Задача пошуку шляхів з використанням спектральної теорії графів

Виконала
Студентка 4-го курсу
групи КА-53
Шульженко Я.О.

Керівник:
к. ф.-м. н.
Стусь О.В.

Актуальність дослідження

- Графи є важливою складовою майже у всіх областях математики. За допомогою графів можна зручно представити взаємозв'язок між елементами різноманітних складних систем.

- Найбільш наочним прикладом використання графів є комунікаційні і транспортні системи, а також маршрутизація даних в Інтернеті. Наприклад, схема метрополітену, вершинами графа є станції метро, а ребрами – лінії між ними. Таку ж структуру мають схеми авіаліній, карти доріг або руху морських суден. Застосування математичного апарату до таких графів дозволяє знайти найкоротший шлях, оптимізувати маршрут транспортного руху.

Об’єкт дослідження:

Шляхи у графі

Предмет дослідження:

Спектральні методи розв’язання задачі пошуку шляхів у графі

Мета дослідження:

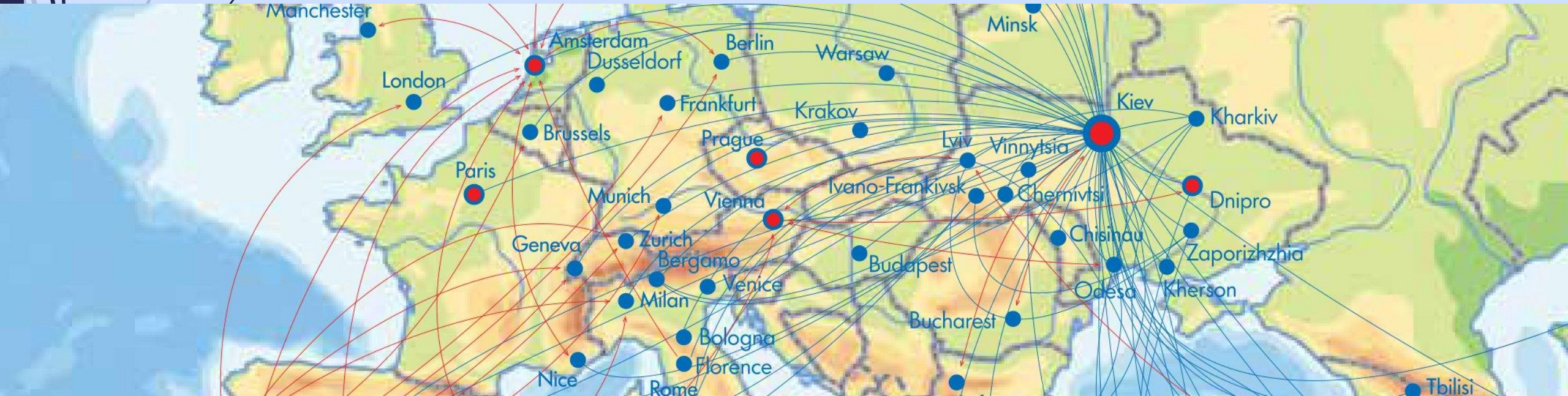
дослідити різні методи розв’язання, розробити програмний продукт, застосувати створений програмний продукт для конкретної задачі

