

➤ Общее задание

Разработать класс для представления неориентированного графа и поиска всех циклов или путей между двумя заданными вершинами.

Должен использоваться метод поиска “в глубину”.

Класс должен включать объект класса "Стек" для представления стека.

Необходимо разработать класс "Стек" для представления стека.

Должен использоваться не рекурсивный алгоритм поиска путей или циклов.

Разработать программу на языке Java, реализующую конкретное задание.

Должны использоваться элементы библиотеки Swing.

Программа должна обеспечивать ввод описания графа с клавиатуры (с помощью мыши).

Программа должна обеспечивать представление исходного графа и результатов в графическом виде.

В программе должны использоваться меню и диалоговые окна.

Программа должна сначала найти и запомнить все пути или циклы, а затем показывать их по запросу.

В описаниях заданий используются следующие обозначения:

- стек_масс – стек на основе массива;
- стек_спис – стек на основе связанного списка;
- граф_матр – для описания графа используется матрица смежности;
- граф_масс – для описания графа используются списки в массивах;
- граф_спис – для описания графа используются связанные списки.

➤ Индивидуальное задание

Найти все самые длинные циклы в графе, не проходящие через заданную вершину (стек_масс, граф_матр).

➤ Описание работы программы

После запуска программы перед пользователем появляется окно. В строке меню две вкладки “Решение” и “Работа”. При наведении курсора мыши на вкладку “Работа” выпадает список с возможными действиями:

“Заданная вершина”, “Очистить”, “Добавить вершины”, “Добавить ребра”. Во вкладке “Решение” две - “Выбрать цикл” и “Изобразить цикл”.

Для начала пользователь выбирает “Добавить вершины”, затем кликает в области окна, на месте клика появляется круг с номером — вершина графа. После добавлении необходимого количества вершин, необходимо соединить наши вершины. Чтобы соединить, кликаем на иконку меню “Добавить ребра”, зажимаем мышью ту вершину, из которой необходимо провести ребро, и проводим до другой вершины, затем отпускаем. В итоге рисуется линия, которая соединяет две вершины графа. Как только граф готов, мы выбираем иконку “Заданная вершина” и вводим соответствующий номер вершины. Чтобы определить какие циклы нашла программа переходим во вкладку “Решение” → “Выбрать цикл”. Если цикл(-ы) найден(-ы), то можно выбрать тот, который нам нужно изобразить, если же циклов нет, то в окне будет отображено “Нет циклов”. Выбрав необходимый цикл, мы изображаем его нажатием на вкладку “Решение” → “Изобразить цикл”.

Программа предусматривает многократное использование, поэтому можно очистить полотно и задать новый граф.

➤ Алгоритмы выполнения основных операций на псевдокоде

Алгоритм поиска в глубину:

ЦИКЛ ($j=0, n-1$)

$M[j]=0$

КОНЕЦ_ЦИКЛ

$vk = vn$ — конец совпадает с началом

$ks=0$

St.Push(vn)

$M[vn]=1$

$L=0$

ЦИКЛ_ПОКА ($ks \geq 0$)

$v=St[ks]$

$Pr=0$

 ЦИКЛ ($j=L, n$)

