

**КІЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА**
Факультет комп'ютерних наук та кібернетики
Кафедра обчислювальної математики

**Кваліфікаційна робота
на здобуття ступеня бакалавра**

за спеціальністю 113 Прикладна математика
на тему:

**АНАЛІЗ ВАРТОСТІ ПРОЖИВАННЯ У ВЕЛИКИХ МІСТАХ
СВІТУ ІЗ ВИКОРИСТАННЯМ МЕТОДІВ КЛАСТЕРИЗАЦІЇ**

Виконав студент 4-го курсу
Пишко Андрій Олександрович

(підпис)

Науковий керівник:
асистент, кандидат фіз.-мат. наук
Затула Дмитро Васильович

(підпис)

Засвідчую, що в цій роботі немає запозичень
з праць інших авторів без відповідних
посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на
засіданні кафедри обчислювальної
математики

«____» _____ 2022 р.,

протокол № ____

Завідувач кафедри
проф. Ляшко С.І.

(підпис)

Київ – 2022

ЗМІСТ

ВСТУП	4
РОЗДІЛ I. ЗАГАЛЬНИЙ ОПИС КЛАСТЕРИЗАЦІЇ	6
1.1. Поняття кластеризації	6
1.2. Етапи кластеризації	6
1.3. Визначення «схожих» об'єктів	6
1.4. Типи алгоритмів кластеризації	8
РОЗДІЛ II. МЕТОД КЛАСТЕРИЗАЦІЇ K-MEANS	9
2.1. Загальний опис методу	9
2.2. Математичний опис методу	9
2.3. Структура алгоритму	10
2.4. Робота методу	12
2.4.1. Місячна оренда 85 м ² мебльованого житла в дорогому районі	12
2.4.2. Місячний квиток на міський транспорт	17
2.4.3. Індекс вартості проживання	21
2.4.4. Аналіз отриманих результатів	26
2.4.5. Стороння реалізація алгоритму	28
РОЗДІЛ III. МЕТОД КЛАСТЕРИЗАЦІЇ BIRCH	31
3.1. Загальний опис методу	31
3.2. Математичний опис методу	31
3.3. Структура алгоритму	36
3.4. Робота методу	38
3.4.1. Місячна оренда 85 м ² мебльованого житла в дорогому районі	39
3.4.2. Місячний квиток на міський транспорт	41
3.4.3. Індекс вартості проживання	43
РОЗДІЛ IV. МЕТОД СПЕКТРАЛЬНОЇ КЛАСТЕРИЗАЦІЇ	45
4.1. Загальний опис методу	45
4.2. Математичний опис методу	45
4.2.1. Граф подібності	45
4.2.2. Введення в термінологію	46
4.2.3. Як побудувати граф подібності?	46
4.2.4. Матриця Кірхгофа	47
4.2.5. Головна ідея алгоритму	51
4.3. Структура алгоритму	55
4.3.1 Підготовчі питання	55
4.3.2. Алгоритм	59
4.4. Робота методу	60
4.4.1. Місячна оренда 85 м ² мебльованого житла в дорогому районі	61
4.4.2. Місячний квиток на міський транспорт	64
4.4.3. Індекс вартості проживання	67
ВИСНОВКИ	70

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**71****ДОДАТКИ****73**

Додаток А. Початкові дані	73
Додаток Б. Отримані дані	74
Метод K-means (5 кластерів)	7
Метод BIRCH (5 кластерів)	82
Спектральна кластеризація (5 кластерів)	90
Додаток В. Код реалізації алгоритмів	98

ВСТУП

Відшукання корисних для аналізу закономірностей у великих обсягах даних з недавніх часів викликає значний інтерес. У зв'язку з цим було введено і стало активно розвиватися таке поняття як кластеризація.

Застосування кластеризації. Кластерний аналіз зарекомендував себе у багатьох відомих галузях людської діяльності, таких як археологія, геологія, медицина, психологія, хімія, біологія, державне регулювання, філологія, антропологія, маркетинг, соціологія та багато інших.

У біології кластеризація має безліч застосувань у різних областях. Наприклад, в біоінформатиці за допомогою неї аналізуються складні мережі взаємодіючих генів, що складаються з сотень або навіть тисяч елементів. Кластерний аналіз дозволяє виділити підмережі, вузькі місця, концентратори та інші приховані властивості досліджуваної системи, що дозволяє в кінцевому рахунку оцінити внесок кожного із генів у формуванні досліджуваного феномена.

