|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**ЛАБОРАТОРНАЯ РАБОТА**

**«Представление графов и операции над ними»**

**ДИСЦИПЛИНА: «Дискретная математика»**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-31Б | |  |  | ( | Суриков Н. С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Никитенко У. В. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

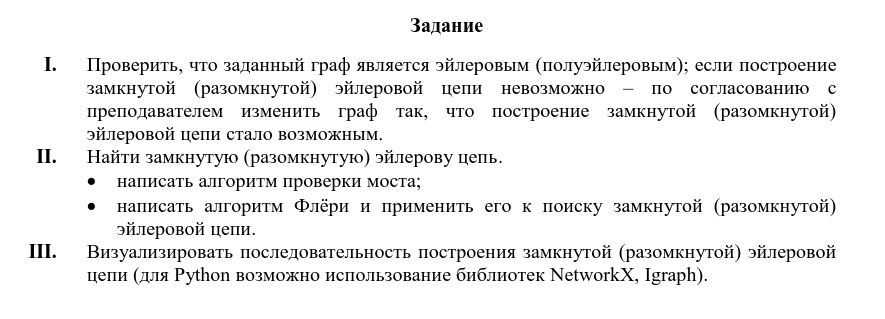
|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

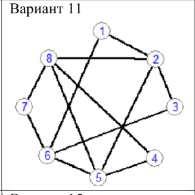
**Цель:** закрепить умения в нахождении эйлеровых циклов

**Задачи:**

1. Hазработать приложение построения замкнутой (разомкнутой) эйлеровой цепи.
2. Используя алгоритм Флёри, визуализировать этапы построения.

**Вариант 11**





**Листинг программы:**

1 import numpy as np

2 import networkx as nx

3 import matplotlib.pyplot as plt

4 import time

5

6

7 class Graph:

8 def \_\_init\_\_(self, adjacency\_matrix):

9 self.adjacency\_matrix = np.array(adjacency\_matrix)

10 self.num\_vertices = len(adjacency\_matrix)

11

12 def is\_eulerian(self):

13 odd\_count = 0

14 for i in range(self.num\_vertices):

15 degree = sum(self.adjacency\_matrix[i])

16 if degree % 2 != 0:

17 odd\_count += 1

18 if odd\_count == 0:

19 return "Замкнутый эйлеров граф"

20 elif odd\_count == 2:

21 return "Полуэйлеров граф"

22 else:

23 return "Не является эйлеровым графом"

24

25 def find\_eulerian\_circuit(self, start\_vertex=None):

26 if self.is\_eulerian() == "Не является эйлеровым графом":

27 return []

28

29 temp\_matrix = self.adjacency\_matrix.copy()

30 eulerian\_circuit = []

31 current\_vertex = start\_vertex if start\_vertex is not None else 0

32

33 stack = [current\_vertex]

34 while stack:

35 current\_vertex = stack[-1]

36 next\_vertex = self.find\_next\_vertex(current\_vertex, temp\_matrix)

37 if next\_vertex is None:

38 eulerian\_circuit.append(current\_vertex + 1)

39 stack.pop()

40 else:

41 temp\_matrix[current\_vertex][next\_vertex] = 0

42 temp\_matrix[next\_vertex][current\_vertex] = 0

43 stack.append(next\_vertex)

44

45 if eulerian\_circuit[0] != eulerian\_circuit[-1]:

46 eulerian\_circuit.append(eulerian\_circuit[0])

47

48 return eulerian\_circuit

49

50 def find\_next\_vertex(self, vertex, temp\_matrix):

51 for next\_vertex in range(self.num\_vertices):

52 if temp\_matrix[vertex][next\_vertex] > 0:

53 return next\_vertex

54 return None

55

56 def visualize\_eulerian\_circuit(self, path):

57 G = nx.Graph(self.adjacency\_matrix)

58 pos = {

59 (2, 3),

60 (2.5, 2),

61 (2.75, 0.75),

62 (2.5, -0.05),

63 (2, -1),

64 (1.5, -0.3),

65 (1.25, 0.75),

66 (1.5, 2),

67 }

68 labels = {

69 i: str(i + 1) for i in range(self.num\_vertices)

70 }

71

72 fig, ax = plt.subplots(figsize=(10, 10))

73 nx.draw(

74 G,

75 pos,

76 with\_labels=True,

77 labels=labels,

78 node\_color="lightblue",

79 node\_size=700,

80 font\_size=15,

81 ax=ax,

82 )

83

84 for i in range(len(path)):

85 path\_edges = [(path[i] - 1, path[(i + 1) % len(path)] - 1)]

