**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 3 (18 варіант) з дисципліни

«Проектування алгоритмів»

„ **Проектування структур даних**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*Спаських М. Д.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2021

Зміст

[1 Мета лабораторної роботи 3](#_Toc51260917)

[2 Завдання 4](#_Toc51260918)

[3 Виконання 9](#_Toc51260919)

[3.1 Програмна реалізація алгоритму 9](#_Toc51260920)

[3.1.1 Вихідний код 9](#_Toc51260921)

[3.1.2 Приклади роботи 9](#_Toc51260922)

[3.2 Тестування алгоритму 10](#_Toc51260923)

[3.2.1 Значення цільової функції зі збільшенням кількості ітерацій 10](#_Toc51260924)

[3.2.2 Графіки залежності розв'язку від числа ітерацій 10](#_Toc51260925)

[Висновок 11](#_Toc51260926)

[Критерії оцінювання 12](#_Toc51260927)

# Мета лабораторної роботи

Мета роботи – вивчити основні підходи формалізації метаеврестичних алгоритмів і вирішення типових задач з їхньою допомогою.

# Завдання

Згідно варіанту, розробити алгоритм вирішення задачі і виконати його програмну реалізацію на будь-якій мові програмування.

Задача, алгоритм і його параметри наведені в таблиці 2.1.

Зафіксувати якість отриманого розв'язку (значення цільової функції) після кожних 20 ітерацій до 1000 і побудувати графік залежності якості розв'язку від числа ітерацій.

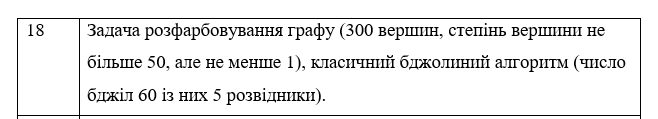
Зробити узагальнений висновок.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |
| --- | --- |
| **№** | **Задача і алгоритм** |
| 1 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 2 | Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 30, починають маршрут в різних випадкових вершинах). |
| 3 | Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники). |
| 4 | Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 5 | Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 35, починають маршрут в різних випадкових вершинах). |
| 6 | Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники). |
| 7 | Задача про рюкзак (місткість P=150, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 8 | Задача комівояжера (200 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм (α = 3, β = 2, ρ = 0,3, Lmin знайти жадібним алгоритмом, кількість мурах М = 45, починають маршрут в різних випадкових вершинах). |
| 9 | Задача розфарбовування графу (150 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 25 із них 3 розвідники). |
| 10 | Задача про рюкзак (місткість P=150, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 11 | Задача комівояжера (250 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм (α = 2, β = 4, ρ = 0,6, Lmin знайти жадібним алгоритмом, кількість мурах М = 45, починають маршрут в різних випадкових вершинах). |
| 12 | Задача розфарбовування графу (300 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 60 із них 5 розвідники). |
| 13 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий 30% і 70%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 14 | Задача комівояжера (250 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 4, β = 2, ρ = 0,3, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (10 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових вершинах). |
| 15 | Задача розфарбовування графу (100 вершин, степінь вершини не більше 20, але не менше 1), класичний бджолиний алгоритм (число бджіл 30 із них 3 розвідники). |
| 16 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий 30%, 40% і 30%, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 17 | Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 2, β = 4, ρ = 0,7, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (15 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових вершинах). |
| 18 | Задача розфарбовування графу (300 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 60 із них 5 розвідники). |
| 19 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення. |
| 20 | Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 3, β = 2, ρ = 0,7, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (10 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах). |
| 21 | Задача розфарбовування графу (200 вершин, степінь вершини не більше 30, але не менше 1), класичний бджолиний алгоритм (число бджіл 40 із них 2 розвідники). |
| 22 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 23 | Задача комівояжера (300 вершин, відстань між вершинами випадкова від 1 до 60), мурашиний алгоритм (α = 3, β = 2, ρ = 0,6, Lmin знайти жадібним алгоритмом, кількість мурах М = 45 (15 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах). |
| 24 | Задача розфарбовування графу (400 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 70 із них 10 розвідники). |
| 25 | Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 26 | Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 4, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 30, починають маршрут в різних випадкових вершинах). |
| 27 | Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники). |
| 28 | Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення. |
| 29 | Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах М = 35, починають маршрут в різних випадкових вершинах). |
| 30 | Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники). |

# Виконання

В мене 18 варіант.



