1. **Цель работы**

   Используя имитационное моделирование, в случайном графе вычислить вероятность существования пути между заданной парой вершин. Построить зависимость вероятности существования пути в случайном графе от вероятности существования ребра. Сравнить результаты моделирования, ускоренного моделирования и полного перебора.

2. **Задание**

   На рисунке 1 изображён случайный граф. Ищем вероятность существования пути из вершины 1 в вершину 6. P1=P2=…=P9

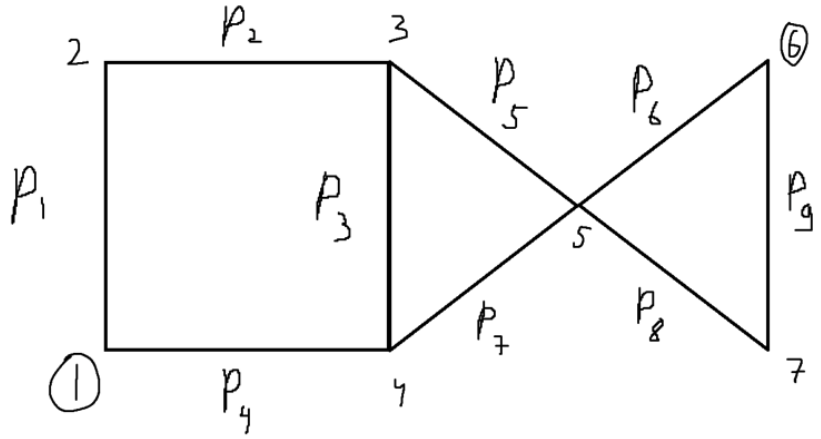

Рисунок 1. Случайный граф

3. **Выполнение задания**

   Для нахождения вероятности существования пути с помощью имитационного моделирования, нужно задать количество проводимых экспериментов N.

   N определяется по формуле N = 9/4 * E^2, где E – это точность моделирования. E примем как 0.1 и 0.01

P ребра	P пути через полный перебор	Моделирование при E = 0.1	Моделирование при E = 0.01
0	0	0	0
0.1	0.001303858	0.0089285714286	0.001511
0.2	0.012724736	0.0133928571429	0.012177
0.3	0.049272894	0.0446428571428	0.049244
0.4	0.1265950719	0.1205357142857	0.124266
0.5	0.25390625	0.2455357142857	0.250177
0.6	0.4272583679	0.4017857142857	0.425422
0.7	0.625948246	0.6448214285714	0.623955
0.8	0.814383104	0.8246428571428	0.811288
0.9	0.9505124819	0.9419642857142	0.958666
1	1	1	1

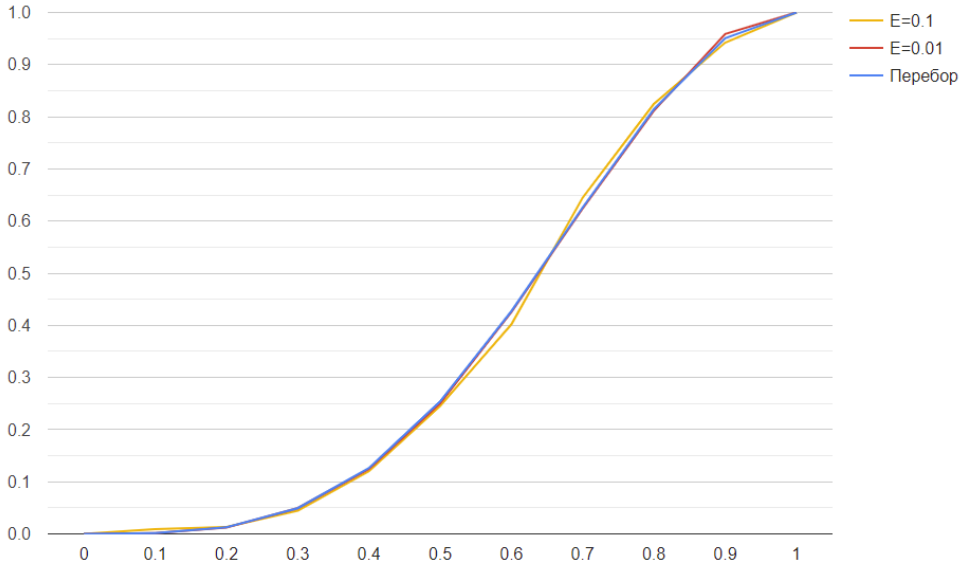

График 1. Зависимость P пути от P ребра
при полном переборе и имитационном моделировании

Далее используем алгоритм ускоренного моделирования, представленный на рисунке 2.

