**

***Міністерство освіти і науки України***

*Національний технічний університет України*

*«Київський політехнічний інститут імені Ігоря Сікорського»*

***Кафедра АСОІУ***

***Спеціальність 122 Комп'ютерні науки та інформаційні технології***

**КУРСОВА РОБОТА**

**з дисципліни**

**“Додаткові розділи дослідження операцій”**

**на тему**

***«Задача побудови шляху максимальної цінності»***

»

|  |  |  |
| --- | --- | --- |
| *Виконала*  студентка групи ІС-71  Алпаєва Ю.С  N зал. кн. ІС-7103  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка) | *Прийняли*  Доц. каф. АСОІУ, к.т.н.  ЖДАНОВА О.Г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис)  Доц. каф. АСОІУ, к.т.н.  ЖУРАКОВСЬКА О.С.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис)  Доц. каф. АСОІУ, к.т.н.  СПЕРКАЧ М.О.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Національний технічний університет України*  *«Київський політехнічний інститут імені Ігоря Сікорського»* | | | | | | | | | | | | | |
| (назва вищого навчального закладу) | | | | | | | | | | | | | |
| Кафедра | АСОІУ | | | | | | | | | | | | |
| Дисципліна | | Додаткові методи дослідження операцій | | | | | | | | | | | |
| Спеціальність | | | | **126 Інформаційні системи та технології** | | | | | | | | | |
| Курс | | | | | 3 | | | Група | | ІС-71 | | Семестр | 6 |
| **Завдання**  НА КУРСОВУ РоБОТУ СТУДЕНТУ | | | | | | | | | | | | | |
| **Алпаєвій Юлії Сергіївні** | | | | | | | | | | | | | |
| 1. Тема роботи | | | Задача побудови шляху максимальної цінності. | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 2. Термін здачі студентом закінченої роботи | | | | | | | | | *30 травня 2019 року* | | | | |
| 3. Завдання курсової роботи | | | | | | | *Побудувати математичну модель задачі.* | | | | | | |
| *Виконати аналіз можливих методів розв’язання. Розробити два алгоритми ()ефективність методів. знаходження* | | | | | | | | | | | | | |
| *алгоритми розв’язання задачі; запропонувати варіанти основних операторів жадібного Виконат* | | | | | | | | | | | | | |
| *алгоритмів. Розробити програмну реалізацію. Аналіз роботи алгоритмів.* | | | | | | | | | | | | | |
| *. Визначити складність розроблених алгоритмів. Дослідити точність алгоритмів.* | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 4. Зміст розрахунково-пояснювальної записки | | | | | | | | | | | | | |
| *Змістовні постановки задач. Математична постановка задачі.* | | | | | | | | | | | | | |
| *Аналіз можливих методів розв’язання. Опис розроблених алгоритмів розв’язання задачі.* | | | | | | | | | | | | | |
| *задачі.* | | | | | | | | | | | | | |
| *Опис програмної реалізації розробленого алгоритму. Приклади застосування алгоритму.* | | | | | | | | | | | | | |
| *Аналіз розроблених алгоритмів. Висновки.* | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 5. Перелік графічного матеріалу (с точним зазначенням обов’язкових креслень) | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 6. Дата видачі завдання | | | | | | *20 лютого 2019 року* | | | | | | | |
|  | | | | | | | | | | |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Назва етапів виконання курсової роботи** | | | **Строк виконання етапів роботи** | **Примітка** |
| 1 | *Опис постановки задачі* | | | *23.03* |  |
| 2 | *Консультація у керівника* | | | *26.03* |  |
| 3 | *Пошук літературних джерел за темою курсової роботи. Порівняльний аналіз існуючих методів розв’язання задачі* | | | *29.03* |  |
| 4 | *Вивчення теоретичних положень алгоритмів* | | | *31.03* |  |
| 5 | *Розробка алгоритмів розв’язання задачі* | | | *10.04* |  |
| 6 | *Розв’язання задач “вручну”* | | | *15.04* |  |
| 7 | *Консультація в керівника – погодження інтерфейсу програми* | | | *18.04* |  |
| 8 | *Розробка програмної реалізації. Відлагодження програм.* | | | *10.05* |  |
| 9 | *Оцінка складності алгоритмів розв’язання задачі* | | | *20.05* |  |
| 10 | *Оформлення пояснювальної записки* | | | *23.05* |  |
| 11 | *Захист курсової роботи* | | | *02.06* |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  | | | | | |
| Студентка | |  | *Алпаєва Ю.С.* | | |
|  | | (підпис) | (прізвище, ініціали) | | |
|  | |
|  | |
| Керівник | |  |  | | |
|  | | (підпис) | (прізвище, ініціали) | | |
| «*21*» *лютого 2020 року* | | | | | |

**ЗМІСТ**

[**ВСТУП** 7](#_Toc42165075)

[**1 ПОСТАНОВКА ЗАДАЧІ** 8](#_Toc42165076)

[**1.1 Змістовна постановка задачі** 8](#_Toc42165077)

[**1.2 Індивідуальна постановка задачі** 8](#_Toc42165078)

[**1.3 Математична постановка задачі** 10](#_Toc42165079)

[**1.3.1 Математична модель в термінах теорії графів** 10](#_Toc42165080)

[**2 ОПИС МЕТОДІВ РОЗВ’ЯЗАННЯ ЗАДАЧІ** 11](#_Toc42165081)

[**2.1 Існуючі методи розв'язання** 11](#_Toc42165082)

[**2.3.1 Розробка алгоритму розв’язання задачі** 12](#_Toc42165095)

[**2.3.2 Оцінка трудомісткості алгоритму** 14](#_Toc42165096)

[**3 ОПИС ПРОГРАМНОГО ПРОДУКТУ** 16](#_Toc42165106)

[**3. 1 Опис функціональної моделі** 16](#_Toc42165107)

[**3. 2 Рішення з інформаційного забезпечення** 16](#_Toc42165108)

[**3.2.1 Вхідні дані** 16](#_Toc42165109)

[**3.2.2 Вихідні дані** 17](#_Toc42165110)

[**3.2.3 Опис інформаційного забезпечення** 17](#_Toc42165112)

[**3.3 Рішення з програмного забезпечення** 18](#_Toc42165113)

[**3.3.1 Опис обраних засобів розробки** 18](#_Toc42165114)

[**3.3.2 Архітектура програмного забезпечення** 19](#_Toc42165115)

[**3.3.2.1 Схема архітектури ПП** 19](#_Toc42165116)

[**3.3.2.2 Діаграма класів (об'єктів)** 19](#_Toc42165117)

[**3.3.2.3 Специфікація функцій** 22](#_Toc42165118)

[**3.4 Керівництво користувача** 23](#_Toc42165119)

[**3.4.1 Інструкція користувача** 23](#_Toc42165120)

[**3.4.2 Методика випробувань** 27](#_Toc42165121)

[**4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕНННЯ АЛГОРИТМІВ** 29](#_Toc42165122)

[**4.1 Приклади розв’язання задач** 29](#_Toc42165123)

[**4.1.1 Розв'язання індивідуальної задачі** 29](#_Toc42165124)

[**4.2 Експерименти та їх результати** 31](#_Toc42165125)

[**4.2.1 Опис обладнання яке використовувалось під час експерементів** 31](#_Toc42165126)

[**4.4 Дослідження другого алгоритму** 32](#_Toc42165127)

[**4.4.1 Дослідження часу виконання алгоритму** 32](#_Toc42165128)

[**4.5 Порівняння точності алгоритмів.** 33](#_Toc42165129)

[**ВИСНОВКИ** 34](#_Toc42165130)

[**ПЕРЕЛІК ПОСИЛАНЬ** 35](#_Toc42165131)

[**Додаток А** 36](#_Toc42165132)

# **РЕФЕРАТ**

Пояснювальна записка до курсової роботи: 37 с., 23 рис., 2 табл., 1 додаток, 10 джерел.

**Предмет дослідження:** Задача побудови шляху максимальної цінності.

**Мета:** дослідження існуючих алгоритмів пошуку шляху максимальної цінності у графі та їх модифікація. Порівняльний аналіз розроблених алгоритмів.

Для вирішення даної задачі в було розроблено два алгоритми точний і жадібний. Ідея першого була взята з алгоритма Дейкстри , а другого з алгоритму пошуку в ширину. Обидва алгоритми були перероблені відповідно до вимог задачі.

Після розробки алгоритмів проведено їх тестування та порівняння часу їх роботи.

Для backend частини було використано мову програмування Java8 і фреймворк Spring Boot , для frontend частини було використано фреймворк Angular 2.

Використовувався архітектурний шаблон MVC.