При аналізі результатів соціологічних досліджень рекомендується здійснювати аналіз методом Уорда, при якому всередині кластерів оптимізується мінімальна дисперсія, і в результаті створюються кластери приблизно рівних розмірів.

Кластеризація результатів пошуку використовується для «інтелектуального» групування результатів при пошуку файлів, веб-сайтів, інших об'єктів, надаючи користувачу можливість швидкої навігації, вибору свідомо більш релевантної підмножини і відсіювання менш релевантної, що може підвищити зручність інтерфейсу в порівнянні з відображенням у вигляді простого сортованого за релевантністю списку.

Також дуже цікавим використанням кластерного аналізу є розбиття цифрового зображення на окремі області з метою розпізнавання об'єктів.

Метою даної роботи є застосування методів кластеризації для розбиття множини міст на групи за даними вартості проживання в них.

Об'єктом дослідження є застосування кластерного аналізу як складової інтелектуального аналізу даних.

Предметом дослідження є кластерний аналіз даних вартості проживання у різних містах.

В якості бази було сформовано вибірку даних по 42 містам Європи, а саме: Рейк'явік (Ісландія), Дублін (Ірландія), Лондон (Великобританія), Париж (Франція), Мадрид (Іспанія), Лісабон (Португалія), Белфаст (Північна Ірландія), Манчестер (Великобританія), Оксфорд (Великобританія), Сарагоса (Іспанія), Барселона (Іспанія), Ніцца (Франція), Ліон (Франція), Женева (Швейцарія), Цюрих (Швейцарія), Мюніх (Швейцарія), Гамбург (Німеччина), Берлін (Німеччина), Прага (Чехія), Віденсь (Австрія), Будапешт (Угорщина), Загреб (Хорватія), Любляна (Словенія), Лугано (Швейцарія), Мілан (Італія), Венеція (Італія), Рим (Італія), Белград (Сербія), Софія (Болгарія), Афіни (Греція), Бухарест (Румунія), Київ (Україна), Краків (Польща), Варшава (Польща), Вільнюс (Литва), Рига (Латвія), Москва (Росія), Таллінн (Естонія), Хельсінкі (Фінляндія), Стокгольм (Швеція), Копенгаген (Данія), Осло (Норвегія). Використані дані було взято із відкритого доступу на сайті expatistan.com [1]. Для аналізу було обрано дані про середню вартість молока (1 літр), яєць (12 штук, великі), місячної оренди житла (площею 85 м² у дорогому районі), місячного квитку для проїзду у міському транспорті, короткого візиту до лікаря (15 хвилин), двох квитків у кіно, місячного абонементу до спортивної залі (у діловому районі) та індекс вартості проживання (визначений ресурсами сайту expatistan.com [1]). Всі ціни переведено за поточним на 10.02.2022 р. курсом у гривні.

РОЗДІЛ I. ЗАГАЛЬНИЙ ОПИС КЛАСТЕРИЗАЦІЇ

1.1. Поняття кластеризації

Кластеризація – це процедура, що реалізує поділ деякої множини об'єктів на окремі групи (кластери), що не перетинаються між собою. Об'єкти, що входять до одного кластеру, мають спільні властивості, які відрізняють їх від об'єктів інших кластерів.

Саме поняття кластера є слабоформалізованим, оскільки спирається на поняття «спільні властивості», що потребує додаткової деталізації. Ця деталізація здебільшого визначається змістом прикладної задачі і реалізується особливостями самого алгоритму. Саме тому існує декілька десятків різних алгоритмів кластеризації та їх модифікацій.

1.2. Етапи кластеризації

У загальному вигляді кластеризація виглядає наступним чином:

1. Визначається множина змінних, за якими будуть оцінюватися об'єкти у вибірці.
2. Необов'язковий етап: нормалізація значень змінних.
3. Обирається вибірка об'єктів для кластеризації.
4. Обчислення значень міри схожості між об'єктами.
5. Застосування методу кластерного аналізу для утворення груп схожих об'єктів (кластерів).
6. Візуалізація результатів аналізу.

