

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра КСУП

Отчет по лабораторной работе
по дисциплине «Дискретная математика»
Тема: «Генерация графа»

Студент гр. 582-1

Полушвайко Константин Николаевич

___ ноября 2023 г.

1. Задание

1. Сгенерировать матрицу смежности произвольной задаваемой размерности n .
2. Генерация должна предусматривать возможность (по выбору) генерации полных графов, графом с кратными ребрами и петлями.
3. Сделать отрисовку графа с помощью NetworkX (или подобные библиотеки)
4. Реализовать функцию преобразования сгенерированной в 1 матрицы смежности в матрицу инцидентности.

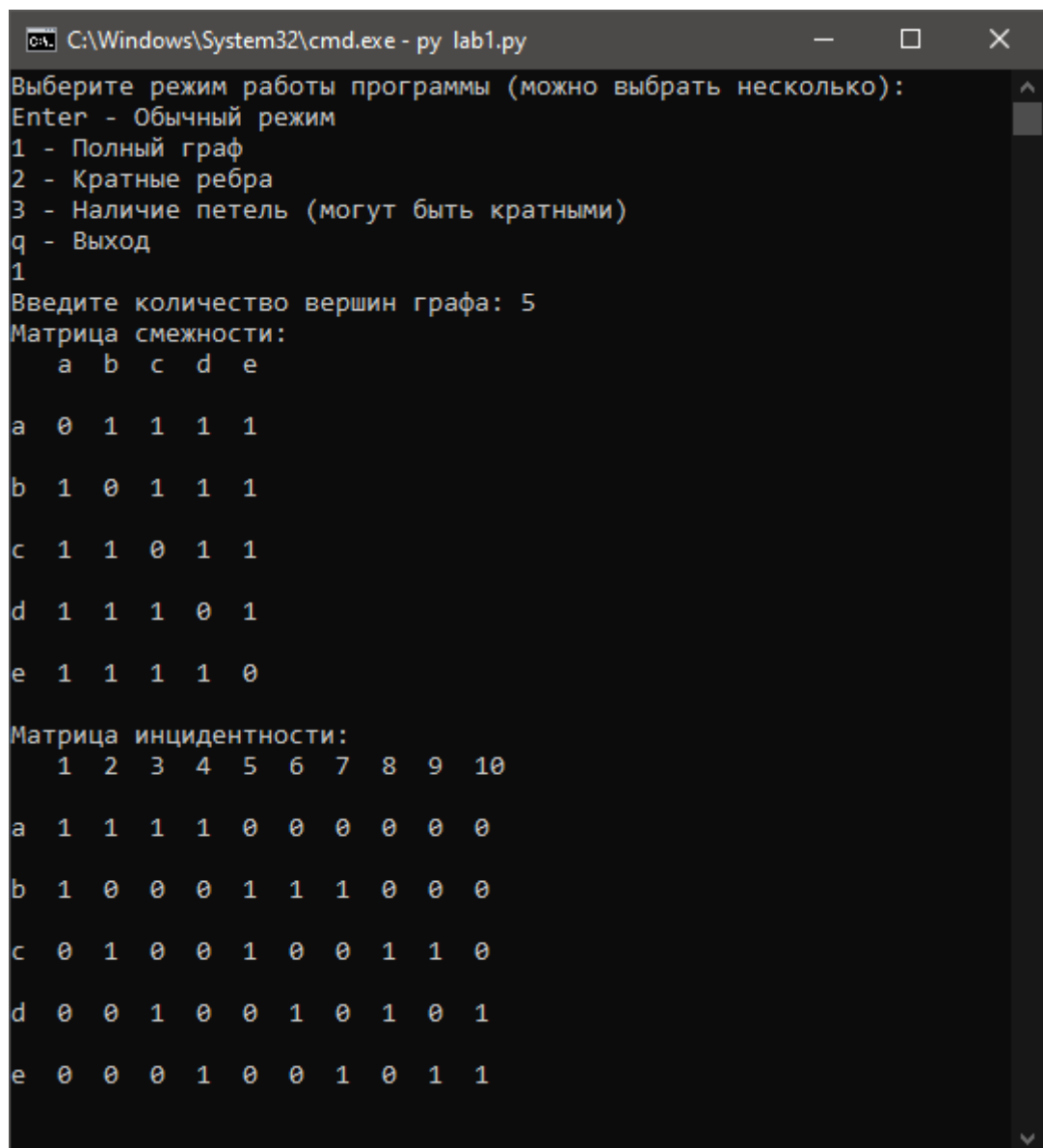
2. Ход работы

В начале, определим язык и структуру кода: будем использовать язык python и придерживаться парадигмы ООП. Создадим класс Graph и определим поля для массива смежности, инцидентности, разных режимов работы с графом и количества вершин и ребер.