ПОШУК ШЛЯХУ МАКСИМАЛЬНОЇ ЦІННОСТІ В ГРАФІ, ЖАДІБНИЙ АЛГОРИТМ, ПОШУК В ГЛИБИНУ, BFS, АЛГОРИТМ ДЕЙКСТРИ

**ВСТУП**

Задача пошуку шляху максимального шляху відноситься до комбінаторних задач. Існує багато алгоритмів для пошуку максимального за цінністю ребер шляху в графі. В даній роботі розглядається задача пошуку максимального за цінністю вершин шляху, початок і кінець якого невідомий. Також існують ліміти на деякі характеристики ребер і вершин, які не має перевищувати шлях.

Для розв’язання цієї задачі не існує готових алгоритмів, отже постала задача розробити їх самостійно. Було розроблено два алгоритми. Ідея першого була взята з алгоритму Дейкстри, для другого було модифіковано пошук в ширину.

# **1 ПОСТАНОВКА ЗАДАЧІ**

# **1.1 Змістовна постановка задачі**

Мандрівник хоче скласти маршрут подорожі містами , кожне місто має естетичну цінність *ci*, і час необхідний для ознайомлення з ним *τi*  . Для кожної пари міст відомі вартості і час, який треба витратити щоб дістатися з однієї вершини в іншу . Також мандрівник має обмежений бюджет С, і ліміт часу Т.

Необхідно знайти маршрут з максимальною естетичною цінністю, час якого не перевищує Т, і вартість не перевищує С.

**Цикл** (в теорії графі) — ланцюг *x*0*u*1*x*1*u*2*x*2…*x*l−1*u*l*x*0, в якому перша та остання вершина збігається з початковою.

**Маршрут** ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *walk*) в графі — скінчена або нескінчена послідовність ребер S = { e, e1, …, en } в якій кожній два сусідні ребра *ei -1* і *ei* мають спільну вершину.

# **1.2 Індивідуальна постановка задачі**

Мандрівник хоче скласти маршрут подорожі містами , кожне місто має естетичну цінність *ci*, і час необхідний для ознайомлення з ним *τi*  . Для кожної пари міст відомі вартості і час,який треба витратити щоб дістатися з однієї вершини в іншу . Також мандрівник має обмежений бюджет С, і ліміт часу Т.

Необхідно знайти маршрут з максимальною естетичною цінністю, час якого не перевищує Т, і вартість не перевищує С.

Нехай С = 120 од. , T = 130 од. Значення для кожної вершини та души зображені на рисунку. На Рисунку 1.1 зображений граф, який відповідає постановці індивідуальної задачі.

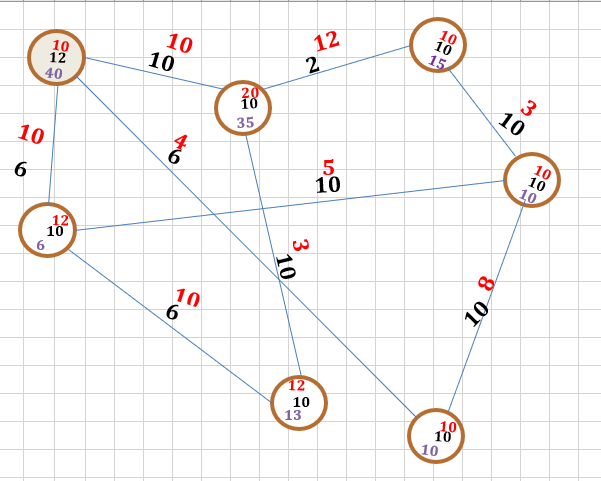


Рисунок.1.1- Постановка індивідуальної задачі

На Рисунку 1.2 і Рисунку 1.3 зображені можливі розв’язки задачі.

Значення цільової функції розв’язку зображеного на Рисунку 1.2 – 110.

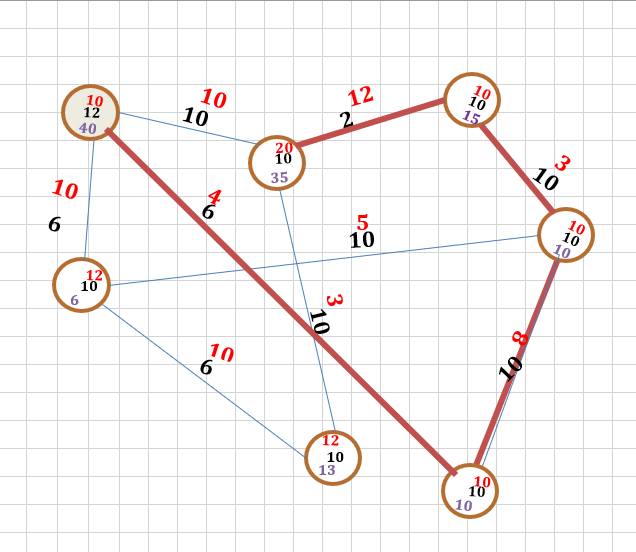


Рисунок 1.2- Можливий розв’язок індивідуальної задачі

Значення цільової функції розв’язку зображеного на Рисунку 1.3 – 106.

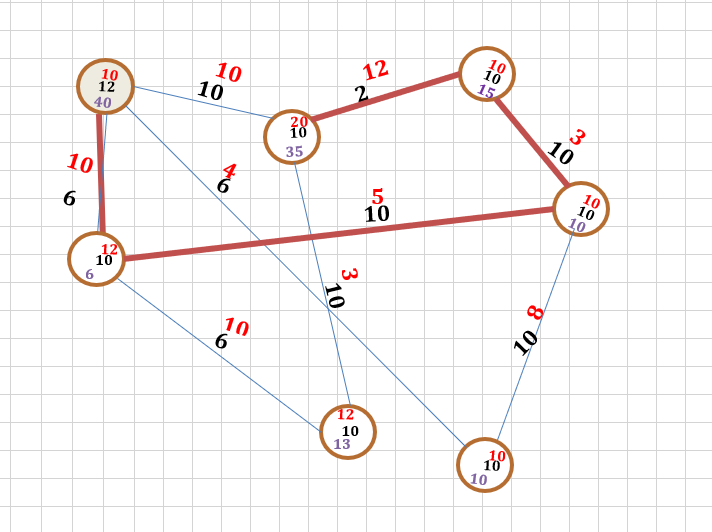


Рисунок 1.3- Можливий розв’язок індивідуальної задачі

# **1.3 Математична постановка задачі**

# *1.3.1 Математична модель в термінах теорії графів*

Маємо граф ,

де - непорожня кінцева множина вершин ,

– множина дуг,

: матриця часу переміщень між вершинами ( )

: – матриця цінностей ребр ( ).

τ : - час перебування в вершині ( ).

D : - естетична цінність ( ).

K: - вартість перебування у вершині ( ).

Знайти: маршрут R = {

З максимальною сумарною цінністю вершин, у якого:

А також для нього досягається:

# **2 ОПИС МЕТОДІВ РОЗВ’ЯЗАННЯ ЗАДАЧІ**

# **2.1 Існуючі методи розв'язання**

Задача дана в курсовій роботі є задачею пошуку шляху у графі.

Методом розв’язання такої задачі може бути алгоритм Дейкстри або Алгоритм Беллмана — Форда . Алгоритм Дейскстри і Алгоритм Беллмана  знаходять мінімальні шляхи від заданої вершини до іншої.

Алгоритм Форда-Беллмана має декілька фаз, на кожній ітерації він намагається покращити поточний розв’язок ,продивляючись всі ребра графа. Наприклад, якщо ми маємо ребро *(a,b)* то алгоритм намагається покращити відповідь для вершини b , покращуючи значення *d[b] d[a]+c.*

А алгоритм Дейсктри обирає вершину , яка має найменшу відстань від поточної і намагається її покращити.

**2.2 Розробка  жадібного алгоритму розв’язання задачі**

2.3 Опис методу (алгоритму)

Алгоритм 1

Для розв’язання задачі пропонується наступний алгоритм.

В задачі не вказано ні стартової вершини ні кінцевої. Отже пропонується знайти для кожної вершини всі шляхи, які відповідають лімітам задачі, а потім серед них обрати максимальний.

Ідея алгоритму пошуку всіх максимальних шляхів для однієї вершини.

Кожній вершині графу присвоюється мітка – естетична цінність вершини.

Мітка самої вершини ,для якої шукаємо шляхи, покладається рівною 0, мітки інших вершин – мінус нескінченності. Всі вершини графа позначаються як невідвідані.

Крок алгоритму: алгоритм завершується якщо все вершини відвідані або якщо утворений шлях перевищує ліміти задачі.