## Програмна реалізація алгоритму

### Вихідний код

using System;

using System.Collections.Generic;

namespace lab3

{

class uzel

{

public int stepin;

public int color;

public int konfl;

public int nomer;

public uzel (int s, int c, int k, int n)

{

stepin = s;

color = c;

konfl = k;

nomer = n;

}

}

class graf

{

public List<uzel> spisok\_uzlov;

public int n = 200; // kilkist vershin

public int m = 50; // max stepin vershini

public int[,] mas; // matrisa sumijnosti grafa

public int max\_color=1;

public List<uzel> spisok\_uzlov\_min\_record;

public int max\_color\_record;

public int k\_rozvidnikiv=5;

public int k\_furjiriv=55;

public graf()

{

mas = new int[n, n];

spisok\_uzlov = new List<uzel>();

for(int i = 0; i<n; i++)

{

uzel u = new uzel(0, 0, 0, i);

spisok\_uzlov.Add(u);

}

}

public void gen ()

{

Random rnd = new Random();

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

mas[i, j] = 0;

}

int s1 = 0;

int s2 = 0;

for (int t = 0; t < i; t++)

if (mas[t, i] == 1) s1++;

for (int t = i + 1; t < n; t++) if (spisok\_uzlov[t].stepin >= m) s2++;

int k\_sv = m - s1;

if (k\_sv > n - i - 1 - s2) k\_sv = n - i - 1 - s2;

if(k\_sv>0) k\_sv = rnd.Next(1,k\_sv);

for (int k = 0; k < k\_sv; k++)

{

int r = rnd.Next(i+1,n);

if (mas[i, r] == 0 && spisok\_uzlov[i].stepin<m && spisok\_uzlov[r].stepin<m) {

mas[i, r] = 1;

spisok\_uzlov[i].stepin++;

spisok\_uzlov[r].stepin++;

}

else k--;

}

}

for(int i=0; i<n; i++)

{

for (int j=i+1; j<n; j++)

{

mas[j, i] = mas[i, j];

}

}

for(int i=0; i<n; i++) // rahuemo stepeni uzliv

{

int s = 0;

for(int j = 0; j < n; j++)

{

if (mas[i, j] == 1) s++;

}

spisok\_uzlov[i].stepin = s;

}

}

public void vivod()

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

Console.Write( $"{ mas[i, j]}" );

}

Console.WriteLine();

}

}

public void farbuemo()

{

List<uzel> rozf = new List<uzel>();

List<uzel> nerozf = new List<uzel>();

for (int i = 0; i < n; i++)

{

nerozf.Add(spisok\_uzlov[i]);

}

for (int t=0; t<n; t++)

{

nerozf.Sort((u, v) => v.konfl - u.konfl); // znahodimo max konfl

uzel v = nerozf[0];

int i;

int j=0;

for(i=1; i<n && j<n; i++) // pidbiraemo kolir

{

for (j = 0; j < n; j++) // pereviraemo vershini sumijni na spibpadinnia koloru

if (mas[v.nomer,j]==1 && i==spisok\_uzlov[j].color && v.nomer!=j)

{

break;

}

}

v.color = i - 1;

if (v.color > max\_color) max\_color = v.color;

max\_color\_record = max\_color;

rozf.Add(v);

nerozf.Remove(v);

for (i = 0; i< nerozf.Count; i++) //rahuemo konflikti

for (j = 0; j<rozf.Count; j++)

{

if (mas[nerozf[i].nomer, rozf[j].nomer] == 1) nerozf[i].konfl++;

}

}

}

public void jadnefarbubania()

{

List<uzel> rozf = new List<uzel>();

List<uzel> nerozf = new List<uzel>();

for (int i = 0; i < n; i++)

