

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)



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №3.2
по дисциплине: Дискретная математика
тема: «Транзитивное замыкание отношения»

Выполнил: ст. группы ПВ-221
Лоёк Никита Викторович

Проверили:
Бондаренко Татьяна Владимировна
Рязанов Юрий Дмитриевич

Белгород 2023 г.

Лабораторная работа № 3.2

Цель работы: изучить и выполнить сравнительный анализ алгоритмов вычисления транзитивного замыкания отношения.

Задания

1. Изучить и программно реализовать алгоритмы объединения степеней и Уоршалла для вычисления транзитивного замыкания отношения.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <windows.h>

using namespace std;

// композиция
vector<vector<bool>> matrixComposition(vector<vector<bool>> &matrix1,
                                     vector<vector<bool>> &matrix2, unsigned &if_counter) {
    vector<vector<bool>> resultMatrix(matrix1.size(), vector<bool>(matrix1[0].size()));
    for (int i = 0; ++if_counter && i < matrix1.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix1[0].size(); j++) {
            bool flag = false;
            for (int z = 0; ++if_counter && !flag && ++if_counter && z < matrix1.size(); z++) {
                if (++if_counter && matrix1[i][z] && ++if_counter && matrix2[z][j]) {
                    flag = true;
                }
            }
            resultMatrix[i][j] = flag;
        }
    }
    return resultMatrix;
}

// объединение
vector<vector<bool>> matrixUnion(vector<vector<bool>> &matrix1,
                               vector<vector<bool>> &matrix2, unsigned &if_counter) {
    vector<vector<bool>> resultMatrix(matrix1.size(), vector<bool>(matrix1[0].size()));
    for (int i = 0; ++if_counter && i < matrix1.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix1[0].size(); j++) {
            resultMatrix[i][j] = matrix1[i][j] || matrix2[i][j];
        }
    }
    return resultMatrix;
}

bool isMatrixComposition(vector<vector<bool>> &matrix, unsigned &if_counter) {
    for (int i = 0; ++if_counter && i < matrix.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix.size(); j++) {
            for (int z = 0; ++if_counter && z < matrix.size(); z++) {
                if (++if_counter && matrix[i][z] && ++if_counter && matrix[z][j]) {
                    if (++if_counter && !matrix[i][j]) {
                        return false;
                    }
                }
            }
        }
    }
    return true;
}
```

```

vector<vector<bool>>> Algorithm1CalculatingTransitivity(vector<vector<bool>>> &matrix, unsigned
&if_counter) {
    vector<vector<bool>>> C_tran = matrix;
    vector<vector<bool>>> buf = matrix;

    for (int i = 1; ++if_counter && i < matrix.size()-1; i++) {
        if (++if_counter && isMatrixComposition(matrix, if_counter)) {
            return C_tran;
        }
        buf = matrixComposition(buf, matrix, if_counter);
        C_tran = matrixUnion(C_tran, buf, if_counter);
    }

    return C_tran;
}

vector<vector<bool>>> Algorithm2CalculatingTransitivity(vector<vector<bool>>> &matrix, unsigned
&if_counter) {
    vector<vector<bool>>> C = matrix;

    for (int i = 0; ++if_counter && i < matrix.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix.size(); j++) {
            for (int k = 0; ++if_counter && k < matrix.size(); k++) {
                C[j][k] = (C[j][k] || C[j][i] && C[i][k]);
            }
        }
    }