В іншому випадку, з ще не відвіданих вершин вибирається вершина u, що має максимальну мітку (цінність). Для кожного сусіда вершини u, крім позначених відвіданими, розглянемо нову цінність шляху, що дорівнює сумі значень поточної мітки u і цінності поточного шляху. Якщо отримане значення довжини більше за значення мітки сусіда, замінимо значення мітки отриманим значенням довжини. Розглянувши всіх сусідів, позначимо вершину u як відвіданих і повторимо крок алгоритму.

Алгоритм 2

Ідея алгоритму базується на тому що на кожному кроці обирається найкраща вершина.

Обхід графа виконується за принципами пошуку в ширину.

Крок алгоритму: на кожному кроці обирається вершина з максимальною цінністю, якщо її ще не розглядали ,кладемо її в чергу, і у список відвіданих вершин.

# *2.3.1 Розробка алгоритму розв’язання задачі*

**Схема алгоритму 1**

1. **Вхід:** с//матриця вартостей переїзду
2. t//матриця часів переїздів
3. d//матриця естетичних цінностей вершин
4. k//матриця вартостей перебування вершин
5. τ//матриця вартостей перебування вершин
6. P //шлях
7. Для всіх вершин графу виконуємо процедуру
8. Procedure
9. **Вхід** a// вершина для якої шукаємо шляхи
10. **Вихід:**  R //шлях
11. **Ініціалізація :** мітка a = 0, всіх інших вершин – INF
12. S ={0} всі вершини непройдені
13. с = 0 вартість шляху
14. R – шлях, пустий список вершин.
15. **While** **do**
16. **Обираємо**
17. **If** {
18. //для кожного сусіда розраховуємо мітку
19. //розглядаємо нову цінність маршруту
20. // if (якщо нова цінність більша за стару){
21. **If** ( ліміти задані задачею виконуються)
22. Додаємо u в шлях R і помічаємо пройденою
23. Змінюємо загальну цінність шляху, з урахуванням нової вершини
24. **Else** помічаємо пройденою обираємо іншого сусіда
25. End
26. **Повторюємо для всіх вершин**

**Обчислюємо** значення ЦФ для кожного шляху R та обираємо серед всіх отриманих R шлях P з максимальною ЦФ.

**Схема алгоритму 2**

1. **Вхід:** с//матриця вартостей переїзду
2. t//матриця часів переїздів
3. d//матриця естетичних цінностей вершин
4. k//матриця вартостей перебування вершин
5. τ//матриця вартостей перебування вершин
6. P //шлях
7. Для всіх вершин графу виконуємо процедуру
8. Procedure
9. **Вхід** a// вершина для якої шукаємо шляхи
10. **Вихід:**  R //шлях
11. **Ініціалізація :**
12. F – черга вершин
13. E – черга пройдених вершин
14. **While** **do**
15. **If** {
16. **Return E**
17. **}**
18. **Беремо перший елемент з F**
19. **Знаходимо його сусіда з максимальною цінністю**
20. **If( елемента немає в E ){**
21. **Додаємо його в F**
22. **Додаємо в E**
23. **}**
24. **Повторюємо для всіх вершин**

**Обчислюємо** значення ЦФ для кожного шляху R та обираємо серед всіх отриманих R шлях P з максимальною ЦФ.

# *2.3.2 Оцінка трудомісткості алгоритму*

Нехай , .

Для алгоритму 1

Оцінка процедури пошуку шляхів для однієї вершини:

Цикл *while* повторюється разів ( де – кількість вершин), для знаходження максимуму серед сусідів виконується *v* операцій, Цикл для обирання сусідів вершини *v* – операцій, .

Загальна складність алгоритму:

Оскільки дана процедура виконується для кожної вершини графу то загальна складність , .

Для алгоритму 2

Алгоритм проходить один раз по вершинам. Цикл *while* повторюється разів. Cкладність алгоритму .

# **2.4 Приклади застосування розроблених алгоритмів**

Умова взята з пункту 1.1.2 Постановка індивідуальної задачі.

На Рисунку 2.1 і Рисунку 2.2 зображені можливі розв’язки задачі.

Значення цільової функції розв’язку зображеного на Рисунку 2.1 – 110.

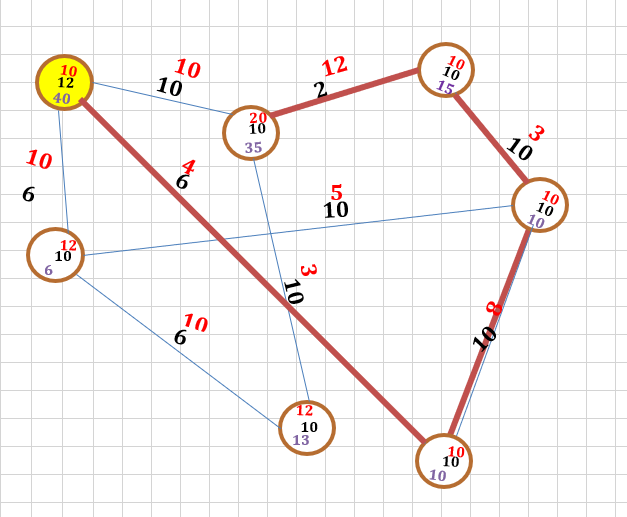


Рисунок 2.1- Можливий розв’язок індивідуальної задачі

Значення цільової функції розв’язку зображеного на Рисунку 2.2 – 106.

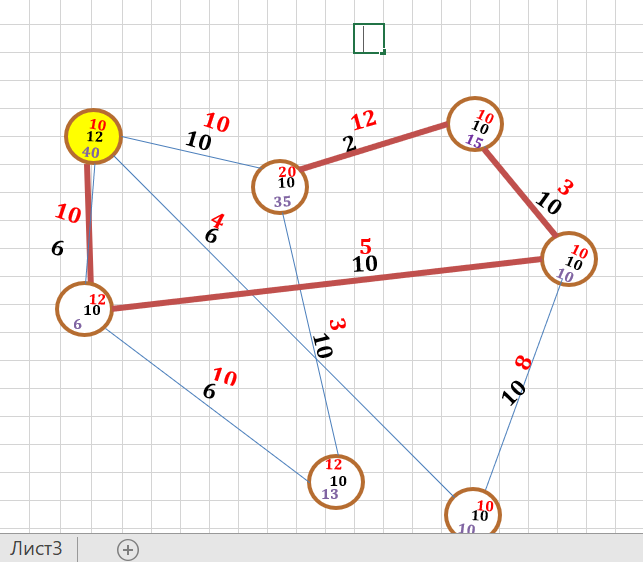


Рисунок 2.2- Можливий розв’язок індивідуальної задачі

# **3 ОПИС ПРОГРАМНОГО ПРОДУКТУ**

# **3. 1 Опис функціональної моделі**

Програмний продукт містить наступні складові:

1. **робота з індивідуальною задачею** (ІЗ):
   1. введення вручну даних ІЗ;
   2. генерація (випадковим чином) даних ІЗ;
   3. розв’язання ІЗ двома алгоритмами, виведення маршруту і значення ЦФ;
2. **експериментальне дослідження розроблених алгоритмів:** 
   1. задання розмірності задачі, кількість ребер та вершин у графі.
   2. генерація ІЗ
   3. розв’язання множини згенерованих ІЗ двома розробленими алгоритмами;
   4. виведення результатів експериментів в консоль:
      * порівняння за точністю;
      * порівняння за часом.

# **3. 2 Рішення з інформаційного забезпечення**

# *3.2.1 Вхідні дані*

Вхідними даними задачі є характеристики вершин і ребер графу.

Для вершини задається ім’я, її попередня вершина (в ході роботи алгоритмів буде присвоюватись нове значення для того щоб побудувати шлях після розрахунків), час перебування у вершині, її естетична цінність, та скільки коштує перебування у вершині. Для ребра задається пара вершин, які воно сполучає ,а також час за який його можна перетнути і скільки коштує переїзд. Також задається величини, які обмежують час подорожі і гроші, які можна витратити на подорож.

# *3.2.2 Вихідні дані*

# Вихідні дані: шлях з максимальною цінністю, який вкладається в обмеження часу вартості.

# *3.2.3 Опис інформаційного забезпечення*

В таблиці 3.1 наведений опис власних типів, які використовувались в ПЗ. Для зручності введення даних ,використовувалось дві моделі представлення графу.