1.3. Визначення «схожих» об'єктів

Для початку потрібно скласти вектор характеристик для кожного об'єкту. Це, зазвичай, може бути множина числових значень, рідше – якісні характеристики. Наприклад, знаючи довготу та широту, можна визначити кластери для відомих географічних об'єктів.

Далі для кожної пари об'єктів вимірюється «відстань» між ними (ступінь схожості). Існує безліч метрик, серед яких виділяють наступні:

1. **Евклідова відстань** – найбільш поширена функція відстані.

Визначається як геометрична відстань між двома точками у просторі $\mathbb{R}^n, n > 1$:

$$\rho(x, x') = \sqrt{\sum_i^n (x_i - x'_i)^2}$$

2. **Квадрат евклідової відстані**, який застосовується для надання більшої «ваги» значно віддаленим один від одного об'єктам. Ця відстань обчислюється наступним чином:

$$\rho(x, x') = \sum_i^n (x_i - x'_i)^2$$

3. **Манхеттенська відстань**. Відповідно до цієї метрики, відстань між двома точками дорівнює сумі модулів різниць їх координат. Для більшості випадків ця міра відстані призводить до таких же результатів, як і звичайна відстань Евкліда. Однак для цієї міри вплив окремих, більш великих, різниць координат зменшується, тому що вони не зводяться в квадрат. Формула для розрахунку манхеттенської відстані:

$$\rho(x, x') = \sum_i^n |x_i - x'_i|$$

4. **Степенева відстань**. Інколи бажають прогресивно збільшувати або зменшувати вагу, що стосується розмірності, для якої відповідні об'єкти значно відрізняються. Це може бути досягнуто з використанням степеневої відстані. Степенева відстань обчислюється за формулою:

$$\rho(x, x') = \sqrt[r]{\sum_i^n (x_i - x'_i)^p}$$

Параметр p відповідає за поступове «зважування» різниць по кожній з координат, параметр r – за прогресивне «зважування» великих відстаней між об'єктами. Якщо $p = r = 2$, то ця відстань збігається з відстанню Евкліда.

Вибір метрики повністю залежить від дослідника, оскільки результати кластеризації можуть істотно відрізнятися при використанні різних мір.

1.4. Типи алгоритмів кластеризації

Ієрархічні та неієрархічні

Ієрархічні алгоритми будують не одне розбиття вибірки на непересічні кластери, а систему вкладених розбиттів. Таким чином, на виході отримується дерево кластерів, коренем якого є вся вибірка, а листям – найбільш дрібні групи.

На відміну від ієрархічних, *неієрархічні* алгоритми будують одне розбиття об'єктів на кластери.

Чіткі та нечіткі

Чіткі (непересічні) алгоритми кожному об'єкту вибірки ставлять у відповідність номер кластера, тобто кожен об'єкт належить тільки одній групі.

Нечіткі (або пересічні) алгоритми кожному об'єкту ставлять у відповідність набір дійсних значень, що показують ступінь відношення об'єкта до відповідних кластерів. Тобто кожен об'єкт відноситься до кожної групи з певною ймовірністю.

РОЗДІЛ II. МЕТОД КЛАСТЕРИЗАЦІЇ K-MEANS

2.1. Загальний опис методу

K-means – це найпростіший, але в той же час досить неточний метод кластеризації в класичній реалізації. Він розбиває безліч елементів векторного простору на заздалегідь відому кількість кластерів. Алгоритм прагне мінімізувати середньоквадратичне відхилення для точок кожного кластера.

Основна ідея методу полягає в тому, що на кожній ітерації переобчислюють центр мас (так званий *центроїд*) для кожного кластера, отриманого на попередньому кроці. Потім вектори розбиваються на кластери знову відповідно до того, який з нових центрів виявився близчим за обраною метрикою. Алгоритм завершується, коли на якийсь ітерації не відбувається зміни кластерів.

Серед недоліків класичного варіанту методу виділяють необхідність завчасного задання кількості кластерів та велику чутливість до вибору початкових центроїдів. Останній факт особливо важливий, адже, у класичному варіанті розглядають випадковий вибір кластерів, що дуже часто є джерелом похибки. Для вирішення цієї проблеми необхідно проводити дослідження об'єкта для більш точного визначення центрів початкових кластерів.

