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: «Метрики графа»

Студент гр. 582-1

Полушвайко Константин Николаевич

\_\_ декабря 2023 г.

2023

## 1. Задание

1. Для сгенерированного графа (из 1 лабораторной работы) реализовать функцию вывода матрицы метрик;
2. Реализовать по матрице метрик функцию нахождения радиуса, диаметра, центральных и периферийных вершин;
3. Для возведения матрицы в степень лучше реализовать функцию умножения матриц (это надо для поиска матрицы метрик)

## 2. Ход работы

Для начала работы определим алгоритм нахождения матрицы метрики:

1. Задаем матрицу метрики µ = (mij) размерности равной размерности матрицы смежности с неопределенными элементами;
2. Создаем матрицу S = R + E, где R – матрица смежности, E – единичная матрица. Элементам главной диагонали матрицы метрики приравниваем 0, так как нет расстояния между одной и той же точкой;
3. Начальное значение степени k = 1. Всем элементам mij, которые неопределенны и выполняется условие sij ≠ 0 в S­­­k приравнивается k, т.е. mij = k;
4. k += 1;
5. Проверяем матрицу метрики на устойчивость: µn == µn+1, где n – это номер итерации в алгоритме. Если матрица неустойчива, то возвращаемся к шагу 3;
6. Всем элементам mij­, которые неопределенны: mij­ = ∞.

Данный алгоритм выполняется в методе MakeMetricMatrix, который можно найти в листинге (пункт 3).

Для нахождения радиуса и диаметра графа пользуемся матрицей метрики: ищем максимумы каждой строки, затем из этих значений ищем минимум – радиус и максимум – диаметр. Если максимум строки матрицы метрики соответствующей вершины равен радиусу – то это центральная вершина, если равна диаметру – то это периферийная вершина. Реализация нахождения этих характеристик представлена в методе FindMetrics.

Конечный код программы приведен в листинге (пункт 3).

На рисунках 2.1 и 2.2 представлен пример работы конечной программы.

