МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Лабораторная работа №3.4

по дисциплине: Дискретная математика
тема: «Упорядоченные множества»

Выполнил: ст. группы ПВ-221
Лоёк Никита Викторович

Проверили:
Бондаренко Татьяна Владимировна
Рязанов Юрий Дмитриевич

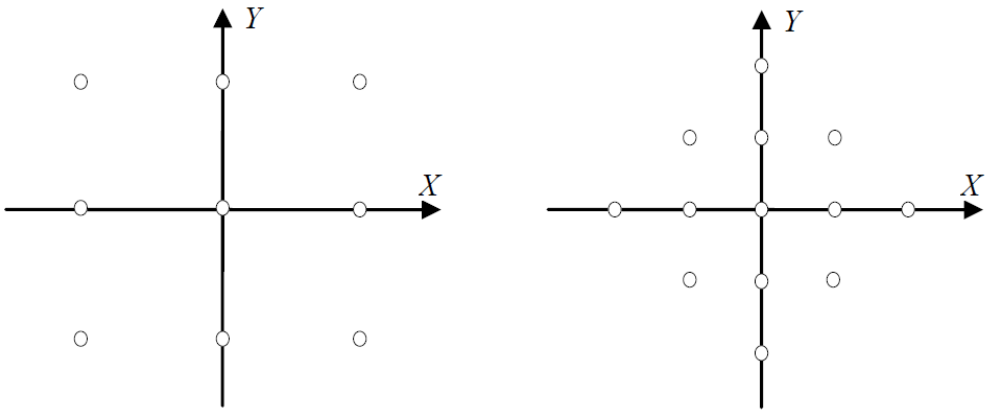
Белгород 2023 г.

Л а б о р а т о р н а я р а б о т а № 3.4

Цель работы: изучить упорядоченные множества, алгоритм топологической сортировки, научиться представлять множества диаграммами Хассе, находить минимальные (максимальные) и наименьшие (наибольшие) элементы упорядоченного множества.

Вариант 12

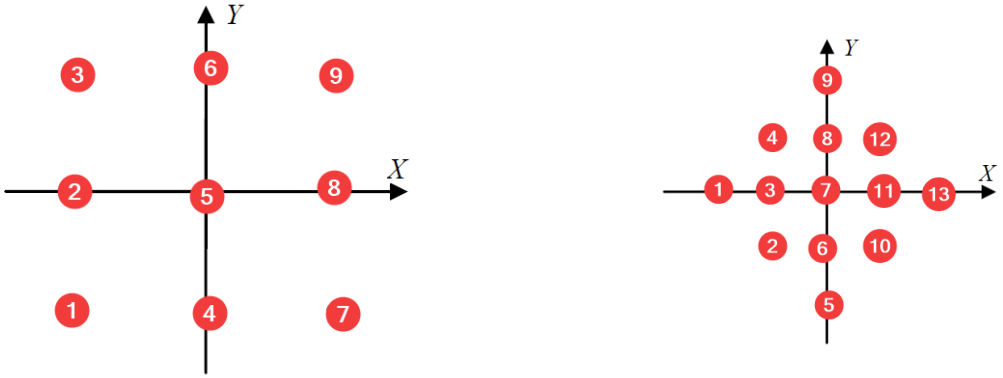
Даны множества точек на плоскости M_1, M_2 и отношение порядка. Для определения отношения на множестве точек примем следующие обозначения: a_x — абсцисса точки a; a_y — ордината a. На рис. 3 координаты правой верхней точки считать (1,1). Координаты самой верхней точки считать (0,2), а координаты самой правой точки считать (2,0).



12 | $A=\{(a,b) \mid a_x \cdot a_y < b_x \cdot b_y\}$

Задания

1. Написать программы, формирующие матрицы отношений в соответствии с вариантом задания, на множествах M_1 и M_2.