86 nx.draw\_networkx\_edges(

87 G, pos, edgelist=path\_edges, edge\_color="red", width=4, ax=ax

88 )

89 plt.draw()

90 plt.pause(0.5)

91 nx.draw(

92 G,

93 pos,

94 with\_labels=True,

95 labels=labels,

96 node\_color="lightblue",

97 node\_size=700,

98 font\_size=15,

99 ax=ax,

100 )

101

102 plt.title("Эйлерова цепь")

103 plt.show()

104

105

106 if \_\_name\_\_ == "\_\_main\_\_":

107 adjacency\_matrix = [

108 [0, 1, 0, 0, 0, 1, 0, 0],

109 [1, 0, 1, 0, 1, 0, 0, 1],

110 [0, 1, 0, 0, 0, 1, 0, 0],

111 [0, 0, 0, 0, 1, 0, 0, 1],

112 [0, 1, 0, 1, 0, 1, 0, 1],

113 [1, 0, 1, 0, 1, 0, 1, 0],

114 [0, 0, 0, 0, 0, 1, 0, 1],

115 [0, 1, 0, 1, 1, 0, 1, 0],

116 ]

117

118 graph = Graph(adjacency\_matrix)

119

120 print(graph.is\_eulerian())

121

122 if graph.is\_eulerian() != "Не является эйлеровым графом":

123 eulerian\_circuit = graph.find\_eulerian\_circuit()

124

125 print("Эйлерова цепь:", eulerian\_circuit)