Постановка задачі

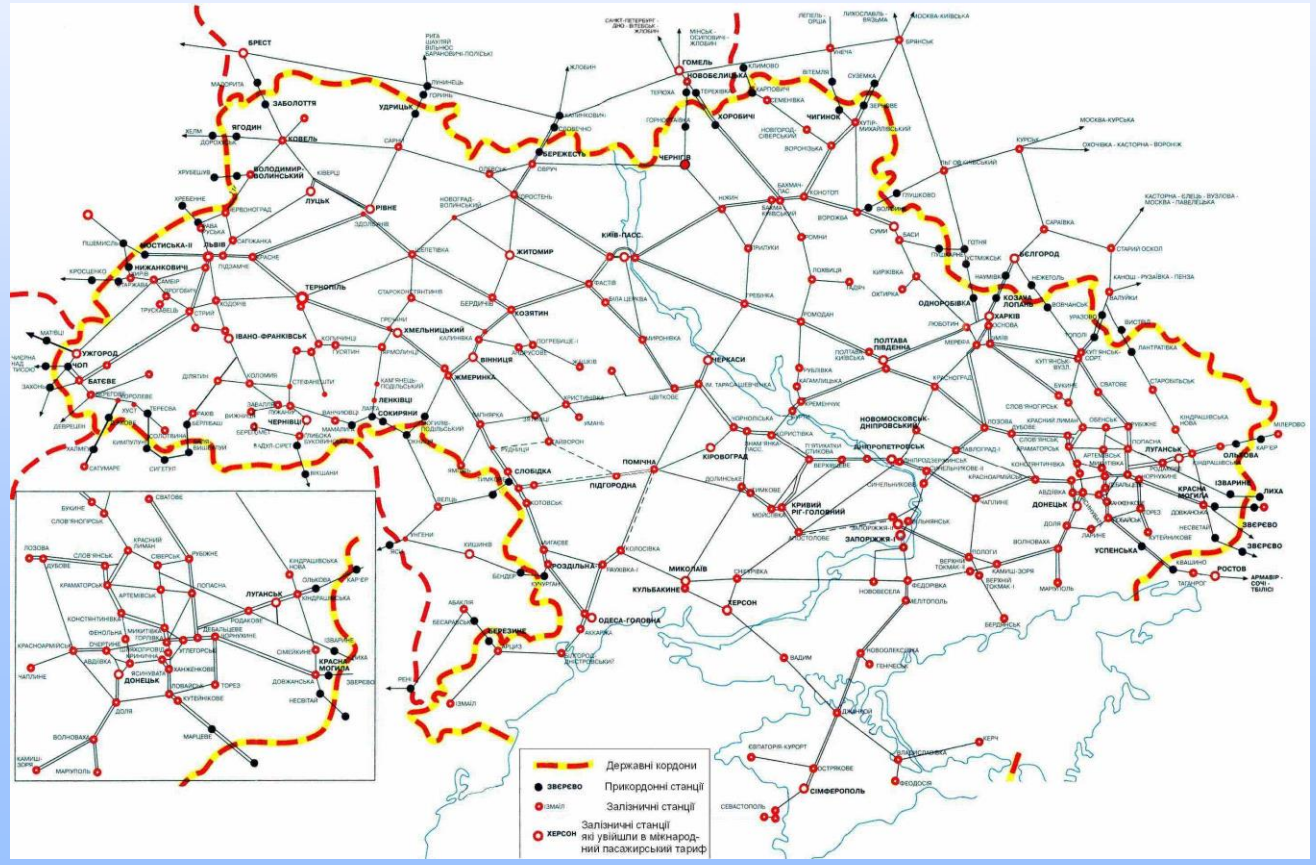
- Проаналізувати різні методи пошуку шляхів у графі.
- Дослідити спектральні характеристики графа для розв'язання задач, пов'язаних з пошуком шляхів.
- Реалізувати алгоритм пошуку всіх шляхів довжини k у графі з використанням спектральної теорії та розробити програмний продукт

Задача пошуку мінімальних відстаней

Маршрутом називається будь -яка послідовність прямуючих один за одним ребер мультиграфа або мультиорграфа у напрямку їх орієнтації. Довжина маршруту – це кількість його ребер.



Шляхом у неорієнтовному графі, що починається у вершині v_1 і закінчується у вершині v_2, називають послідовність вершин та ребер вигляду $v_1 e_{i1} v_{i1} e_{i2} v_{i2} e_{i3} \dots v_{in-1} e_{in} v_2$, де кожне ребро інцидентне обом вершинам, які є для нього сусідніми в послідовності.



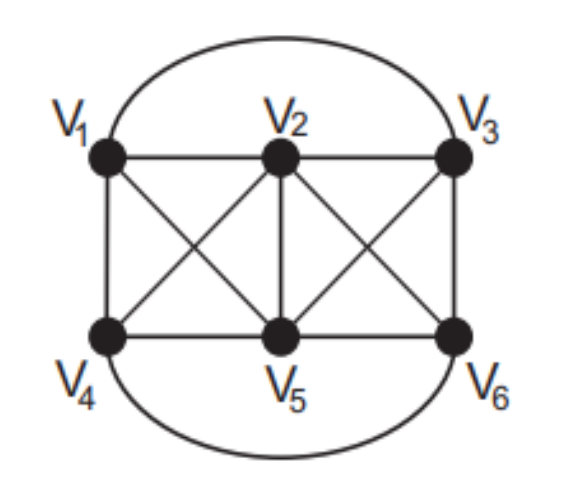
Алгоритми знаходження маршрутів у графі

- **Алгоритм Беллмана-Форда**. Знаходить найкоротші шляхи з однієї вершини зваженого графа до всіх інших, при цьому вага ребер може бути від'ємною, але немає бути циклів з від'ємною сумою ваг ребер.
- **Алгоритм Флойда-Уоршелла.** Знаходить найкоротші шляхи між усіма вершинами зваженого орієнтовного графа. Ваги можуть бути від'ємними, але немає бути циклів з від'ємною сумою ваг ребер.
- **Хвильовий алгоритм (Алгоритм Лі).** Знаходить шлях між двома заданими вершинами графа, який містить найменшу кількість проміжних вершин.