{

nerozf.Add(spisok\_uzlov[i]);

}

for (int t = 0; t < n; t++)

{

//nerozf.Sort((u, v) => v.konfl - u.konfl); // znahodimo max konfl

uzel v = nerozf[0];

int i;

int j = 0;

for (i = 1; i < n && j < n; i++) // pidbiraemo kolir

{

for (j = 0; j < n; j++) // pereviraemo vershini sumijni na spibpadinnia koloru

if (mas[v.nomer, j] == 1 && i == spisok\_uzlov[j].color && v.nomer != j)

{

break;

}

}

v.color = i - 1;

if (v.color > max\_color) max\_color = v.color;

max\_color\_record = max\_color;

rozf.Add(v);

nerozf.Remove(v);

for (i = 0; i < nerozf.Count; i++) //rahuemo konflikti

for (j = 0; j < rozf.Count; j++)

{

if (mas[nerozf[i].nomer, rozf[j].nomer] == 1) nerozf[i].konfl++;

}

}

}

public void glupe\_farbuvania()

{

int i;

for ( i = 0; i < n; i++) spisok\_uzlov[i].color = i + 1;

max\_color = i;

max\_color\_record = max\_color;

}

public List<int> rozvidka ()

{

List<int> rez = new List<int>();

Random rnd = new Random();

for (int k=0; k<k\_rozvidnikiv; k++)

{

int n\_dil = rnd.Next(n);

if (!rez.Contains(n\_dil)) rez.Add(n\_dil);

else k--;

}

//Console.Write($"rozvidka: ");

//for (int k = 0; k < rez.Count; k++) { Console.Write($"{rez[k]} "); if (rez[k] == 199) Console.Write("!!!!!!!!!!!!!!!!!"); }

//Console.WriteLine();

return rez;

}

public List<int> rozpodil\_furajiriv (List<int> nomera\_tsiley)

{

List<int> rez = new List<int>();

int s = 0;

for (int i = 0; i < nomera\_tsiley.Count; i++) s += spisok\_uzlov[nomera\_tsiley[i]].stepin;

for (int i = 0; i < nomera\_tsiley.Count; i++) rez.Add(1+ spisok\_uzlov[nomera\_tsiley[i]].stepin\*(k\_furjiriv-k\_rozvidnikiv)/s);

//Console.Write($"rozpodil\_furajiriv: ");

//for (int k = 0; k < rez.Count; k++) { Console.Write($"{rez[k]} ");}

//Console.WriteLine();

return rez;

}

public void poshuk\_v\_okolitsiah(List<int> tsili, List<int> rozpodileni\_bdjoli)

{

//List<int> tsili=rozvidka();

//List<int> rozpodileni\_bdjoli = rozpodil\_furajiriv(tsili);

Random rnd = new Random();

spisok\_uzlov\_min\_record = spisok\_uzlov;

for (int i=0; i < tsili.Count; i++)

{

List < uzel > okil = new List<uzel>();

for(int j=0;j<n;j++) //formuemo okil

{

uzel u = new uzel(spisok\_uzlov[j].stepin, spisok\_uzlov[j].color, spisok\_uzlov[j].konfl, spisok\_uzlov[j].nomer);

okil.Add(u);

}

for (int k=0; k<rozpodileni\_bdjoli[i]; k++) // zapusk furajiriv

{

int tsil = tsili[i];

List<int> sumijni\_vershini\_z\_tsillu = new List<int>();

for (int j=0; j<n; j++)

{

if (mas[tsil, j] == 1) sumijni\_vershini\_z\_tsillu.Add(j);

}

int n2 = sumijni\_vershini\_z\_tsillu[rnd.Next(sumijni\_vershini\_z\_tsillu.Count)];

//Console.WriteLine($"n2={n2}");

List<int> sumijni\_vershini\_z\_n2 = new List<int>();

for (int j = 0; j < n; j++)

{

if (mas[n2, j] == 1) sumijni\_vershini\_z\_n2.Add(j);

}

bool flag = true;

for (int j=0; j<sumijni\_vershini\_z\_n2.Count; j++)

