

Отчёт по расчётной работе по дисциплине ПиОИВИС

Тема: Графы

Цель

Определить, является ли вводимый граф — графом Паппа.

Задача

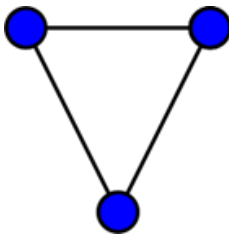
Создать алгоритм, который будет брать из файла данные и проверять по матрице инцидентности, является ли этот граф графом Паппа.

Вариант

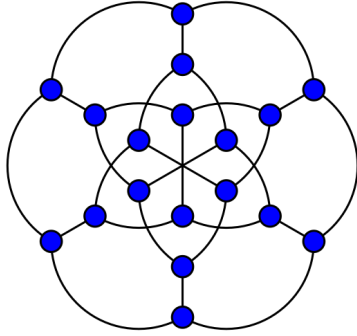
1.20 (ми)

Список ключевых понятий (определения)

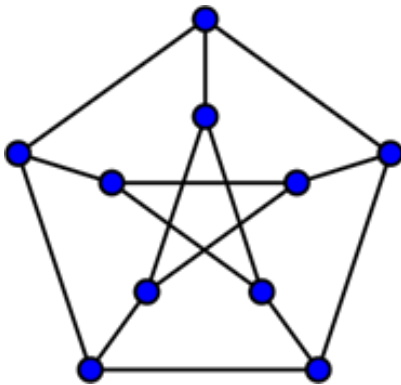
- **Граф** — это совокупность непустого множества вершин и множества пар вершин.



- **Граф Паппа** — двудольный 3-регулярный неориентированный граф с 18 вершинами и 27 рёбрами. Является единственным кубическим симметричным графом с 18 вершинами.



- **Кубический граф** — граф, в котором все вершины имеют степень три.



- **Инцидентность** — понятие, используемое только в отношении ребра и вершины. Две вершины или два ребра не могут быть инцидентны.
- **Матрица инцидентности** — одна из форм представления графа, в которой указываются связи между инцидентными элементами графа (ребро и вершина). Столбцы матрицы соответствуют рёбрам, строки — вершинам. Ненулевое значение в ячейке матрицы указывает связь между вершиной и ребром (их инцидентность).

V	1-2	1-3	2-4	2-5	3-5
1	1	1	0	0	0
2	1	0	1	1	0
3	0	1	0	0	1
4	0	0	1	0	0
5	0	0	0	1	1

Файлы с содержанием матрицы инцидентности

- graph.txt

0	0	0	0	0	0	0	1	0	0	0	1	1	0
0	0	0	0	0	0	1	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	0	0	0	0	0	0	1	0	0
0	0	1	0	1	0	1	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1

Алгоритм

1. Выбрать файл с матрицей инцидентности и открыть его.
2. Посчитать количество элементов и строк в файле.
3. Проверить размеры матрицы: матрица имеет n строк (число вершин) и m столбцов (число рёбер).
4. Проверить значения матрицы: все элементы матрицы принадлежат множеству $\{0, 1\}$. Если матрица содержит иные числа, вывести сообщение: «Матрица должна состоять из 0 и 1».
5. Проверить столбцы: каждый столбец должен содержать ровно 2 единицы. Если нет, вывести сообщение: «Каждый столбец должен содержать ровно 2 единицы».
6. Создать матрицу $M \times N$, где N — количество строк, а M — количество столбцов, и считать данные из файла в эту матрицу.
7. Проверить, соответствует ли $N = 18$.
8. Проверить симметричность матрицы: $a_{ij} = a_{ji}$.
9. Проверить, чтобы каждая вершина имела ровно 3 связи.
10. Вывести матрицу инцидентности.
11. Вывести сообщение: является или нет данный граф графом Палпа.