Определим методы для класса: конструктор, заполнение матрицы смежности случайным образом, вывод матрицы смежности и инцидентности, вывод графа, создание матрицы инцидентности, подсчет ребер графа, а также определим главную функцию (исполняет роль менеджера-меню).

Конечный код программы приведен в листинге (пункт 3).

На рисунках 2.1-4 представлены примеры работы конечной программы.



```
C:\Windows\System32\cmd.exe - py lab1.py
Выберите режим работы программы (можно выбрать несколько):
Enter - Обычный режим
1 - Полный граф
2 - Кратные ребра
3 - Наличие петель (могут быть кратными)
q - Выход
1
Введите количество вершин графа: 5
Матрица смежности:
  a b c d e
a 0 1 1 1 1
b 1 0 1 1 1
c 1 1 0 1 1
d 1 1 1 0 1
e 1 1 1 1 0
Матрица инцидентности:
  1 2 3 4 5 6 7 8 9 10
a 1 1 1 1 0 0 0 0 0 0
b 1 0 0 0 1 1 1 0 0 0
c 0 1 0 0 1 0 0 1 1 0
d 0 0 1 0 0 1 0 1 0 1
e 0 0 0 1 0 0 1 0 1 1
```

Рисунок 2.1 – Первый запуск программы

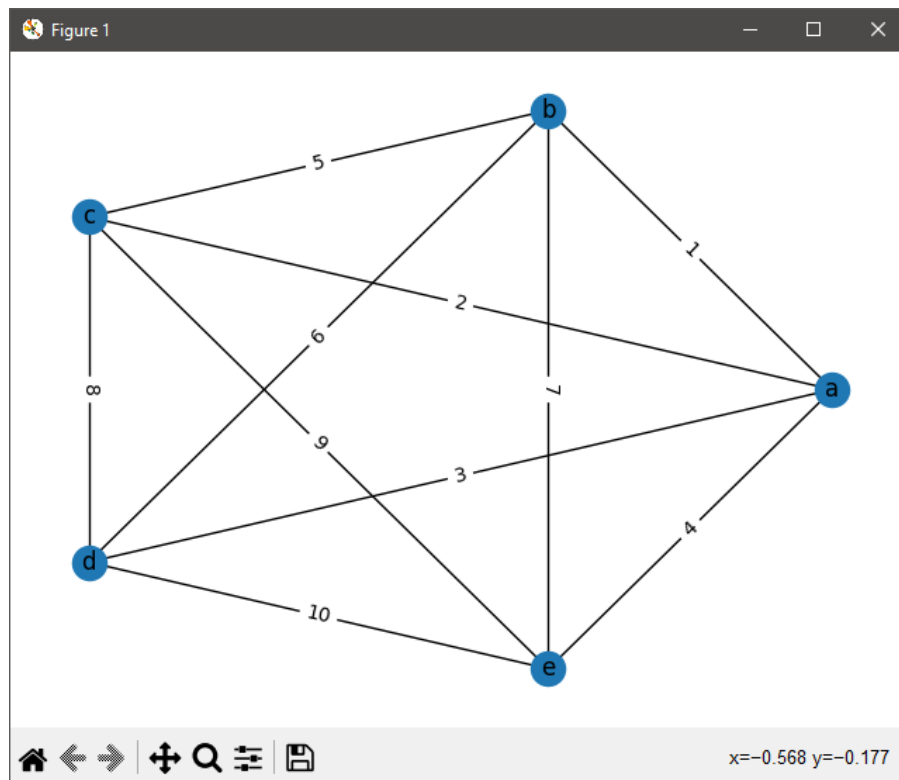


Рисунок 2.2 – Первый граф

```

C:\Windows\System32\cmd.exe - py lab1.py
Выберите режим работы программы (можно выбрать несколько):
Enter - Обычный режим
1 - Полный граф
2 - Кратные ребра
3 - Наличие петель (могут быть кратными)
q - Выход
23
Введите количество вершин графа: 3
Матрица смежности:
  a  b  c
a  2  2  3
b  2  2  0
c  3  0  3

Матрица инцидентности:
  1  2  3  4  5  6  7  8  9  10  11  12
a  2  2  1  1  1  1  1  0  0  0  0  0
b  0  0  1  1  0  0  0  2  2  0  0  0
c  0  0  0  0  1  1  1  0  0  2  2  2
  