    return C;
}

```

3. Разработать и написать программу, которая генерирует 1000 отношений на множестве мощности N с заданным числом пар, для каждого отношения вычисляет транзитивное замыкание двумя алгоритмами и подсчитывает количество k выполнений тела самого вложенного цикла. Значение k при обработке различных отношений на множестве мощности N с заданным числом пар может быть разным, поэтому программа должна определять минимальное и максимальное значение k . Отношение, при обработке которого получено минимальное (максимальное) k , и его транзитивное замыкание, сохранить в файле. Выполнить программу при $N = 5, 10$ и 15 . Результат для каждого N представить в виде таблицы (табл. 3), а сохраненные отношения — в виде графа.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <windows.h>

using namespace std;

// композиция
vector<vector<bool>> matrixComposition(vector<vector<bool>> &matrix1,
                                     vector<vector<bool>> &matrix2, unsigned &if_counter) {
    vector<vector<bool>> resultMatrix(matrix1.size(), vector<bool>(matrix1[0].size()));
    for (int i = 0; ++if_counter && i < matrix1.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix1[0].size(); j++) {
            bool flag = false;
            for (int z = 0; ++if_counter && !flag && ++if_counter && z < matrix1.size(); z++) {
                if (++if_counter && matrix1[i][z] && ++if_counter && matrix2[z][j]) {
                    flag = true;
                }
            }
            resultMatrix[i][j] = flag;
        }
    }
    return resultMatrix;
}

// объединение
vector<vector<bool>> matrixUnion(vector<vector<bool>> &matrix1,
                               vector<vector<bool>> &matrix2, unsigned &if_counter) {
    vector<vector<bool>> resultMatrix(matrix1.size(), vector<bool>(matrix1[0].size()));
    for (int i = 0; ++if_counter && i < matrix1.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix1[0].size(); j++) {
            resultMatrix[i][j] = matrix1[i][j] || matrix2[i][j];
        }
    }
    return resultMatrix;
}

bool isMatrixComposition(vector<vector<bool>> &matrix, unsigned &if_counter) {
    for (int i = 0; ++if_counter && i < matrix.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix.size(); j++) {
            for (int z = 0; ++if_counter && z < matrix.size(); z++) {
                if (++if_counter && matrix[i][z] && ++if_counter && matrix[z][j]) {
                    if (++if_counter && !matrix[i][j]) {
                        return false;
                    }
                }
            }
        }
    }
    return true;
}
```

```

vector<vector<bool>> Algorithm1CalculatingTransitivity(vector<vector<bool>> &matrix, unsigned
&if_counter) {
    vector<vector<bool>> C_tran = matrix;
    vector<vector<bool>> buf = matrix;

    for (int i = 1; ++if_counter && i < matrix.size()-1; i++) {
        if (++if_counter && isMatrixComposition(matrix, if_counter)) {
            return C_tran;
        }
        buf = matrixComposition(buf, matrix, if_counter);
        C_tran = matrixUnion(C_tran, buf, if_counter);
    }

    return C_tran;
}

vector<vector<bool>> Algorithm2CalculatingTransitivity(vector<vector<bool>> &matrix, unsigned
&if_counter) {
    vector<vector<bool>> C = matrix;

    for (int i = 0; ++if_counter && i < matrix.size(); i++) {
        for (int j = 0; ++if_counter && j < matrix.size(); j++) {
            for (int k = 0; ++if_counter && k < matrix.size(); k++) {
                C[j][k] = (C[j][k] || C[j][i] && C[i][k]);
            }
        }
    }

    return C;
}

void matrixToFile(ofstream &file, vector<vector<bool>> &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            file << matrix[i][j] << ' ';
        }
        file << endl;
    }
    file << endl;
}

void writingToFile(int n, int m,
    unsigned k1_min, unsigned k1_max, vector<vector<vector<bool>>> &matrix1_1,
    unsigned k2_min, unsigned k2_max, vector<vector<vector<bool>>> &matrix1_2) {
    string fileName = "At_N_" + to_string(n) + "_couple_" + to_string(m) + ".txt";
    ofstream file(fileName);
    file << "При n = " << n << " и числе пар в отношении " << m << ":" << endl << endl;

    file << "Алгоритм объединения степней:" << endl;
    file << "min: k = " << k1_min << endl;
    file << "Matrix:" << endl;
    matrixToFile(file, matrix1_1[0]);
    file << "Matrix_transitivity:" << endl;
    matrixToFile(file, matrix1_1[1]);
    file << "#####" << endl << endl;
    file << "max: k = " << k1_max << endl;
    file << "Matrix:" << endl;
    matrixToFile(file, matrix1_1[2]);
    file << "Matrix_transitivity:" << endl;
    matrixToFile(file, matrix1_1[3]);

    file << "#####" << endl;
    file << "#####" << endl;
    file << "#####" << endl << endl;
}

```

```

file << "Алгоритм Уоршалла:" << endl;
file << "min: k = " << k2_min << endl;
file << "Matrix:" << endl;
matrixToFile(file, matrix1_2[0]);
file << "Matrix_transitivity:" << endl;
matrixToFile(file, matrix1_2[1]);
file << "#####" << endl << endl;
file << "max: k = " << k2_max << endl;
file << "Matrix:" << endl;
matrixToFile(file, matrix1_2[2]);
file << "Matrix_transitivity:" << endl;
matrixToFile(file, matrix1_2[3]);

file.close();
}

void genBinNum_(int n, int m, vector<bool> num, vector<vector<bool>> &rez, unsigned long long
lim) {
    if (n != 0 && rez.size() < lim) {
        num.push_back(false);
        genBinNum_(n-1, m, num, rez, lim);
        num.pop_back();
    }
    if (m != 0 && rez.size() < lim) {
        num.push_back(true);
        genBinNum_(n, m-1, num, rez, lim);
        num.pop_back();
    }
    if (n == 0 && m == 0 || rez.size() >= lim) {
        rez.push_back(num);
        return;
    }
}

vector<vector<bool>> genBinNum(int n, int m, int lim=-1) {
    vector<vector<bool>> rez(0);

    genBinNum_(n*n-m, m, vector<bool>(0), rez, lim);

    return rez;
}

vector<vector<vector<bool>>> genMatrices(int n, int m, int lim=0) {
    vector<vector<vector<bool>>> generation;
    vector<vector<bool>> binNums = genBinNum(n, m, lim);

    for (unsigned long long mask = 0; mask < lim && mask < binNums.size(); mask++) {
        int mk = 0;
        vector<vector<bool>> matrix(n, vector<bool>(n));
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                matrix[i][j] = binNums[mask][mk++];
            }
        }
        generation.push_back(matrix);
    }

    return generation;
}

void getComparisonCounter(int n, int m) {
    unsigned k1_min = -1, k1_max = 0;
    unsigned k2_min = -1, k2_max = 0;
    vector<vector<vector<bool>>> matrix_1(4, vector<vector<bool>>(n, vector<bool>(n)));