Таблиця 3.1 – Опис власних типів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва типу | Поле | Тип поля | Призначення  поля | Призначення типу |
|  | name | String | Ім’я вершини |  |
| Vertex | adjacencies | List<Edge> | Список ребер,які з неї виходять | Вершина |
|  | maxValuePath | Double | Для розрахунків (естетична цінність шляху, який веде у цю вершину). |  |
|  | previous | Vertex | Вершина ,яка передує даній |  |
|  | Cost | Int | Ціна перебування у вершині |  |
|  | Time | Int | Час перебування у вершині |  |
|  | value | int | Естетична цінність |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | from | Vertex | Вершина, з якої ребро виходить | Ребро |
| Edge | to | Vertex | Вершина, у яку ребро входить |  |
|  | Time | Int | Час переїзду |  |
|  | cost | int | Вартість переїзду |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Graph | Vertexes | List<Vertex> | Список вершин | Представлення графу front-end. |
|  | edges | List<Edges> | Список ребер |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GraphDto | Vertexes | List<Vertex> | Список вершин | Представлення графу back-end. |

# **3.3 Рішення з програмного забезпечення**

# *3.3.1 Опис обраних засобів розробки*

Для розробки ПП використовувались наступні технології :

Для backend застосунку:

* Java 8 (мова програмування);
* Spring boot 2 (фреймворк);

Для frontend застосунку:

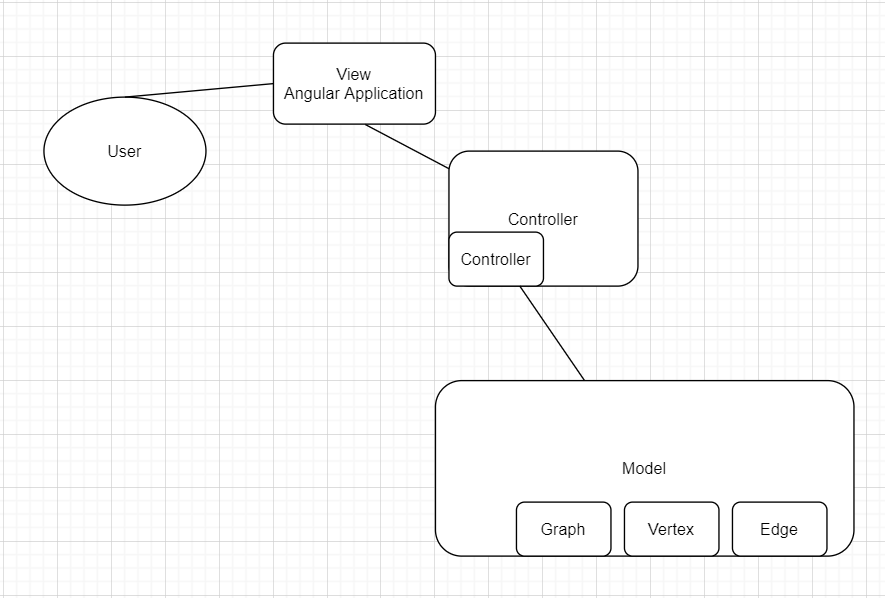
* Angular 2 (фреймворк);

Angular має надає багато засобів для розробки, зручний у використанні для розробки frontend. Оскільки frontend і beckend це два самостійних застосунки, які передають дані використовуючи REST, зручним є те що Angular і Spring boot мають засоби (анотації та методи) для реалізації такої передачі даних.

# **3.3.2 Архітектура програмного забезпечення**

# *3.3.2.1 Схема архітектури ПП*

Додаток був спроектований за моделлю MVC(Model View Controller). На Рисунку 3.1 зображена схема архітектури застосунку.



*Рисунок 3.1 –* Схема архітектури ПЗ

# *3.3.2.2 Діаграма класів (об'єктів)*

На Рисунку 3.2 зображена повна діаграма класів застосунку.

*Изображение выглядит как текст, карта

Автоматически созданное описание*

*Рисунок 3.2 –* Діаграма всіх класів системи.

На Рисунку 3.3 зображені класи застосунку, які відносяться до моделі в паттерні MVC

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.3 – Діаграма класів з головними сутностями системи.

# 

# *3.3.2.3 Специфікація функцій*

В таблиці 3.2 наведений опис функцій, які використовувались у проекті.

Таблиця 3.2 – Опис функцій

|  |  |  |
| --- | --- | --- |
| Метод | Клас | Призначення |
| private ArrayList<Vertex> computePaths(Vertex source,GraphDto graphDto) | PathService | Розрахунок цінності можливих шляхів в графі, для першого алгоритму. |
| private List<Vertex> getPathTo(Vertex target) | PathService | Побудова шляхів ,на базі розрахунків функції, описаної вище. |
| public List<Vertex> findPathServiceDejkstra(GraphDto graph, int costLimit, int timeLimit) { | PathService | Розрахунок шляху для кожної вершини, для всіх вершин графу точним алгоритмом |
| public List<Vertex> findPathServiceGreedy(GraphDto graph, int costLimit, int timeLimit) { | PathService | Розрахунок шляху для кожної вершини, для всіх вершин графу жадібним алгоритмом |
| public List<Vertex> greedy(Vertex source,int costLimit, int timeLimit) | PathService | Пошук шляхів з однієї вершини жадібним алгоритмом. |
| private List<Vertex> findPathWithMaxCost(List<List<Vertex>> paths) | PathService | Вибір шляху з максимальної цінністю, серед найдених алгоритмами. |
| public static boolean checkLimits(List<Vertex> path, int costLimit, int timeLimit) | LimitService | Перевірка, чи відповідає шлях заданим лімітам |
| private static boolean checkCost(List<Vertex> path, int costLimit) | LimitService | Перевірка виконання ліміту грошей |
| private static boolean checkTime(List<Vertex> path, int timeLimit) | LimitService | Перевірка виконання ліміту часу |
| public static GraphDto generateService(int sizeVertexes, int sizeEdges) | InputService | Генерація значень рандомним чином |

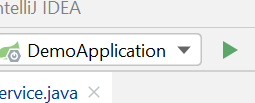
# **3.4 Керівництво користувача**

# *3.4.1 Інструкція користувача*

Проект можна запустити з інтегрованого середовища розробки Intellij IDEA, натиснувши на кнопку з піктограмою у вигляді зеленого трикутника.

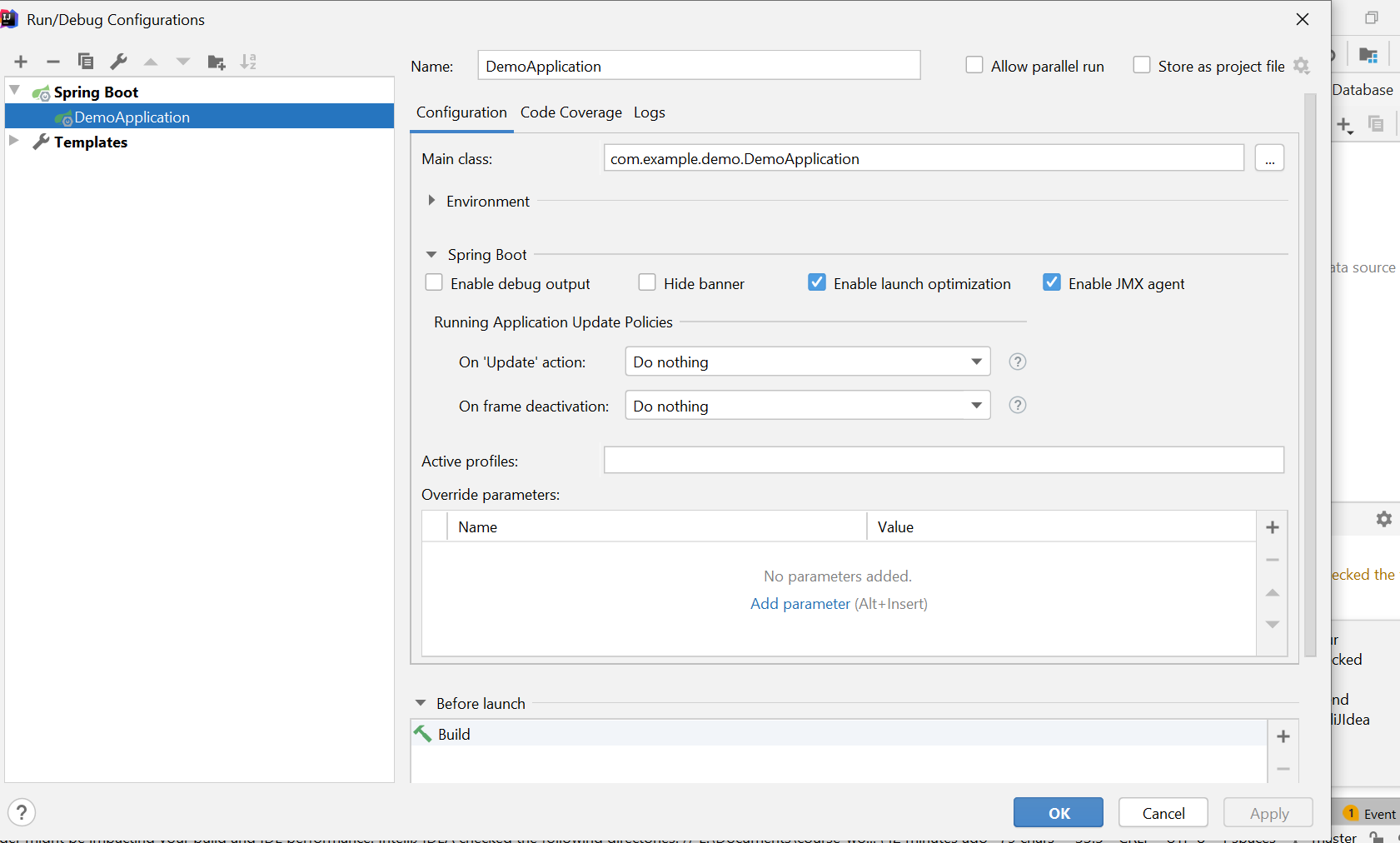
Для backend застосунку:

На Рисунку 3.3 зображена кнопка запуску для backend застосунку.