```

Рисунок 2.3 – Второй запуск программы

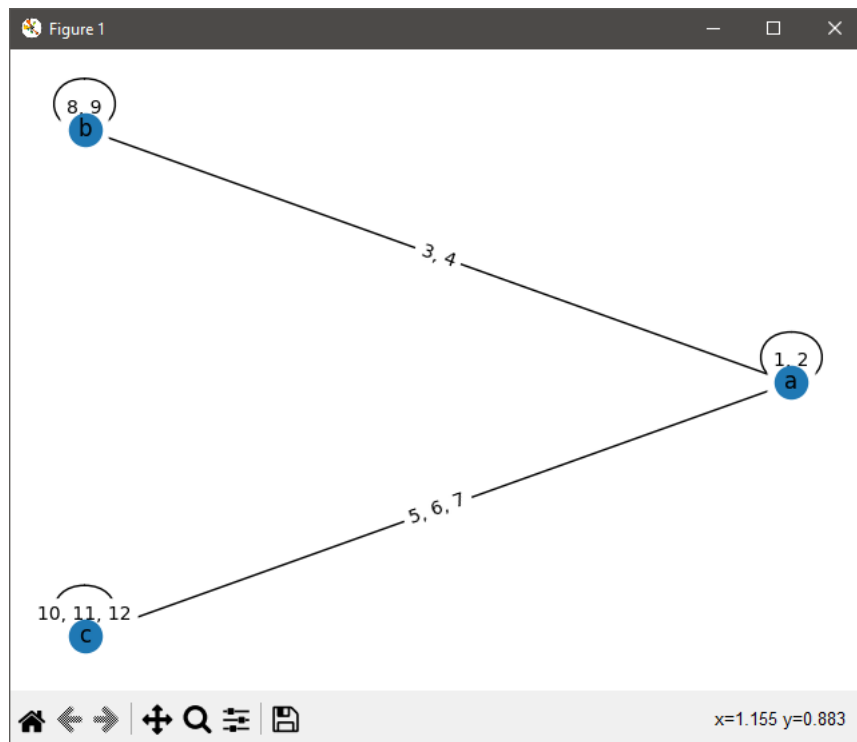


Рисунок 2.4 – Второй граф

3. Листинг

```
1: import networkx as nx
2: import matplotlib.pyplot as plt
3: import numpy as np
4: import random
5: import os
6:
7: AINDEX = 97
8:
9: class Graph:
10:     # Конструктор класса
11:     def __init__(self, size, bFullGraph = False, bMultiedge = False, bLoop = False):
12:         self._nodes = size
13:         self._bFullGraph = bFullGraph
14:         self._bMultiedge = bMultiedge
15:         self._bLoop = bLoop
16:         self._imatix = None
17:         self._amatix = list()
18:         for i in range(self._nodes):
19:             self._amatix.append(list())
20:             for j in range(self._nodes):
21:                 self._amatix[i].append(0)
22:
23:     # Вывод матрицы смежности
24:     def showAdjacencyMatrix(self):
25:         print("Матрица смежности: ")
26:         for i in range(self._nodes * 2 + 1):
27:             for j in range(self._nodes * 2 + 1):
28:                 if j % 2 == 1:
29:                     print(end = " ") # |
30:                 elif i % 2 == 1:
31:                     print(end = " ") # -
32:                 elif (i == 0 and j != 0) or (j == 0 and i != 0):
33:                     print(end = f"{chr(AINDEX + (i + j) // 2 - 1)}")
34:                 elif i // 2 > 0 and j // 2 > 0:
35:                     print(end = f"{self._amatix[i // 2 - 1][j // 2 - 1]}")
36:                 else:
37:                     print(end = " ")
38:             print()
39:
40:     # Вывод графа (при помощи networkx)
41:     def showGraph(self):
42:         nodeMap = dict()
43:         edgeMap = dict()
44:         loopMap = dict()
45:         for i in range(0, self._nodes):
46:             nodeMap.update({i: chr(AINDEX + i)})
47:
48:         count = 1
49:         for i in range(self._nodes):
50:             for j in range(i, self._nodes):
51:                 if self._amatix[i][j] != 0:
52:                     edgeName = ""
53:                     if self._amatix[i][j] > 1:
54:                         for k in range(self._amatix[i][j] - 1):
55:                             edgeName += f"{count}, "
56:                             count += 1
57:                         edgeName += f"{count}"
58:                         count += 1
59:                     else:
60:                         edgeName = f"{count}"
61:                         count += 1
62:                     if i == j:
63:                         edgeName += "\n\n"
64:                         loopMap.update({(chr(AINDEX + i), chr(AINDEX + j)): edgeName})
65:                     else:
66:                         edgeMap.update({(chr(AINDEX + i), chr(AINDEX + j)): edgeName})
```

```

67:
68:     G = nx.Graph(np.array(self._amatrix))
69:     nx.relabel_nodes(G, nodeMap, False)
70:     pos = nx.circular_layout(G)
71:     nx.draw(G, pos, with_labels = True)
72:     nx.draw_networkx_edge_labels(G, pos, edge_labels = edgeMap)
73:     nx.draw_networkx_edge_labels(G, pos, edge_labels = loopMap)
74:     plt.show()
75:
76:     # Заполнение матрицы смежности при помощи рандома
77:     def setRandomMatrix(self):
78:         deltaIndex = 0 if self._bLoop else 1
79:         minEdges = 1 if self._bFullGraph else 0
80:         maxEdges = 3 if self._bMultiedge else 1
81:
82:         for i in range(0, self._nodes):
83:             for j in range(i + deltaIndex, self._nodes):
84:                 self._amatrix[i][j] = self._amatrix[j][i] = random.randint(minEdges,
