Отчёт по расчётной работе по дисциплине ПиОИвИС

Тема: Графы

Цель

Определить, является ли вводимый граф — графом Паппа.

Задача

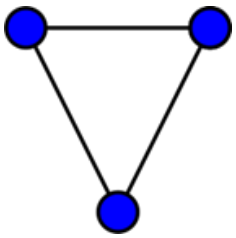
Создать алгоритм, который будет брать из файла данные и проверять по матрице инцидентности, является ли этот граф графом Паппа.

Вариант

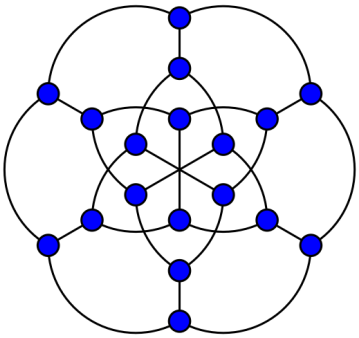
1.20 (ми)

Список ключевых понятий (определения)

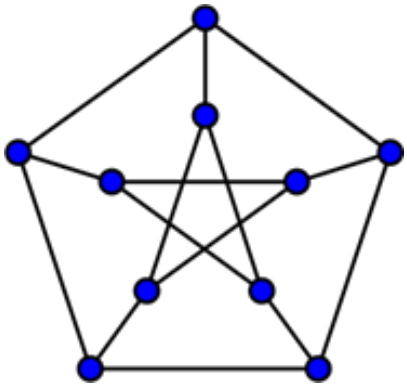
- **Граф** — математическая абстракция реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств — множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества 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.



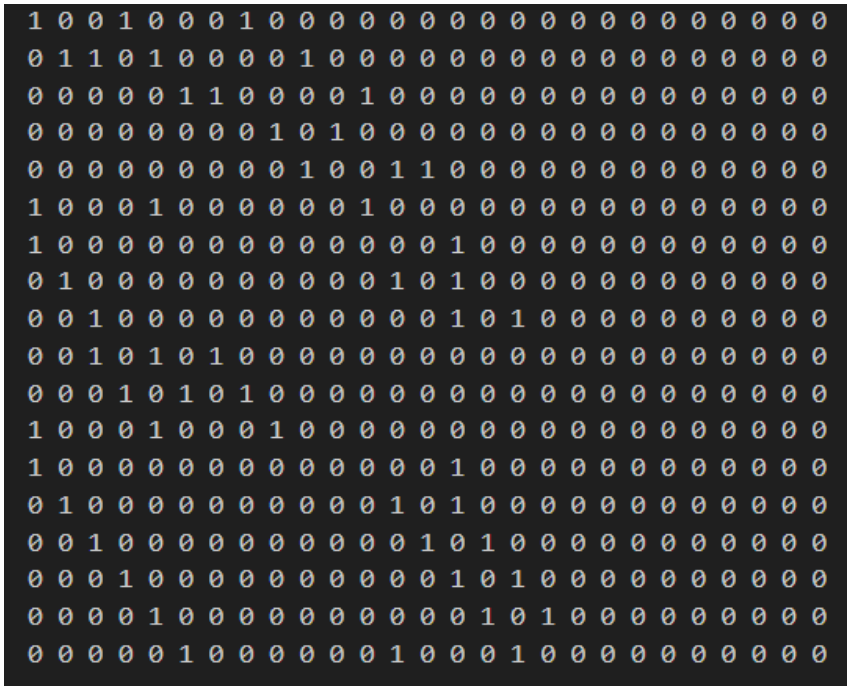
- **Инцидентность** — понятие, используемое только в отношении ребра и вершины. Две вершины или два ребра не могут быть инцидентны.

- **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).