```

```

vector<vector<vector<bool>>> matrix_2(4, vector<vector<bool>>(n, vector<bool>(n)));

const int lim = 10000;
vector<vector<vector<bool>>> generatedMatrices = genMatrices(n, m, lim);
for (unsigned long long mask = 0; mask < generatedMatrices.size(); mask++) {
    vector<vector<bool>> matrix = generatedMatrices[generatedMatrices.size()-mask-1];

    unsigned k = 0;
    vector<vector<bool>> matrixTransitivity = Algorithm1CalculatingTransitivity(matrix, k);
    if (k < k1_min || k > k1_max) {
        if (k < k1_min) {
            k1_min = k;
            matrix_1[0] = matrix;
            matrix_1[1] = matrixTransitivity;
        }
        if (k > k1_max) {
            k1_max = k;
            matrix_1[2] = matrix;
            matrix_1[3] = matrixTransitivity;
        }
    }

    k = 0;
    matrixTransitivity = Algorithm2CalculatingTransitivity(matrix, k);
    if (k < k2_min || k > k2_max) {
        if (k < k2_min) {
            k2_min = k;
            matrix_2[0] = matrix;
            matrix_2[1] = matrixTransitivity;
        }
        if (k > k2_max) {
            k2_max = k;
            matrix_2[2] = matrix;
            matrix_2[3] = matrixTransitivity;
        }
    }
}
writingToFile(n, m, k1_min, k1_max, matrix_1, k2_min, k2_max, matrix_2);
}

void writeToFile(int n) {
    getComparisonCounter(n, 1);
    getComparisonCounter(n, pow(n, 2) / 4);
    getComparisonCounter(n, pow(n, 2) / 2);
    getComparisonCounter(n, (2 * (pow(n, 2))) / 3);
    getComparisonCounter(n, pow(n, 2));
}

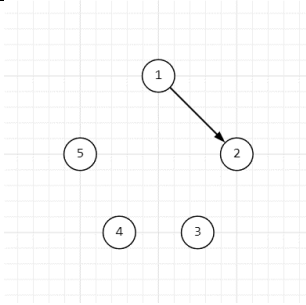
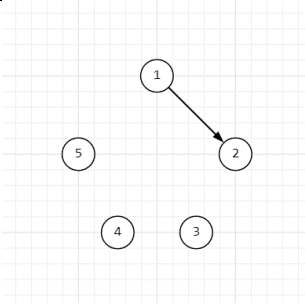
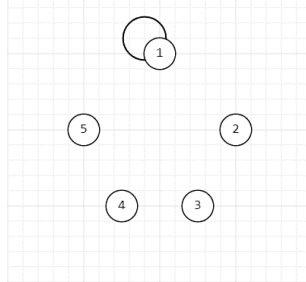
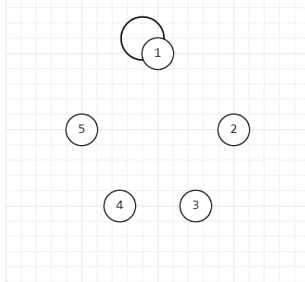
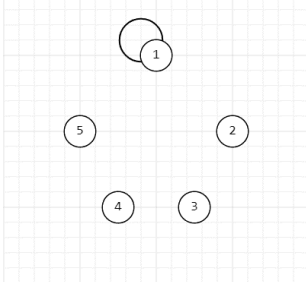
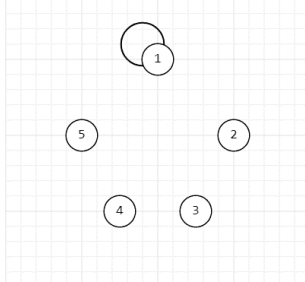
int main() {
    SetConsoleCP(CP_UTF8);

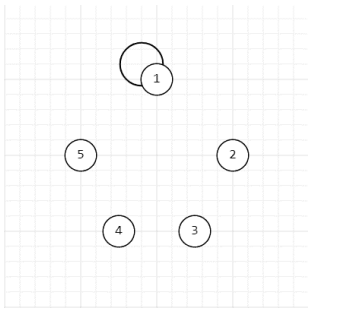
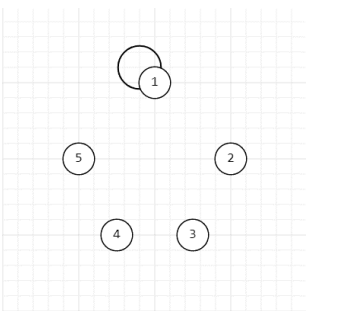
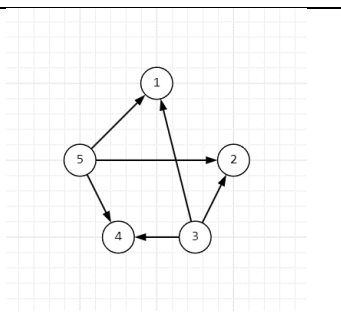
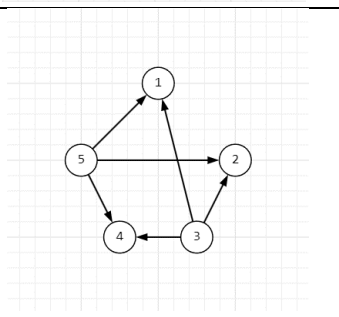
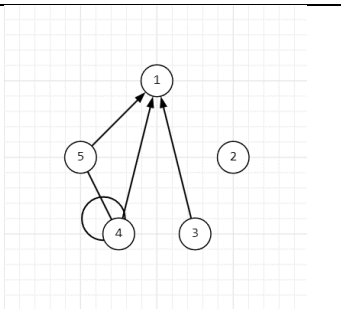
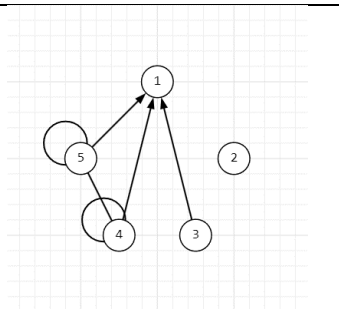
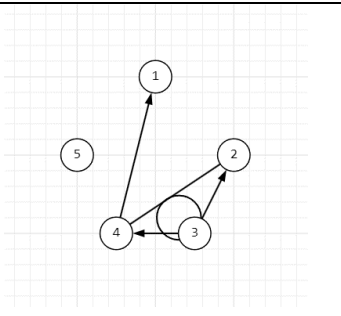
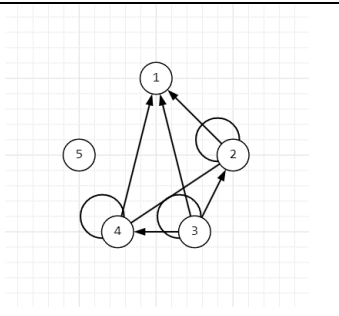
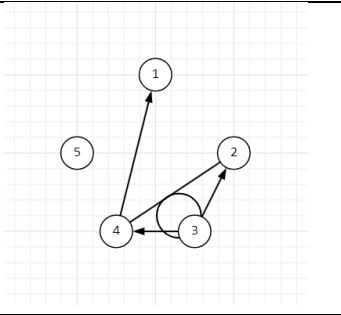
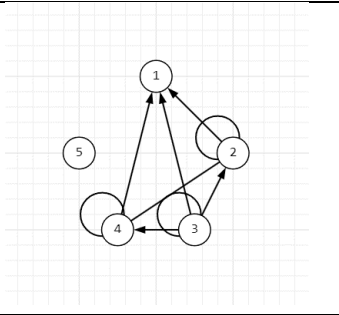
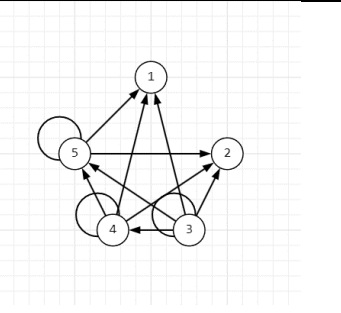
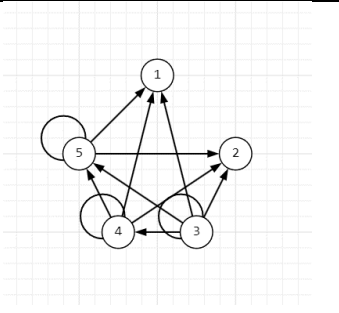
    writeToFile(5);
    writeToFile(10);
    writeToFile(15);

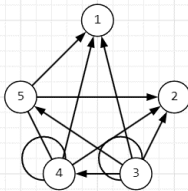
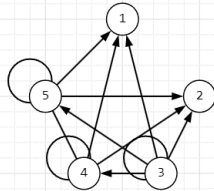
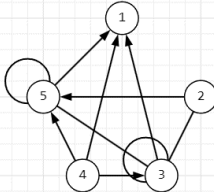
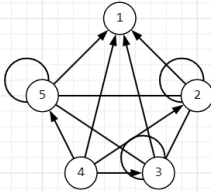
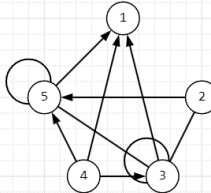
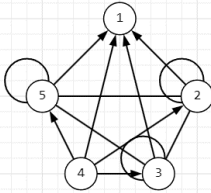
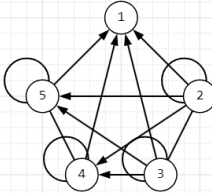
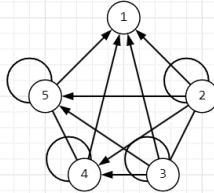
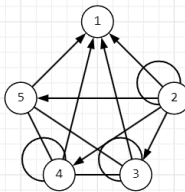
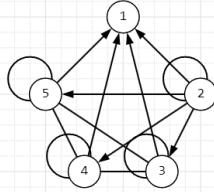
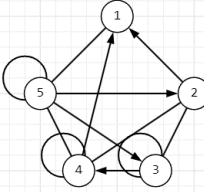
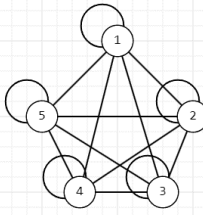
    return 0;
}

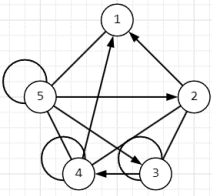
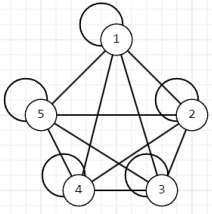
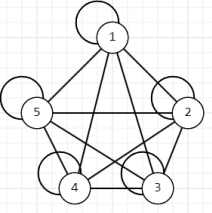
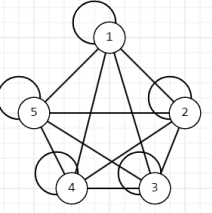
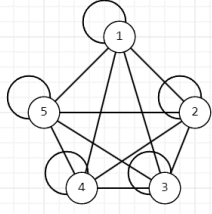
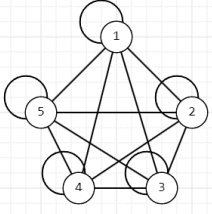
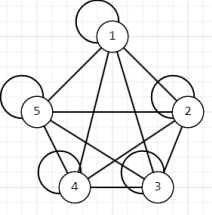
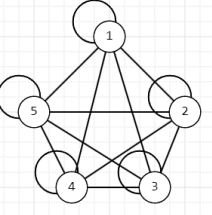
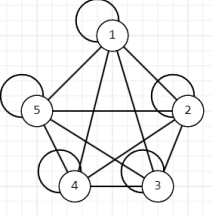
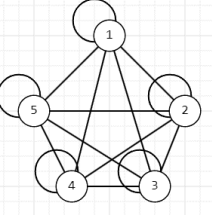
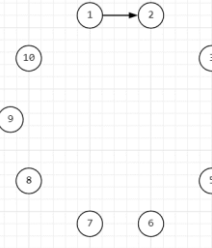
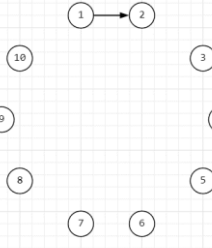
```