2.2. Математичний опис методу

Нехай $X = \{x_i, i = 1, \dots, N\}$, $x_i \in R_n$ – множина об'єктів, що підлягає кластеризації, та $Y = \{y_j, j = 1, \dots, K\}$ – мітки майбутніх кластерів. Як правило, мітками виступають натуральні числа. Тут K – кількість кластерів. Задача кластеризації полягає в побудові відображення $T: X \rightarrow Y$, за яким кожному елементу $x_i \in X$ ставиться у відповідність мітка y_j за правилом:

$$T(x_i) = y_j$$

Метод базується на знаходженні для кожного кластера його центроїда $\omega_j \in R_n$, $j = 1, \dots, K$ та подальшого розподілу кожного з об'єктів $x_i \in X$ до кластерів за принципом мінімальної відстані до відповідного центроїда y_j :

$$T(x_i) = y_j = \arg \min_{s \in Y} \rho(x_i, \omega_s),$$

де $\rho(x_i, \omega_s)$ – відстань між двома векторами в просторі R_n . Процес обчислення положення центроїдів є ітераційним, і тому передбачає задання деяких початкових наближень положень центроїдів. Найпростіший варіант вибору початкових наближень центроїдів – вибрати K довільних елементів множини X . Але цей метод дуже часто призводить до повільної збіжності, або збіжності до неприйнятного результату.

2.3. Структура алгоритму

1. Вибір початкових центроїдів

- a. З множини X випадковим чином обираємо 5-10% векторів $X_0 \subset X$
- b. Формуємо центроїди двох перших кластерів ω_1, ω_2 , обираючи їх серед точок множини X_0 . Обираємо точки, відстань між якими є максимальною:

$$\omega_1, \omega_2 = \arg \max_{(x_i, x_j \in X_0)} \rho(x_i, x_j)$$

- c. Формуємо центроїди усіх інших кластерів, починаючи з третього. Нехай вже отримано центроїди $\omega_1, \omega_2, \dots, \omega_j$, $j < K$. Тоді центроїд ω_{j+1} знаходимо серед точок множини X_0 за правилом:

$$\omega_{j+1} = \arg \max_{(x_i \in X_0)} \min_{(1 \leq s \leq j)} \rho(x_i, \omega_s)$$

Тобто, для кожного елементу X_0 обчислюється відстань до

кожного з відомих центроїдів і серед них обирається мінімальне значення. Ці мінімальні значення утворюють послідовність. А той елемент множини X_0 , для якого послідовність набуває максимального значення, і обирається в якості наступного центроїда. Таким чином будується множину точок з максимальною середньою відстанню.

2. Розподіляємо точки множини X по кластерам за принципом мінімальної відстані до центроїда:

$$T(x_i) = y_j = \arg \min_{s \in Y} \rho(x_i, \omega_s)$$

Утворюємо кластери $X_j = \{x_i \in X, T(x_i) = y_j\}$.

3. Обчислюємо нові значення центроїдів для кожного кластера:

$$\hat{\omega}_j = \frac{1}{n_j} \sum_{x_i \in X_j} x_i, j = 1..K,$$

де n_j – кількість елементів кластера X_j .

4. Перевіряємо виконання критерію зупинки ітераційного процесу:

$$\frac{||\Omega - \hat{\Omega}||}{||\hat{\Omega}||} \leq \varepsilon,$$

де $\Omega, \hat{\Omega}$ – матриці, рядками яких є вектори центроїдів ω_j , $\hat{\omega}_j, j = 1, \dots, K$.