Программа:

```
#include <iostream>
#include <vector>
#include <cassert>

using namespace std;

vector<vector<bool>> createRelationship(vector<vector<int>> &matrix) {
    vector<vector<bool>> newMatrix(matrix.size(),
                                   vector<bool>(matrix.size()));
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            int ax = matrix[i][0];
            int ay = matrix[i][1];
            int bx = matrix[j][0];
            int by = matrix[j][1];
            newMatrix[i][j] = ax*ay<bx*by;
        }
    }

    return newMatrix;
}

int main() {
    vector <vector<int>> m1(9, vector<int>(2));
    int index = 0;
    for (int x = -1; x <= 1; x++) {
        for (int y = -1; y <= 1; y++) {
            m1[index++] = {x, y};
        }
    }

    vector <vector<int>> m2(13, vector<int>(2));
    index = 0;
    for (int x = -2; x <= 2; x++) {
        for (int y = -2; y <= 2; y++) {
            if (abs(x) <= 1 && abs(y) <= 1 ||
                (y == 0 || x == 0)) {
                m2[index++] = {x, y};
            }
        }
    }

    vector <vector<bool>> matrixRel1 = createRelationship(m1);
    vector <vector<bool>> matrixRel2 = createRelationship(m2);

    cout << "M1:" << endl;
    matrixOutput(matrixRel1);
    cout << "M2:" << endl;
    matrixOutput(matrixRel2);

    return 0;
}
```

Результат работы программы:

M1:

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

M2:

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

2. Написать программы, формирующие матрицы отношения доминирования по матрицам отношения порядка.

```
#include <iostream>
#include <vector>
#include <cassert>

using namespace std;

vector <vector<bool>> createDominance(vector <vector<bool>> matrix) {
    vector <vector<bool>> newMatrix(matrix.size(),
                                    vector<bool>(matrix.size()));

    for (int i = 0; i < matrix.size(); i++) {
            matrix[i][i] = 0;
    }

    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            if (matrix[i][j] == 1) {
                bool flag = true;
                for (int k = 0; k < matrix.size(); k++) {
                    if (matrix[i][k] && matrix[k][j]) {
                        flag = false;
                        break;
                    }
                }
                if (flag) {
                    newMatrix[i][j] = 1;
                } else {
                    newMatrix[i][j] = 0;
                }
            } else {
                newMatrix[i][j] = 0;
            }
        }
    }

    return newMatrix;
}


int main() {
    vector <vector<int>> m1(9, vector<int>(2));
    int index = 0;
    for (int x = -1; x <= 1; x++) {
        for (int y = -1; y <= 1; y++) {
            m1[index++] = {x, y};
        }
    }

    vector <vector<int>> m2(13, vector<int>(2));
    index = 0;
    for (int x = -2; x <= 2; x++) {
        for (int y = -2; y <= 2; y++) {
            if (abs(x) <= 1 && abs(y) <= 1 ||
                (y == 0 || x == 0)) {
                m2[index++] = {x, y};
            }
        }
    }

    vector <vector<bool>> matrixRel1 = createRelationship(m1);
```

```
    vector <vector<bool>> matrixRel2 = createRelationship(m2);

    vector <vector<bool>> matrixDom1 = createDominance(matrixRel1);
    vector <vector<bool>> matrixDom2 = createDominance(matrixRel2);

    cout << "M1 Dominance:" << endl;
    matrixOutput(matrixDom1);
    cout << "M2 Dominance:" << endl;
    matrixOutput(matrixDom2);

    return 0;
}
```

Результат работы программы:

```
M1 Dominance:
0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1
0 1 0 1 1 1 0 1 0
1 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 1
0 1 0 1 1 1 0 1 0
1 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0

M2 Dominance:
0 1 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 1 0
1 0 1 0 1 1 1 1 1 0 1 0 1
0 1 0 0 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1 0
1 0 1 0 1 1 1 1 1 0 1 0 1
0 1 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 1 0
```

3. Написать программу, реализующую алгоритм топологической сортировки по матрице отношения доминирования.

```
#include <iostream>
#include <vector>
#include <cassert>


using namespace std;


vector<int> sortTopological(vector <vector<bool>> &matrix) {
    vector<int> newMatrix(matrix.size());

    for (int i = 0; i < matrix.size(); i++) {
        int counter = 0;
        for (int j = 0; j < matrix.size(); j++) {
            counter += matrix[j][i];
        }
        newMatrix[i] = counter;
    }

    return newMatrix;
}


void sortTopologicalOutput(vector<int> &matrix1, vector <vector<int>>
&matrix2) {
    assert(matrix1.size() == matrix2.size());
    for (int i = 0; i < matrix1.size(); i++) {
        bool flagOfNull = false;
        for (int j = 0; j < matrix1.size(); j++) {
            if (matrix1[j] == 0) {
                flagOfNull = true;
            }
        }
        if (flagOfNull) {
            for (int j = 0; j < matrix1.size(); j++) {
                matrix1[j]--;
            }
        } else {
            for (int j = 0; j < matrix1.size(); j++) {
                if (matrix1[j] > 0) {
                    matrix1[j]--;
                }
            }
        }
    }

    int val = -1;
    for (int i = 0; i < matrix1.size(); i++) {
        bool flag = false;
        for (int j = 0; j < matrix1.size(); j++) {
            if (matrix1[j] == val) {
                flag = true;
                break;
            }
        }

        if (flag) {
            cout << i << " level: ";
            for (int j = 0; j < matrix1.size(); j++) {
                if (matrix1[j] == val) {
```