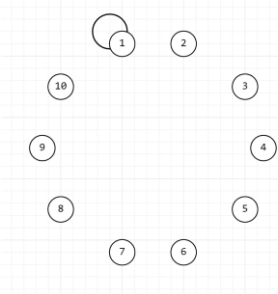
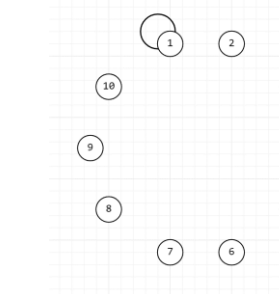
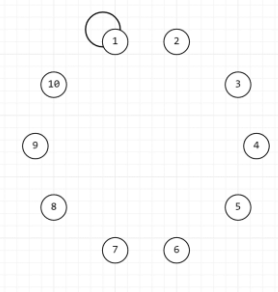
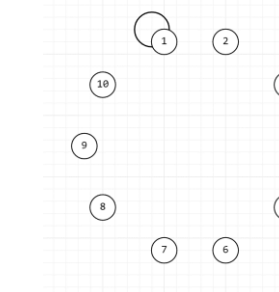
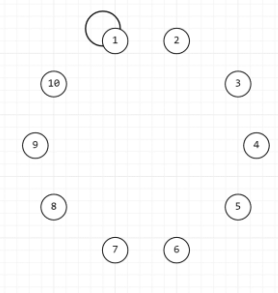
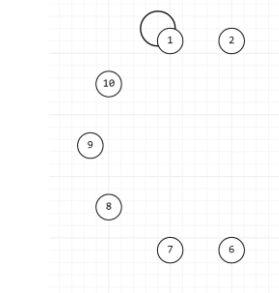
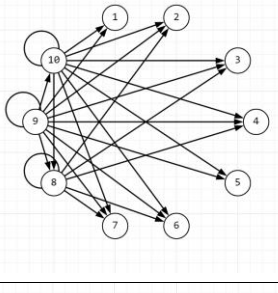
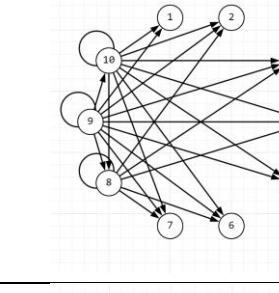
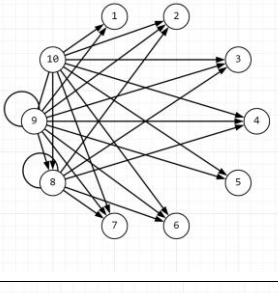
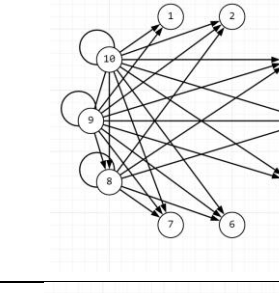
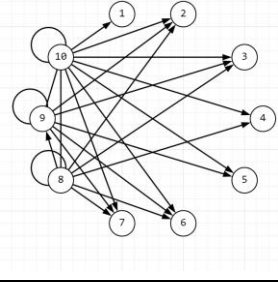
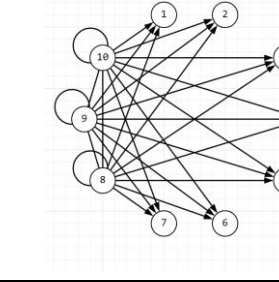
N	Число пар в отношении	1		N ² /4		N ² /2		N ² *(2/3)		N ²	
		min	max	min	max	min	max	min	max	min	max
5	Алгоритм объединения степеней	318	319	343	2533	395	2604	437	2517	563	563
	Алгоритм Уоршалла	186	186	186	186	186	186	186	186	186	186
10	Алгоритм объединения степеней	2233	2234	2519	48217	2973	31582	25017	25601	4223	4223
	Алгоритм Уоршалла	1221	1221	1221	1221	1221	1221	1221	1221	1221	1221
15	Алгоритм объединения степеней	7248	7249	215520	216053	168868	169830	10983	122842	13983	13983
	Алгоритм Уоршалла	3856	3856	3856	3856	3856	3856	3856	3856	3856	3856