- **Алгоритм Джонсона.** Знаходить найкоротші шляхи між усіма парами вершин зваженого орієнтовного графа, при цьому повинні бути відсутніми цикли з від'ємною вагою.
- **Алгоритм пошуку A* (A з зірочкою).** Дозволяє знайти шлях із найменшою вартістю між двома вершинами за допомогою алгоритму пошуку по першому найкращому збігу на графі.
- **Алгоритм Дейкстри.** Знаходить найкоротший шлях у графі від заданої вершини до всіх інших, при цьому ваги ребер не можуть бути від'ємними.

Спектри графів і основні визначення

Матриця суміжності A мультиграфа (мультиорграфа) G з множиною вершин $\{v_1, v_2 \ldots, v_n\}$ – це квадратна матриця порядку n, у якій значення елементу a_{ij}, розташованого на місці *(i, j)*, дорівнює кількості ребр, з початком у вершині v_i і кінцем у вершині v_j.



$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Характеристичний многочлен і спектр графа

Характеристичний многочлен $|\lambda I\text{-}A|$ матриці суміжності A графа G називається характеристичним многочленом графа G і позначається $P_G(\lambda)$. Власні значення матриці A (нулі многочлена $|\lambda I\text{-}A|$) і спектр матриці A (складається з власних значень) називаються відповідно власними значеннями та спектром графа G. Якщо $\lambda_1, \ldots, \lambda_n$ – власні значення графа G, то весь спектр позначається через $S_P(\mathrm{G}) = [\lambda_1, \ldots, \lambda_n]$.

$$P_G(\lambda) = \begin{vmatrix} \lambda & -1 & -1 & -1 & -1 & 0 \\ -1 & \lambda & -1 & -1 & -1 & -1 \\ -1 & -1 & \lambda & 0 & -1 & -1 \\ -1 & -1 & 0 & \lambda & -1 & -1 \\ -1 & -1 & -1 & -1 & \lambda & -1 \\ 0 & -1 & -1 & -1 & -1 & \lambda \end{vmatrix} = \lambda^6 - 13\lambda^4 - 24\lambda^3 - 12\lambda^2$$

$$S_P(\mathrm{G}) = \left[-2, -1, 0, \frac{3}{2} - \frac{\sqrt{33}}{2}, \frac{3}{2} + \frac{\sqrt{33}}{2}\right]$$

Про кількість маршрутів

Якщо $x_1, x_2, x_3 \ldots, x_n$ – повна система взаємно ортогональних нормованих власних векторів матриці A, що відповідають спектру $[\lambda_1, \ldots, \lambda_n]$, $\mathrm{X} = (x_1, x_2, x_3 \ldots, x_n) = (x_{ij})$ і $\Lambda = (\delta_{ij}\lambda_i)$. X – ортогональна матриця ($\mathrm{X}^{-1} = \mathrm{X}^{\mathrm{T}}$) і $\mathrm{A} = \mathrm{X}\Lambda\mathrm{X}^{\mathrm{T}}$.

Нехай A – матриця суміжності мультиорграфа G з вершинами *1*, *2*, …, n і $A^k = \left(a_{ij}^k\right)$; нехай $N_k(i,j)$ позначає число маршрутів довжини k, які починаються у вершині i та закінчуються у вершині j. Тоді

$$N_k(i,j) = a_{ij}^{(k)} \ (\mathrm{k} = 0, 1, 2, \ldots)$$

Нехай тепер G позначає мультиграф і $X = (x_1, x_2, x_3 \dots, x_n) = (x_{ij})$ – ортогональна матриця власних векторів матриці A. Тоді

