.subgraph(node for node in component)  
 all\_subgraphs.append(temp) # Все компоненты нашего графа  
for component in all\_subgraphs:  
 print('...')  
 t = nx.diameter(component)  
 if t > maximum:  
 maximum = t  
print('Диаметр компоненты связности: ' + str(maximum))  
  
print('\n12 ВОПРОС')  
find\_de\_way(g, 807, 216)  
  
print('\n13 ВОПРОС')  
find\_de\_way(g, 463, 908)  
  
print('\n14 ВОПРОС')  
find\_de\_way(g, 817, 37)