**Міністерство освіти і науки України**

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

**Факультет інформатики та обчислювальної техніки**

**Кафедра автоматизованих систем обробки інформації**

**і управління**

**Звіт**

з лабораторної роботи № 3 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

**«Евристичні алгоритми»**

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-01 Галько Міла Вячеславівна*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2021

**Лабораторна робота № 3**

**«Евристичні алгоритми»**

**Мета роботи** – вивчити основні підходи формалізації евристичних алгоритмів і вирішення типових задач з їх допомогою.

**Завдання**

****

****

1. Для **задачі розфарбовування графа**, вибрати 15 адміністративних одиниць (областей, районів) в Албанії і записати для них суміжність один з одним. Для визначення суміжностей рекомендується використовувати інтернет сервіси (наприклад Google Maps).
   1. Псевдокод
   2. Код
   3. Висновок
2. +1 додатковий бал можна отримати за програмне формування таблиць відстаней, суміжностей, тощо (за допомогою API інтернет сервісів) або за графічну демонстрацію роботи алгоритмів (на графі за допомогою десктопного інтерфейсу), отримати можна лише +1 бал.

1.1) Псевдокод «Степенева Евристика»:

1.ПОЧАТОК

2. ЦИКЛ проход по всім V не мають кольору:

2.1. V не має кольору (тобто колір №0).

3. ЦИКЛ Перебрати усі V від з найбільшою степеню до найменшої:

3.1. Порівняти обрану V з її суміжними:

3.1.1. ЯКЩО Обрана V має такий самий колір як і суміжна, ТО:

3.1.1.1. Присвоїти обраній V наступний колір.

3.1.1.2. ЯКЩО номер наступного кольору став найбільшим, ТО вважати його хроматичним числом.

3.1.1.3. Повторити Цикл (3.) з початку.

4. RETURN хроматичне число.

1.1) Псевдокод «Степенева Евристика використовуючи алгоритм Пошук з поверненням» :

1.ПОЧАТОК

2. ЦИКЛ проход по всім V не мають кольору:

2.1. V не має кольору (тобто колір №0).

3.ПОКИ усі V не будуть зафарбовані:

3.1.Обрати V (за найбільшою степеню з самого початку).

3.2.ЯКЩО V зафарбована, ТО:

3.2.1.Переходимо до наступної V з такою самою степеню.

3.2.2.ЯКЩО усі V з такою степеню зафарбовані, ТО обираємо ту що має найбільшу степінь, але степінь менша за попередню.

3.3.Додаємо обрану V у стек.

3.4.V фарбуємо у колір №1.

3.5.ПОКИ не перевіримо суміжність V з усіма іншими V:

3.5.1.ЯКЩО Обрана V суміжна з іншою V1 та V1 не була вийнята зі стеку, ТО:

3.5.1.1.ЯКЩО V та V1 мають однаковий колір, ТО:

3.5.1.1.1.Присвоїти обраній V наступний колір.

3.5.1.1.2.ЯКЩО наступній колір став більший за максимальну кількість кольорів, ТО:

3.5.1.1.2.1.ВИКОНАТИ:

3.5.1.1.2.1.1.Позначити V кольором №0.  
 3.5.1.1.2.1.2.Вилучити V зі стеку.

3.5.1.1.2.1.2.1.ЯКЩО зі стеку було вилучено дно, ТО:

3.5.1.1.2.1.2.1.1.Обрати дном стека наступну V з найб. степеню

3.5.1.1.2.1.2.1.1.1.ЯКЩО усі V з найб. степеню раніше вже були у стеці, ТО «Рішення не існує».

