

1) Цель работы

Получение практических навыков оценки надежности вычислительных сетей.

2) Вариант задания: 12

Задан случайный граф $G(X,Y,P)$, где $X = \{x_i\}$ – множество вершин, $Y = \{(x_i, x_j)\}$ – множество ребер, $P = \{p_{ij}\}$ – множество вероятностей существования ребер. Вероятности существования ребер равны между собой и равны p . В ходе выполнения лабораторной работы необходимо выполнить следующие действия.

1) Вычислить вероятность существования пути между заданной парой вершин $x_i = 2$, $x_j = 4$ в графе G .

2) Построить зависимость вероятности существования пути в случайном графе от вероятности существования ребра..

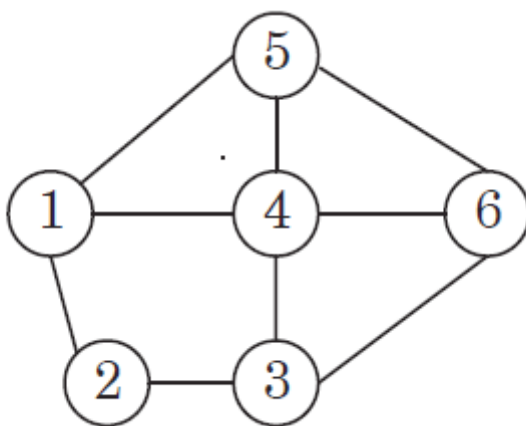


Рис. 1 – Топология исходного графа

3) Вывод формулы существования пути методом декомпозиции

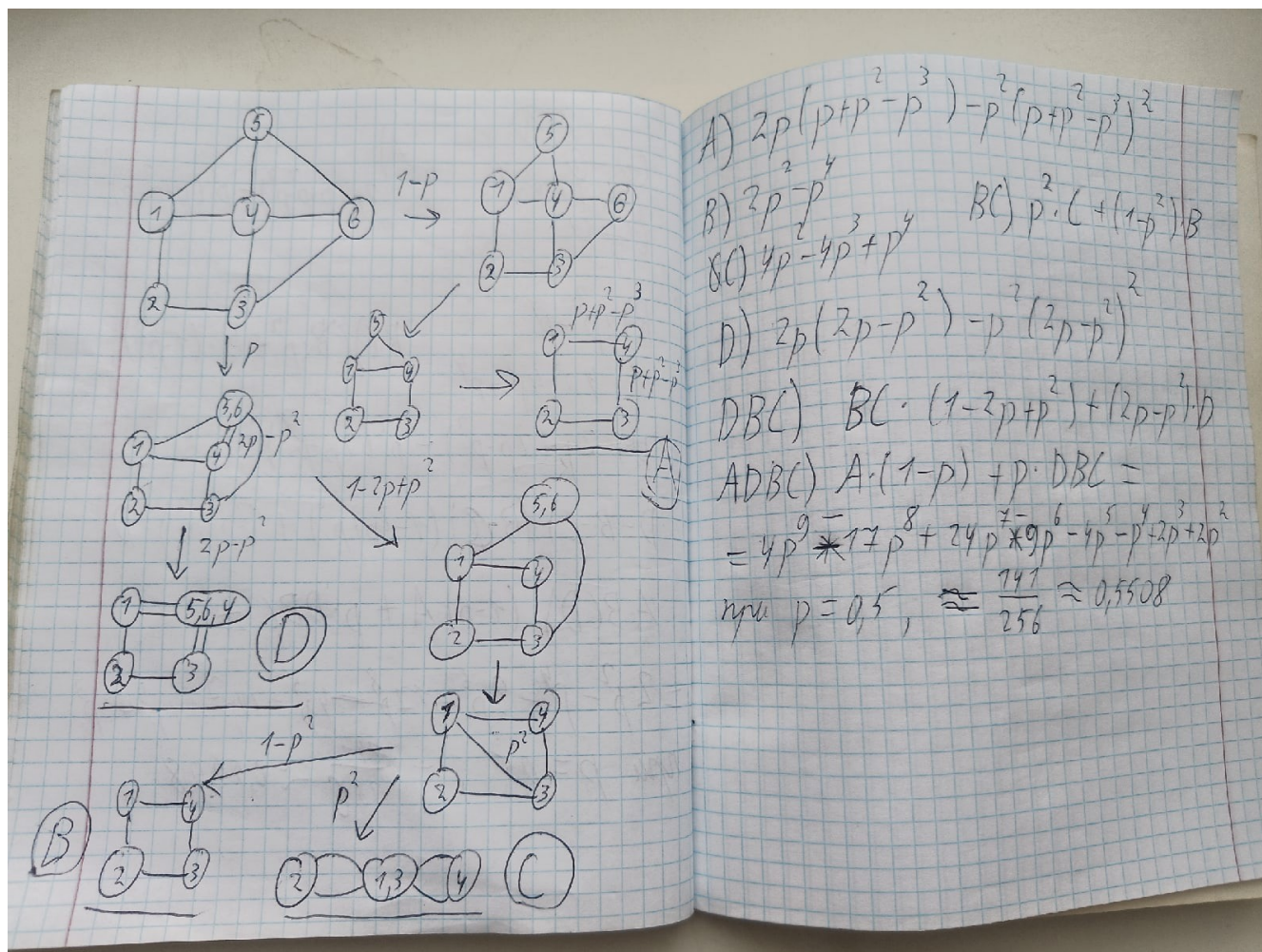


Рис. 2 – Вычисление искомой формулы

Итоговая формула принимает вид

$$P = 4p^9 - 17p^8 + 24p^7 - 9p^6 - 4p^5 - p^4 + 2p^3 + 2p^2$$

4) Описание программы

Программа выполняет вычисление вероятности наличия искомого пути в два шага: в начале выполняется перебор всех возможных графов при заданной топологии, в которых существует путь между заданными вершинами, что проверяется с помощью обхода в глубину (DFS), после суммируются вероятности у найденных графов.