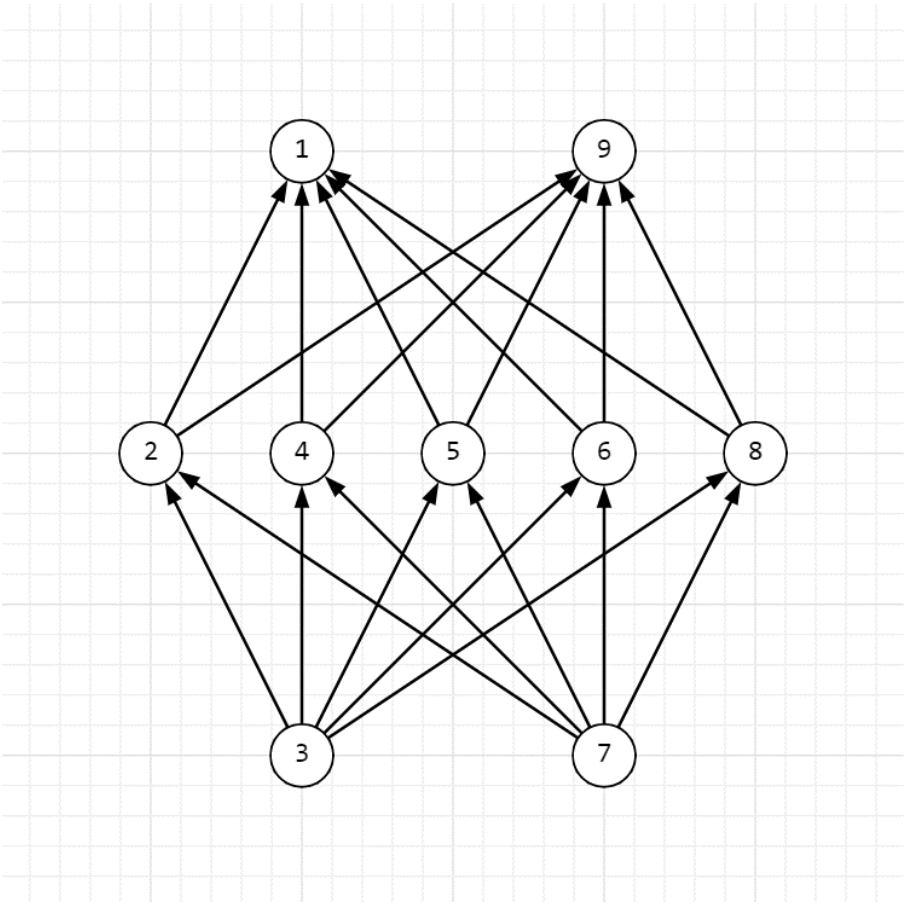
```
                    cout << j + 1 << " ";
                }
            }
            cout << endl;
        }

        val--;
    }
}

int main() {
    vector <vector<int>> m1(9, vector<int>(2));
    int index = 0;
    for (int x = -1; x <= 1; x++) {
        for (int y = -1; y <= 1; y++) {
            m1[index++] = {x, y};
        }
    }

    vector <vector<int>> m2(13, vector<int>(2));
    index = 0;
    for (int x = -2; x <= 2; x++) {
        for (int y = -2; y <= 2; y++) {
            if (abs(x) <= 1 && abs(y) <= 1 ||
                (y == 0 || x == 0)) {
                m2[index++] = {x, y};
            }
        }
    }

    vector <vector<bool>> matrixRel1 = createRelationship(m1);
    vector <vector<bool>> matrixRel2 = createRelationship(m2);

    vector <vector<bool>> matrixDom1 = createDominance(matrixRel1);
    vector <vector<bool>> matrixDom2 = createDominance(matrixRel2);

    vector<int> matrixTop1 = sortTopological(matrixDom1);
    vector<int> matrixTop2 = sortTopological(matrixDom2);

    cout << "M1 Topological Sort:" << endl;
    sortTopologicalOutput(matrixTop1, m1);
    cout << "M2 Topological Sort:" << endl;
    sortTopologicalOutput(matrixTop2, m2);

    return 0;
}
```

