# **ТЕХНІЧНЕ ЗАВДАННЯ**

## **1 ВСТУП**

## **1.1 НАЙМЕНУВАННЯ ПРОГРАМИ**

В даному проекті програма для порівняння графів буде мати назву «Зрівняння графів» (далі по тексту ПО).

## **1.2 СФЕРА ВИКОРИСТАННЯ**

Данна програма буде корисною у виді бібліотеки, яку можна використовувати в великих проектах. Такими проектами можуть бути: гра «шахи», рейси поїздів та літаків, розклад

поїздів та метрополітену, програми навігатори типу «Google Maps».

## **2 ПІДСТАВИ ДЛЯ РОЗРОБКИ**

Підставою для розробки є обов’язковий курсовий проект, у другому семестрі 2 курсу для узагальнення та демонстрації здобутих знать після 2 років вивчення дисципліни «Технології програмування».

## **3 ПРИЗНАЧЕННЯ РОЗРОБКИ**

## **3.1 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ ПРОГРАМИ АБО ПО**

Данна програма зберігає та порівнює графи, тому ця програма може використовуватися в міській інфраструктурі при вираховуванні шляхів метро або поїздів. Також ця програма може використовуватися для побудування кожного разу нових шляхів для інкасаторських автомобілів, задля безпеки поїздки.

## **3.2 ЕКСПЛУАТАЦІЙНЕ ПРИЗНАЧЕННЯ ПРОГРАМИ АБО ПО**

Програма орієнтована на наступне використання:

1. Запуск;
2. Завантаження графів;
3. Обробка графів;
4. Завершення програми.

Програма може використовуватися як в спеціальних умовах (в диспетчерських) так і в домашніх умовах.

## **4 ВИМОГИ ДО ПРОГРАМИ АБО ПРОГРАМНОГО ВИРОБУ**

## **4.1 ВИМОГИ ДО ФУНКЦІОНАЛЬНИХ ХАРАКТЕРИСТИК**

## **4.1.1 ВИМОГИ ДО СКЛАДУ ВИКОНУВАНИХ ФУНКЦІЙ**

Для порівняння графів, вони повинні бути неорієнтовані та у представленні FO. В процесі порівняння будуть порівнюватися вектори повних підграфів потужності від N до N-2.

У рамках задачі порівняння графів, система повинна виконувати такі функції:

1. Графічна побудова графу:

Додавання нової вершини;

Додавання нового ребра;

Видалити весь граф;

Видалення вершини та прилежних, до неї вершин.

1. Збереження поточного графу в бібліотеці;

Граф зберігається у виді екземпляру класу Graph, в якому в виді полів зберігаються необхідні данні, а саме поля: GraphInFO, N, Tops, Ribs. У свою чергу колекції “Tops” і “Ribs” складаються з екземплярів класів “Top” та “Rib” відповідно.

1. Збереження поточного графу у пам’ять у форматі \*.JSON;
2. Завантаження графу з пам’яті у форматі \*.JSON.

## **4.1.2 ВИМОГИ ДО ОРГАНІЗАЦІЇ ВХІДНИХ ДАНИХ**

Вхідні данні представлені в форматі \*.JSON файлу;

У файлі знаходиться серіалізований граф. Розглянемо приклад файлу, зображений на рис.1.

