

Лабораторная работа №1 ОПРЕДЕЛЕНИЕ СЛУЧАЙНОГО ГРАФА

1. Цель работы

Вычислить вероятность существования пути между заданной парой вершин в графе. А также построить зависимость вероятности существования пути в случайном графе от вероятности существования ребра.

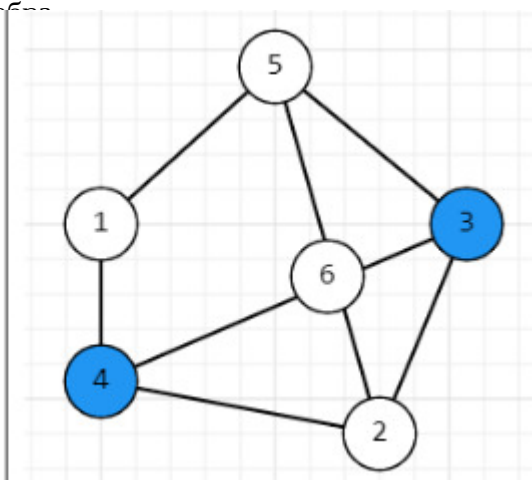
Вариант 22

2. Ход выполнения

2.1. Этап 1

Вывод формулы вероятности существования пути

1. Топология исходного графа



вероятности существования пути от р.

и 4, 3.

Рисунок 1 – Заданный граф

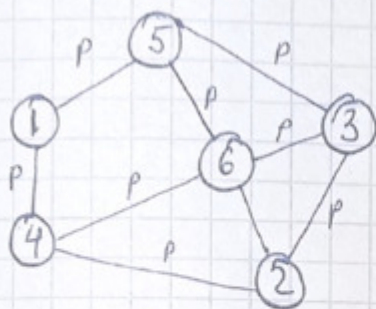
2. Процесс вывода формулы:

Лабораторная работа N1.

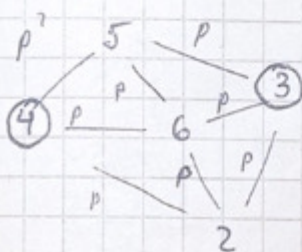
Вар. 22

Пути (4-3) ?

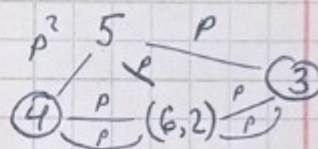
$2p-p^2$



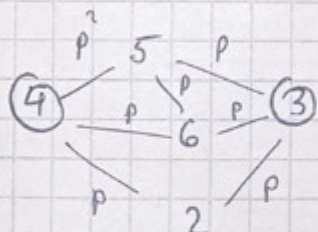
$\Downarrow (4,1) \cup (1,5)$



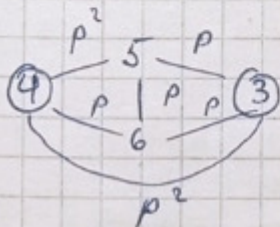
$(6,2)$ склеиваются



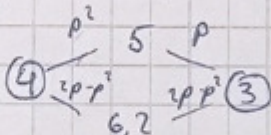
нет (6,2) \downarrow



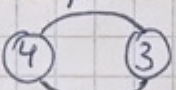
$\Downarrow (4,2) \cup (2,3)$



нет (5,6)