Результат работы программы:

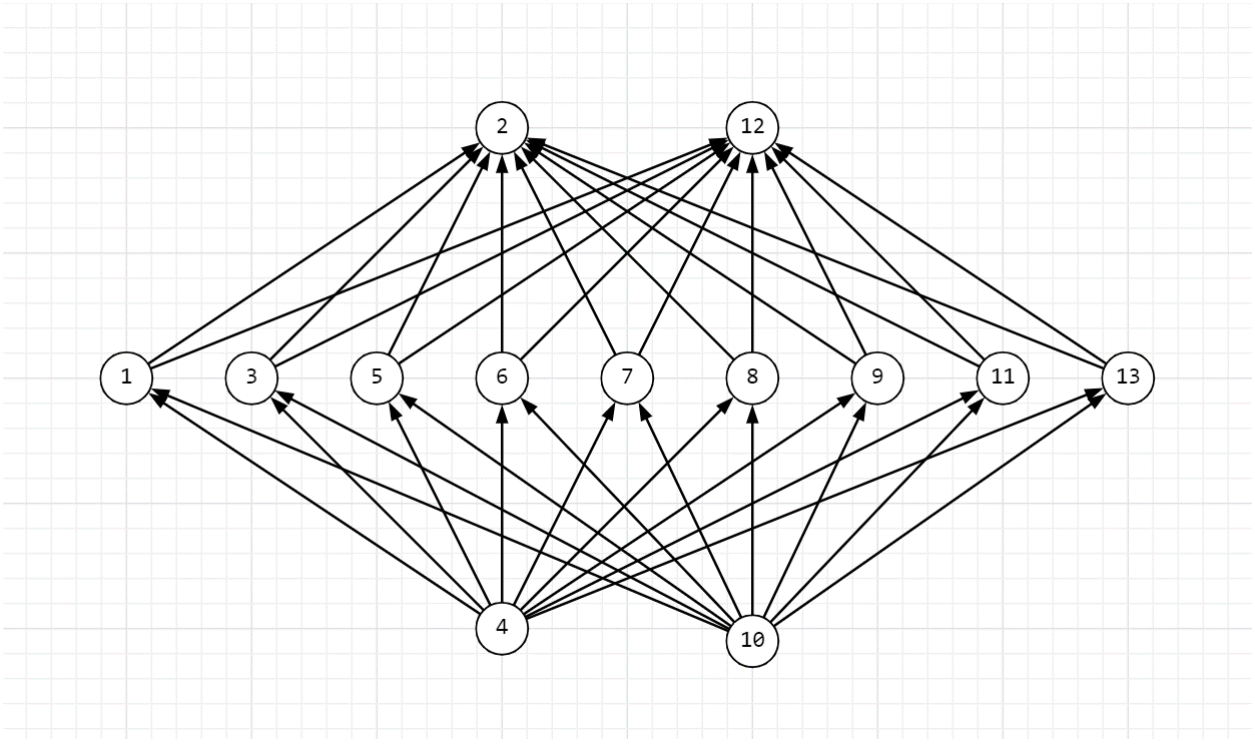
```
M1 Topological Sort:
0 level: 1 9
1 level: 2 4 5 6 8
2 level: 3 7
M2 Topological Sort:
0 level: 2 12
1 level: 1 3 5 6 7 8 9 11 13
2 level: 4 10
```

4. Изобразить диаграмму Хассе отношения доминирования на множествах M_1 и M_2.

M_1:



M_2:



5. Найти минимальные и максимальные элементы множеств M_1 и M_2.

	Минимальный элемент	Максимальный элемент
M1	3, 7	1, 9
M2	4, 10	2, 12

6. Найти, если существуют, наименьший и наибольший элементы множеств M_1 и M_2.

	Наименьший элемент	Наибольший элемент
M1	нет	нет
M2	нет	нет

Вывод

Вывод: в ходе работы я изучил упорядоченные множества, алгоритм топологической сортировки, научился представлять множества диаграммами Хассе, находить минимальные (максимальные) и наименьшие (наибольшие) элементы упорядоченного множества.