**Рисунок 3.3

Опис конфігурації для запуску:

На Рисунку 3.4 зображена конфігурація запуску для backend застосунку.

*Рисунок 3.4*

Для frontend застосунку:

Виконати команду npm install в папці з frontend. Запуск команда – ng serve

І перейти за посиланням <http://localhost:4200/>

Або за допомогою IDE.

На Рисунку 3.5 зображена кнопка запуску фронтенду застосунку.

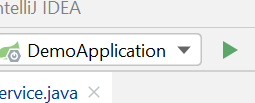
**

Рисунок 3.5

На Рисунку 3.6 зображений опис конфігурації для запуску frontend:

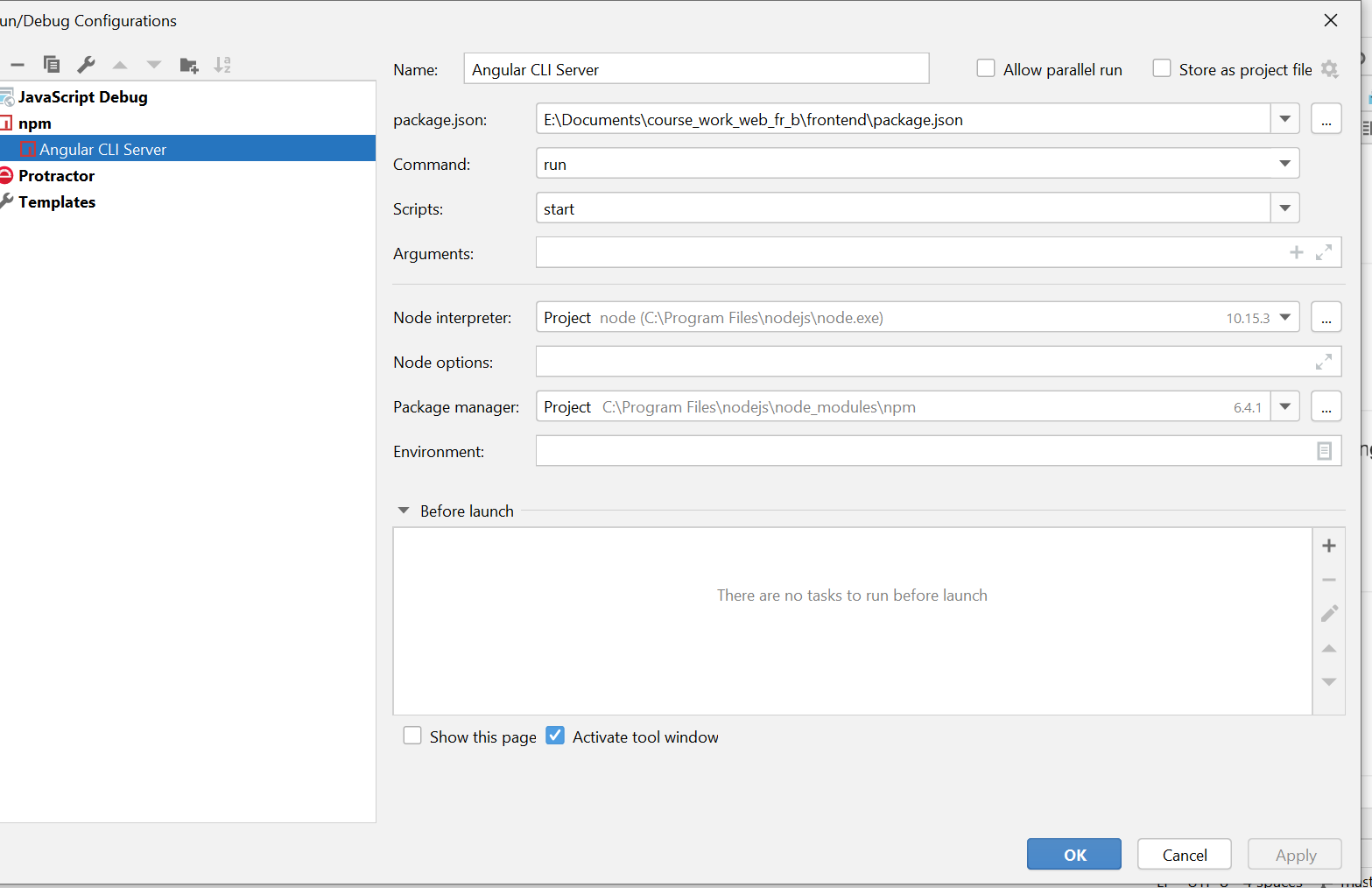


Рисунок 3.6 -Опис конфігурації запуску frontend застосунку.

Інструкція по роботі користувача з системою:

На Рисунку 3.7 зображена головна сторінка користувача системи.

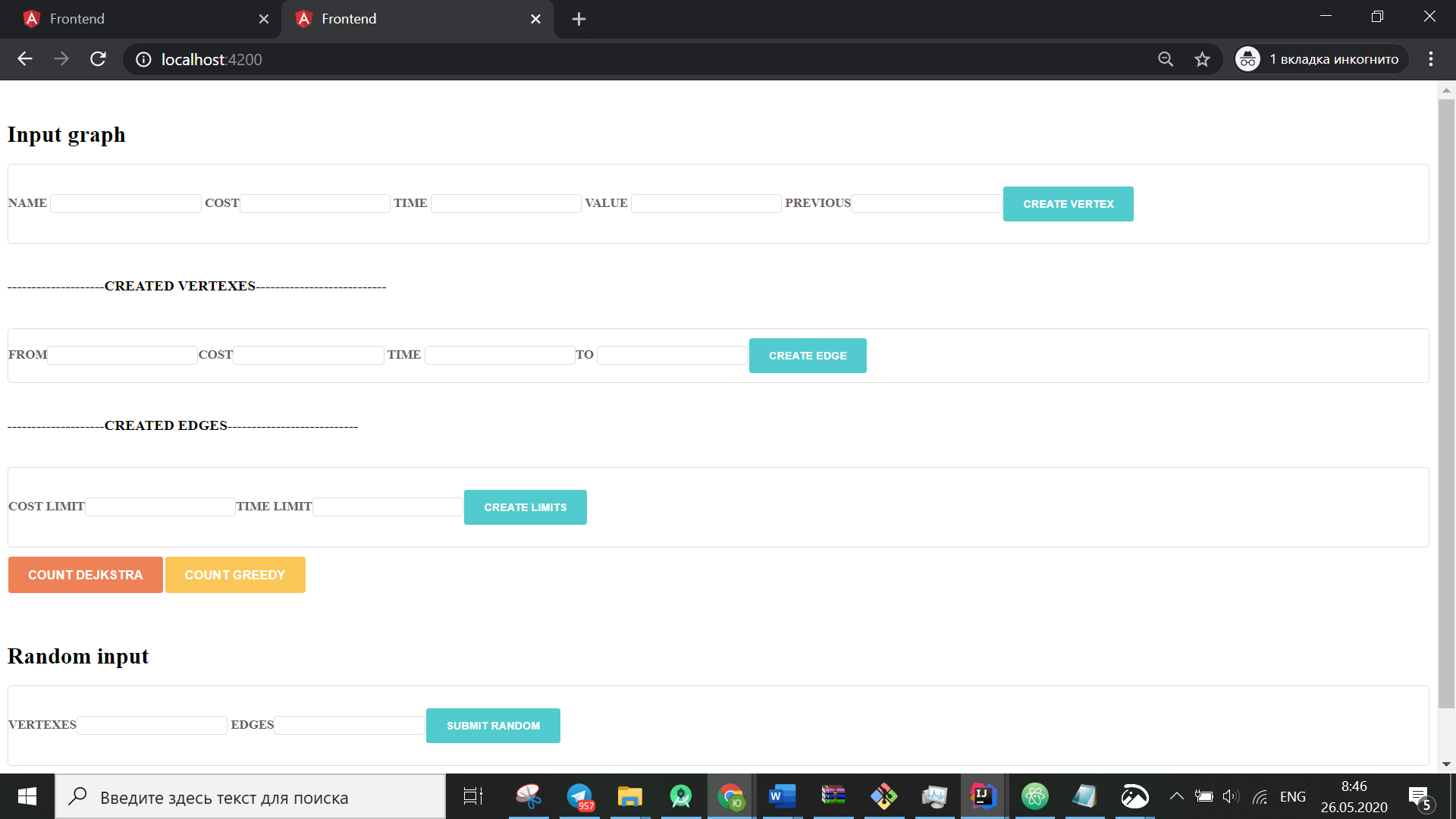


Рисунок 3.7 – Головна сторінка застосунку

На Рисунку 3.8 зображене введення даних для розрахунку шляхів.