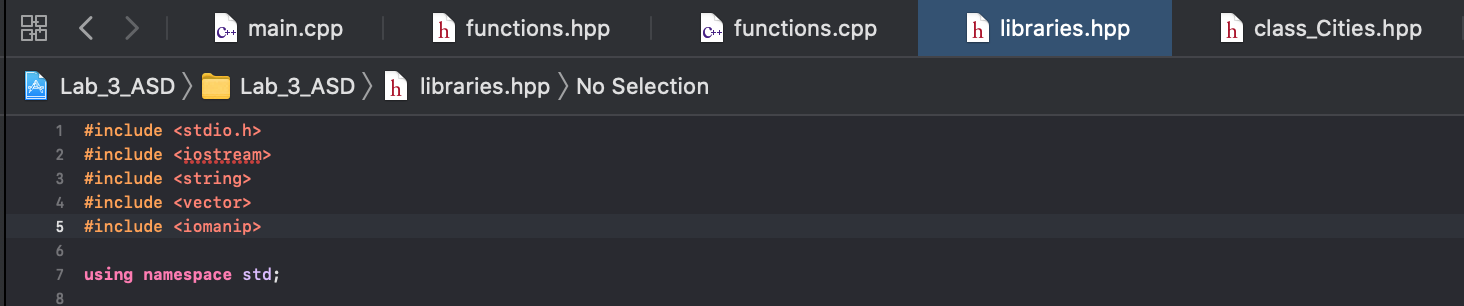
3.5.1.1.2.2.ПОКИ «Рішення не існує» АБО наступній колір вершини стека менший за максимальну кількість кольорів.