Также программа выполняет подсчет по формуле найденной в пункте (3).

Листинг исходного кода представлен в Приложении 1.

| | | |
|-------------------|--------------------|--|
| p = 0 | | |
| probability real: | 0 | |
| probability ther: | 0 | |
| | | |
| p = 0.1 | | |
| probability real: | 0.021853234 | |
| probability ther: | 0.021853234 | |
| | | |
| p = 0.2 | | |
| probability real: | 0.0928097280000002 | |
| probability ther: | 0.092809728 | |
| | | |
| p = 0.3 | | |
| probability real: | 0.213831162 | |
| probability ther: | 0.213831162 | |
| | | |
| p = 0.4 | | |
| probability real: | 0.373805056 | |
| probability ther: | 0.373805056 | |
| | | |
| p = 0.5 | | |
| probability real: | 0.55078125 | |
| probability ther: | 0.55078125 | |
| | | |
| p = 0.6 | | |
| probability real: | 0.718078464000002 | |
| probability ther: | 0.718078464 | |
| | | |
| p = 0.7 | | |
| probability real: | 0.852680458 | |
| probability ther: | 0.852680458 | |
| | | |
| p = 0.8 | | |
| probability real: | 0.942292992 | |
| probability ther: | 0.942292991999999 | |
| | | |
| p = 0.9 | | |
| probability real: | 0.987835986 | |
| probability ther: | 0.987835985999999 | |
| | | |
| p = 1 | | |
| probability real: | 1 | |
| probability ther: | 1 | |