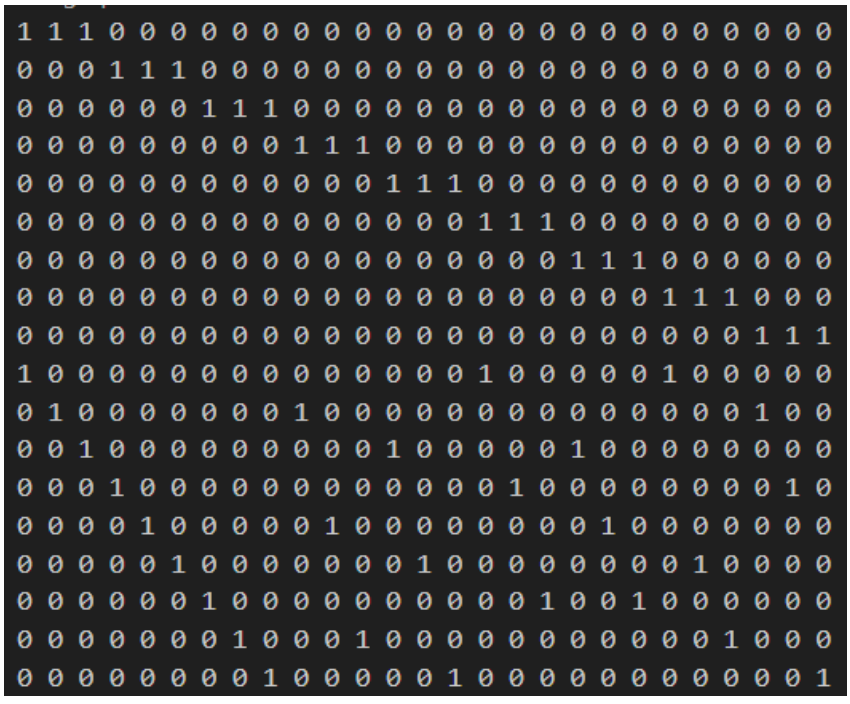
V	1-2	1-3	2-4	2-5	3-5
1	1	1	0	0	0
2	1	0	1	1	0
3	0	1	0	0	1
4	0	0	1	0	0
5	0	0	0	1	1