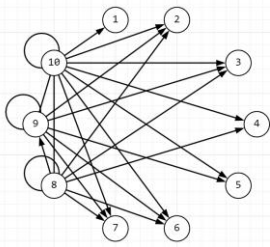
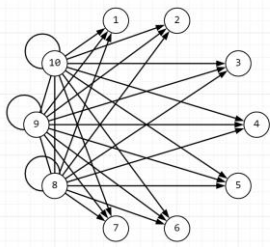
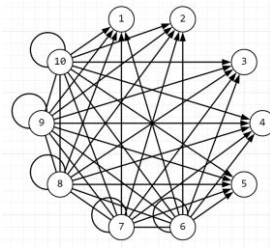
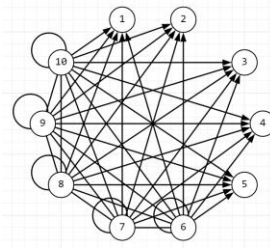
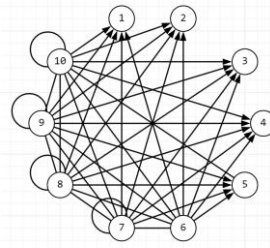
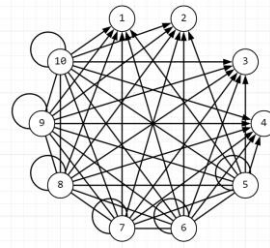
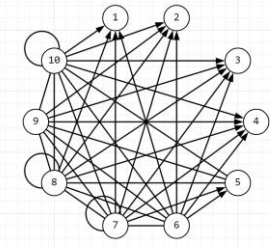
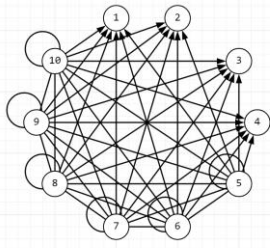
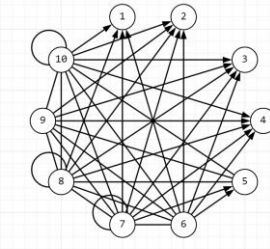
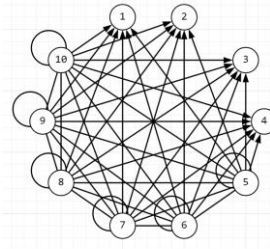
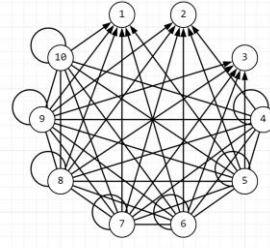
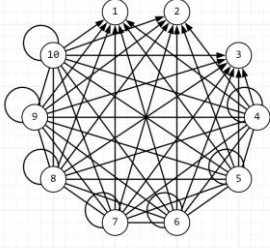