{

if (spisok\_uzlov[tsil].color == spisok\_uzlov[sumijni\_vershini\_z\_n2[j]].color && sumijni\_vershini\_z\_n2[j] != tsil) flag = false;

}

for (int j = 0; j < sumijni\_vershini\_z\_tsillu.Count; j++)

{

if (spisok\_uzlov[n2].color == spisok\_uzlov[sumijni\_vershini\_z\_tsillu[j]].color && sumijni\_vershini\_z\_tsillu[j]!=n2) flag = false;

}

if (flag) // zamina vuzliv

{

int tmp = okil[n2].color;

okil[n2].color = okil[tsil].color;

okil[tsil].color = tmp;

//Console.WriteLine("zamina !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"); // test

//okil[n2].color = 0; // zmenshuemo kolori;

int j1;

int t1=0;

for (j1 = 1; j1 <= max\_color && t1 < sumijni\_vershini\_z\_n2.Count; j1++)

{

for (t1 = 0; t1 < sumijni\_vershini\_z\_n2.Count; t1++)

{

if (okil[sumijni\_vershini\_z\_n2[t1]].color == j1) break;

}

}

okil[n2].color = j1 - 1;

//Console.WriteLine($"v {n2} kolir {j1 - 1}"); //test

//okil[tsil].color = 0;

t1 = 0;

for (j1 = 1; j1 <= max\_color && t1 < sumijni\_vershini\_z\_tsillu.Count; j1++)

{

for (t1 = 0; t1 < sumijni\_vershini\_z\_tsillu.Count; t1++)

{

if (okil[sumijni\_vershini\_z\_tsillu[t1]].color == j1) break;

}

}

okil[tsil].color = j1 - 1;

//Console.WriteLine($"v {tsil} kolir {j1 - 1}"); //test

if (otsinka\_korisnosti\_dilanki(okil) < otsinka\_korisnosti\_dilanki(spisok\_uzlov\_min\_record)) // onovlenia rekordu

{

spisok\_uzlov\_min\_record = okil;

//Console.WriteLine($"korisnist rekordu {otsinka\_korisnosti\_dilanki(spisok\_uzlov\_min\_record)}");

}

}

}

}

spisok\_uzlov = spisok\_uzlov\_min\_record;

}

public int otsinka\_korisnosti\_dilanki( List<uzel> dilianka)

{

int rez = 0;

for (int i = 0; i < dilianka.Count; i++) rez += dilianka[i].color;

//Console.WriteLine($"otsinka\_korisnosti\_dilanki = {rez}");

return rez;

}

public bool test()

{

bool flag = true;

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

if (mas[i,j]==1 && spisok\_uzlov[i].color==spisok\_uzlov[j].color)

{

flag = false;

return flag;

}

}

}

return flag;

}

public void roi (int k = 10)

{

for (int i=1; i<=k; i++ )

{

List<int> tsili = rozvidka();

poshuk\_v\_okolitsiah(tsili, rozpodil\_furajiriv(tsili));

max\_color\_record = 0;

for (int j=0; j< spisok\_uzlov\_min\_record.Count; j++)

{

if (spisok\_uzlov\_min\_record[j].color > max\_color\_record) max\_color\_record = spisok\_uzlov\_min\_record[j].color;

}

if(i%20==0) Console.Write ($"№ {i} koloriv {max\_color\_record} \t");

if (i % 100 == 0) Console.WriteLine();

}

}

public void vivod\_rozkraski()

{

Console.WriteLine("rozkraska: ");

for (int i = 0; i < n; i++)

{

Console.Write($"v[{i}].col={spisok\_uzlov[i].color} \t");

if ((i+1) % 5 == 0) Console.WriteLine();

}

Console.WriteLine($"vikoristano {max\_color\_record} koloriv");

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

graf g = new graf();

g.gen();

g.vivod();

g.jadnefarbubania();

//g.glupe\_farbuvania();

g.vivod\_rozkraski();

g.roi(1000);

g.vivod\_rozkraski();