 ЕСЛИ ($A[v,j]=1$) ТО

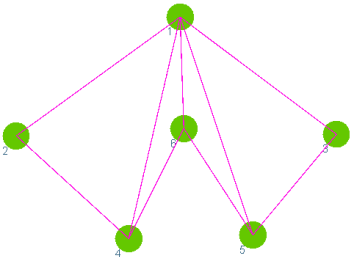
 ЕСЛИ ($j=vk$) ТО

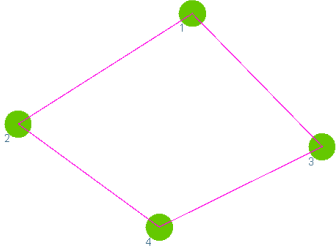
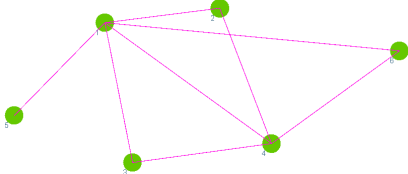
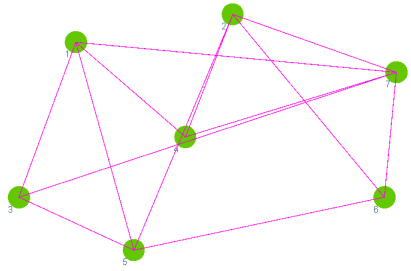
```

        ЕСЛИ ( t не принадлежит стеку ) ТО
            Вывод “Путь содержит вершины”
            Вывод (St[i],i=0,ks),vk
            КОНЕЦ_ЕСЛИ
        ИНАЧЕ
            ЕСЛИ ( M[j]=0 ) ТО
                Pr=1
                Прервать цикл по j
            КОНЕЦ_ЕСЛИ
        КОНЕЦ_ЕСЛИ
    КОНЕЦ_ЕСЛИ
КОНЕЦ_ЦИКЛ
ЕСЛИ ( Pr=1 ) ТО
    ks=ks+1
    St.SM_length = ks
    St.Push(j)
    L=0
    M[j]=1
ИНАЧЕ
    L=v+1
    M[v]=0
    ks=ks-1
    St.Pop()
КОНЕЦ_ЕСЛИ КОНЕЦ_ЦИКЛ

```

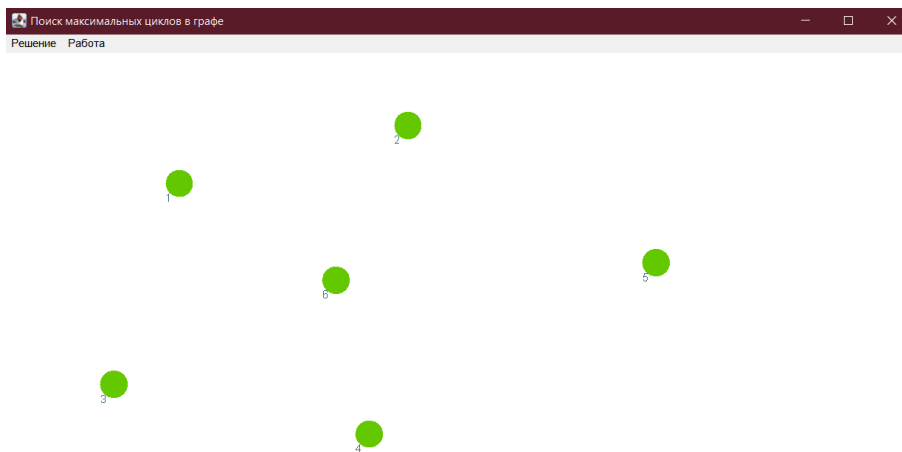
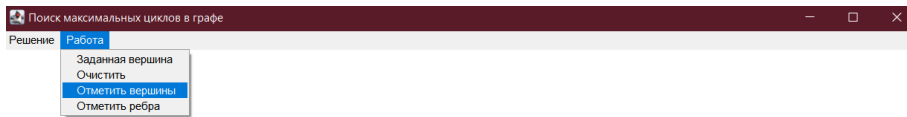
➤ Тесты

№	Входные данные	Выходные данные
1	 <p>Заданная вершина: 6</p>	<div style="border: 1px solid black; padding: 10px;"> <pre> 1)1->2->4->1-> 2)1->3->5->1-> </pre> </div>

2	 <p>Заданная вершина: 1</p>	<div data-bbox="751 219 1042 577" style="border: 1px solid black; padding: 5px;"> <p>Нет циклов</p> </div>
3	 <p>Заданная вершина: 5</p>	<div data-bbox="751 633 1114 992" style="border: 1px solid black; padding: 5px;"> <p>1)1->2->4->3->1-> 2)1->2->4->6->1-> 3)1->3->4->6->1-></p> </div>
4	 <p>Заданная вершина: 3</p>	<div data-bbox="751 1048 1225 1406" style="border: 1px solid black; padding: 5px;"> <p>1)1->4->2->5->6->7->1-></p> </div>