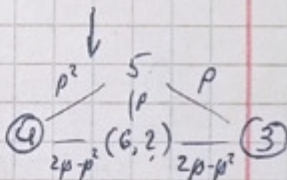
$p^3 \downarrow$



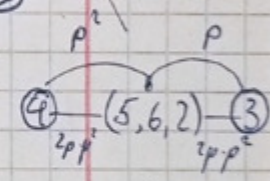
$$4p^2 - 4p^3 + p^4$$

$$4p^2 - 3p^3 + p^4 - 4p^5 + 4p^6 - p^7$$

III

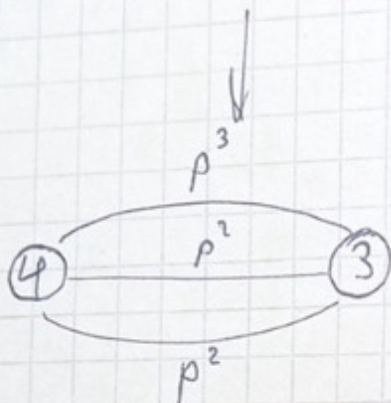
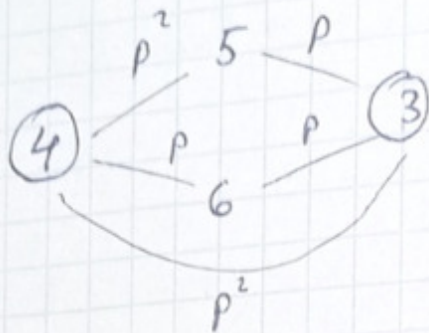


IV



$$2p - 2p^3 + p^4 - 3p^2 - 2p^3 + p^4$$

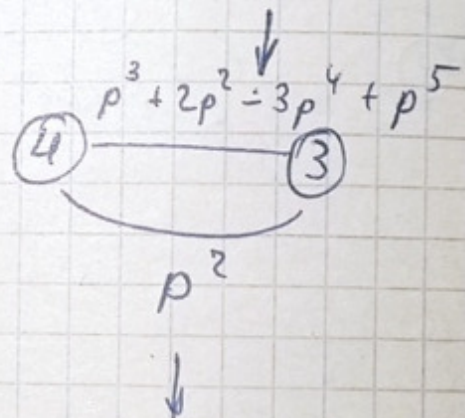
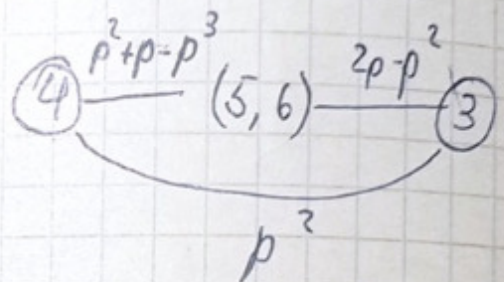
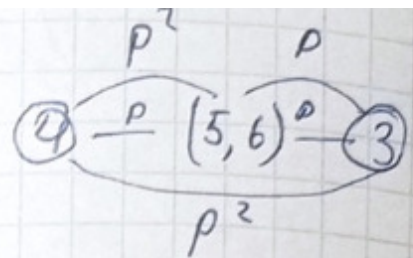
$$6p^2 - 6p^3 - 4p^4 + 6p^5 - 5p^6 + p^7$$