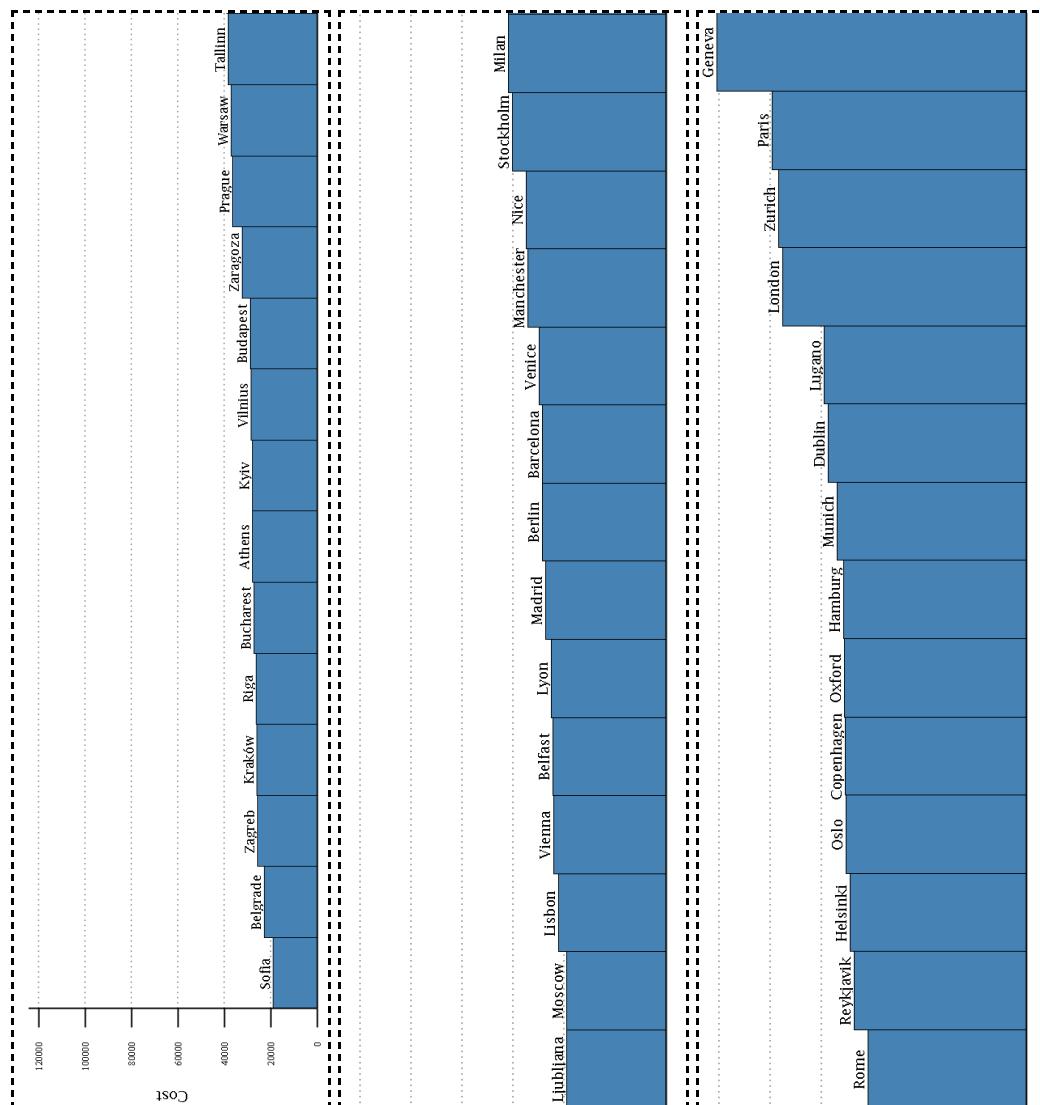
Якщо остання умова не виконується, кроки 2-4 повторюються до виконання умови. Додатковою умовою закінчення ітераційного процесу є обмеження кількості ітерацій.