➤ Распечатки экранов при работе программы

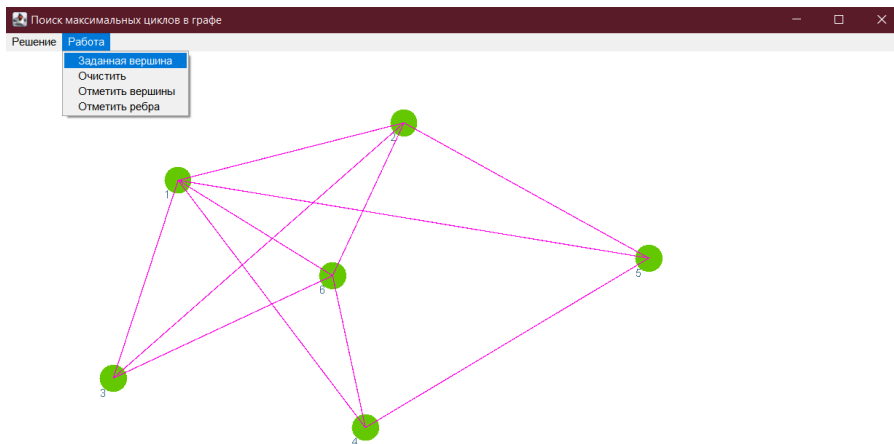
Изобразим 6 вершин:

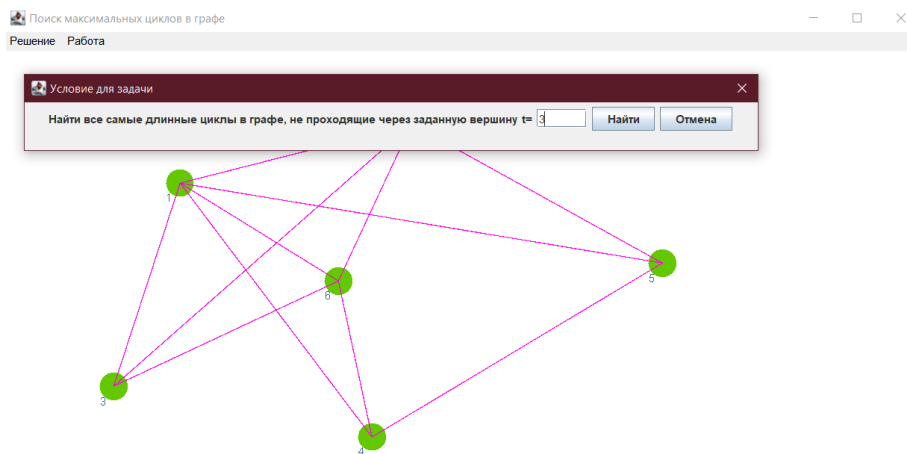


Соединим некоторые вершины:

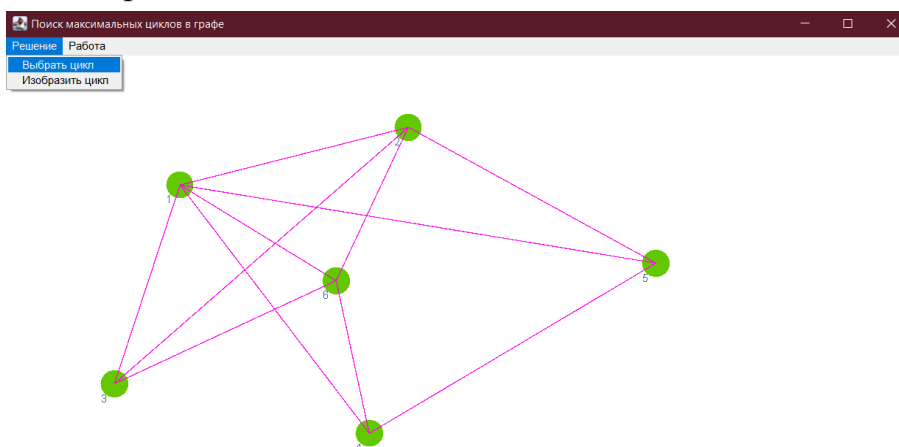


Зададим вершину, через которую циклы не должны проходить:

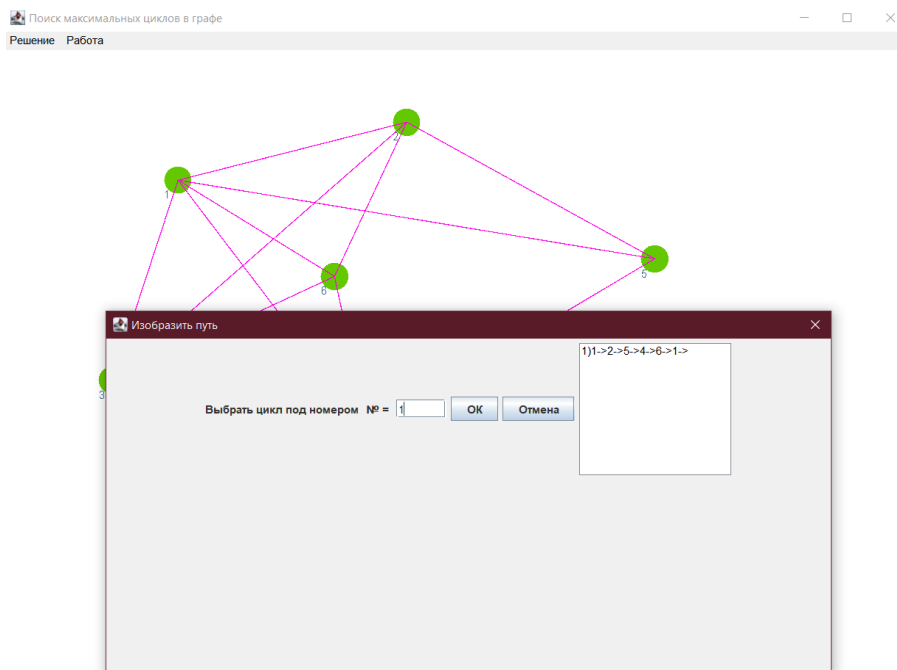




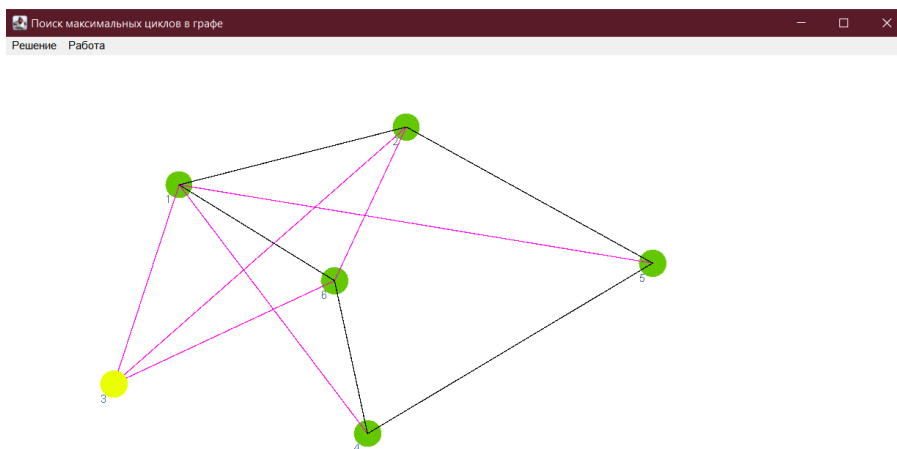
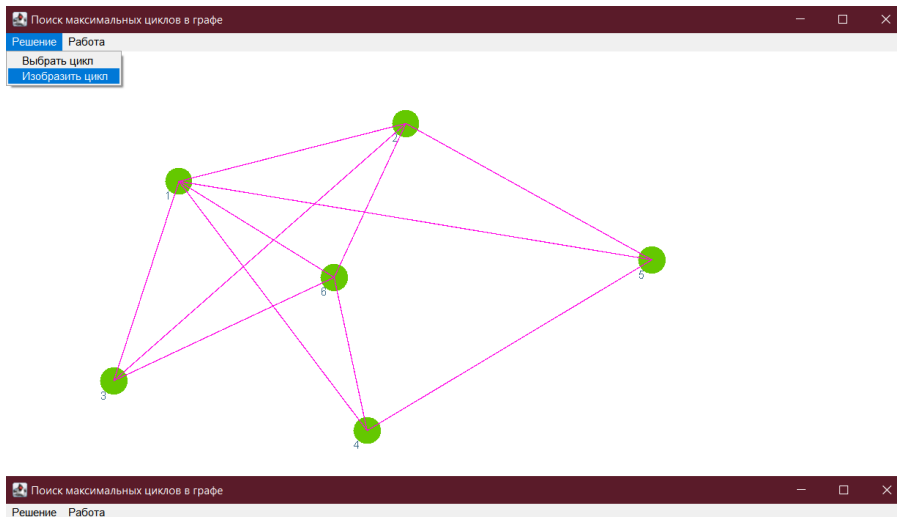
Посмотрим на найденные циклы:



Найден самый длинный путь, выберем его:



Изобразим:



➤ ЛИСТИНГ

Main.java

```
package LW_8;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Parametr {
    int t;
    boolean flagOk;
}

class Graf {
    int[][] Rez = new int[2550][2550];
    int kol = 0;
    int[][] M = new int[1000][1000];
    int[] L = new int[1000];
    Graph F = new Graph();
}

class Num_of_cycle {
    int numer;
    boolean flagOk;
}

class SolveDialog extends Dialog {
    JTextField tf3;
    List lst;

    SolveDialog(Graf g, Num_of_cycle par, Frame parent, String title) {
        super(parent, title, true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
                setVisible(false);
            }
        });
        StringBuilder way_in_cycle;
        int i, j;
        setLayout(new FlowLayout());
        setSize(800, 400);
        lst = new List(10);
        if (g.F.size_g != 0) {
            for (i = 0; i < g.F.size_g; i++) {
                way_in_cycle = new StringBuilder(i + 1 + "");
                j = 0;
                while (g.Rez[i][j] != 0) {
                    way_in_cycle.append(g.Rez[i][j]).append("->");
                    j++;
                }
            }
        }
    }
}
```

```

        }
        lst.add(way_in_cycle.toString());
    }
} else {
    lst.add("Нет циклов");
}

JButton btOk, btCancel;
JLabel lbT, lbS;
lbT = new JLabel("№ = ");
lbS = new JLabel("Выбрать цикл под номером ");
tf3 = new JTextField(5);
btOk = new JButton("OK");
btCancel = new JButton("Отмена");
add(lbS);
add(lbT);
add(tf3);
add(btOk);
add(btCancel);
add(lst);
btOk.addActionListener(new OkListener(par, this));
btCancel.addActionListener(new CancelListener(par));
}

class OkListener implements ActionListener {
    Num_of_cycle param;
    SolveDialog pd;

    OkListener(Num_of_cycle par, SolveDialog pard) {
        param = par;
        pd = pard;
    }

    public void actionPerformed(ActionEvent ae)
        throws NumberFormatException {
        int x;
        x = Integer.parseInt(tf3.getText());
        param.numer = x;
        param.flagOk = true;
        pd.dispose();
        pd.setVisible(false);
    }
}

class CancelListener implements ActionListener {
    Num_of_cycle param;

    CancelListener(Num_of_cycle par) {
        param = par;
    }

    public void actionPerformed(ActionEvent ae) {
        param.flagOk = false;
    }
}