N	Число пар в отношении	Алгоритм	min/max	Матрица	Транзитивная к ней
5	1	Алгоритм объединения степеней	min		
			max		
		Алгоритм Уоршалла	min		

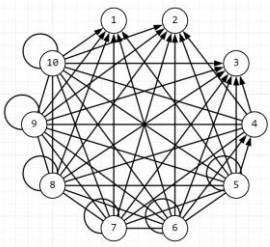
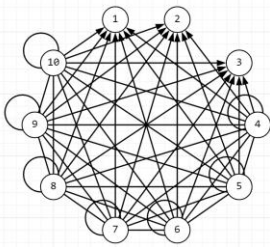
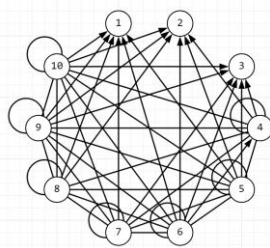
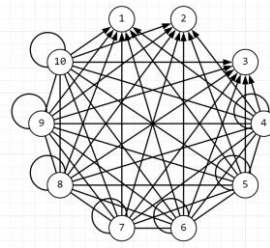
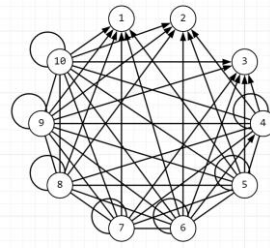
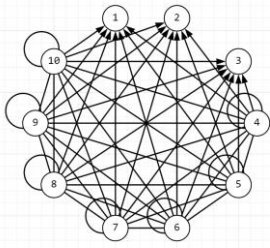
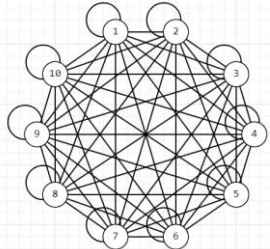
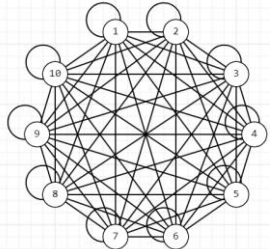
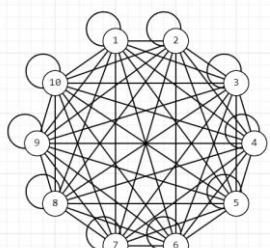
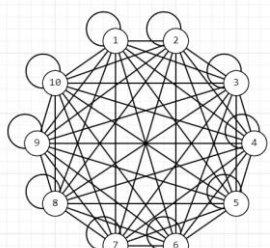
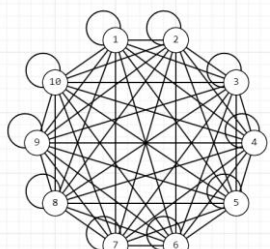
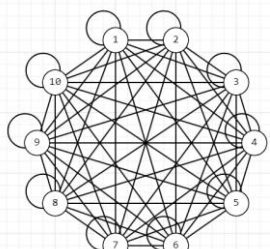
			max		
N^2/4	Алгоритм объединения степеней	min			
		max			
	Алгоритм Уоршалла	min			
		max			
N^2/2	Алгоритм объединения степеней	min			

