**Міністерство освіти та науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра** **ІПІ (ІСТ)**

**Звіт**

з лабораторної роботи № 7 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

**„ Проектування і аналіз алгоритмів пошуку”**

**Виконав:**

*Студент I курсу*

*гр. ІП-з31*

Ткаченко К.О.

**Перевірила:**

к.т.н., Халус Олена Андріївна

2024

ЗМІСТ

1 **МЕТА ЛАБОРАТОРНОЇ РОБОТИ** ..................................................... 3

2 З**АВДАННЯ**............................................................................................... 4

3 В**ИКОНАННЯ** .......................................................................................... 4

3.1 ПСЕВДОКОД АЛГОРИТМІВ ....................................................................... 8

3.2 ПРОГРАМНА РЕАЛІЗАЦІЯ ......................................................................... 8

3.2.1 Вихідний код .................................................................................... 8

3.2.2 Приклади роботи............................................................................ 8

**ВИСНОВОК**..................................................................................................... 9

**КРИТЕРІЇ ОЦІНЮВАННЯ**........................................................................ 10

**3.1 ПСЕВДОКОД АЛГОРИТМІВ**

1. Структура Точка:

- x: дійсне число

- y: дійсне число

2. Функція відстань(p1: Точка, p2: Точка) -> дійсне число:

- Повертає відстань між точками p1 і p2.

3. Функція зчитати\_граф(filename: рядок) -> вектор<Точка>:

- Відкриває файл із ім'ям filename.

- Зчитує кількість точок (n) із файлу.

- Створює вектор graph розміром n.

- Для кожної точки i від 0 до n-1:

- Зчитує координати x і y із файлу і додає їх до графу.

4. Функція tsp(graph: вектор<Точка>) -> пара<дійсне число, вектор<ціле число>>:

- n = розмір(graph)

- Створює вектор path розміром n.

- Створює вектор visited розміром n, всі значення якого є False.

- visited[0] = True

- Додає 0 до path.

- totalDistance = 0.0

- Для i від 1 до n-1:

- nearest = -1

- minDist = максимальне\_дійсне\_число

- Для j від 0 до n-1:

- Якщо not visited[j]:

- dist = відстань(graph[path.останній()], graph[j])

- Якщо dist < minDist:

- minDist = dist

- nearest = j

- totalDistance += minDist

- Додає nearest до path.

- visited[nearest] = True

- totalDistance += відстань(graph[path.останній()], graph[0])

- Додає 0 до path.

- Повертає {totalDistance, path}.

5. Функція записати\_результат(filename: рядок, distance: дійсне число, path: вектор<ціле число>):

- Відкриває файл із ім'ям filename.

- Якщо файл не відкритий:

- Виводить "Помилка: Неможливо відкрити файл" разом із ім'ям filename.

- Виходить з програми.

- Записує distance у файл.

- Для кожного елементу i у path:

- Записує i у файл разом з пробілом.

- Записує перенос рядка у файл.

6. Функція головна():

- inputFilename = "шлях\_до\_вхідного\_файлу"

- outputFilename = "шлях\_до\_вихідного\_файлу"

- graph = зчитати\_граф(inputFilename)

- [distance, path] = tsp(graph)

- записати\_результат(outputFilename, distance, path)

- Вивести "Найкоротший шлях:", distance

- Вивести "Шлях:"

- Для кожного елементу i у path:

- Вивести i разом з пробілом.

- Вивести перенос рядка.

**3.2 ПРОГРАМНА РЕАЛІЗАЦІЯ**

**3.2.1 Вихідний код**

struct Point {

double x, y;

};

double distance(const Point& p1, const Point& p2) {

return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));

}

vector<Point> readGraph(const string& filename) {

ifstream file(filename);

if (!file.is\_open()) {

cerr << "Error: Unable to open file " << filename << endl;

exit(1);

}

int n;

file >> n;

vector<Point> graph(n);

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

file >> graph[i].x >> graph[i].y;

}

file.close();

return graph;

}

pair<double, vector<int>> tsp(const vector<Point>& graph) {

int n = graph.size();

vector<int> path;

path.reserve(n);

vector<bool> visited(n, false);

visited[0] = true;

path.push\_back(0);

double totalDistance = 0.0;

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

int nearest = -1;

double minDist = numeric\_limits<double>::max();

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

if (!visited[j]) {

double dist = distance(graph[path.back()], graph[j]);

if (dist < minDist) {

minDist = dist;

nearest = j;

}

}

}

totalDistance += minDist;

path.push\_back(nearest);

visited[nearest] = true;

}

totalDistance += distance(graph[path.back()], graph[0]);

path.push\_back(0);

return {totalDistance, path};

}

void writeResult(const string& filename, double distance, const vector<int>& path) {

ofstream file(filename);

if (!file.is\_open()) {

cerr << "Error: Unable to open file " << filename << endl;

exit(1);

}

file << distance << endl;

for (int i : path) {

file << i << " ";

}

file << endl;

file.close();

}

int main() {

string inputFilename = R"(P:\KPI-Works 2\Algorithms\Laba8\input.txt)";

string outputFilename = R"(P:\KPI-Works 2\Algorithms\Laba8\output.txt)";

vector<Point> graph = readGraph(inputFilename);

auto [distance, path] = tsp(graph);

writeResult(outputFilename, distance, path);

cout << "Shortest distance: " << distance << endl;

cout << "Path: ";

for (int i : path) {

cout << i << " ";

}

cout << endl;

return 0;

}

**3.2.2 Приклади роботи**

**Вхідний файл:**

5

1380 939

2848 96

3510 1671

457 334

3888 666

**Вихідний файл:**

Shortest distance: 8018.39

Path: 0 3 1 4 2 0