2.4. Робота методу

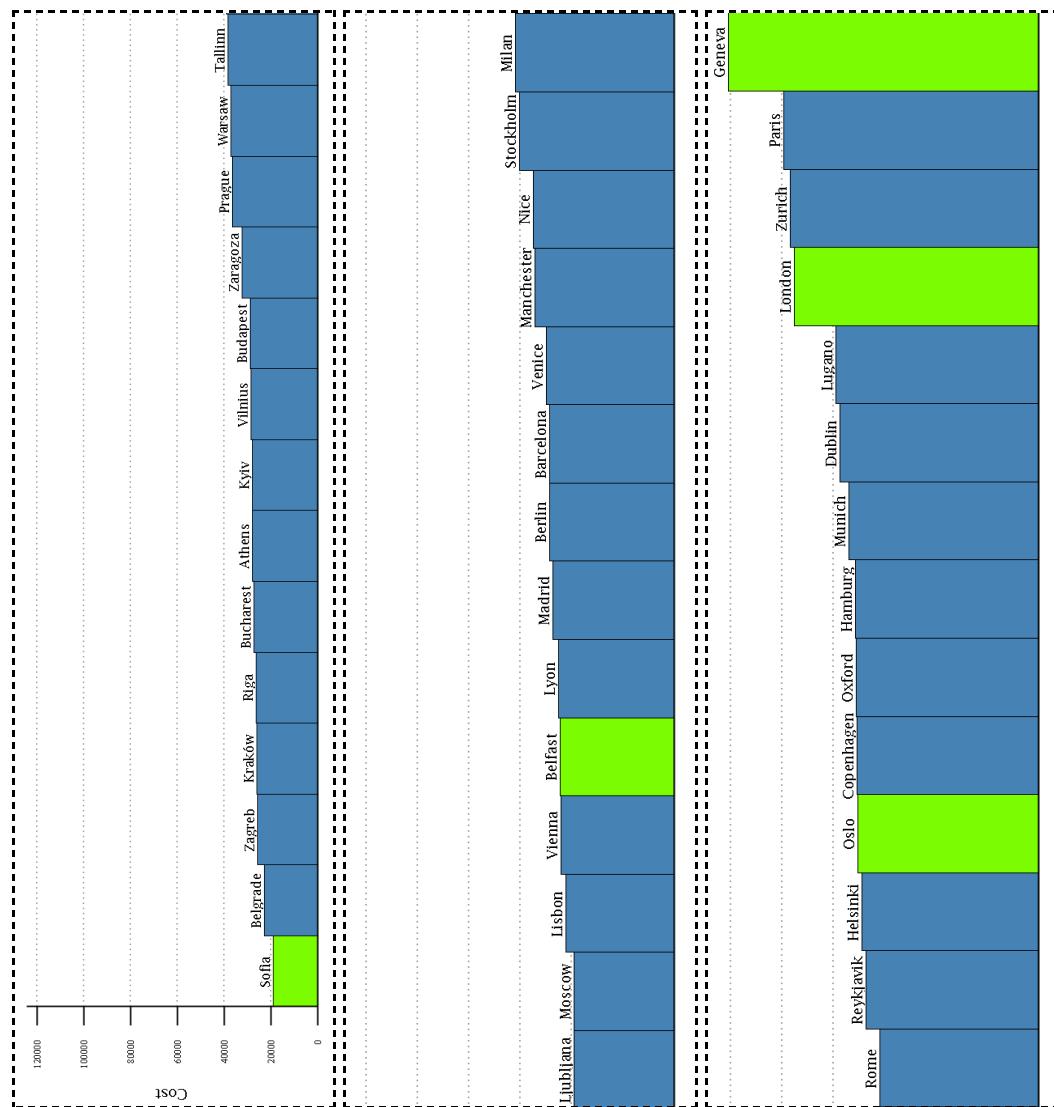
Робота методу K-means буде продемонстрована на розбитті множини міст за величиною місячної оренди 85 m^2 мебльованого житла в дорогому районі, місячною вартістю квитка на міський транспорт та за величиною індексу вартості проживання. Алгоритм був реалізований за допомогою програмного забезпечення Maple. Обрані дані для вибірки див. у Додатку А.

2.4.1. Місячна оренда 85 m^2 мебльованого житла в дорогому районі

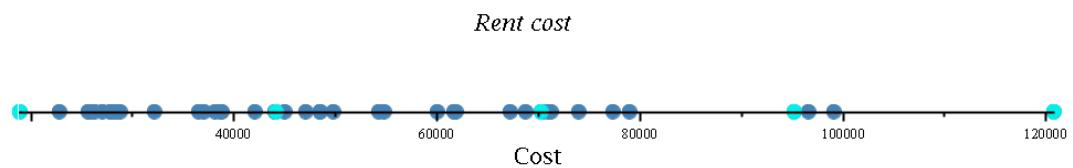
Початкові дані, відображені у вигляді гістограм