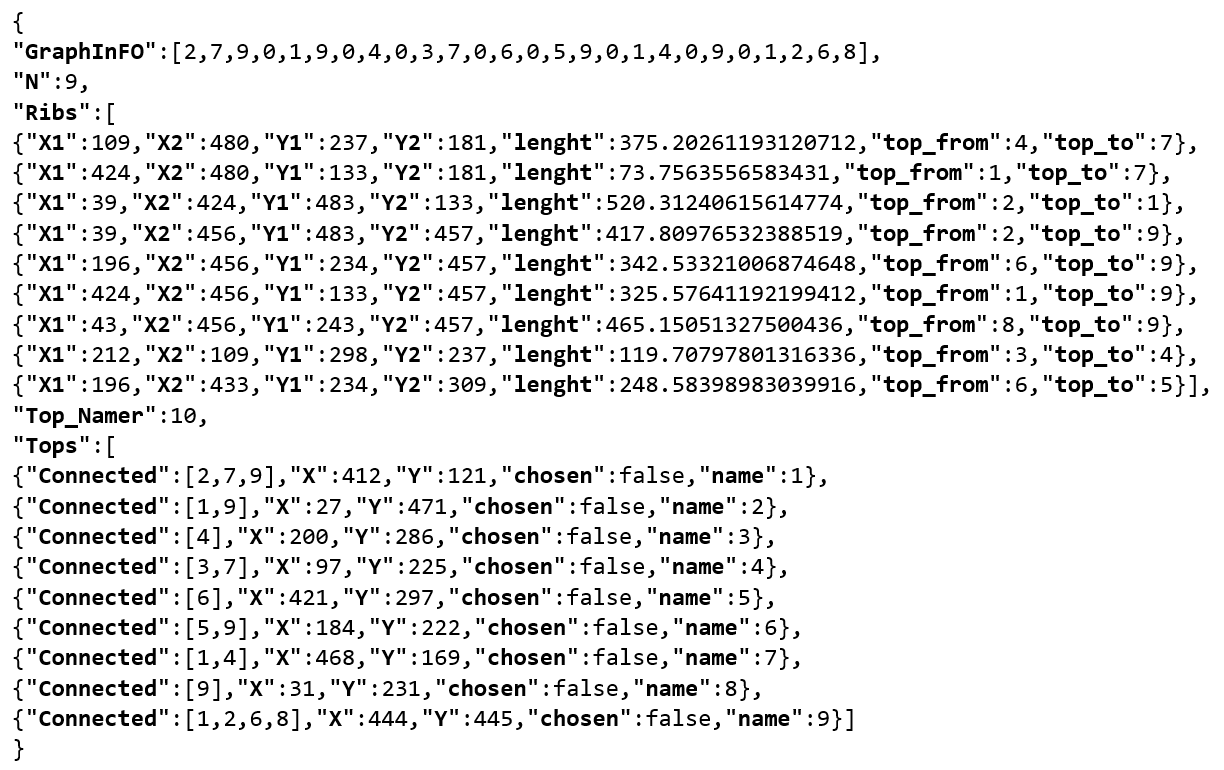


Рисунок 4.1.1 – Приклад серіалізованого графа

**GraphInFO** – Колекція, яка відображає граф у виді FO;

**N** – Кількість вершин графа;

**Ribs** – Колекція ребер графу;

**X1** – Координата X першої вершини ребра;

**X2** – Координата X другої вершини ребра;

**Y1** – Координата Y першої вершини ребра;

**Y2** – Координата Y другої вершини ребра;

**length** – Довжина ребра (обчислюється відносно координатів, представлений вище);

**top\_from** – Ім’я вершини, з якої починається ребро;

**top\_to** – Ім’я вершини, в якій закінчується ребро;

**Top\_Namer** – Поле, яке слідкує за неймінгом нових вершин;

**Tops** – Колекція вершин графа;

**name** – Назва вершини (номер);

**Connected** – Колекція вершин, з якими має спільне ребро вершина “name”;

**X** – Координата X вершини;

**Y** – Координата Y вершини;

**chosen** – Чи зображати цю вершину як “обрану” при побудуванні ребра.

Данні повинні знаходитися на робочому комп’ютері та вказуються через діалогове вікно.

## **4.1.3 ВИМОГИ ДО ОРГАНІЗАЦІЇ ВИХІДНИХ ДАНИХ**

Вихідні данні представлені в форматі \*.JSON файлу. У файлі знаходиться серіалізований граф. Приклад файлу зображений на рис.1. Місце збереження результатів обробки вказується через діалогове вікно.

## **4.1.4 ВИМОГИ ДО ТИМЧАСОВИХ ХАРАКТЕРИСТИКАМ**

Програма повинна виконуватися не більше 1 хвилини на комп’ютері з 8 Гб ОЗУ, процесором i5 8250U на ОС Windows 7 та вище.

## **4.2 ВИМОГИ ДО СКЛАДУ І ПАРАМЕТРІВ ТЕХНІЧНИХ ЗАСОБІВ**

Домашній або робочий комп’ютер;

ОС Windows 7 х32;

Процесор і3 2100;

ОЗУ 2 Гб.

## **4.3 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ І ПРОГРАМНОЇ СУМІСНОСТІ**

## **4.3.1 ВИМОГИ ДО ВИХІДНОГО КОДУ І МОВ ПРОГРАМУВАННЯ**

Програма повинна бути написана на мові програмування C#. В процесі розробки буде використано інструмент для розробки Visual Studio 2019 версії 16.5.2 та Framework 4.8.

## **4.3.2 ВИМОГИ ДО ПРОГРАМНИХ ЗАСОБІВ, ЯКІ ВИКОРИСТОВУЮТЬСЯ ПРОГРАМОЮ**

Для функціювання програми потрібна ОС Windows 7 та вище.

## **5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ**

У результаті розробки програми повинна бути представлена ​​наступна програмна документація:

а) технічне завдання;

б) схеми алгоритмів і даних;

в) текст програми;

г) результати тестування;

д) пояснювальна записка;

е) керівництво користувача.

## **6 СТАДІЇ І ЕТАПИ РОЗРОБКИ**

## **6.1 СТАДІЇ РОЗРОБКИ**

1. Розробка технічного завдання;
2. Робоче проектування;
3. Впровадження ПО.

## **6.2 ЕТАПИ РОЗРОБКИ**

1. Розробка програми;
2. Розробка програмної документації;
3. Випробування програми.

## **6.3 ЗМІСТ РОБІТ ПО ЕТАПАХ**

1) Постановка задачі;

2) Визначення та уточнення вимог до технічних засобів;

3) Визначення вимог до програми;

4) Визначення стадій, етапів і термінів розробки програми і

документації на неї;

5) Вибір мов програмування;

6) Узгодження і затвердження технічного завдання.