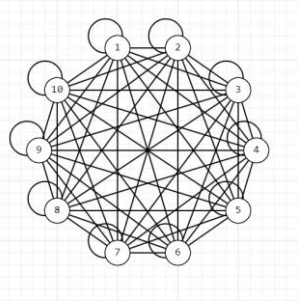
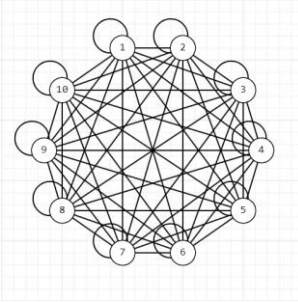
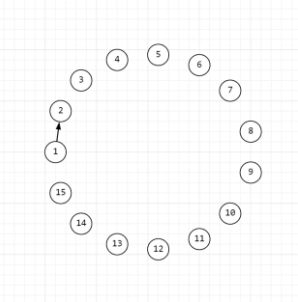
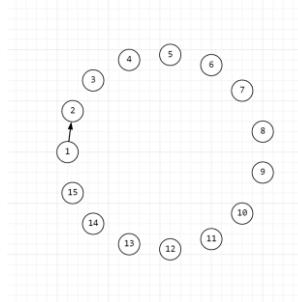
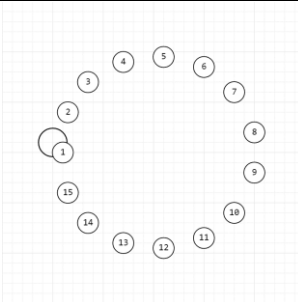
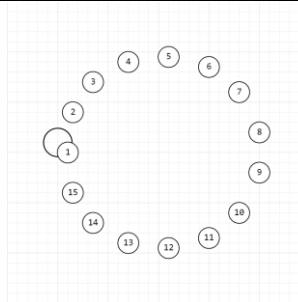
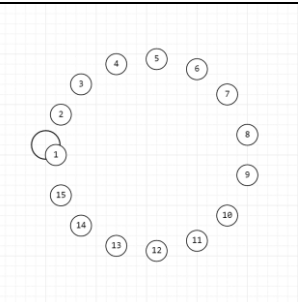
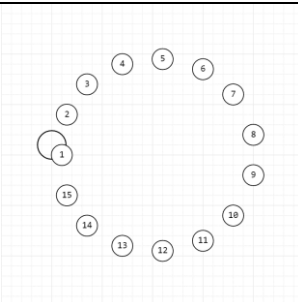
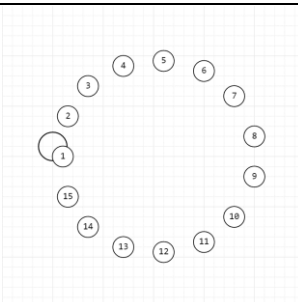
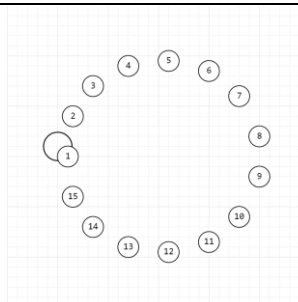
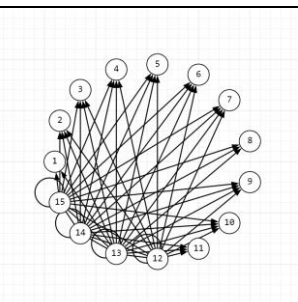
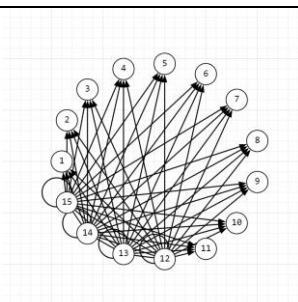
			max		
N^2*(2/3)	Алгоритм Уоршалла	min			
		max			
	Алгоритм объединения степеней	min			
		max			
	Алгоритм Уоршалла	min			

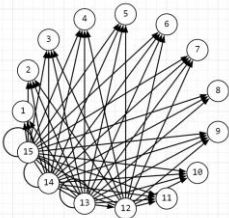
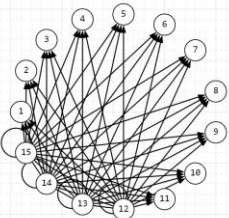
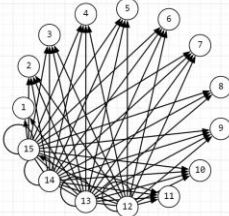
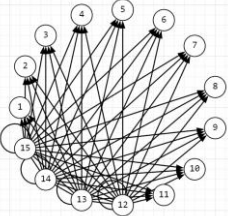
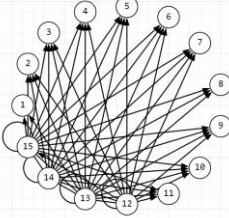
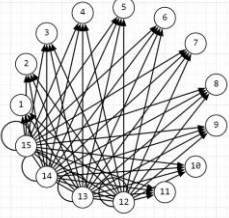
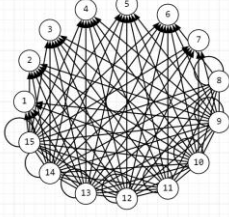
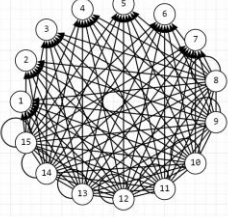
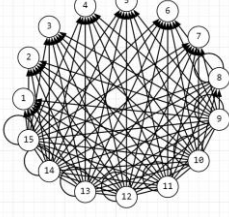
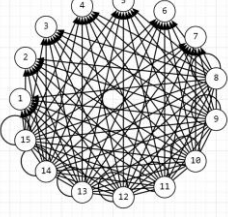
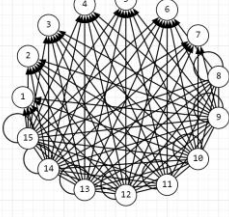
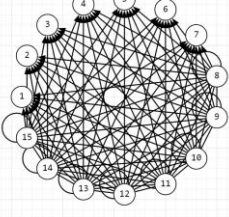
			max		
N^2		Алгоритм объединения степеней	min		
			max		
		Алгоритм Уоршалла	min		
			max		
10	1	Алгоритм объединения степеней	min		