Файлы с содержанием матрицы инцидентности

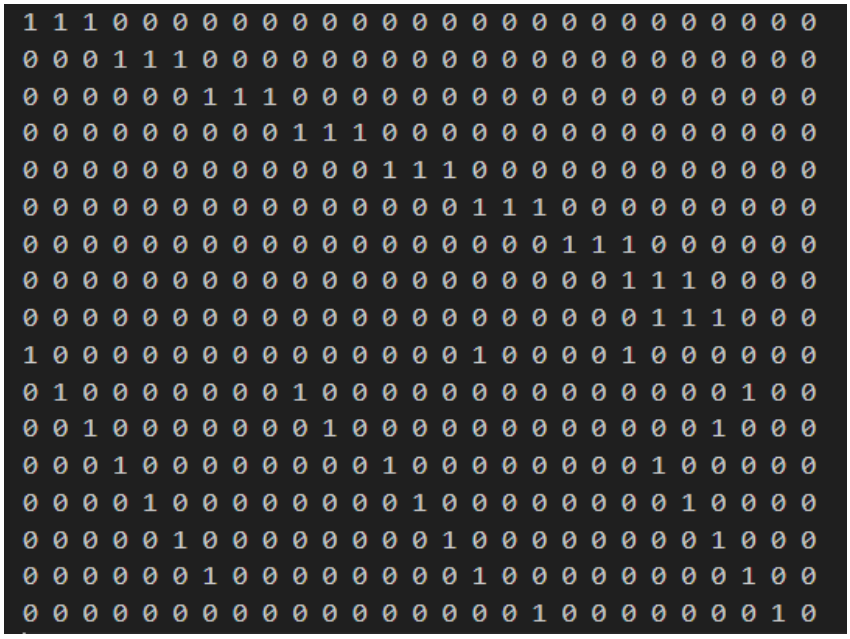
- `graph.txt`



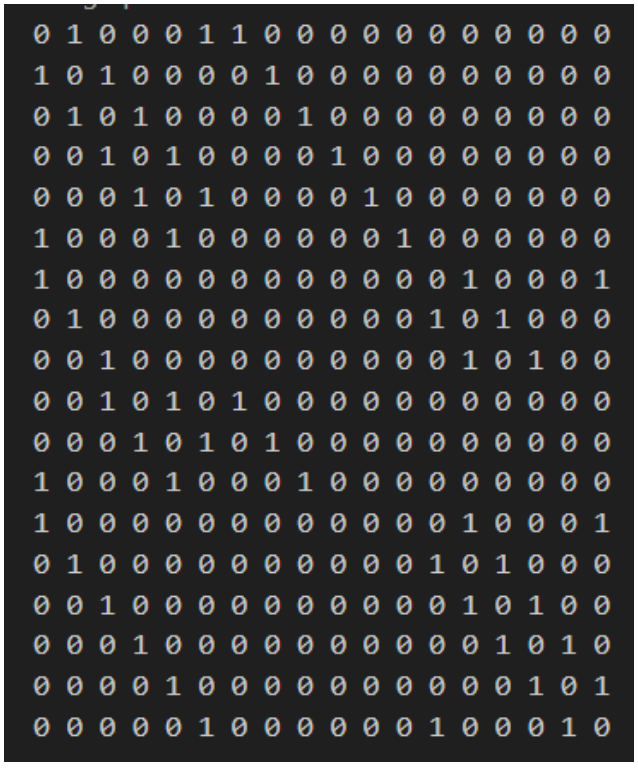
- graph2.txt



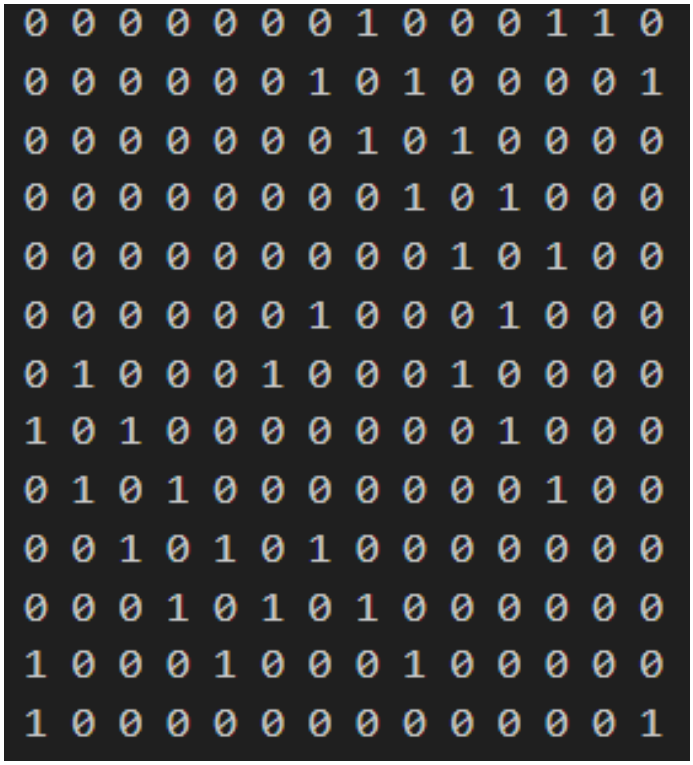
- graph3.txt



- graph4.txt



- graph5.txt



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 Паппа.

Код

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main() {
    setlocale(LC_ALL, "rus");

    // Input file name
    string nameOfFile;
    int numberOfGraph;
    cout << "Enter the number of the file containing the graph: ";
    cin >> numberOfGraph;

    switch (numberOfGraph) {
    case 1: nameOfFile = "graph.txt"; break;
    case 2: nameOfFile = "graph2.txt"; break;
    case 3: nameOfFile = "graph3.txt"; break;
    case 4: nameOfFile = "graph4.txt"; break;
    case 5: nameOfFile = "graph5.txt"; break;
    default:
        cout << "Invalid file number." << endl;
        return 0;
    }

    ifstream fin(nameOfFile);
    if (!fin.is_open()) {
        cout << "Failed to open the file!" << endl;
        return 0;
    }

    // Count rows (vertices) and columns (edges)
    const int MAX_ROWS = 100, MAX_COLS = 100; // Maximum dimensions
    int matrix[MAX_ROWS][MAX_COLS] = { 0 };     // Incidence matrix
    int rowCount = 0, colCount = 0;

    char temp;
    int value, colInRow = 0;
    bool hasError = false;  // Error flag

    // Flags for each type of error
    bool sizeError = false, matrixValueError = false, columnValueError = false,
        rowCountError = false, connectivityError = false, edgeCountError = false;