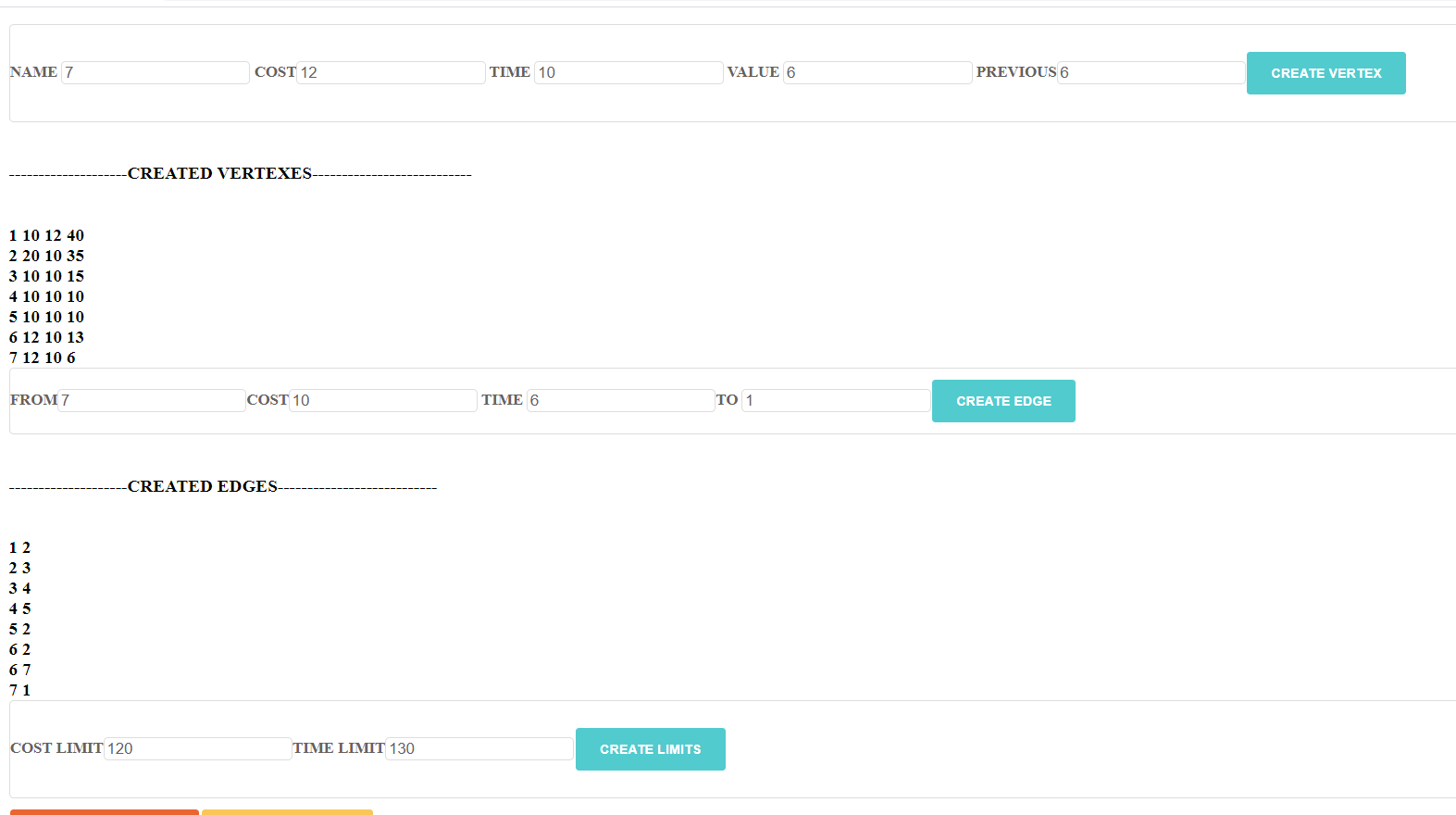


Рисунок 3.7 – Введення даних

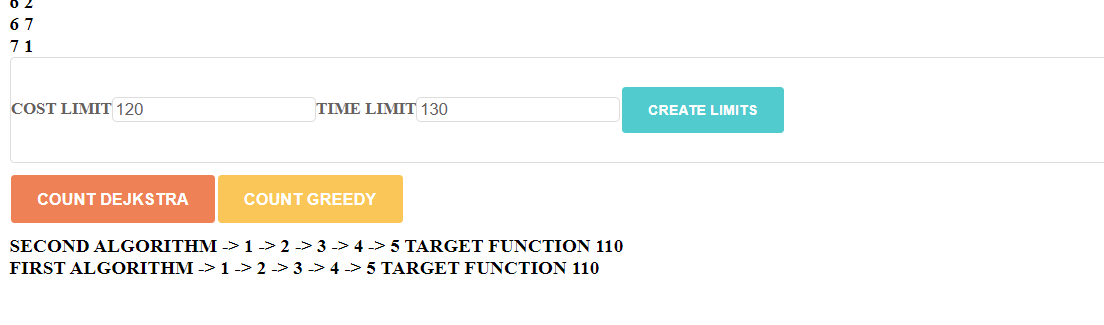


Рисунок 3.8 – Отримання результату

Генерація даних рандомним чином:

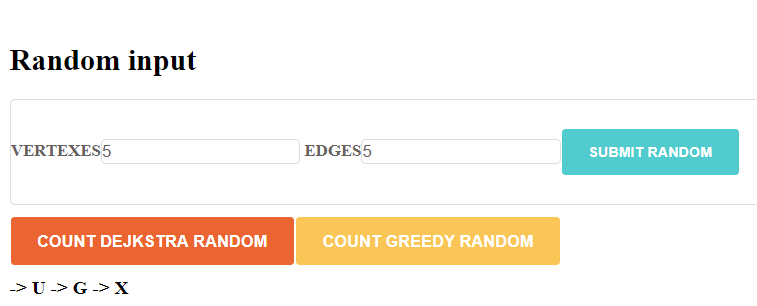


Рисунок 3.9– Отримання результату

# *3.4.2 Методика випробувань*

Для тестування ПЗ було застосовано методи чорної та білої скрині. Для обох видів тестувань спочатку тестування проводилось на задачі розмірності 3 (три вершини ,три ребра), перевірялась коректність роботи методів розрахунку (наприклад, розрахунок величин для перевірки виконання лімітів, розрахунок цільової функції), далі тестування проводилось для задач розмірності 5 і більше , перевірялась правильність роботи алгоритмів.

Тестування API проводилось за допомогою Postman. На Рисунку 3.10 зображена сторінка застосунку Postman, запит і результат виконання запиту.