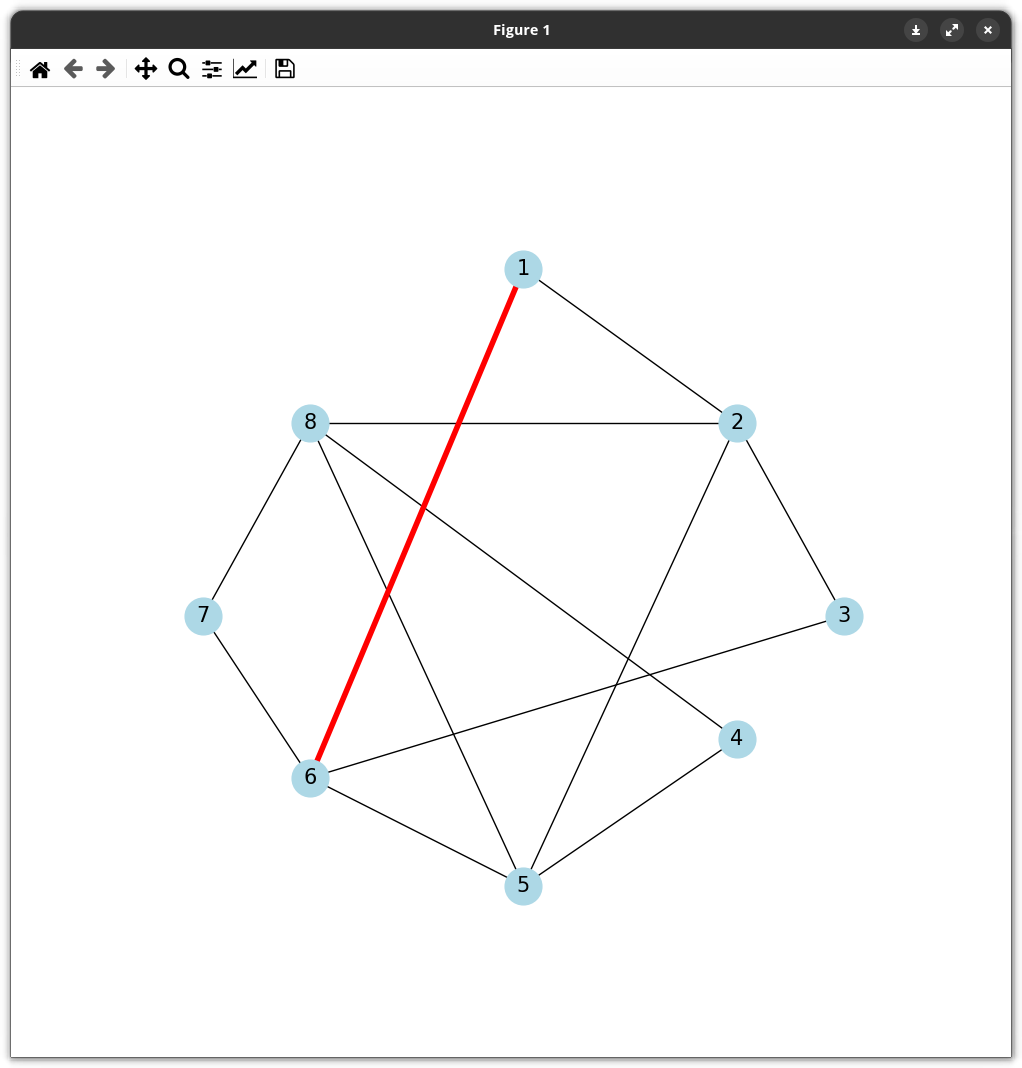
126

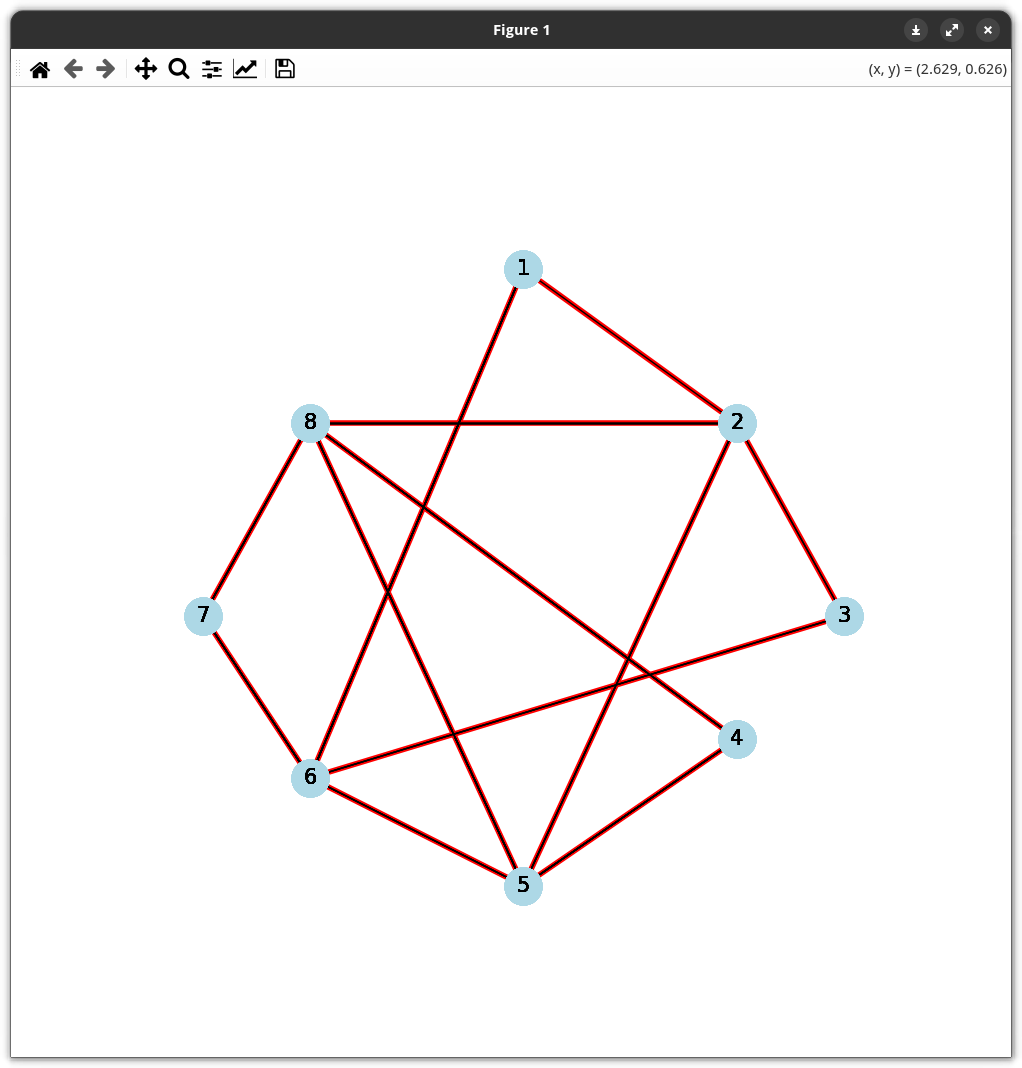
127 graph.visualize\_eulerian\_circuit(eulerian\_circuit)

128 else:

129 print("Невозможно найти эйлерову цепь.")

**Результат работы:**





*В результате мы получаем последовательное построение эйлеровой цепи на экране, и её текстовой отображение в консоли.*

**Вывод:** В ходе работы были закреплены умения в нахождении эйлеровых циклов.