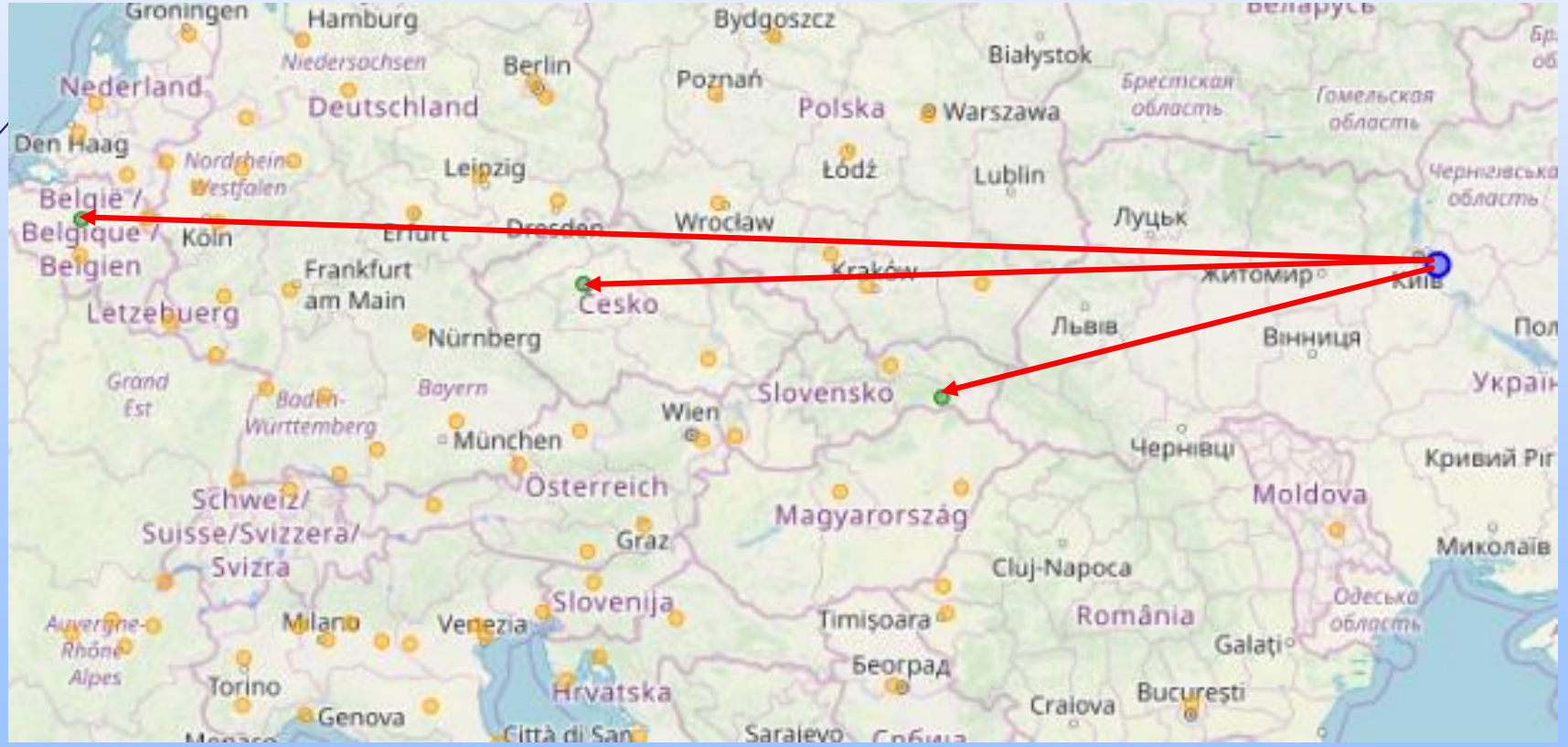
$$a_{ij}^{(k)} = \sum_{\nu=1}^{n} x_{i\nu} x_{j\nu} \lambda_\nu^k$$

Число всіх маршрутів довжини k в графі G

$$N_k = \sum_{i,j} N_k(i,j) = \sum_{i,j} a_{ij}^{(k)} = \sum_{\nu=1}^{n} (\sum_{i=1}^{n} x_{i\nu})^2 \lambda_\nu^k$$

Практичне застосування

Мета роботи полягає у тому, щоб визначити яка мінімальна кількість пересадок знадобиться, щоб потрапити з одного міста в інше, якщо прямого рейсу не існує.



Вхідні дані

Нехай дано 11 міст: Київ (0), Париж (1), Прага (2), Брюссель (3), Мюнхен (4), Москва (5), Рим (6), Лондон (7), Осло (8), Варшава (9), Дублін (10) – вершини графу.