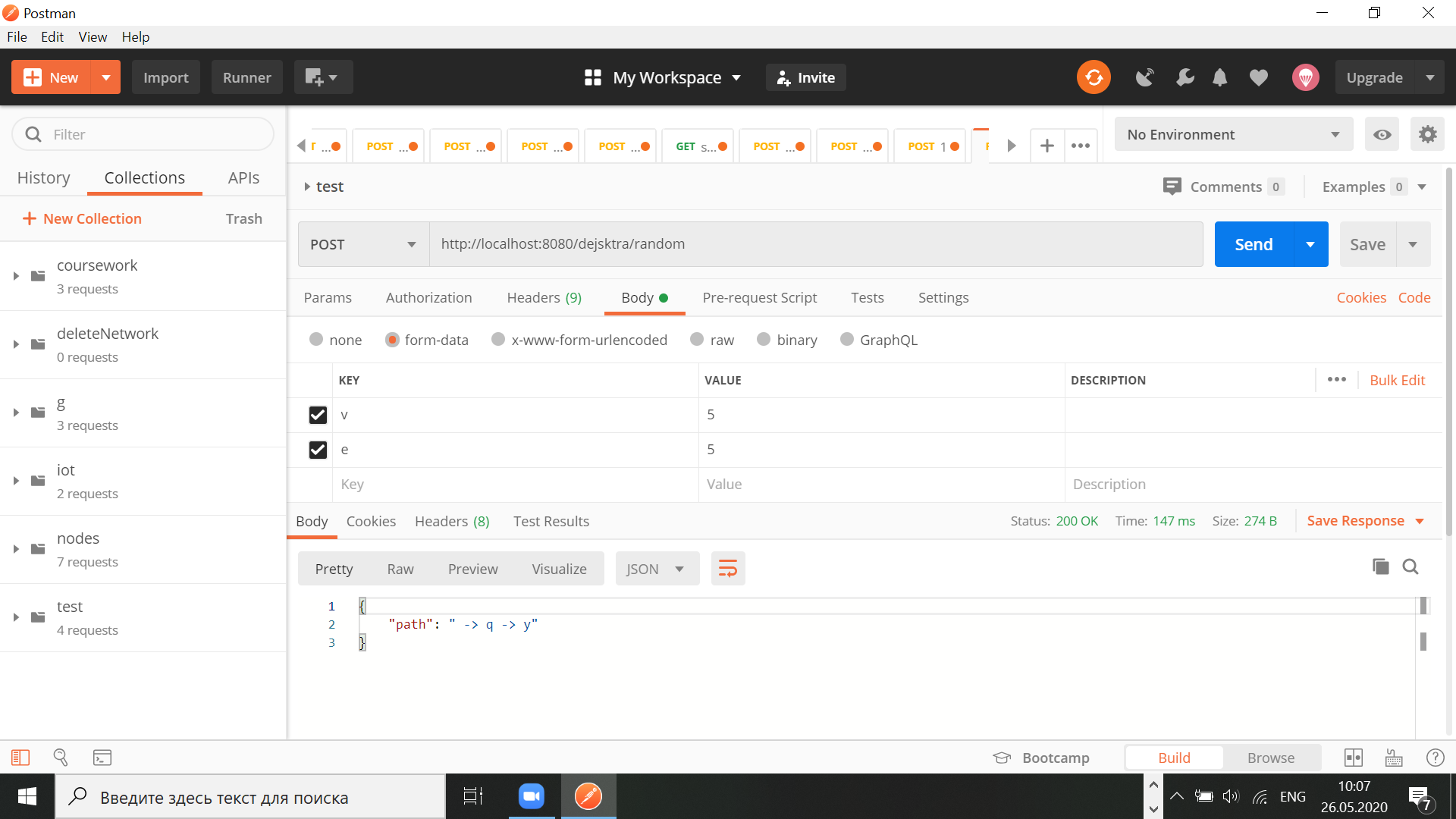


Рисунок 3.10 – тестуваня API

# **4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕНННЯ АЛГОРИТМІВ**

Для того, щоб експеримент мав статистичну значимість, буде використовуватись випадковий спосіб генерування задач.

1. Розмірності для першого алгоритму будуть від 5 до 30 з кроком 1. Сумарна кількість задач – 25.
2. Розмірності для другого від 10 до 710 з кроком 20.

Сумарна кількість задач – 35.

Така разбіжність викликана тим ,що час розрахунку шляхів другого алгоритму зростає досить повільно. Тому щоб провести аналіз було проведено більше випробувань на більших розмірностях задач.

# **4.1 Приклади розв’язання задач**

# *4.1.1 Розв'язання індивідуальної задачі*

На Рисунку 4.1 представлена умова індивідуальної задачі.

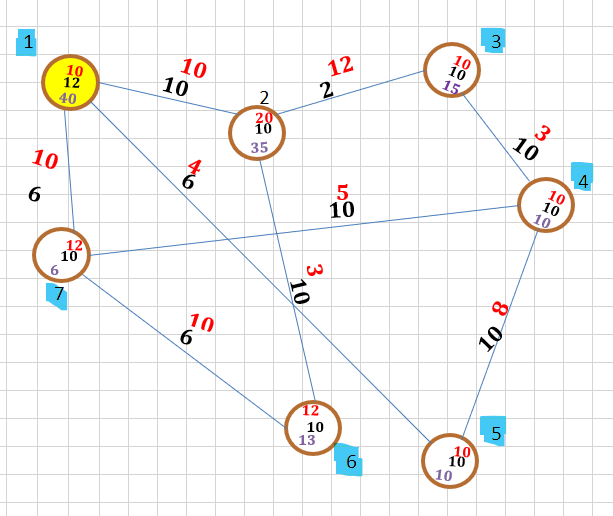


Рисунок 4.1 Умова індивідуальної задачі

На Рисунку 4.2 зображена головна сторінка застосунку, і введення даних задачі.

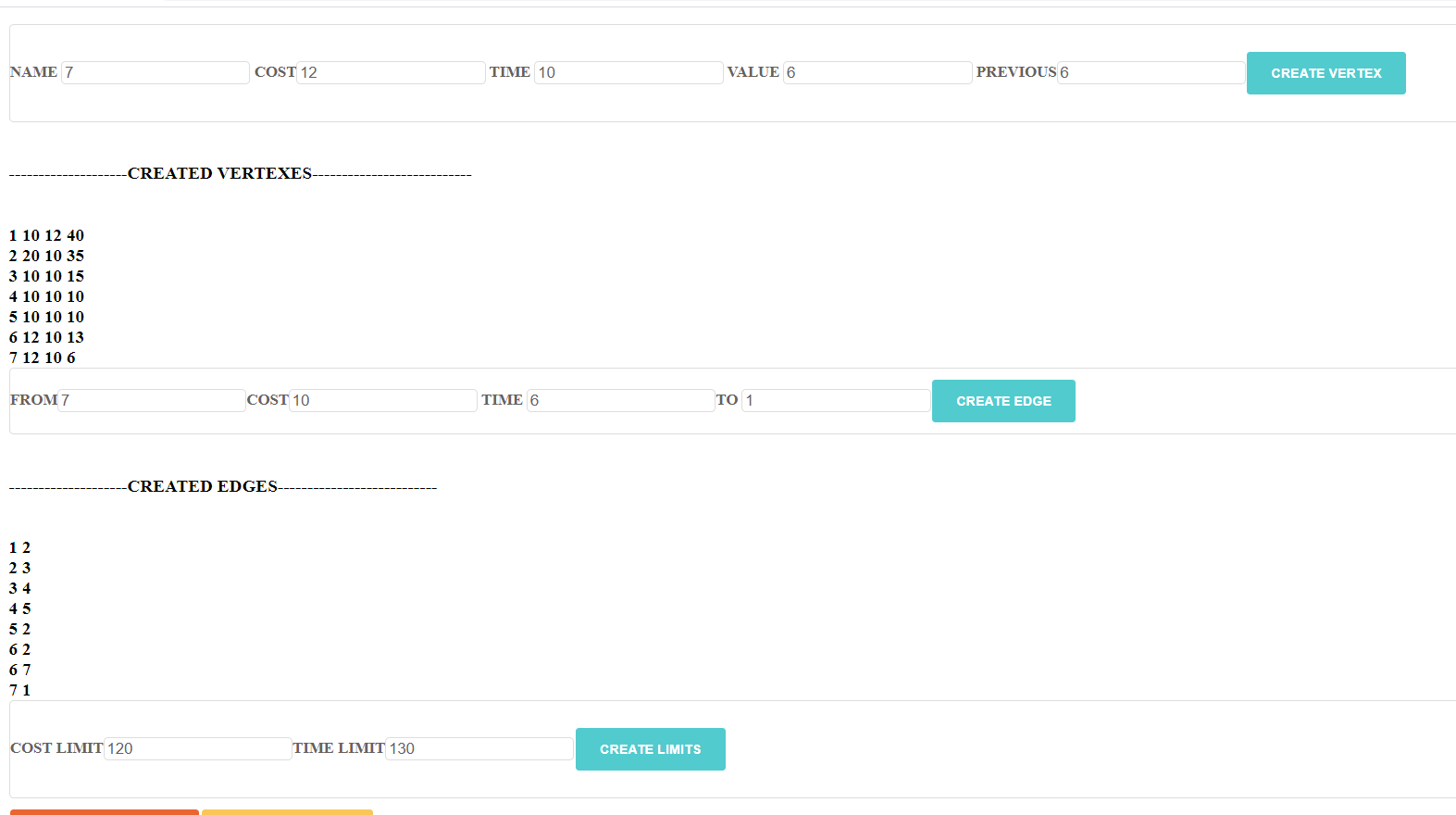


Рисунок 4.2 Форма з умовою індивідуальної задачі

На Рисунку 4.3 зображений результат роботи програми.