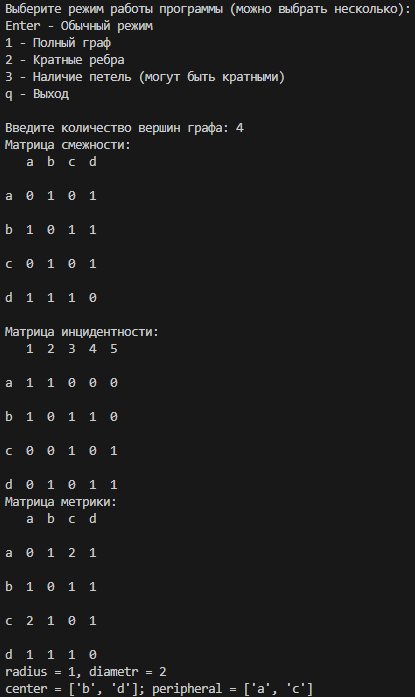


Рисунок 2.1 – Меню и вывод программы

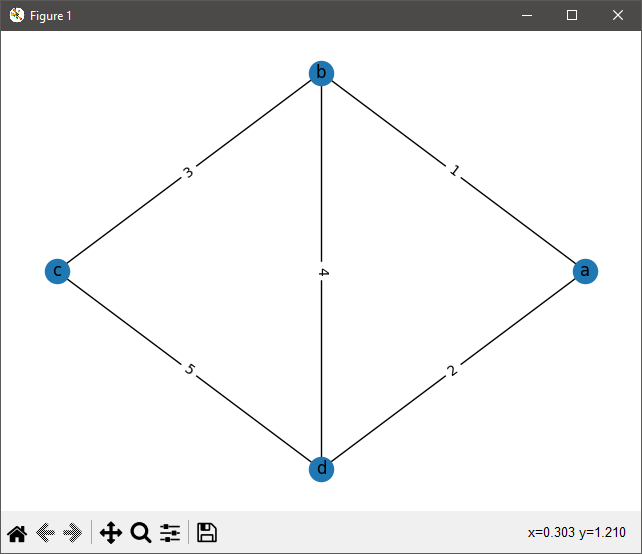


Рисунок 2.2 – Визуализация графа

## 3. Листинг

1. import networkx as nx
2. import matplotlib.pyplot as plt
3. import matplotlib as mpl
4. import numpy as np
5. import random
6. import math
7. import os
8. AINDEX = 97
9. # Создание матрицы размером (n x n)
10. def MakeMatrix(n):
11. matrix = list()
12. for i in range(n):
13. matrix.append(list())
14. for j in range(n):
15. matrix[i].append(0)
16. return matrix
17. # Возведение матрицы в степень
18. def PowerMatrix(matrix, n):
19. newMatrix = MakeMatrix(n)
20. for row in range(n):
21. for col in range(n):
22. for i in range(n):
23. newMatrix[row][col] +=  matrix[row][i] \* matrix[i][col]
25. return newMatrix
26. def EqualMatrix(a, b, n):
27. if a == None or b == None:
28. return False
29. for i in range(n):
30. for j in range(n):
31. if a[i][j] != b[i][j]:
32. return False
33. return True
34. def CopyMatrix(matrix, n):
35. newMatrix = MakeMatrix(n)
36. for i in range(n):
37. for j in range(n):
38. newMatrix[i][j] = matrix[i][j]
39. return newMatrix
40. class Graph:
41. # Конструктор класса
42. def \_\_init\_\_(self, size, bFullGraph = False, bMultiedge = False, bLoop = False):
43. self.\_nodes = size
44. self.\_bFullGraph = bFullGraph
45. self.\_bMultiedge = bMultiedge
46. self.\_bLoop = bLoop
47. self.\_imatrix = None
48. self.\_amatrix = MakeMatrix(self.\_nodes)