$$p^3 + 2p^2 - p^5 - p^5 - p^4 + p^7$$

Ⓘ

$Pr\{\text{нет } (6,2) \text{ и нет } (5,6)\}$



$$p^3 + 2p^2 - 3p^4 + p^5 - p^5 - 2p^4 + 3p^6 - p^7$$

Ⓜ

$Pr\{\text{нет } (6,2) \text{ и есть } (5,6)\}$


$$[\text{Ⓘ}(1-p) + \text{Ⓜ} \cdot p](1-p) + [\text{Ⓜ} \cdot (1-p) + \text{Ⓜ} \cdot p] \cdot p$$


```

Using edges: 1 2 3 4 5 7 8 Edge list: (1-5) (2-3) (2-4) (2-6) (3-5) (4-6) (5-6) true
Using edges: 1 2 3 4 6 7 8 Edge list: (1-5) (2-3) (2-4) (2-6) (3-6) (4-6) (5-6) true
Using edges: 1 2 3 5 6 7 8 Edge list: (1-5) (2-3) (2-4) (3-5) (3-6) (4-6) (5-6) true
Using edges: 1 2 4 5 6 7 8 Edge list: (1-5) (2-3) (2-6) (3-5) (3-6) (4-6) (5-6) true
Using edges: 1 3 4 5 6 7 8 Edge list: (1-5) (2-4) (2-6) (3-5) (3-6) (4-6) (5-6) true
Using edges: 2 3 4 5 6 7 8 Edge list: (2-3) (2-4) (2-6) (3-5) (3-6) (4-6) (5-6) true
Subgraph count = 36
Subgraphs with ways = 36
Result on current step: 0.04954883400000011
Using edges: 0 1 2 3 4 5 6 7 Edge list: (1-4) (1-5) (2-3) (2-4) (2-6) (3-5) (3-6) (4-6) true
Using edges: 0 1 2 3 4 5 6 8 Edge list: (1-4) (1-5) (2-3) (2-4) (2-6) (3-5) (3-6) (5-6) true
Using edges: 0 1 2 3 4 5 7 8 Edge list: (1-4) (1-5) (2-3) (2-4) (2-6) (3-5) (4-6) (5-6) true
Using edges: 0 1 2 3 4 6 7 8 Edge list: (1-4) (1-5) (2-3) (2-4) (2-6) (3-6) (4-6) (5-6) true
Using edges: 0 1 2 3 5 6 7 8 Edge list: (1-4) (1-5) (2-3) (2-4) (3-5) (3-6) (4-6) (5-6) true
Using edges: 0 1 2 4 5 6 7 8 Edge list: (1-4) (1-5) (2-3) (2-6) (3-5) (3-6) (4-6) (5-6) true
Using edges: 0 1 3 4 5 6 7 8 Edge list: (1-4) (1-5) (2-4) (2-6) (3-5) (3-6) (4-6) (5-6) true
Using edges: 0 2 3 4 5 6 7 8 Edge list: (1-4) (2-3) (2-4) (2-6) (3-5) (3-6) (4-6) (5-6) true
Using edges: 1 2 3 4 5 6 7 8 Edge list: (1-5) (2-3) (2-4) (2-6) (3-5) (3-6) (4-6) (5-6) true
Subgraph count = 9
Subgraphs with ways = 9
Result on current step: 0.2217357180000003
Using edges: 0 1 2 3 4 5 6 7 8 Edge list: (1-4) (1-5) (2-3) (2-4) (2-6) (3-5) (3-6) (4-6) (5-6) true
Subgraph count = 1
Subgraphs with ways = 1
Result on current step: 0.6091562070000003
Ended with 0.8999999999999999 probability

```

Рисунок 2 – Вывод работы программы

 Result.txt – Блокнот

Файл Правка Формат Вид Справка

```

0.1 0.023530744000000003
0.2 0.103500288000000004
0.300000000000000004 0.241045272000000006
0.4 0.419516416000000006
0.5 0.609375
0.6 0.778074624
0.7 0.901238968
0.7999999999999999 0.970801152
0.8999999999999999 0.996576696
0.9999999999999999 1.0

```

Рисунок 3 – Результат работы программы (полный перебор)

```
Current P = 0.1, current probability = 0.023530744000000006
Current P = 0.2, current probability = 0.103500288000000002
Current P = 0.30000000000000004, current probability = 0.2410452720000001
Current P = 0.4, current probability = 0.4195164160000001
Current P = 0.5, current probability = 0.609375
Current P = 0.6, current probability = 0.7780746239999999
Current P = 0.7, current probability = 0.901238968
Current P = 0.7999999999999999, current probability = 0.970801152
Current P = 0.8999999999999999, current probability = 0.996576696
Current P = 0.9999999999999999, current probability = 1.0000000000000009
```

Рисунок 4 – Результат работы программы (проверка полинома)

Критерием правильности выполнения ЛР является совпадение результатов первого и второго этапов. Данные, полученные разными способами, совпадают, исходя из этого, можем говорить о том, что лабораторная работа выполнена верно.