```

```

        dispose();
        setVisible(false);
    }
}

class ParamDialog extends Dialog {
    JTextField tft;

    ParamDialog(Parametr par, Frame parent, String title) {
        super(parent, title, true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
                setVisible(false);
            }
        });
        setLayout(new FlowLayout());
        setSize(800, 100);
        JButton btOk, btCancel;
        JLabel lbT, lbS;
        lbT = new JLabel("t=");
        lbS = new JLabel("Найти все самые длинные циклы в графе, не прохо\n" +
            "дящие через заданную вершину");
        tft = new JTextField(5);
        btOk = new JButton("Найти");
        btCancel = new JButton("Отмена");
        add(lbS);
        add(lbT);
        add(tft);
        add(btOk);
        add(btCancel);
        btOk.addActionListener(new OkListener(par, this));
        btCancel.addActionListener(new CancelListener(par));
    }

    class OkListener implements ActionListener {
        Parametr param;
        ParamDialog pd;

        OkListener(Parametr par, ParamDialog pard) {
            param = par;
            pd = pard;
        }

        public void actionPerformed(ActionEvent ae)
            throws NumberFormatException {
            int x;
            x = Integer.parseInt(tft.getText());
            param.t = x;
            param.flagOk = true;
            pd.dispose();
        }
    }
}

```

```

        pd.setVisible(false);
    }
}

class CancelListener implements ActionListener {
    Parametr param;

    CancelListener(Parametr par) {
        param = par;
    }

    public void actionPerformed(ActionEvent ae) {
        param.flagOk = false;
        dispose();
        setVisible(false);
    }
}

}

// Класс главного окна приложения
class DialogDemo extends Frame {
    Graf G;
    Parametr t_task;
    Num_of_cycle Nofc;
    int[] y_vert = new int[1000];
    int[] x_vert = new int[1000];
    int num_vert = 0;
    int i = 1;

    DialogDemo(String title) {
        super(title);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
        t_task = new Parametr();
        Nofc = new Num_of_cycle();
        G = new Graf();

        MenuBar mb = new MenuBar();
        setMenuBar(mb);
        Menu draw = new Menu("Работа");
        Menu solve = new Menu("Решение");
        MenuItem draw_cycle = new MenuItem("Выбрать цикл");
        solve.add(draw_cycle);
        MenuItem drawCycle = new MenuItem("Изобразить цикл");
        solve.add(drawCycle);
        mb.add(solve);
        MenuItem parr = new MenuItem("Заданная вершина");
        draw.add(parr);
        MenuItem Cl = new MenuItem("Очистить");
        draw.add(Cl);
    }
}

```

```

MenuItem Mark_vert = new MenuItem("Отметить вершины");
draw.add(Mark_vert);
MenuItem Mark_edge = new MenuItem("Отметить ребра");
draw.add(Mark_edge);
mb.add(draw);

drawCycle.addActionListener(new Dcycle());
draw_cycle.addActionListener(new DrawCcl(this));
Cl.addActionListener(new Clean());
Mark_vert.addActionListener(new markVert());
Mark_edge.addActionListener(new MarkEdge());
parr.addActionListener(new HandParr(this));
}

class Dcycle implements ActionListener {
    int start, finish, j, k, n;
    boolean flag = true;

    public void actionPerformed(ActionEvent ae) {
        System.out.println(Nofc.number);
        Graphics g = getGraphics();

        for (k = 0; k < num_vert; k++) {
            for (n = 0; n < num_vert; n++) {
                if (G.F.G[k][n] == 1) {
                    j = 0;
                    flag = true;
                    while (G.Rez[Nofc.number - 1][j + 1] != 0 && flag) {
                        if (k == G.Rez[Nofc.number - 1][j] && n ==
G.Rez[Nofc.number - 1][j + 1] || n == G.Rez[Nofc.number - 1][j] && j ==
G.Rez[Nofc.number - 1][j + 1]) {
                            flag = false;
                        }
                        j++;
                    }
                    if (flag) {
                        g.setColor(new Color(255, 3, 230));
                        g.drawLine(x_vert[k], y_vert[k], x_vert[n],
y_vert[n]);
                    }
                }
            }
        }
        j = 0;
        g.setColor(new Color(0, 0, 0));
        while (G.Rez[Nofc.number - 1][j + 1] != 0) {
            start = G.Rez[Nofc.number - 1][j] - 1;
            finish = G.Rez[Nofc.number - 1][j + 1] - 1;
            g.drawLine(x_vert[start], y_vert[start], x_vert[finish],
y_vert[finish]);
            j++;
        }
    }
}