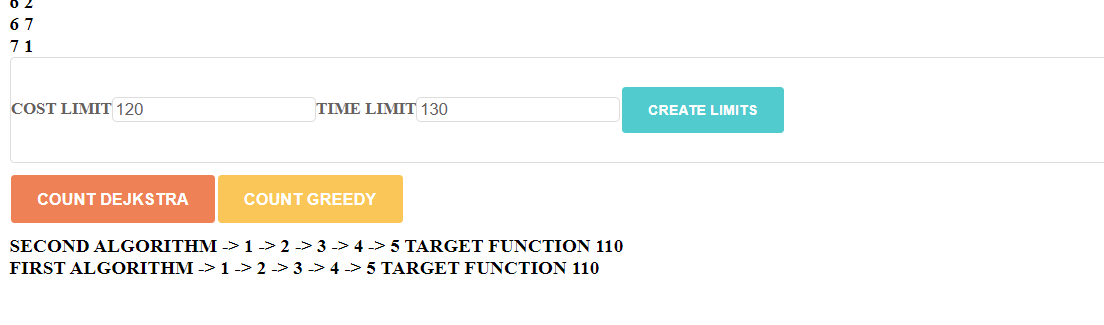


Рисунок 4.3 Результат роботи програми

Як бачимо результати роботи першого і другого алгоритму співпадають для цієї індивідуальної задачі.

# **4.2 Експерименти та їх результати**

# *4.2.1 Опис обладнання яке використовувалось під час експерементів*

Операційна система: Windows 10 x64

Процесор: 2.28 GHz

ОЗП: 16 GB

# **4.3 Дослідження першого алгоритму**

# *4.3.1 Дослідження часу виконання алгоритму*

На Рисунку 4.4 зображена залежність часу роботи першого алгоритму від розмірності задачі. Тестування проводилось на розмірностях задачі від 5 до 30 з кроком 1.

Як ми бачимо з графіку час розрахунку швидко зростає. Це зумовлено тим, що при розрахунку максимального шляху, алгоритм для кожної вершини обчислює цінність шляху, який веде в цю вершину. Також після розрахунків цінностей для кожної вершини, відбувається відновлення шляхів, це додає часу роботі алгоритму. Через те що дані генерувались рандомним чином, графік має скачки.

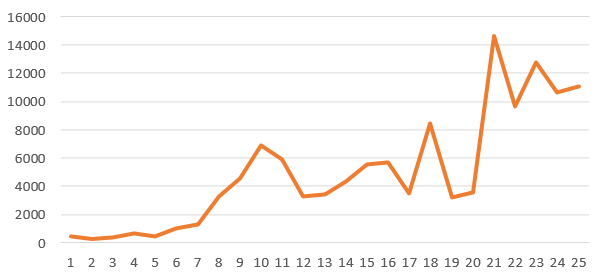


Рисунок 4.4 Графік залежності часу від розмірності задачі для першого

алгоритму.

# **4.4 Дослідження другого алгоритму**

# *4.4.1 Дослідження часу виконання алгоритму*

На Рисунку 4.5 зображена залежність часу роботи другого алгоритму від розмірності задачі. Тестування проводилось на розмірностях задачі від 10 до 700. Дослідження на таких великих розмірностях задач (впорівнянні з дослідженням першого алгоритму) зумовлене тим, що час алгоритму повільно зростає, для того щоб побачити залежність було вирішено провести випробування на розмірностях задач більших ніж в випадку першого алгоритму.

Час зростає повільніше ніж в випадку першого алгоритму через те, що алгоритм на кожному кроці обирає вершину з найвищім значенням естетичної цінності на поточному кроці. Алгоритм не розраховує значення для цілого шляху, як в випадку першого алгоритму, а лише порівнює вершини на поточному кроці за естетичною цінністю. Також алгоритм одразу будує шлях, і не має необхідності додатково проходити граф. Через те що дані генерувались рандомним чином, графік має скачки.

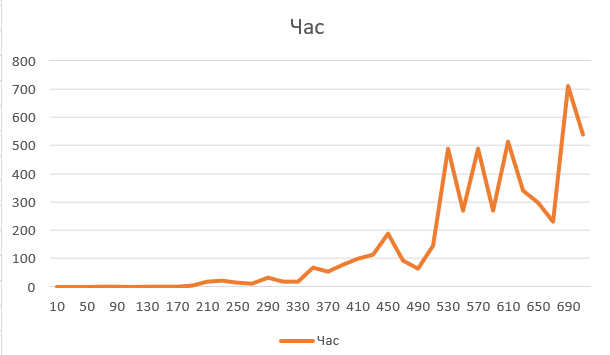
**

Рисунок 4.5 Графік залежності часу від розмірності задачі для другого

алгоритму.

# **4.5 Порівняння точності алгоритмів**

На Рисунку 4.6 зображена залежність різниці цільової функції першого та другого алгоритмів від розмірності задачі. Ми вважаємо перший алгоритм більш точним за другий , через те що перший обходить весь граф і для кожної вершини розраховує максимальне можливе значення естетичної цінності, отже алгоритм аналізує більш повну інформацію про граф, а не просто обирає кращу на поточному кроці вершину як другий. З графіку легко бачити ,що спочатку різниця між результатами роботи мінімальна, але при збільшені розмірностей задачі різниця збільшується.

**

Рисунок 4.6 Графік залежності різниці цільової функції двох алгоритмів від розмірності задачі.

# 

# **ВИСНОВКИ**

В ході даної лабораторної роботи було вирішено задачу про шлях максимальної цінності. Для цього було розроблено два алгоритми.

Оскільки ми застосовуємо два різні алгоритми для розв’язання нашої задачі, то знайдені шляхи можуть відрізнятись за складом вершин. Перший алгоритм дає більш точний результат розв’язання задачі, це викликано тим , що алгоритм має більш повну інформацію про граф, а другий алгоритм лише аналізує вершини сусідні до поточної.

З проведеного вище дослідження можемо дійти висновку, що на малих розмірностях задач перший і другий алгоритм мають невелику розбіжність у значенні цільової функції, але час роботи першого алгоритму навіть на невеликих задачах суттєво більший за час роботи другого алгоритму. Це викликано тим, що алгоритм розраховує цінність поточного шляху для кожної вершини, а також не зберігає шлях, отже після розрахунків потрібно його відновити. Чим більше розмірність задачі тим більша розбіжність між значеннями цільових функцій. Час роботи другого алгоритму зростає повільно, тому навіть на великих розмірностях задач алгоритм працює досить швидко.

В ході курсової роботи розроблено програмний продукт для розрахунку шляху з максимальною естетичною цінністю. Програмне забезпечення створено за шаблоном MVC. Для візуального представлення було використано Angular 2.

В ході розробки зіткнулася з проблемами через рекурсивну залежність даних один від одного, цю проблему було вирішено розділенням моделі даних окремо для backend і frontend*.*

# **ПЕРЕЛІК ПОСИЛАНЬ**

1. Документація до фреймворку Angular 2. [Електронний ресурс] // Режим доступу: <https://angular.io/docs>
2. Документація до фреймворку Spring Boot. [Електронний ресурс] // Режим доступу: <https://docs.spring.io/spring-boot/docs/current/reference/html/>
3. Алгоритм Дейкстри. [Електронний ресурс] // Режим доступу:

<http://e-maxx.ru/algo/dijkstra#2>

1. Пошук в ширину. [Електронний ресурс] // Режим доступу:

<http://opendatastructures.org/versions/edition-0.1e/ods-java/12_3_Graph_Traversal.html#SECTION001531000000000000000>

1. Оцінка складності алгоритмів. [Електронний ресурс] // Режим доступу:

<https://introprogramming.info/english-intro-csharp-book/read-online/chapter-19-data-structures-and-algorithm-complexity/>

1. Оцінка складності алгоритмів. [Електронний ресурс] // Режим доступу:

[https://www.khanacademy.org/computing/computer-science/algorithms/breadth-first- search/a/the-breadth-first-search-algorithm](https://www.khanacademy.org/computing/computer-science/algorithms/breadth-first-%20%20%20%20search/a/the-breadth-first-search-algorithm)

1. Longest path between any pair of vertices [Електронний ресурс] // Режим доступу:

<https://www.geeksforgeeks.org/longest-path-between-any-pair-of-vertices/>

1. Алгоритми на C++. [Текст] / Роберт Седжвік, Кевін Уейн, 2017 – 633 с.
2. Алгоритмы построение и анализ [Текст] /Т. Кормен , 2017 – 489 с.
3. Шляхи максимальної довжини в графі. [Електронний ресурс] // Режим доступу:

<https://studfile.net/preview/4430358/page:4/>

# **Додаток А**

«Вихідні коди ПЗ»

Код був завантажений на хостинг GitHub. Отримати доступ до репозиторію можна за наступним посиланням: <https://github.com/jalpv1/RoutesInGraphCourseWork>