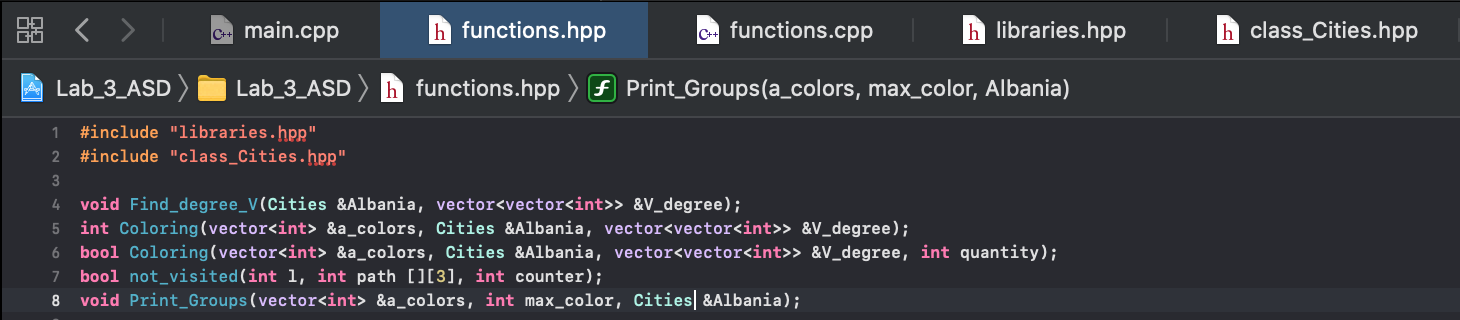
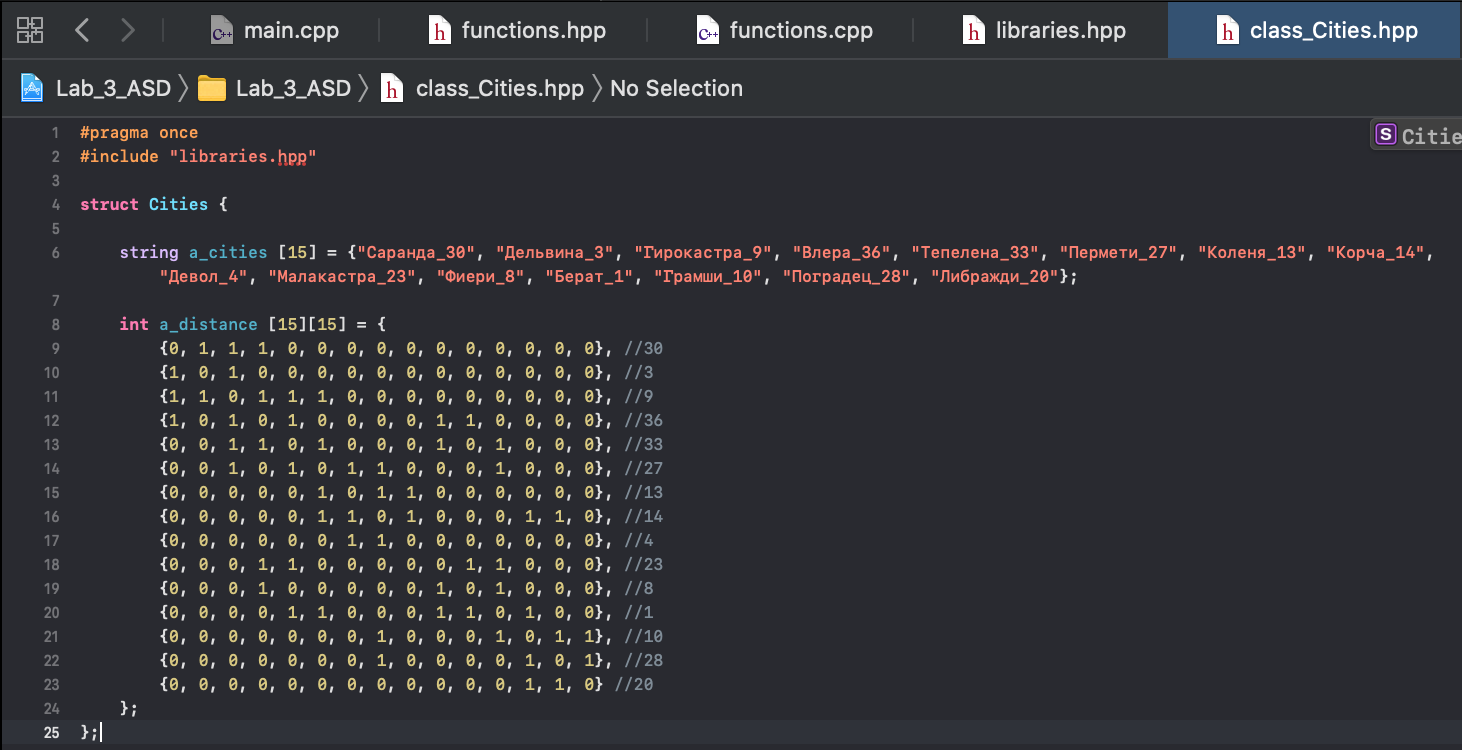
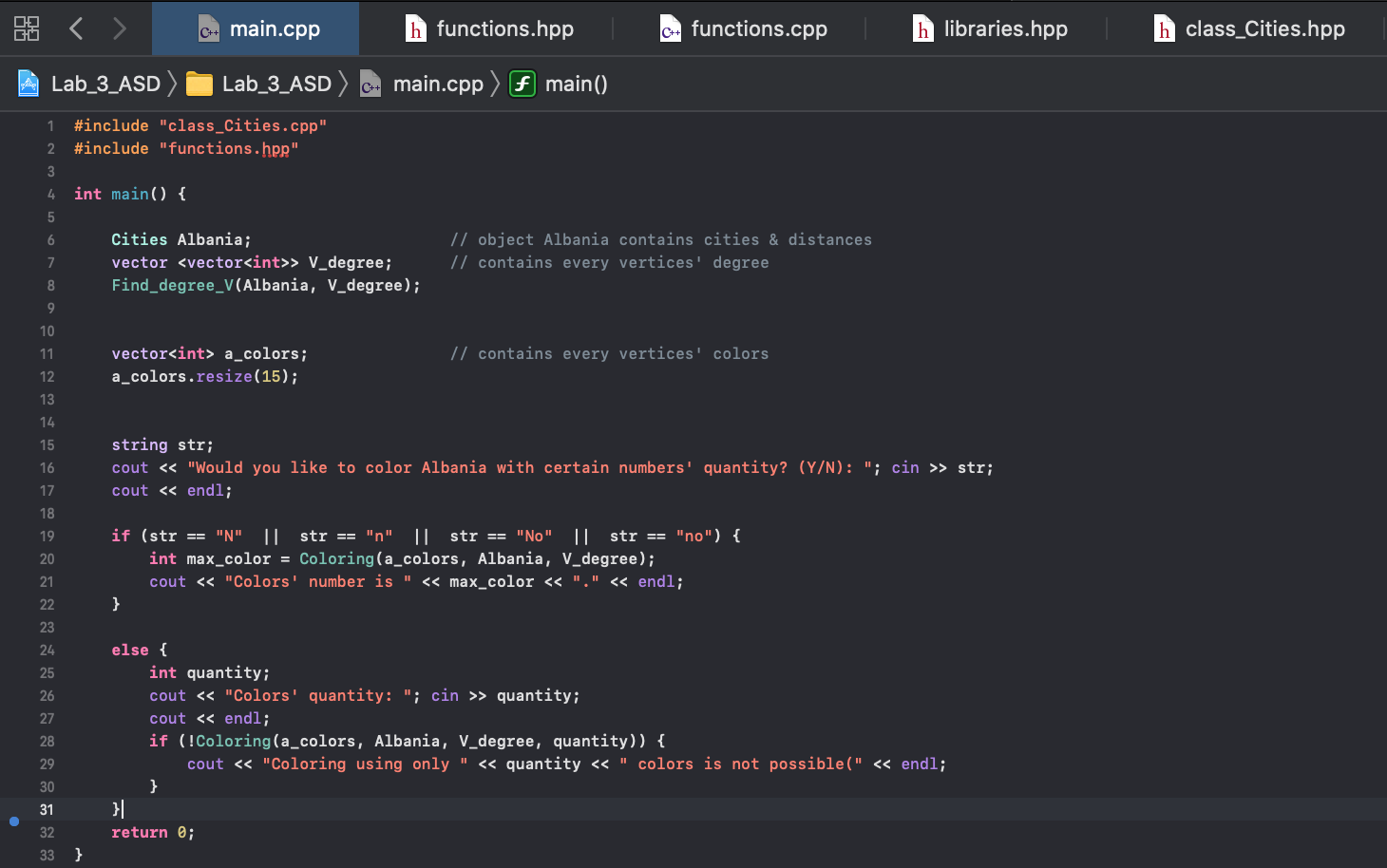
3.5.1.1.2.3.ЯКЩО стек не очищувався повністю, ТО вершині стеку змінити колір на наступний. ІНАКШЕ break;