```

```

        g.setColor(new Color(234, 255, 0));
        g.fillOval(x_vert[t_task.t - 1] - 15, y_vert[t_task.t - 1] - 15,
30, 30);
        g.setColor(new Color(50, 100, 131));
        String S = Integer.toString(t_task.t);
        g.drawString(S, x_vert[t_task.t - 1] - 15, y_vert[t_task.t - 1] +
20);
    }
}

```

```

class markVert implements ActionListener {
    int x, y;

    public void actionPerformed(ActionEvent ae) {
        addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent me) {
                x = me.getX();
                y = me.getY();
                y_vert[i - 1] = y;
                x_vert[i - 1] = x;
                num_vert++;
                String S = Integer.toString(i);
                i++;
                Graphics g = getGraphics();
                g.setColor(new Color(100, 200, 0));
                g.fillOval(x - 15, y - 15, 30, 30);
                g.setColor(new Color(50, 100, 131));
                g.drawString(S, x - 15, y + 20);
                G.F.Add_vertex(num_vert);
                System.out.println(i);
            }
        });
    }
}

```

```

class Clean implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
        t_task = new Parametr();
        Nofc = new Num_of_cycle();
        G = new Graf();
        y_vert = new int[1000];
        x_vert = new int[1000];
        num_vert = 0;
        i = 1;
        repaint();
    }
}

```

```

class MarkEdge implements ActionListener {
    int x_s, y_s, b_1, b_2, x_f, y_f;
    boolean flag_1, flag_2;

    public void actionPerformed(ActionEvent ae) {

```

```

addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent me) {
        flag_1 = false;
        b_1 = 0;
        x_s = me.getX();
        y_s = me.getY();
        while (!flag_1 && b_1 < num_vert) {
            if (Math.pow(x_s - x_vert[b_1], 2) + Math.pow(y_s -
y_vert[b_1], 2) <= 900) {
                flag_1 = true;
            } else
                b_1++;
        }
    }

    public void mouseReleased(MouseEvent me) {
        flag_2 = false;
        b_2 = 0;
        x_f = me.getX();
        y_f = me.getY();
        while (!flag_2 && b_2 < num_vert) {
            if (Math.pow(x_f - x_vert[b_2], 2) + Math.pow(y_f -
y_vert[b_2], 2) <= 900) {
                flag_2 = true;
            } else
                b_2++;
        }
        if (flag_1 && flag_2) {
            Graphics g = getGraphics();
            g.setColor(new Color(255, 3, 230));
            g.drawLine(x_vert[b_1], y_vert[b_1], x_vert[b_2],
y_vert[b_2]);

            G.F.Add_edge(b_1 + 1, b_2 + 1);
            System.out.println(G.F.G[b_1][b_2]);
            System.out.println(G.F.G[b_2][b_1]);
        }
    }
});
}

class HandParr implements ActionListener {
    DialogDemo mainfr;

    HandParr(DialogDemo mf) {
        mainfr = mf;
    }

    public void actionPerformed(ActionEvent ae) {
        ParamDialog d = new
            ParamDialog(t_task, mainfr, "Условие для задачи");
        d.setVisible(true);
    }
}

```

```

    }

}

class DrawCcl implements ActionListener {
    DialogDemo mainfr;

    DrawCcl(DialogDemo mf) {
        mainfr = mf;
    }

    public void actionPerformed(ActionEvent ae) {
        G.Rez = G.F.Search_max_cycles(num_vert, t_task.t - 1, G.kol, G.M,
G.L);

        SolveDialog d = new
            SolveDialog(G, NoFc, mainfr, "Изобразить путь");
        d.setVisible(true);
    }

}

public static void main(String[] args) {
    DialogDemo f = new DialogDemo("Поиск максимальных циклов в графе");
    f.setSize(1000, 800);
    f.setVisible(true);
    System.out.println("Абобус");
}
}