    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!"
                            << endl;
57                      sizeError = true;
58                  }
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!" <<
                        endl;
71                  sizeError = true;
72              }
73          }
74          rowCount++;
75      }
76      fin.close();
77
78      // Output the matrix size
79      cout << "Matrix size: " << rowCount << "x" << colCount << endl
            ;
80
81      // Output the initial incidence matrix
82      cout << "Incidence matrix:" << endl;
83      for (int i = 0; i < rowCount; i++) {
84          for (int j = 0; j < colCount; j++) {
85              cout << matrix[i][j] << " ";
86          }
87          cout << endl;
88      }
89
90      // Check matrix values (should be 0 or 1)
91      for (int i = 0; i < rowCount; i++) {
92          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!
                            " << endl;
96                      matrixValueError = true;
97                  }
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
                        !" << endl;
113                 columnValueError = true;
114             }
115         }
116     }
117
118     // Check the number of rows (vertices)
119     if (rowCount != 18) {
120         if (!rowCountError) {
121             cout << "The matrix should contain exactly 18 rows (
                    vertices)!" << endl;
122             rowCountError = true;
123         }
124     }
125
126     // Check cubicity (each vertex should have exactly 3
        connections)
127     for (int i = 0; i < rowCount; i++) {
128         int connections = 0;
129         for (int j = 0; j < colCount; j++) {
130             if (matrix[i][j] == 1) {
131                 connections++;
132             }
133         }
134         if (connections != 3) {
135             if (!connectivityError) {
136                 cout << "Each vertex should have exactly 3
                        connections!" << endl;
137                 connectivityError = true;
138             }
```

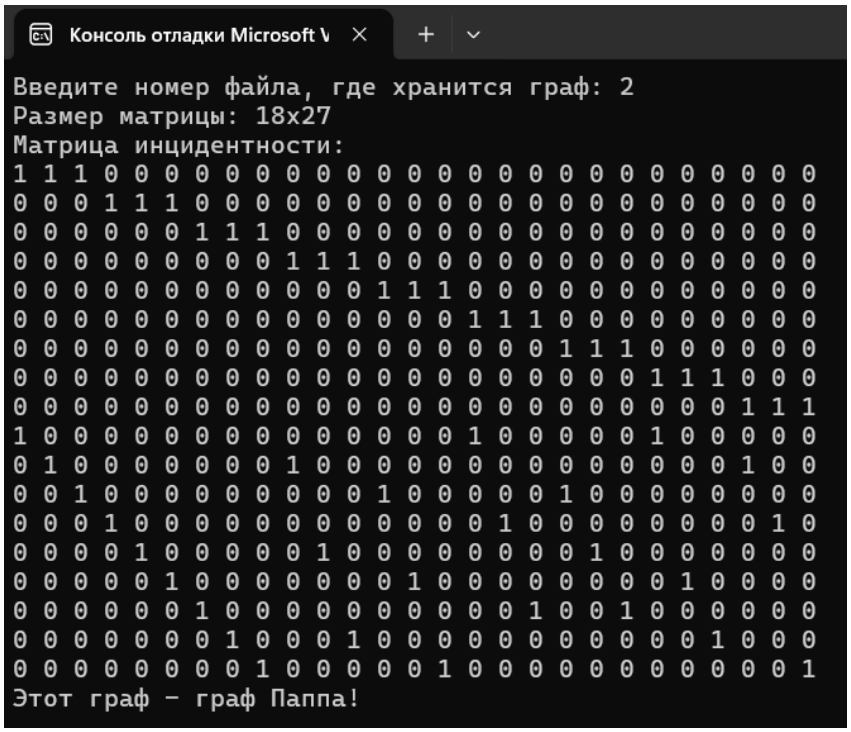
```
        }
    }

    // Check the number of edges
    if (colCount != 27) {
        if (!edgeCountError) {
            cout << "The matrix should contain exactly 27 columns (edges)!" << endl;
            edgeCountError = true;
        }
    }

    // If at least one error was detected
    if (sizeError || matrixValueError || columnValueError || rowCountError ||
        connectivityError || edgeCountError) {
        cout << "This is not the Petersen graph!" << endl;
    }
    else {
        cout << "This graph is the Petersen graph!" << endl;
    }

    return 0;
}
```

Пример работы кода



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.