Код

```
1 #include <iostream>
2 #include <fstream>
3
4 using namespace std;
5
6 int main() {
7     setlocale(LC_ALL, "rus");
8
9     // Input file name
10    string nameOfFile;
11    int numberOfGraph;
12    cout << "Enter the number of the file containing the graph: ";
13    cin >> numberOfGraph;
14
15    switch (numberOfGraph) {
16        case 1: nameOfFile = "graph.txt"; break;
17        case 2: nameOfFile = "graph2.txt"; break;
18        case 3: nameOfFile = "graph3.txt"; break;
19        case 4: nameOfFile = "graph4.txt"; break;
20        case 5: nameOfFile = "graph5.txt"; break;
21        default:
22            cout << "Invalid file number." << endl;
23            return 0;
24    }
25
26    ifstream fin(nameOfFile);
27    if (!fin.is_open()) {
28        cout << "Failed to open the file!" << endl;
29        return 0;
30    }
31
32    // Count rows (vertices) and columns (edges)
33    const int MAX_ROWS = 100, MAX_COLS = 100; // Maximum
34        dimensions
35    int matrix[MAX_ROWS][MAX_COLS] = { 0 }; // Incidence
36        matrix
37    int rowCount = 0, colCount = 0;
38
39    char temp;
40    int value, colInRow = 0;
41    bool hasError = false; // Error flag
42
43    // Flags for each type of error
44    bool sizeError = false, matrixValueError = false,
45        columnValueError = false,
46        rowCountError = false, connectivityError = false,
47        edgeCountError = false;
48
49    while (fin.get(temp)) {
```

```

46     if (temp == '0' || temp == '1') {
47         value = temp - '0';
48         matrix[rowCount][colInRow++] = value;
49     }
50     else if (temp == '\n') {
51         if (rowCount == 0) {
52             colCount = colInRow; // Store the number of
                                   // columns after the first row
53         }
54         else if (colInRow != colCount) {
55             if (!sizeError) {
56                 cout << "The matrix has incorrect dimensions!"
57                     << endl;
58                 sizeError = true;
59             }
60             rowCount++;
61             colInRow = 0;
62         }
63     }
64     if (colInRow > 0) { // For the last row without a newline
65         if (rowCount == 0) {
66             colCount = colInRow;
67         }
68         else if (colInRow != colCount) {
69             if (!sizeError) {
70                 cout << "The matrix has incorrect dimensions!" <<
71                     endl;
72                 sizeError = true;
73             }
74         }
75         rowCount++;
76     }
77     fin.close();
78
79     // Output the matrix size
80     cout << "Matrix size: " << rowCount << "x" << colCount << endl
81         ;
82
83     // Output the initial incidence matrix
84     cout << "Incidence matrix:" << endl;
85     for (int i = 0; i < rowCount; i++) {
86         for (int j = 0; j < colCount; j++) {
87             cout << matrix[i][j] << " ";
88         }
89         cout << endl;
90     }
91
92     // Check matrix values (should be 0 or 1)
93     for (int i = 0; i < rowCount; i++) {
94         for (int j = 0; j < colCount; j++) {

```

```

93         if (matrix[i][j] != 0 && matrix[i][j] != 1) {
94             if (!matrixValueError) {
95                 cout << "The matrix should consist of 0 and 1!"
96                     << endl;
97                 matrixValueError = true;
98             }
99         }
100     }
101
102     // Check each column (should contain exactly 2 ones)
103     for (int j = 0; j < colCount; j++) {
104         int onesCount = 0;
105         for (int i = 0; i < rowCount; i++) {
106             if (matrix[i][j] == 1) {
107                 onesCount++;
108             }
109         }
110         if (onesCount != 2) {
111             if (!columnValueError) {
112                 cout << "Each column should contain exactly 2 ones!"
113                     << endl;
114                 columnValueError = true;
115             }
116         }
117     }
118
119     // Check the number of rows (vertices)
120     if (rowCount != 18) {
121         if (!rowCountError) {
122             cout << "The matrix should contain exactly 18 rows (vertices)!"
123                 << endl;
124             rowCountError = true;
125         }
126     }
127
128     // Check cubicity (each vertex should have exactly 3
129     // connections)
130     for (int i = 0; i < rowCount; i++) {
131         int connections = 0;
132         for (int j = 0; j < colCount; j++) {
133             if (matrix[i][j] == 1) {
134                 connections++;
135             }
136         }
137         if (connections != 3) {
138             if (!connectivityError) {
139                 cout << "Each vertex should have exactly 3 connections!"
140                     << endl;
141                 connectivityError = true;
142             }
143         }
144     }

```



```

139     }
140 }
141
142 // Check the number of edges
143 if (colCount != 27) {
144     if (!edgeCountError) {
145         cout << "The matrix should contain exactly 27 columns
146             (edges)!" << endl;
147         edgeCountError = true;
148     }
149 }
150
151 // If at least one error was detected
152 if (sizeError || matrixValueError || columnValueError ||
153     rowCountError ||
154     connectivityError || edgeCountError) {
155     cout << "This is not the Petersen graph!" << endl;
156 }
157 else {
158     cout << "This graph is the Petersen graph!" << endl;
159 }
160
161 return 0;
162 }

```

Вывод

В результате выполнения данной работы были получены следующие практические навыки:

- Изучены основы теории графов.
- Изучены способы представления графов.
- Изучены базовые алгоритмы для работы с графами.