```

Graph.java

```

package LW_8;

public class Graph {
    int[][] G;
    int Num_vertex;
    int size_g;
    public Graph() {
        G = new int[2550][2550];
        for (int i = 0; i < 2550; i++) {
            for (int j = 0; j < 2550; j++) {
                G[i][j] = 0;
            }
        }
        Num_vertex = 0;
    }

    public void Add_vertex(int vert) {
        Num_vertex = vert;
    }

    public void Add_edge(int a, int b) {
        G[a - 1][b - 1] = G[b - 1][a - 1] = 1;
    }
}

```



```

public int[][] Search_max_cycles(int n, int t, int kolp, int[][] RezMatr,
int[] Rezlen) {
    for (int d = 0; d < n; d++) {
        Stack St = new Stack(0);
        int[] M = new int[1000];
        int v, i, j, L, Pr, Prt, ks, vk, vn;
        vn = d;
        vk = vn;
        L = 0;
        for (j = 0; j < n; j++) {
            M[j] = 0;
        }
        ks = 0;
        St.Push(vn);
        M[vn] = 1;
        while (ks >= 0) {
            Pr = 0;
            v = St.Elem(ks);
            for (j = L; j < n; j++) {
                if (G[v][j] == 1) {
                    if (j == vk) {
                        Prt = 0;
                        for (i = 0; i <= ks; i++) {
                            if (St.Elem(i) == t)
                                Prt = 1;
                        }
                        if (Prt == 0) {
                            for (i = 0; i <= ks; i++) {
                                RezMatr[kolp][i] = St.Elem(i) + 1;
                                System.out.println("TyT");
                            }
                            RezMatr[kolp][ks + 1] = vk + 1;
                            Rezlen[kolp] = ks + 2;
                            kolp++;
                        }
                    }
                    else {
                        if (M[j] == 0) {
                            Pr = 1;
                            break;
                        }
                    }
                }
            }
        }
        if (Pr == 1) {
            ks++;
            St.SM_length = ks;
            St.Push(j);
            L = 0;
            M[j] = 1;
        } else {
            L = v + 1;
            M[v] = 0;
            ks--;
        }
    }
}

```

```

        St.Pop();
    }
}

int max_length_cycle = 4;
for (int l = 0; l < kolp; l++) {
    if (max_length_cycle < Rezlen[l]) {
        max_length_cycle = Rezlen[l];
    }
}

int[][] Rez = new int[2550][2550];
int h = 0;
for (int l = 0; l < kolp; l++) {
    if (max_length_cycle == Rezlen[l] &&
!Check_result(Rez, RezMatr[l], max_length_cycle, h)) {
        Rez[h] = RezMatr[l];
        h++;
    }
}

size_g = h;
return Rez;
}

public boolean Check_result(int[][] M, int[] V, int max, int size_M) {
    if(size_M == 0)
        return false;
    else {
        int i = 0;
        int j;
        int k;
        int rez;
        boolean flag = true;
        boolean flag_1;
        while (i < size_M && flag) {
            j = 0;
            rez = 1;
            while (j < max && flag) {
                k = 0;
                flag_1 = true;
                while (k < max && flag_1) {
                    if (M[i][j] == V[k]) {
                        rez++;
                        flag_1 = false;
                    }
                    k++;
                }
                if (rez == max+1)
                    flag = false;
                j++;
            }
            i++;
        }
        return !flag;
    }
}

```

```

    }
}
}
Stack.java
package LW_8;

public class Stack {
    int[] SM;
    public int SM_length;

    public Stack(int a) {
        SM = new int[2550];
        for (int i = 0; i < 2550; i++) {
            SM[i] = 0;
        }
        SM_length = a;
    }

    public void Push(int elem) {
        if (SM_length == 0) {
            SM[0] = elem;
            SM_length = 1;
        } else {
            SM[SM_length] = elem;
            SM_length++;
        }
    }

    public int Pop() {
        int i;
        if (SM_length == 0) {
            return -1;
        } else {
            i = SM[SM_length - 1];
            SM[SM_length - 1] = 0;
            SM_length--;
            return i;
        }
    }

    public int Elem(int i){
        return SM[i];
    }
}

```