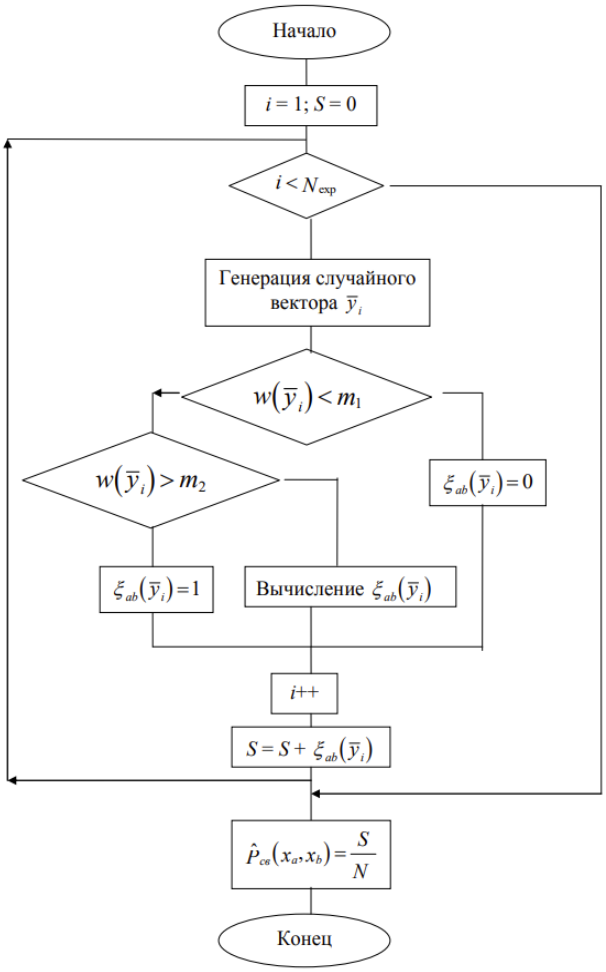

Рисунок 2. Алгоритм ускоренного моделирования

Для рассматриваемого графа m1 = 3, m2 = 7

P ребра	P пути через полный перебор	Моделирование при E = 0.1	Моделирование при E = 0.01
0	0	0	0
0.1	0.001303858	0	0.000622
0.2	0.012724736	0.008928571	0.009911
0.3	0.049272894	0.035714286	0.044844
0.4	0.1265950719	0.138392857	0.122444
0.5	0.25390625	0.276785714	0.250977
0.6	0.4272583679	0.424107143	0.427022
0.7	0.625948246	0.638392857	0.626044
0.8	0.814383104	0.803571429	0.814844
0.9	0.9505124819	0.950892857	0.9512
1	1	1	1