Рис. 3 – Результат работы программы

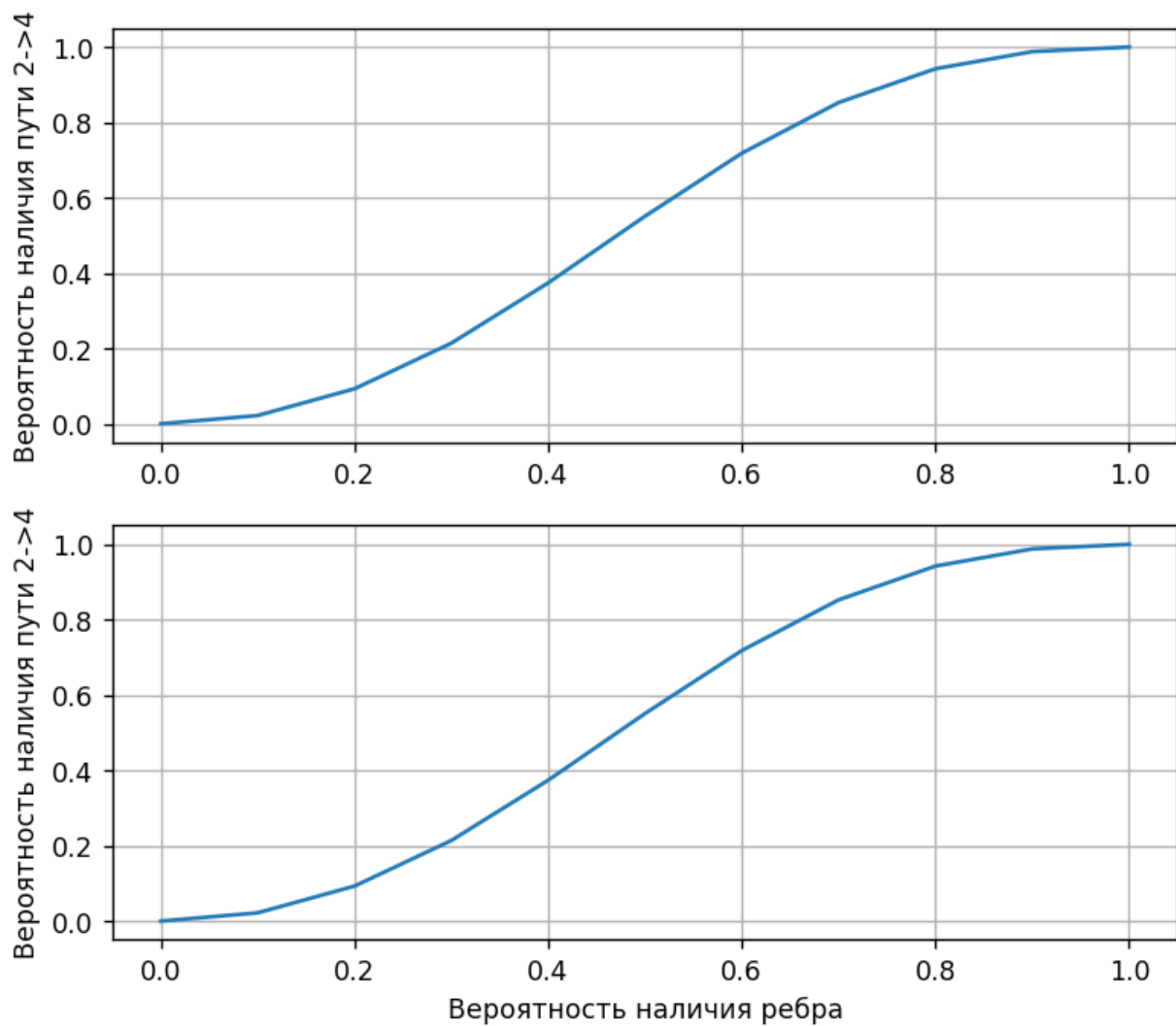


Рис. 4 – Графики зависимости наличия искомого пути от вероятности существования ребра по формуле и расчету соответственно

5) Вывод

Результаты полного перебора и формулы сошлись с точностью до 6 знака после запятой, что свидетельствует о правильности расчётов и высокой точности вычисления при использовании полного перебора

Листинг исходного кода на языке C++

```
#include <iostream>
#include <cmath>
#include <iomanip>

int mask[1000] = { 0 };
int matrix[11][11] = { 0,0 };
double sum[11] = {};

constexpr int N = 6;
constexpr int m[N][N] = {
    {0,1,0,1,1,0},
    {0,0,1,0,0,0},
    {0,0,0,1,0,1},
    {0,0,0,0,1,1},
    {0,0,0,0,0,1},
    {0,0,0,0,0,0}
};

bool vis[10] = { 0 };
void dfs(int i, int max_edge_count) {
    for (int j = 0; j < max_edge_count; j++)
        if (matrix[i][j] == 1 && vis[j] == 0) { vis[j] = 1; dfs(j,
max_edge_count); }
}

void brute_force(int x, int y, int max_edge_count){
    int flag = 0, mzx = pow(2, max_edge_count) - 1;
    while (flag != mzx) {
        for (int i = 0; i < N; i++) vis[i] = 0;

        for (long i = max_edge_count - 1, carry = 1; i >= 0; i--)
            if (carry) {
                if (mask[i]) mask[i] = 0;
                else { mask[i] = 1; carry = 0; }
            }

        int st = 0;
        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++)
                if (m[i][j]) {
                    matrix[i][j] = mask[st];
                    matrix[j][i] = mask[st++];
                }

        vis[x] = 1;
        dfs(x, max_edge_count);

        double pr[11] = { 1,1,1,1,1,1,1,1,1,1,1 };
    }
}
```

```

        if (vis[y]) {
            for (int i = 0; i < max_edge_count; i++)
                if (!mask[i]) for (double i = 0, pi = .0; i <
11; i++, pi += .1) pr[(int)i] *= 1.0 - pi;
                else for (double i = 0, pi = .0; i < 11; i++, pi
+= .1) pr[(int)i] *= pi;

            for (int i = 0; i < 11; i++) sum[i] += pr[i];
        }
        flag++;
    }
}

int main() {
    int x, y, max_edge_count = 0;
    std::cin >> x >> y;

    for (int i = 0; i < N; i++)
        for (int j = i + 1; j < N; j++)
            if (m[i][j] == 1) max_edge_count++;

    brute_force(x, y, max_edge_count);

    double sumPR[11] = {};
    double ks[] = { 0, 0, 2, 2, -1, -4, -9, 24, -17, 4 };
    for (double i = 0, pi = .0, cur_p = 1; i < 11; i++, pi += .1, cur_p = 1)
        for (int ii = 0; ii < 10; ii++, cur_p *= pi) sumPR[(int)i] +=
cur_p * ks[ii];

    std::cout << std::setprecision(6);
    for (double i = 0, pi = .0; i < 11; i++, pi += .1)
        std::cout << "p = " << pi << "\nprobability real:\t" <<
sum[(int)i]
        << "\nprobability ther:\t" << sumPR[(int)i] << "\n\n";
}

```