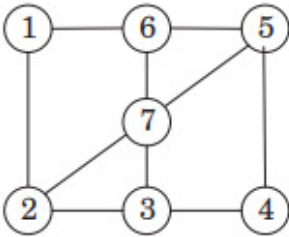


# 1. Цель работы

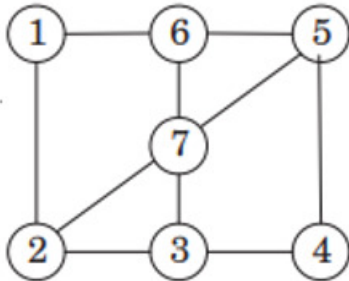
Получение практических навыков оценки надежности вычислительных сетей.

## 2. Вариант задания

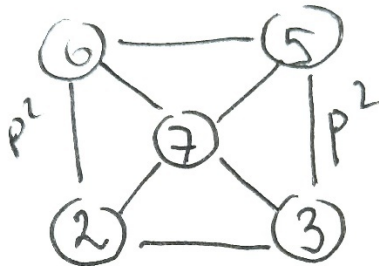
Вариант 14		$x_i=2$ $x_j=5$
---------------	---	--------------------

## 3. Вывод формулы вероятности существования пути в случайном графе.

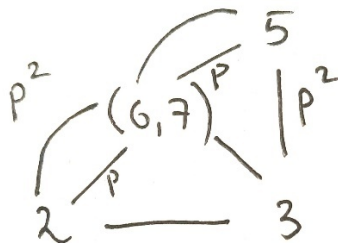
Исходный граф:



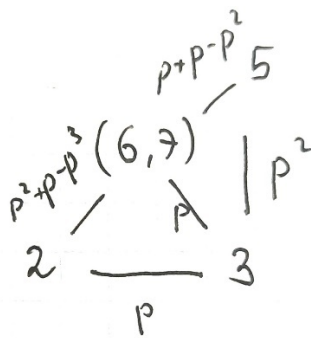
- 1) Упрощение структуры графа  
соединение ребер (2,1) и (1,6) в одно (2,6)  
соединение ребер (3,4) и (4,5) в одно (3,5)



- 2) Декомпозиция ребра (6,7)
- 2.1) (6,7) есть с вероятностью  $p$

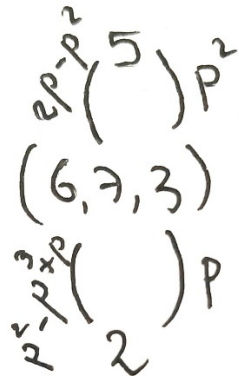


- 2.1.1) Упрощение структуры графа  
Соединение двух ребер (2,(6,7)) в одно  
Соединение двух ребер ((6,7),5) в одно

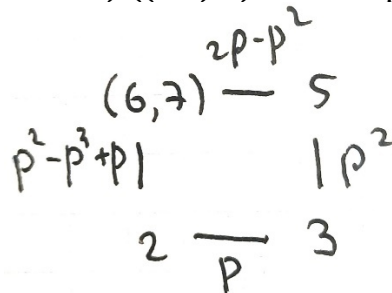


2.1.2) Декомпозиция ребра  $((6,7), 3)$

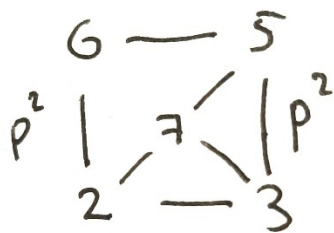
2.1.2.1)  $((6,7),3)$  есть с вероятностью  $p$



2.1.2.2)  $((6,7),3)$  нет с вероятностью  $1-p$

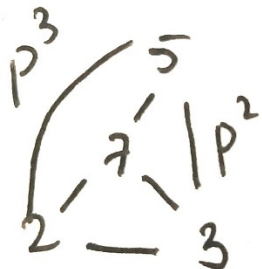


2.2)  $(6,7)$  нет с вероятностью  $(1-p)$



2.2.1) Упрощение структуры графа

соединение ребер  $(2,6)$  и  $(6,5)$  в одно  $(2,5)$

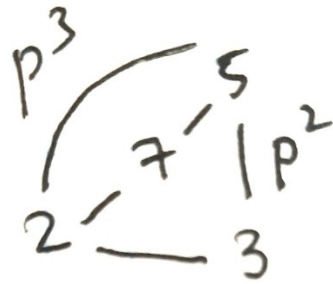


2.2.2) Декомпозиция ребра  $(7,3)$

2.2.2.1)  $(7,3)$  есть с вероятностью  $p$

$$p^3 \binom{(5)p^2}{(7,3)}_2$$

2.2.2.2) (7,3) нет с вероятностью  $1-p$



Формула вероятности:

4. Описание программы вычисления вероятности существования пути в случайном графе с использованием алгоритма полного перебора.  
 Результаты работы программы.  
 Алгоритм работы программы:



Результаты работы программы:

```
p=0
enumeration: 0
formula: 0

p=0.1
enumeration: 0.0139995
formula: 0.0139995

p=0.2
enumeration: 0.0704112
formula: 0.0704112

p=0.3
enumeration: 0.182481
formula: 0.182481

p=0.4
enumeration: 0.346271
formula: 0.346271

p=0.5
enumeration: 0.539063
formula: 0.539063

p=0.6
enumeration: 0.725971
formula: 0.725971

p=0.7
enumeration: 0.872984
formula: 0.872984

p=0.8
enumeration: 0.961231
formula: 0.961231

p=0.9
enumeration: 0.99538
formula: 0.99538

p=1
enumeration: 1
formula: 1
```

## 5. Графики зависимости $p_{ij}(p)$

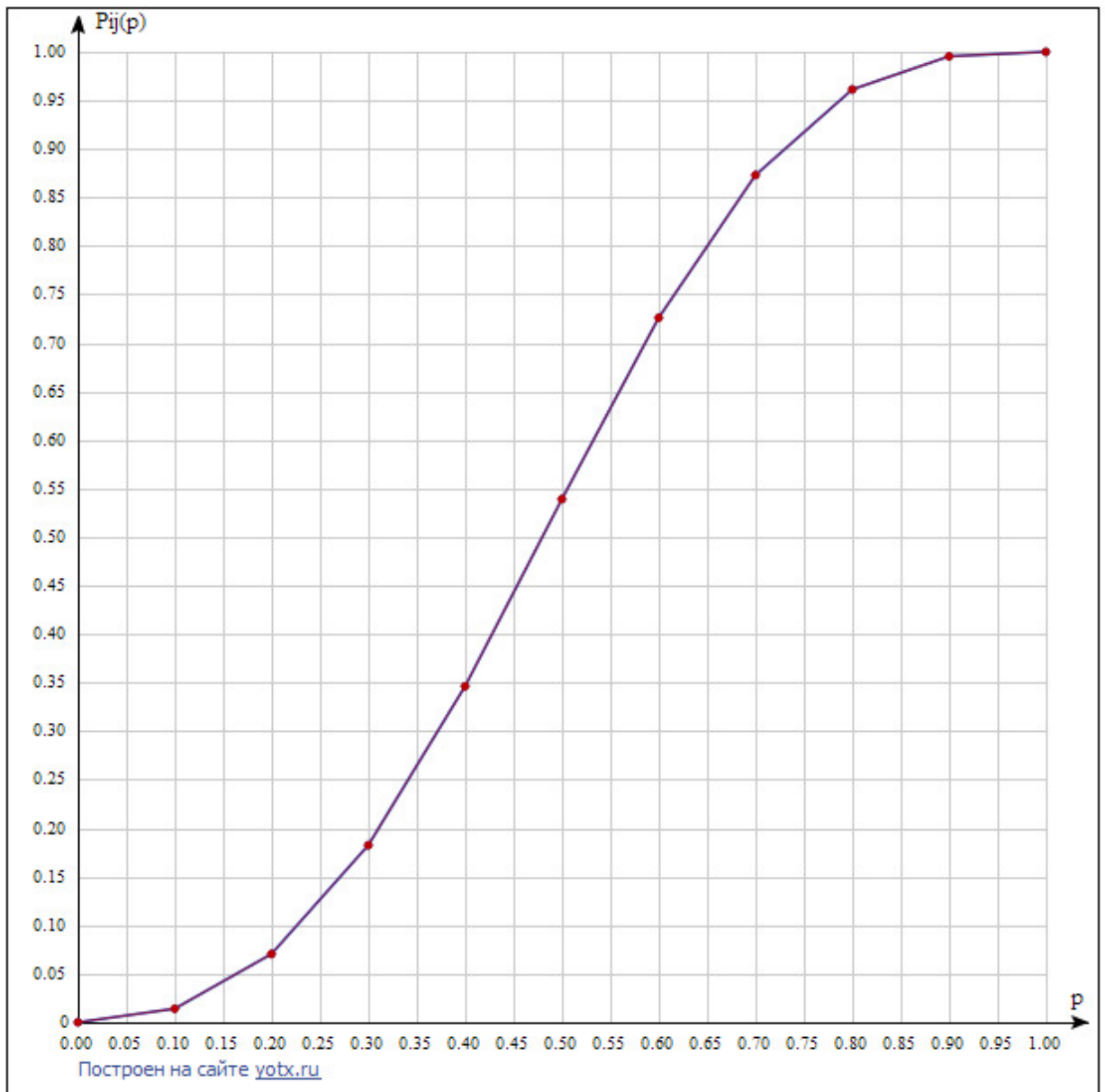


Рисунок 1 - Графики зависимостей  $P_{ij}(p)$ . Синяя линия - полный перебор, красная - выведенная формула

## 6. Выводы

В ходе лабораторной работы были получены вероятности существования пути в графе методом полного перебора графа и методом упрощения графа с использованием декомпозиции и перебора по путям. Вероятности данных методов совпадают.

## 7. Листинг кода

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <vector>
#include <bitset>
#include <fstream>

#define EDGES 10
#define VERTEX 7
```

```

using namespace std;
class Graph {
public:
    double probability = 0;
    bool* visited;
    bool isWay = false;
    vector<int> o;

    int mtx[7][7] = {
        {0,1,0,0,0,1,0},
        {1,0,1,0,0,0,1},
        {0,1,0,1,0,0,1},
        {0,0,1,0,1,0,0},
        {0,0,0,1,0,1,1},
        {1,0,0,0,1,0,1},
        {0,1,1,0,1,1,0},
    };

    Graph() {
        visited = new bool[VERTEX];
        for (int i = 0; i < VERTEX; i++) {
            visited[i] = 0;
        }
        for (int i = 0; i < VERTEX - 1; i++) {
            for (int j = i + 1; j < VERTEX; j++) {
                o.push_back(mtx[i][j]);
            }
        }
    }

    void fill() {
        for (int i = 0; i < VERTEX; i++) {
            for (int j = 0; j < VERTEX; j++) {
                cin >> mtx[i][j];
            }
        }
    }