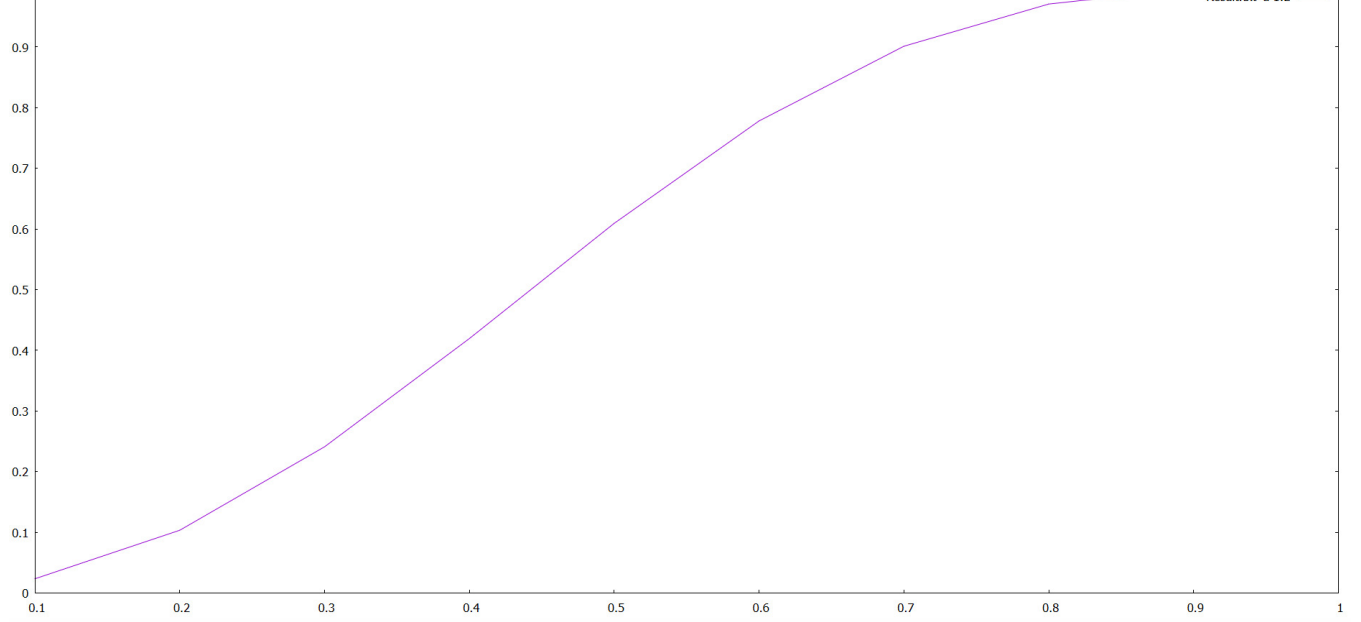


Рисунок 5 – График зависимости вероятности существования ребра от вероятности пути между заданными вершинам

3. Выводы

В ходе выполнения лабораторной работы были изучены способы вычисления вероятности существования пути между заданной парой вершин в графе. Научились находить вероятность полным перебором по подграфам случайного графа, а также методом декомпозиции. Результаты вычисления обоими способами совпали.

Помимо этого, построили зависимость вероятности существования пути в случайном графе от вероятности существования ребра. И сделали вывод о том, что с ростом вероятности существования ребра, увеличивается и вероятность пути между заданными вершинами.

Листинг программы

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.lang.Math.*;

public class Graph {

    private int n; //Вершины
    private int l; //Рёбра
    private int v1; //Вершина 1
    private int v2; //Вершина 2
    private double p; //Вероятность существования ребра
    private ArrayList<Pair> edgeList = new ArrayList<>(); //Массив рёбер
    private boolean[] visited;
    private double result;

    public Graph() {
        n = 6;
        l = 9;
        v1 = 3;
        v2 = 4;
        p = 0.1;
        visited = new boolean[n + 1];
        edgeList.add(new Pair(1, 4));
        edgeList.add(new Pair(1, 5));
        edgeList.add(new Pair(2, 3));
        edgeList.add(new Pair(2, 4));
        edgeList.add(new Pair(2, 6));
        edgeList.add(new Pair(3, 5));
        edgeList.add(new Pair(3, 6));
        edgeList.add(new Pair(4, 6));
        edgeList.add(new Pair(5, 6));
    }