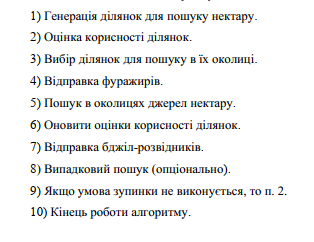
Console.WriteLine("perevirka vidsutnosti konfliktiv koloriv: "+g.test());

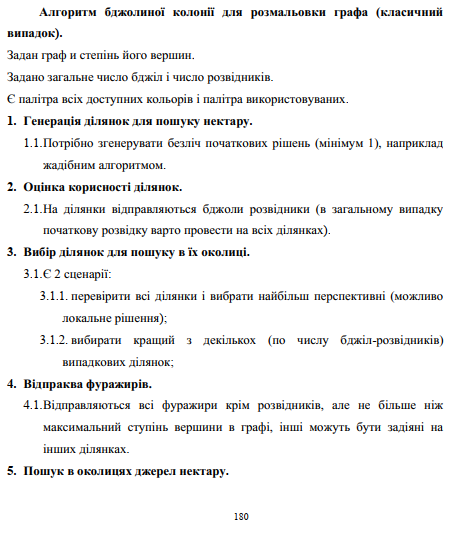
}

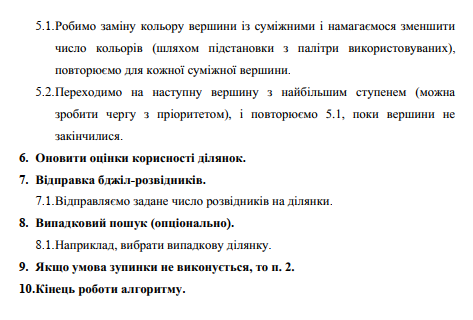
}

}

Словесний опис алгоритму (в інтернеті не знайшов псевдокоду, тому писав програму по лекції Максима Миколайовича):







Алгоритм працює дуже ефективно, навіть при дуже поганому початковому розв'язку (якщо навмисно всі вершини пофарбувати в різні кольори), він знаходить дуже якісні розв'язки за 1000 ітерацій. Схожі на розв’язки за 1000 ітерацій з початковим станом сформованим жадібним алгоритмом.