maxEdges)
85:
86:     # Подсчет ребер графа
87:     def updateEdges(self):
88:         edges = 0
89:         for i in range(self._nodes):
90:             for j in range(i, self._nodes):
91:                 edges += self._amatrix[i][j]
92:         self._edges = edges
93:
94:     # Создание матрицы инцидентности
95:     def makeIncidenceMatrix(self):
96:         self.updateEdges()
97:
98:         self._imatrix = list()
99:         for i in range(self._nodes):
100:             self._imatrix.append(list())
101:             for j in range(self._edges):
102:                 self._imatrix[i].append(0)
103:
104:         edgeIndex = 0
105:         for i in range(self._nodes):
106:             for j in range(i, self._nodes):
107:                 if self._amatrix[i][j] != 0:
108:                     value = 1
109:                     if i == j:
110:                         value = 2
111:
112:                     for k in range(self._amatrix[i][j]):
113:                         self._imatrix[i][edgeIndex] = self._imatrix[j][edgeIndex] =
value
114:                         edgeIndex += 1
115:
116:     # Вывод матрицы инцидентности
117:     def showIncidenceMatrix(self):
118:         if self._imatrix is None:
119:             self.makeIncidenceMatrix()
120:         print("Матрица инцидентности: ")
121:         for i in range(self._nodes * 2 + 1):
122:             for j in range(self._edges * 2 + 1):
123:                 if j % 2 == 1:
124:                     print(end = " ") # |
125:                 elif i % 2 == 1:
126:                     print(end = " ") # -
127:                 elif j == 0 and i != 0:
128:                     print(end = f"{chr(AINDEX + (i + j) // 2 - 1)}")
129:                 elif i == 0 and j != 0:
130:                     print(end = f"{j // 2}")
131:                 elif i // 2 > 0 and j // 2 > 0:
132:                     print(end = f"{self._imatrix[i // 2 - 1][j // 2 - 1]}")

```

```

133:             if j >= 20:
134:                 print(end = " ")
135:             else:
136:                 print(end = " ")
137:         print()
138:
139:
140: def main():
141:     menu = "Выберите режим работы программы (можно выбрать несколько):\n"
142:     menu += "Enter - Обычный режим\n1 - Полный граф\n2 - Кратные ребра\n3 - Наличие\nпетель (могут быть кратными)\nq - Выход\n"
143:
144:     mode = input(menu)
145:
146:     while ('q' not in mode):
147:         NodeNumber = int(input("Введите количество вершин графа: "))
148:         graph = Graph(NodeNumber, '1' in mode, '2' in mode, '3' in mode)
149:         graph.setRandomMatrix()
150:         graph.showAdjacencyMatrix()
151:         print()
152:         graph.showIncidenceMatrix()
153:         graph.showGraph()
154:         os.system("cls")
155:         mode = input(menu)
156:     os.system("cls")
157:
158:
159:
160: if __name__ == "__main__":
161:     main()

```


4. Заключение

В ходе выполнения лабораторной работы изучили матрицы смежности, инцидентности, графы на практике. Научились строить графы в python, используя библиотеку networkx, а также переводить матрицу смежности в матрицу инцидентности.

Затраченное время:

- 6 часов на написание конечного кода (с учетом рефакторинга);
- 1 час на отчет.

Итого: затрачено на работу 7 часов.

Больше всего времени заняла индексация ребер и вершин, и их правильный вывод в матрицах и особенно в графе, так как пришлось обращаться к документации библиотеки network, а также немного к numpy и matplotlib.