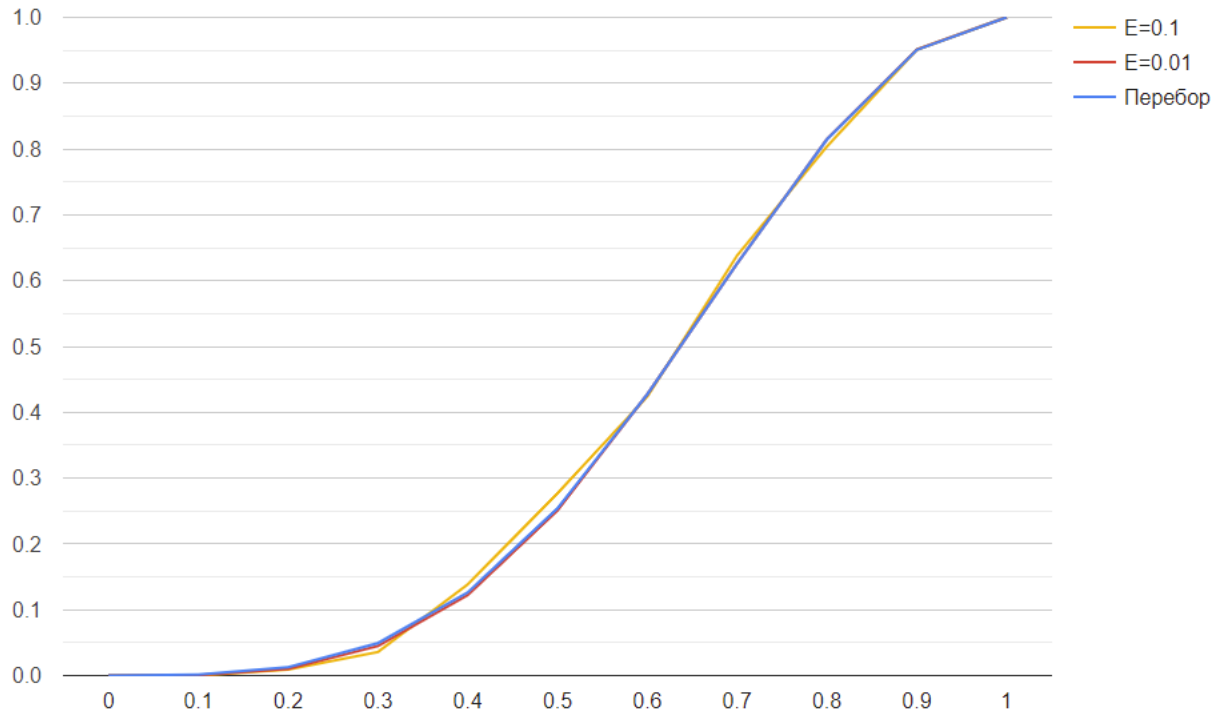

График 2. Зависимость P пути от P ребра
при полном переборе и ускоренном имитационном моделировании

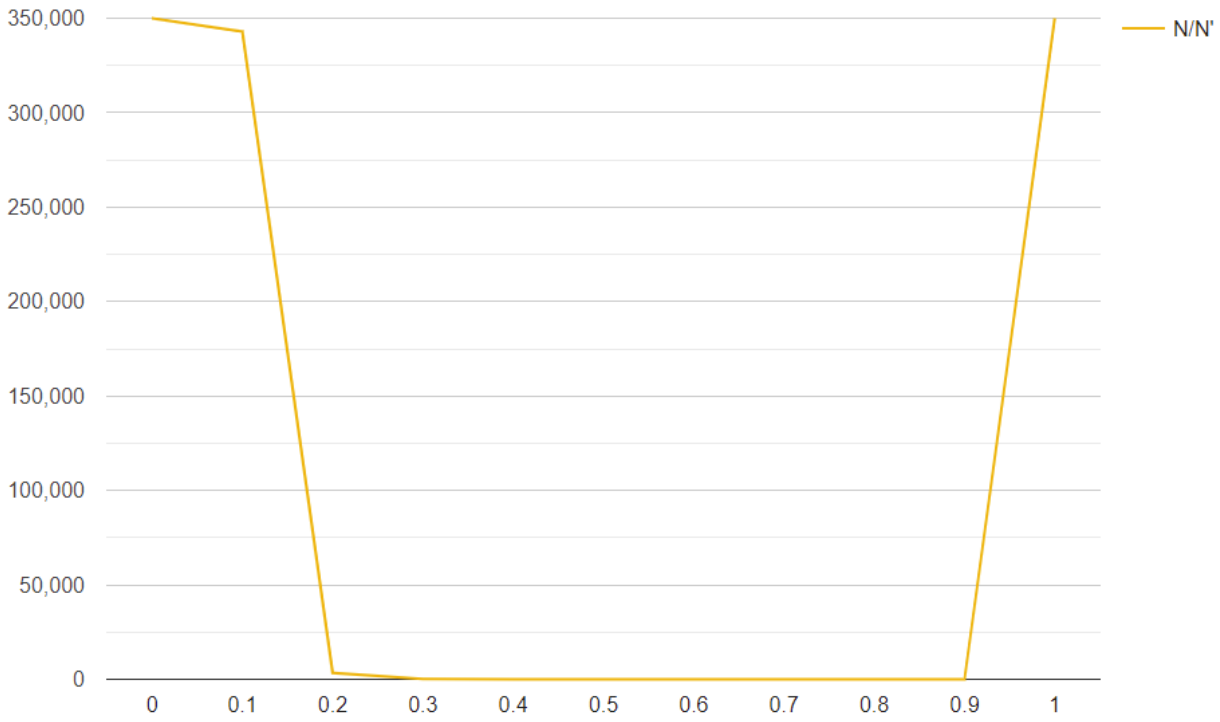

График 3. Выигрыш при использовании ускоренного моделирования (350000 тут принято за бесконечность)