3.5.1.1.3.Почати спочатку ПОКИ ().

4.RETURN «Рішення існує».

1.2)





FUNCTIONS (1):

#include "functions.hpp"

void Find\_degree\_V(Cities &Albania, vector<vector<int>> &V\_degree) {

int vertices = 15;

int counter;

int v\_degree;

int max = 100;

int last = V\_degree.size() - 1;

while (vertices != 0) {

v\_degree = 0;

for (int i = 0; i < 15; i++) {

counter = 0;

for (int j = 0; j < 15; j++) {

if (Albania.a\_distance[i][j] == 1) {

counter ++;

}

}

if (v\_degree < counter && counter < max) {

v\_degree = counter;

}

}

V\_degree.resize(V\_degree.size() + 1);

last = V\_degree.size() - 1;

for (int i = 0; i < 15; i++) {

counter = 0;

for (int j = 0; j < 15; j++) {

if (Albania.a\_distance[i][j] == 1) {

counter ++;

}

}

if (counter == v\_degree) {

vertices -= 1;

V\_degree[last].push\_back(i);

}

}

max = v\_degree;

}

}

FUNCTIONS (2):

int Coloring(vector<int> &a\_colors, Cities &Albania, vector<vector<int>> &V\_degree){

for (int i = 0; i < 15; i++) {

a\_colors[i] = 0;

}

int color = 1;

int max\_color = 1;

for (int i = 0; i < V\_degree.size(); i++) {

for (int j = 0; j < V\_degree[i].size(); j++) {

int l = 0;

while (l < 15){

if (Albania.a\_distance[V\_degree[i][j]][l] == 1){

if (a\_colors[l] == color) {

color++;

l = 0;

if (max\_color < color) {

max\_color = color;

}

continue;

}

}

l++;

}

a\_colors[V\_degree[i][j]] = color;

color = 1;

}

}

Print\_Groups(a\_colors, max\_color, Albania);

return max\_color;

}

FUNCTIONS (3):

bool Coloring(vector<int> &a\_colors, Cities &Albania, vector<vector<int>> &V\_degree, int quantity) {

for (int i = 0; i < 15; i++) {

a\_colors[i] = 0;

}

int path [15][3];

int i = 0;

int j = 0;

int l = 0;

int deleted = -1;

int counter = 1;

bool the\_end = false;

int all\_V = 0;

while (all\_V != 15) {

if (!not\_visited(V\_degree[i][j], path, counter)) {

if (j == V\_degree[i].size() - 1) {

i++; j = 0;

if (i == V\_degree.size()) {

break;

}

}

else{

j++;

}

continue;

}

path[counter - 1][0] = V\_degree[i][j];

path[counter - 1][1] = i;

path[counter - 1][2] = j;

a\_colors[path[counter - 1][0]] = 1;

l = 0;

while (l < 15){

if (Albania.a\_distance[path[counter - 1][0]][l] == 1 && deleted != l) {

if (a\_colors[l] == a\_colors[path[counter - 1][0]]) {

a\_colors[path[counter - 1][0]] += 1;

if (a\_colors[path[counter - 1][0]] > quantity) {

do {

deleted = path[counter - 1][0];

path[counter - 1][0] = 0;

counter --;

all\_V--;

if (counter == 1) {

i = 0;

j = path[counter - 1][2] + 1;

if (j == 7) {

return 0;