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

```
Given matrix has such eigenvalues and eigenvectors:

Eigenvalue:
5.134840020278077
Eigenvector:
[0.14679567 0.40844684 0.3676721  0.38610018 0.32059341 0.20923061
 0.38052523 0.35683889 0.14929879 0.28557937 0.06949367]

Eigenvalue:
1.833631372234814
Eigenvector:
[-0.38493589  0.17551953 -0.2692787  -0.43655181  0.14497917 -0.30586919
  0.04994963  0.40812985 -0.21083964  0.42460998  0.2225801 ]

Eigenvalue:
-2.337196104822094
Eigenvector:
[ 0.37634771  0.35130687 -0.38817737 -0.49142102  0.04926678  0.35526827
  0.33955881 -0.02381399  0.06497624 -0.30648625  0.01018913]

Eigenvalue:
-2.4094768689719506
Eigenvector:
[ 0.03312702  0.11249062 -0.42170693  0.34188813 -0.52475687  0.25091574
 -0.22556294  0.50323451 -0.0482782   0.05586054 -0.20885633]

Eigenvalue:
0.6831030051803955
Eigenvector:
[ 0.38461303 -0.05957672  0.43360654 -0.17087623 -0.33373708 -0.10394738
 -0.07611379  0.32209901 -0.36157067 -0.21567549  0.47152334]
```

```
Eigenvalue:
0.2241346574472292
Eigenvector:
[ 0.27028898  0.28620333  0.19282848 -0.13224735 -0.12080151 -0.26867949
  0.03672077 -0.14906669 -0.42620177  0.23671439 -0.66507647]

Eigenvalue:
-0.26646598479491096
Eigenvector:
[-0.62788677  0.02739595  0.387025   -0.21971453 -0.27749425  0.41350039
  0.25012272  0.05345324 -0.11411659 -0.20069227 -0.20060062]

Eigenvalue:
-0.8160015997197447
Eigenvector:
[ 0.26271113 -0.64552958  0.04268591 -0.25705861 -0.0542613   0.32920769
  0.17772936  0.09810823  0.09721702  0.51954957 -0.12023044]

Eigenvalue:
-1.7563410519027631
Eigenvector:
[-0.03173487 -0.37040581 -0.2519035   0.30764075  0.20573344 -0.14887239
  0.53878888  0.1470119  -0.48192783 -0.29671253 -0.0837035 ]

Eigenvalue:
-1.2902274449290563
Eigenvector:
[-0.04786499  0.1617784  -0.15337712  0.21513384 -0.31647571  0.19742177
  0.21199642 -0.54057408 -0.33105036  0.37456572  0.4189758 ]
```

```
Eigenvalue:
1.0000000000000002
Eigenvector:
[-2.59739719e-16  2.06043951e-16 -2.88327665e-16  2.85879459e-17
  5.00000000e-01  5.00000000e-01 -5.00000000e-01 -5.03001176e-17
 -5.00000000e-01 -4.93456508e-18 -2.64301447e-16]
```

Київ(0) – Париж(1)

```
Paths of which length should be found?
2

i:
0
j:
1

The numbers of paths from city (0) to city (1) lengths 2
2.0
```

Київ – Прага – Париж

Київ – Брюссель – Париж

Дублін(10) – Брюссель(3)

```
Paths of which length should be found?
3

i:
10
j:
3

The numbers of paths from city (10) to city (3) lengths 3
4.0
```

Дублін – Лондон – Париж – Брюссель

Дублін – Лондон – Прага – Брюссель

Дублін – Лондон – Мюнхен – Брюссель

Дублін – Лондон – Рим – Брюссель

Висновки

У дипломній роботі було реалізовано метод пошуку шляхів з використанням спектральної теорії графів. Розроблено програмний продукт, який маючи, як вхідні дані, матрицю суміжності графа, у результаті виводить спектр графа, власні вектори та кількість усіх шляхів довжиною k.

Було наведено приклад використання задачі для схем low-cost авіаліній, а саме, яка мінімальна кількість пересадок потрібна, щоб потрапити з одного міста в інше, якщо прямого рейсу не існує.

Подальші дослідження

- Дослідити більш складну схему, з більшою кількістю міст.
- Додати до вхідних даних ціни авіарейсів для того, щоб знайти не тільки найкоротший маршрут, а й найдешевший.
- Провести аналіз тривалості польоту та часу очікування у місті, де буде здійснюватися пересадка. Таким чином знайти маршрут з найменшою тривалістю.

Дякую за увагу!