### Приклади роботи

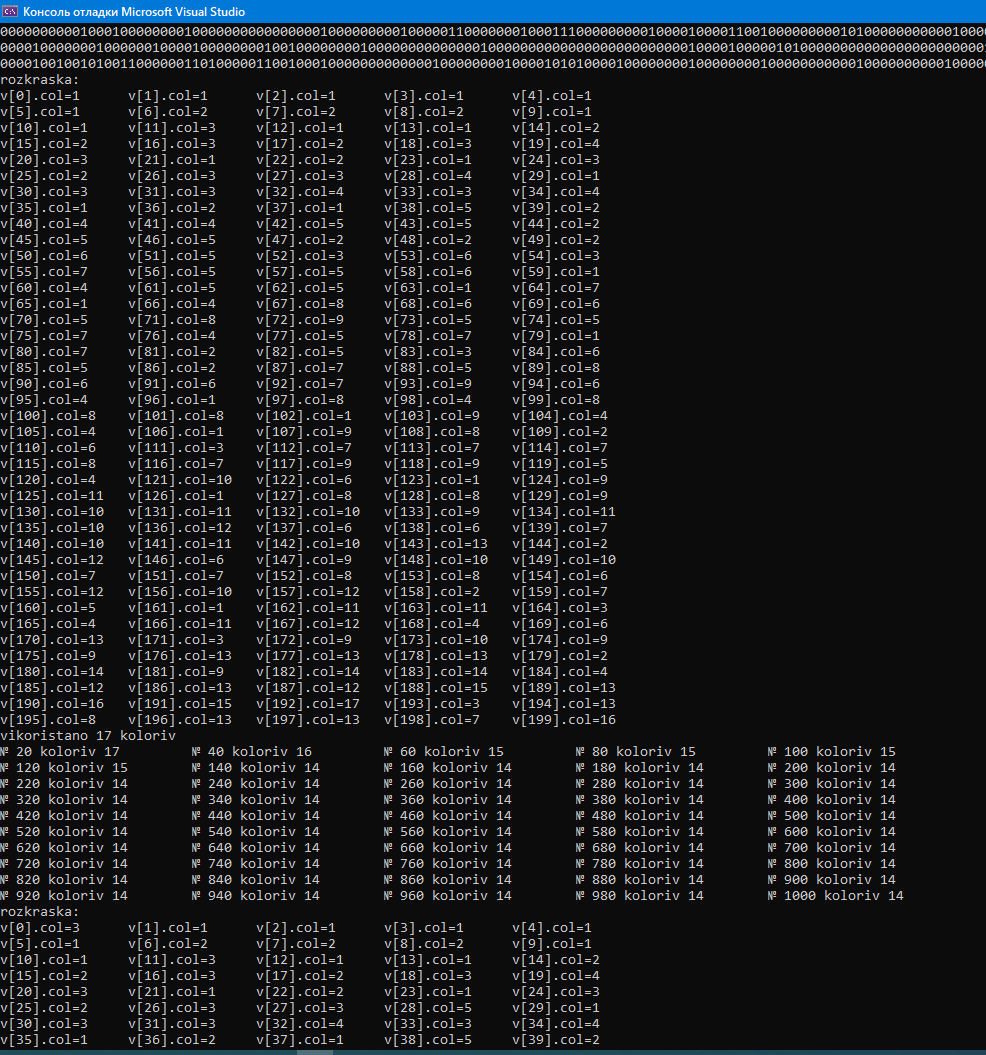
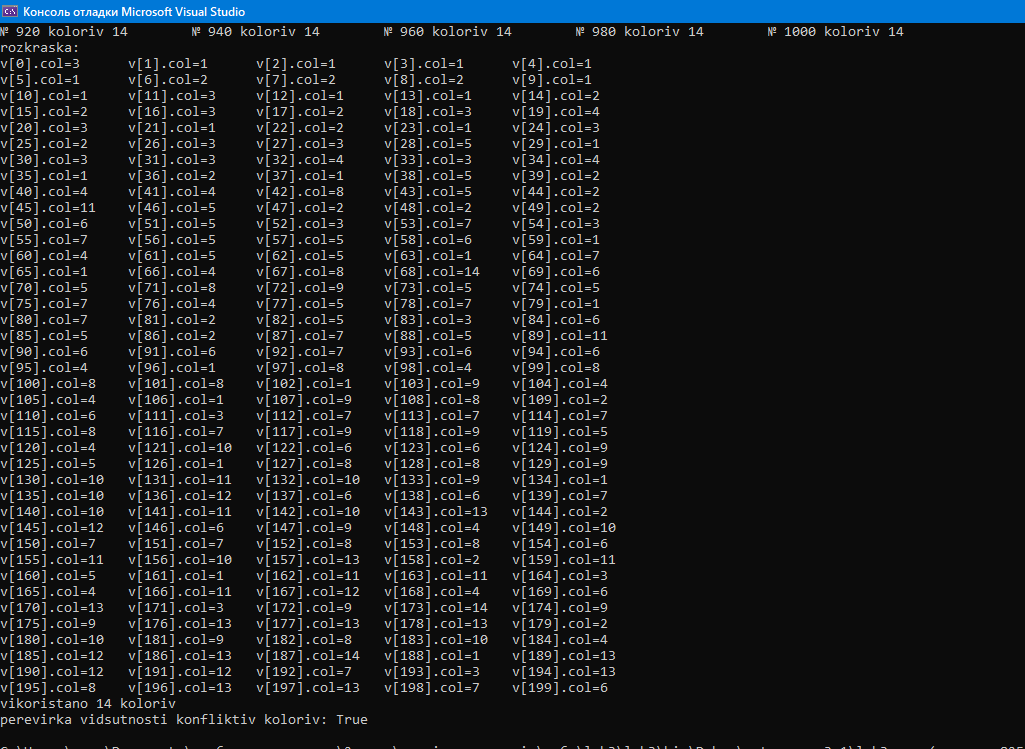
На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 – Матриця суміжності 200\*200:





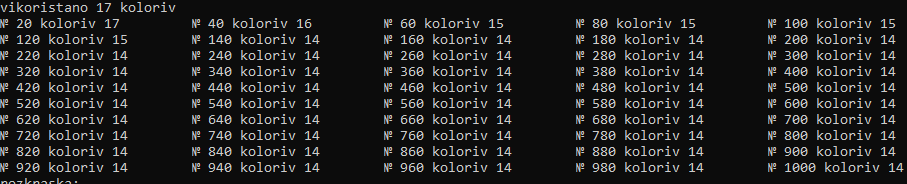
Рисунок 3.2 – початкова созкраска, зміна числа кольорів після кожної ітерації з кроком 20 для 1000 ітерацій, кінцева розкраска, перевірка відсутності конфліктів кольорів.

## Тестування алгоритму

### Значення цільової функції зі збільшенням кількості ітерацій

У на малюнку нижче наведено значення цільової функції (використаного числа кольорів) зі збільшенням кількості ітерацій.

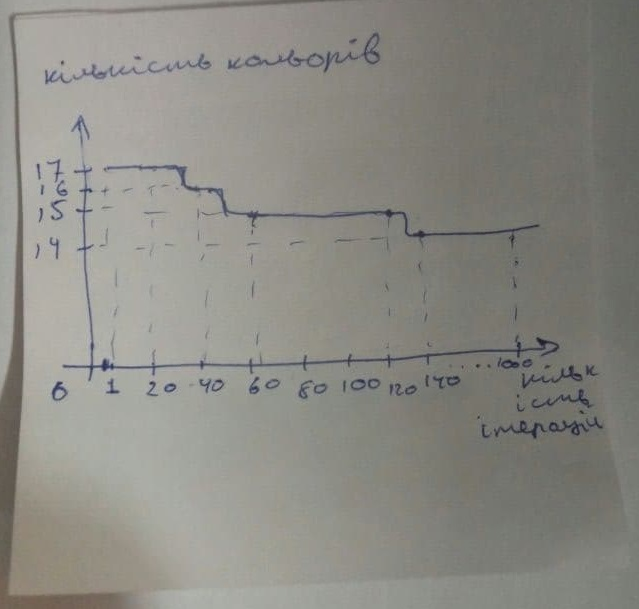


Насправді в якості оцінки корисності ділянок в програмі я використовував сумарне значення номерів кольорів всіх вершин, бо воно більш детальне і допомагає відрізнити яка ділянка краще з однаковим числом кольорів. Але на малюнку вивів, в якості оцінки, число використаних кольорів, бо воно більш зрозуміле для всіх.

### Графіки залежності розв'язку від числа ітерацій

На рисунку 3.3 наведений графік, який показує якість отриманого розв'язку.

Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій



Висновок

В рамках даної лабораторної роботи номер 3 я створив програму яка генерує граф з 200 вершинами, степінь кожної від 1 до 50. Розфарбував його жадібним алгоритмом. І удосконалив розфарбування класичним алгоритмом бджолиного рою. Використав 5 бджіл розвідників і 55 бджіл фуражирів. Зазвичай алгоритм покращує рішення не значно – на 0-3 числа кольорів, в залежності від якості початково згенерованого розв’язку. Але якщо, наприклад, розфарбувати всі вершини в різні кольори (в 200 кольорів) в початковому розв’язку (в мене там функція glupe\_farbuvania так фарбувати може) , то він значно покращує розв’язок за 1000 ітерацій, приблизно до такого ж значення як і при гарно згенерованому початковому розв’язку. Алгоритм працює досить швидко: 1000 ітерація на моєму графі робить за кілька секунд. Рішення кращі від фарбування алгоритмом по максимальній кількості конфлікті. Я також пробував ним генерувати початкові стани, і бджоли їх удосконалювали. Нового в цій роботі, на відміну від попередніх, використовувалися ймовірнісні величини в генерації випадкових чисел.

Ссилка на гідхаб з лабами: https://github.com/kolya-sp/alg\_2