51. # Вывод матрицы смежности
52. def showAdjacencyMatrix(self):
53. print("Матрица смежности: ")
54. for i in range(self.\_nodes \* 2 + 1):
55. for j in range(self.\_nodes \* 2 + 1):
56. if j % 2 == 1:
57. print(end = "  ") # |
58. elif i % 2 == 1:
59. print(end = " ") # -
60. elif (i == 0 and j != 0) or (j == 0 and i != 0):
61. print(end = f"{chr(AINDEX + (i + j) // 2 - 1)}")
62. elif i // 2 > 0 and j // 2 > 0:
63. print(end = f"{self.\_amatrix[i // 2 - 1][j // 2 - 1]}")
64. else:
65. print(end = " ")
66. print()
68. # Вывод матрицы смежности
69. def showMetricMatrix(self):
70. print("Матрица метрики: ")
71. for i in range(self.\_nodes \* 2 + 1):
72. for j in range(self.\_nodes \* 2 + 1):
73. if j % 2 == 1:
74. print(end = "  ") # |
75. elif i % 2 == 1:
76. print(end = " ") # -
77. elif (i == 0 and j != 0) or (j == 0 and i != 0):
78. print(end = f"{chr(AINDEX + (i + j) // 2 - 1)}")
79. elif i // 2 > 0 and j // 2 > 0:
80. print(end = f"{self.\_mmatrix[i // 2 - 1][j // 2 - 1]}")
81. else:
82. print(end = " ")
83. print()
84. # Вывод графа (при помощи networkx)
85. def showGraph(self):
86. nodeMap = dict()
87. edgeMap = dict()
88. loopMap = dict()
89. for i in range(0, self.\_nodes):
90. nodeMap.update({i: chr(AINDEX + i)})
91. count = 1
92. for i in range(self.\_nodes):
93. for j in range(i, self.\_nodes):
94. if self.\_amatrix[i][j] != 0:
95. edgeName = ""
96. if self.\_amatrix[i][j] > 1:
97. for k in range(self.\_amatrix[i][j] - 1):
98. edgeName += f"{count}, "
99. count += 1
100. edgeName += f"{count}"
101. count += 1
102. else:
103. edgeName = f"{count}"
104. count += 1
105. if i == j:
106. edgeName += "\n\n"
107. loopMap.update({(chr(AINDEX + i), chr(AINDEX + j)): edgeName})
108. else:
109. edgeMap.update({(chr(AINDEX + i), chr(AINDEX + j)): edgeName})
110. G = nx.Graph(np.array(self.\_amatrix))
111. nx.relabel\_nodes(G, nodeMap, False)
112. pos = nx.circular\_layout(G)
113. nx.draw(G, pos, with\_labels = True)
114. nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels = edgeMap)
115. nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels = loopMap)
116. plt.show()
117. # Заполнение таблицы смежности при помощи рандома
118. def setRandomMatrix(self):
119. deltaIndex = 0 if self.\_bLoop else 1
120. minEdges = 1 if self.\_bFullGraph else 0
121. maxEdges = 3 if self.\_bMultiedge else 1
122. for i in range(0, self.\_nodes):
123. for j in range(i + deltaIndex, self.\_nodes):
124. value = random.randint(1, maxEdges) if random.randint(minEdges, 1) == 1 else 0
125. self.\_amatrix[i][j] = self.\_amatrix[j][i] = value
127. # Подсчет ребер графа
128. def updateEdges(self):
129. edges = 0
130. for i in range(self.\_nodes):
131. for j in range(i, self.\_nodes):
132. edges += self.\_amatrix[i][j]
133. self.\_edges = edges
135. # Создание матрицы инцидентности
136. def makeIncidenceMatrix(self):
137. self.updateEdges()
138. self.\_imatrix = list()
139. for i in range(self.\_nodes):
140. self.\_imatrix.append(list())
141. for j in range(self.\_edges):
142. self.\_imatrix[i].append(0)
144. edgeIndex = 0
145. for i in range(self.\_nodes):
146. for j in range(i, self.\_nodes):
147. if self.\_amatrix[i][j] != 0:
148. value = 1
149. if i == j:
150. value = 2
152. for k in range(self.\_amatrix[i][j]):
153. self.\_imatrix[i][edgeIndex] = self.\_imatrix[j][edgeIndex] = value
154. edgeIndex += 1
155. # Вывод матрицы инцидентности
156. def showIncidenceMatrix(self):
157. if self.\_imatrix is None:
158. self.makeIncidenceMatrix()
159. print("Матрица инцидентности: ")
160. for i in range(self.\_nodes \* 2 + 1):
161. for j in range(self.\_edges \* 2 + 1):
162. if j % 2 == 1:
163. print(end = "  ") # |
164. elif i % 2 == 1:
165. print(end = " ") # -
166. elif j == 0 and i != 0:
167. print(end = f"{chr(AINDEX + (i + j) // 2 - 1)}")
168. elif i == 0 and j != 0:
169. print(end = f"{j // 2}")
170. elif i // 2 > 0 and j // 2 > 0:
171. print(end = f"{self.\_imatrix[i // 2 - 1][j // 2 - 1]}")
172. if j >= 20:
173. print(end = " ")
174. else:
175. print(end = " ")
176. print()
178. def MakeMetricMatrix(self):
179. self.\_mmatrix = None
180. tempMatrix = MakeMatrix(self.\_nodes)
181. smatrix = MakeMatrix(self.\_nodes)
182. k = 1
183. for i in range(self.\_nodes):
184. for j in range(self.\_nodes):
185. smatrix[i][j] = 1 if self.\_amatrix[i][j] != 0 else 0
186. if i == j:
187. smatrix[i][j] += 1
188. while not EqualMatrix(self.\_mmatrix, tempMatrix, self.\_nodes):
189. self.\_mmatrix = CopyMatrix(tempMatrix, self.\_nodes)
190. for i in range(self.\_nodes):
191. for j in range(self.\_nodes):
192. if (not i==j and smatrix[i][j] != 0 and tempMatrix[i][j] == 0):
193. tempMatrix[i][j] += k
194. smatrix = PowerMatrix(smatrix, self.\_nodes)
195. k+=1
197. def FindMetrics(self):
198. maxRow = list()
199. for i in range(self.\_nodes):
200. maxRow.append(max(self.\_mmatrix[i]))
201. self.radius = min(maxRow) if min(maxRow) != 0 else math.inf
202. self.diametr = max(maxRow) if self.radius != math.inf else math.inf
203. print (f"radius = {self.radius}, diametr = {self.diametr}")
204. self.peripheral = list()
205. self.central = list()
206. for i in range(self.\_nodes):
207. if max(self.\_mmatrix[i]) == self.radius:
208. self.central.append(chr(AINDEX + i))
209. if max(self.\_mmatrix[i]) == self.diametr:
210. self.peripheral.append(chr(AINDEX + i))
211. print (f"center = {self.central}; peripheral = {self.peripheral}")
212. def main():
213. menu = "Выберите режим работы программы (можно выбрать несколько):\n"
214. menu += "Enter - Обычный режим\n1 - Полный граф\n2 - Кратные ребра\n3 - Наличие петель (могут быть кратными)\nq - Выход\n"
215. mode = input(menu)
216. while ('q' not in mode):
217. NodeNumber = int(input("Введите количество вершин графа: "))
218. graph = Graph(NodeNumber, '1' in mode, '2' in mode, '3' in mode)
219. graph.setRandomMatrix()
220. graph.showAdjacencyMatrix()
221. print()
222. graph.showIncidenceMatrix()
223. graph.MakeMetricMatrix()
224. graph.showMetricMatrix()
225. graph.FindMetrics()
226. graph.showGraph()
227. os.system("cls")
228. mode = input(menu)
229. os.system("cls")
230. if \_\_name\_\_ == "\_\_main\_\_":
231. main()

## 4. Заключение

В ходе выполнения лабораторной работы изучили алгоритм создания матрицы метрики, получили периферийные и центральные точки графа, а также нашли радиус и диаметр.