    // Вероятность существования count подграфов с k ребрами в которых путь есть
    public void findProb(int count, int k) {
        double tmp = count * (Math.pow(p, k) * Math.pow((1 - p), 1 - k));
        result += tmp;
    }

    // Очистка посещённых вершин
    public void clearVisited() {
        for (int i = 0; i < visited.length; i++) {
            visited[i] = false;
        }
    }

    // Перебор подграфов
    private boolean findCombination(int[] a, int n, int k) {
        for (int i = k - 1; i >= 0; --i) {
            if (a[i] < n - k + i) {
                a[i]++;
                for (int j = i + 1; j < k; ++j) {
                    a[j] = a[j - 1] + 1;
                }
                return true;
            }
        }
        return false;
    }

    // Построение подграфа по массиву из номеров рёбер
    public ArrayList<Pair> createSubEdgeList(int[] a, int n) {
        ArrayList<Pair> subgraphEdgeList = new ArrayList<>();
        System.out.print("Using edges: ");
        for (int i = 0; i < n; i++) {
```



```

        subgraphEdgeList.add(edgeList.get(a[i]));
        System.out.print(a[i] + " ");
    }
    System.out.print("Edge list: ");
    for (Pair pair : subgraphEdgeList) {
        System.out.print(pair + " ");
    }
    return subgraphEdgeList;
}

// Проверка наличия пути
public boolean isConnect(int v1, int v2, boolean[] visited, ArrayList<Pair>
subEdgeList) {
    if (v1 == v2) {
        return true;
    }
    visited[v1] = true;
    for (int i = 0; i < subEdgeList.size(); i++) {
        int v = 0;
        if (subEdgeList.get(i).first() == v1) {
            v = subEdgeList.get(i).second();
        }
        if (subEdgeList.get(i).second() == v1) {
            v = subEdgeList.get(i).first();
        }
        if (v != 0 && !visited[v]) {
            if (isConnect(v, v2, visited, subEdgeList)) {
                return true;
            }
        }
    }
    return false;
}

public void printToFile() {
    try {
        FileWriter file = new FileWriter("Result.txt", true);
        StringBuilder str = new StringBuilder();
        str.append(p).append(" ").append(result).append("\n");
        file.write(str.toString());
        file.flush();
    } catch (IOException exception) {
        System.out.println(exception.getMessage());
    }
}

public void bruteForce() {
    while (p <= 1) {
        for (int k = 0; k <= 1; k++) {

            int num = 1;
            int count = 0;
            int[] edges = new int[1];
            for (int i = 0; i < 1; i++) {
                edges[i] = i;
            }

            boolean isConnected = isConnect(v1, v2, visited,
createSubEdgeList(edges, k));
            clearVisited();
            System.out.println(isConnected);
            if (isConnected) {
                count++;
            }

            if (1 >= k) {
                while (findCombination(edges, 1, k)) {
                    isConnected = isConnect(v1, v2, visited,

```

```

createSubEdgeList(edges, k));
        clearVisited();
        System.out.println(isConnected);
        if (isConnected) {
            count++;
        }
        num++;
    }
    }
    System.out.println("Subgraph count = " + num);
    System.out.println("Subgraphs with ways = " + count);
    System.out.println("Result on current step: " + result);
    findProb(count, k);
}

printToFile();
System.out.println("Ended with " + p + " probability");

result = 0;
p += 0.1;
}

}

public void count_polynom()
{
    double tmp;
    double p = 0.1;
    System.out.println("");
    System.out.println("");
    for(p = 0.1; p < 1; p+=0.1)
    {
        tmp = Math.pow(p, 9) * 4 + Math.pow(p, 8) * -16 + Math.pow(p, 7) * 19 +
Math.pow(p, 6) * -1 + Math.pow(p, 5) * -7 + Math.pow(p, 4) * -4 + Math.pow(p, 3) *
4 + Math.pow(p, 2) * 2;
        System.out.println("Current P = " + p + ", current probability = " +
tmp);
    }
}

}

```