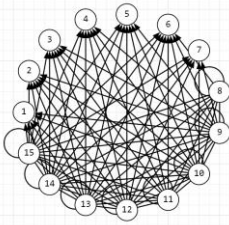
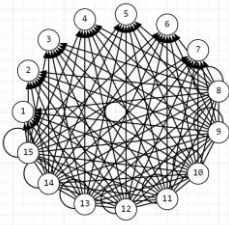
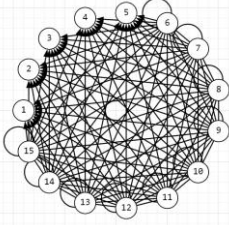
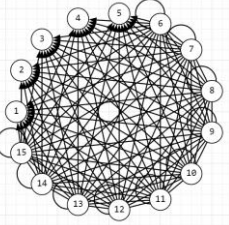
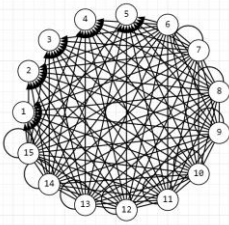
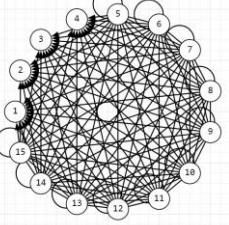
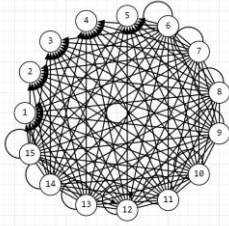
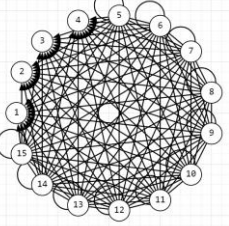
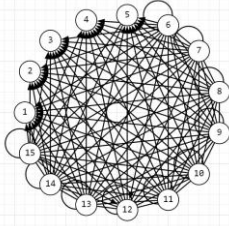
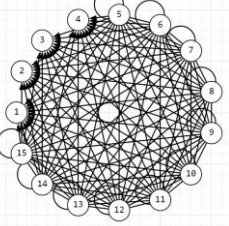
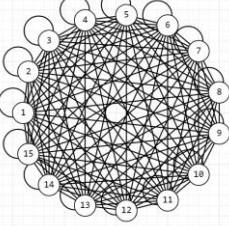
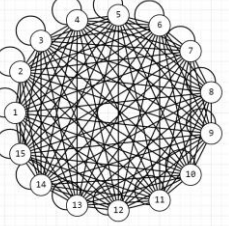
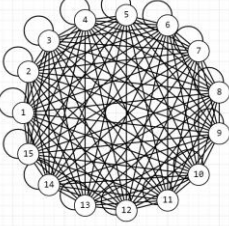
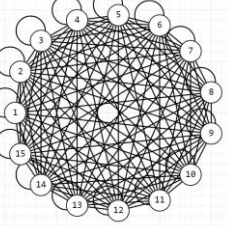
			max		
		Алгоритм Уоршалла	min		
			max		
	N ² /4	Алгоритм объединения степеней	min		
			max		
		Алгоритм Уоршалла	min		

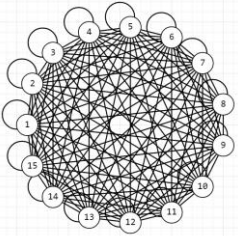
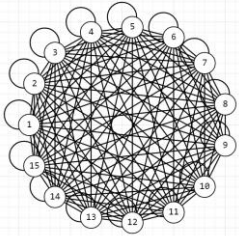
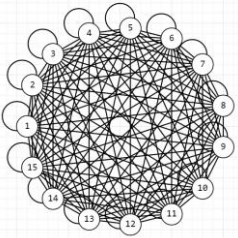
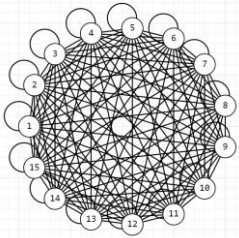
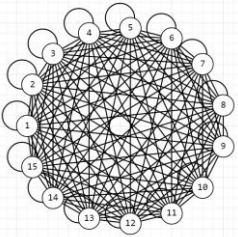
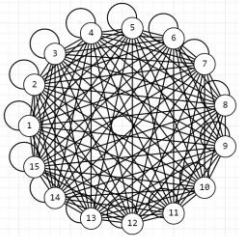
			max		
$N^2/2$	Алгоритм объединения степеней	min			
		max			
	Алгоритм Уоршалла	min			
		max			
	$N^2 \cdot (2/3)$	min			
		max			

				max		
		Алгоритм Уоршалла		min		
				max		
	N^2	Алгоритм объединения степеней		min		
				max		
		Алгоритм Уоршалла		min		

			max		
15	1	Алгоритм объединения степеней	min		
			max		
		Алгоритм Уоршалла	min		
			max		
	N^2/4	Алгоритм объединения степеней	min		

			max		
		Алгоритм Уоршалла	min		
			max		
	N ² /2	Алгоритм объединения степеней	min		
			max		
		Алгоритм Уоршалла	min		

			max		
$N^{2*(2/3)}$	Алгоритм объединения степеней	min			
		max			
	Алгоритм Уоршалла	min			
		max			
	Алгоритм объединения степеней	min			
N^2	Алгоритм объединения степеней	min			

			max		
		Алгоритм Уоршалла	min		
			max		

4. Определить порядок функции временной сложности алгоритмов вычисления транзитивного замыкания.

Алгоритм объединения степеней – $O(n^4)$,
Алгоритм Уоршалла – $O(n^3)$.

Вывод

Вывод: в ходе работы были изучен и выполнен сравнительный анализ алгоритмов вычисления транзитивного замыкания отношения.