}

counter = 0;

the\_end = true;

}

else {

i = path[counter - 1][1];

j = path[counter - 1][2];

}

} while (quantity < a\_colors[V\_degree[i][j]] + 1 && !the\_end);

if (!the\_end) {

a\_colors[V\_degree[i][j]] = 1 + a\_colors[V\_degree[i][j]];

all\_V++;

}

else {break;}

}

l = 0;

continue;

}

}

l++;

}

deleted = -1;

counter++;

if (the\_end) {

the\_end = false;

continue;

}

i = 0;

j = 0;

}

FUNCTIONS (4):

Print\_Groups(a\_colors, quantity, Albania);

return 1;

}

bool not\_visited(int l, int path [][3], int counter) {

bool result = true;

for (int i = 0; i < counter - 1; i++) {

if (path[i][0] == l) {

result = false;

}

}

return result;

}

FUNCTIONS (5):

void Print\_Groups(vector<int> &a\_colors, int max\_color, Cities &Albania) {

int color = 1;

int i = 0;

while (color < max\_color + 1) {

cout << "Color #" << color << ": ";

while (i < 15) {

if (a\_colors[i] == color) {

cout << Albania.a\_cities[i] << ", ";

}

i++;

}

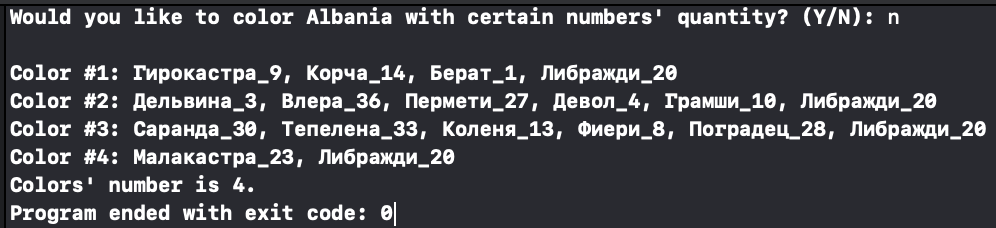
cout << endl;

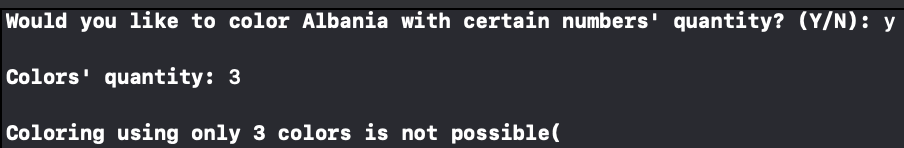
color++;

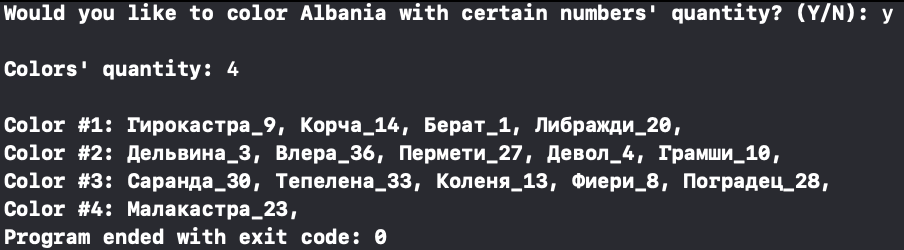
i = 0;

}

}







1.3) Висновок:

В даній лабораторній роботі мною був досліджений алгоритм «Пошук з поверненням» степеневої Евристики на прикладі округів Албанії. В задачі я використала 15 округів, що межують одне з одним для більш інформативного рішення задачі «Розфарбовування графа».

В ході роботи:

1. написаний псевдокод по 2 варіантам рішення задачі;

2. були розроблені 2 варіанта рішення:

2.1. Степенева евристика із поверненням хроматичного числа;

2.2. Степенева евристика алгоритм «Backtracking», де кількість кольорів вводиться користувачем;

3. перевірений результат (найменше можливе хроматичне число дорівнює 4 для обох варіантів);