4. Вывод

В ходе работы была выведена вероятность существования пути между заданной парой вершин с помощью обычного и ускоренного имитационного моделирования, правильность которой была подтверждена результатами программного полного перебора всех возможных подграфов случайного графа. Также был составлен график выигрыша ускоренного имитационного моделирования над обычным.

```
public static void main(String args[]) {
        int n = 7;
        int l = 9;
        int from = 0;
        int find = 5;
        int lMin = 3;
        int lMax = l - 2;

        double p;
        double tmP = 0;
        LinkedList<Double> P = new LinkedList<>();

        for (int i = 0; i <= 10; ++i) {
            P.add(tmP);
            tmP += 0.1;
        }

        LinkedList<Pair> lp = new LinkedList<>();
        lp.add(new Pair(0, 1)); //p1
        lp.add(new Pair(0, 3)); //p2
        lp.add(new Pair(1, 2)); //p3
        lp.add(new Pair(2, 3)); //p4
        lp.add(new Pair(2, 4)); //p5
        lp.add(new Pair(3, 4)); //p6
        lp.add(new Pair(4, 5)); //p7
        lp.add(new Pair(4, 6)); //p8
        lp.add(new Pair(5, 6)); //p9

        Graph subGraph = new Graph(n);

        subGraph.addVertex('A'); //1
        subGraph.addVertex('B'); //2
        subGraph.addVertex('C'); //3
        subGraph.addVertex('D'); //4
        subGraph.addVertex('E'); //5
        subGraph.addVertex('F'); //6
        subGraph.addVertex('G'); //7

        Gray grayCode = new Gray(l);
        int count = 0;
        int trueCount = 0;
        double start = System.currentTimeMillis();
        double sum = 0;
        // полный перебор
        LinkedList<Double> bigP = new LinkedList<>();
        for (Double aDouble : P) {
            p = aDouble;
            for (int i = 0; i < grayCode.code.size(); ++i) {
                LinkedList<String> tmp = grayCode.code.get(i); //000000000

                for (int j = 0; j < l; ++j) {
                    if (tmp.get(j).equals("1")) {
                        subGraph.addEdge(lp.get(j).f, lp.get(j).s);
                        trueCount++;
                    }
                }
                if (subGraph.dfs(from, find)) {
                    count++;
                    sum += Math.pow(p, trueCount) * Math.pow((1-p), (l-trueCount));
```

```
                }

                //del
                subGraph.clear();
                trueCount = 0;
            }
            bigP.add(sum);
            sum = 0;
        }
        double end = System.currentTimeMillis();
        System.out.println();
        System.out.println("Всего " + grayCode.code.size() + " подграфов");
        System.out.println("Из них в " + count/11 + " есть путь от " + ++from + " до
" + ++find);
        for (int i = 0; i < P.size(); ++i) {
            System.out.println("Вероятность равна: " + bigP.get(i) + " при p равном "
+ P.get(i));
        }
        System.out.println("Время работы полного перебора: " + (end-start) + "ms");


        bigP.clear();
        double E = 0.1;
        int N = (int) (9/(4*E*E));
        RandY randY = new RandY();
        start = System.currentTimeMillis();

        LinkedList<Double> gain = new LinkedList<>();
        double C = 0;
        count = 0;
        Cmn cmn = new Cmn();
        for (Double aDouble : P) {
            p = aDouble;

            for (int g = lMin; g <= lMax; ++g) {
                C = cmn.Cpr(l, g) * Math.pow(p, g) * Math.pow((1 - p), (l - g));
            }
            gain.add(1/C);

            for (int i = 0; i < N; ++i) {
                LinkedList<Integer> code = randY.randCode(l, p);
                for (int j = 0; j < code.size(); ++j) {
                    if (code.get(j) == 1) {
                        subGraph.addEdge(lp.get(j).f, lp.get(j).s);
                        trueCount++;
                    }
                }
                //обычное моделирование

//                if (subGraph.dfs(from, find)) {
//                    count++;
//                    sum += 1;
//                }

                // ускоренное моделирование
                if (lMin < trueCount) {
                    if (subGraph.dfs(from, find)) {
                        count++;
                        sum += 1;
                    }
                }
                else if (lMax < trueCount) {
```

```
                    count++;
                    sum += 1;
                }

                //del
                subGraph.clear();
                trueCount = 0;
            }
            bigP.add(sum/N);
            sum = 0;
        }
        end = System.currentTimeMillis();

        System.out.println("------------------------------------------------------------
--");
        System.out.println("Всего " + grayCode.code.size() + " подграфов");
        for (int i = 0; i < P.size(); ++i) {
            System.out.println("Вероятность равна: " + bigP.get(i) + " при p равном "
+ P.get(i));
        }
        System.out.println("Время работы: " + (end-start) + "ms");


System.out.println("************************************************************");
        for (int i = 0; i < P.size(); ++i) {
            System.out.println("Выигрыш равен моделирования: " + gain.get(i) + " при
p равном " + P.get(i));
        }
    }
```