    void dfs(int st, int en) {
        visited[st] = true;
        if (st == en) {
            isWay = true;
            return;
        }
        for (int i = 0; i < VERTEX; i++) {
            if ((mtx[st][i] == 1) && (visited[i] == false)) {
                dfs(i, en);
            }
        }
    }

    void track(int v1, int v2, double p) {
        probability = 0;
        for (int k = 0; k < pow(2, EDGES); k++) {
            bitset<EDGES> bs(k);
            double prop = 1.0;
            int idx = 0;
            int oidx = 0;
            for (int i = 0; i < VERTEX - 1; i++) {
                for (int j = i + 1; j < VERTEX; j++) {
                    if (o[oidx] == 1) {
                        mtx[i][j] = bs[idx];
                        mtx[j][i] = bs[idx];
                        prop *= bs[idx] ? p : (1.0 - p);
                        idx++;
                    }
                    else {
                        mtx[i][j] = 0;
                        mtx[j][i] = 0;
                    }
                }
            }
        }
    }
}

```

```

        oidx++;
    }
}
isWay = false;
dfs(v1, v2);
if (isWay) {
    probability += prop;
}
for (int i = 0; i < VERTEX; i++) {
    visited[i] = 0;
}
}
}
};
double formula(double p) {
    return 4 * pow(p, 10) - 14 * pow(p, 9) + 12 * pow(p, 8) + 4 * pow(p, 7) - pow(p, 6) - 10 * pow(p, 5) + pow(p,
4) + 4 * pow(p, 3) + p * p;
}
double formuladan(double p) {
    return 3 * pow(p, 9) - 13 * pow(p, 8) + 17 * pow(p, 7) - pow(p, 6) - 11 * pow(p, 5) + 2 * pow(p, 4) + 3 *
pow(p, 3) + p * p;
}
int main()
{
    Graph g = Graph();
    ofstream out;
    out.open("out.txt");
    for (double p = 0; p <= 1; p += 0.1) {
        g.track(1, 4, p); //1 4
        if (out.is_open()) {
            out << "p=" << p << endl;
            out << "enumeration: " << g.probability << endl;
            out << "formula: " << formula(p) << endl;
            out << endl;
        }
        cout << "p=" << p << endl;
        cout << "enumeration: " << g.probability << endl;
        cout << "formula: " << formula(p) << endl;
        cout << endl;
    }
    out.close();
}

```