Перше наближення центроїдів

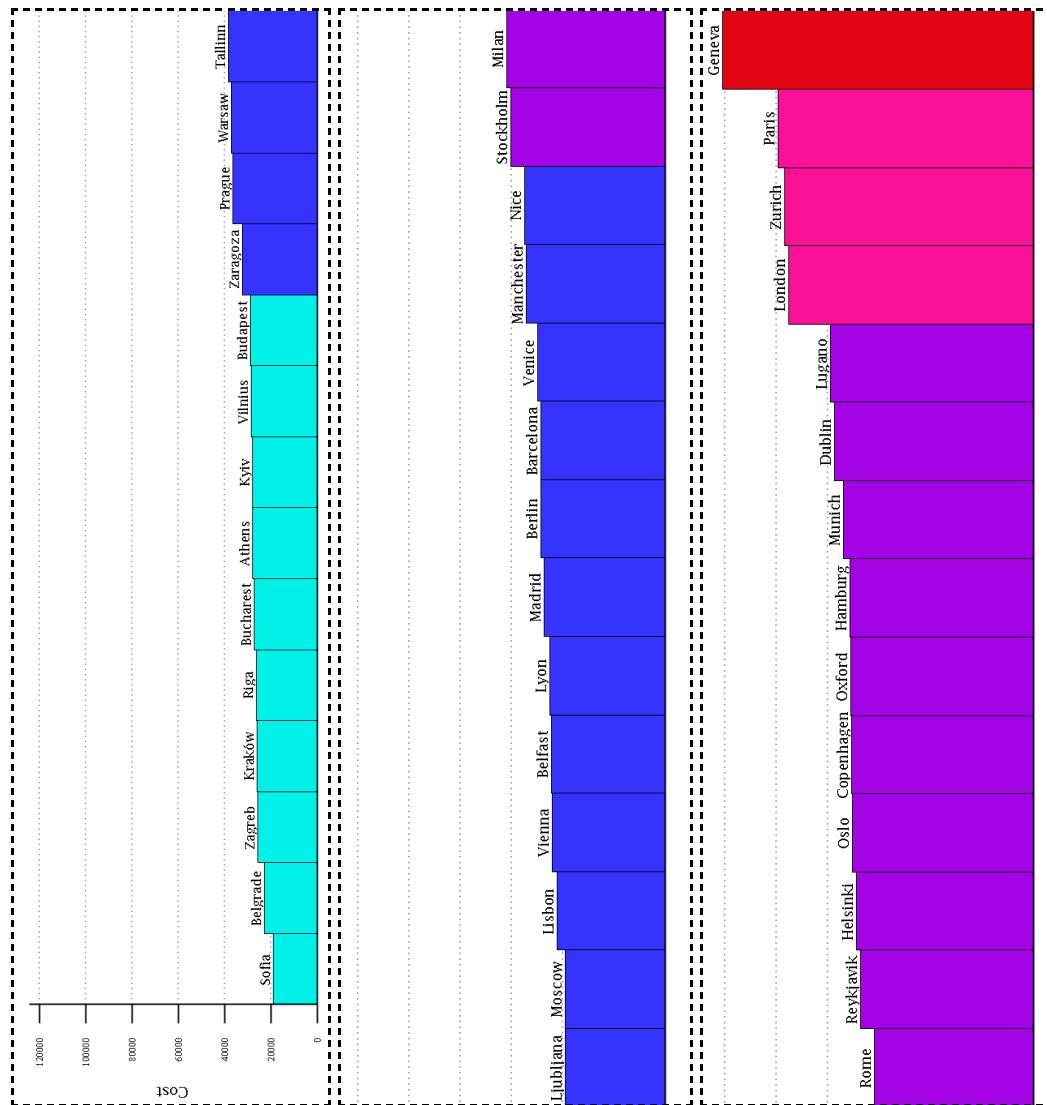


Зеленим кольором зображені міста, що були обрані центроїдами.

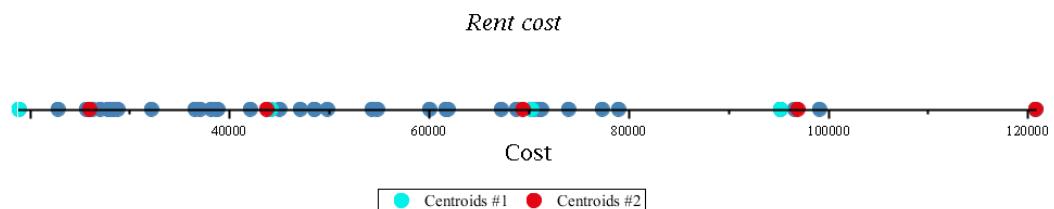


На даному рисунку синім кольором зображені усі точки множини, блакитним – центроїди.

Визначення кластерів (перше наближення)

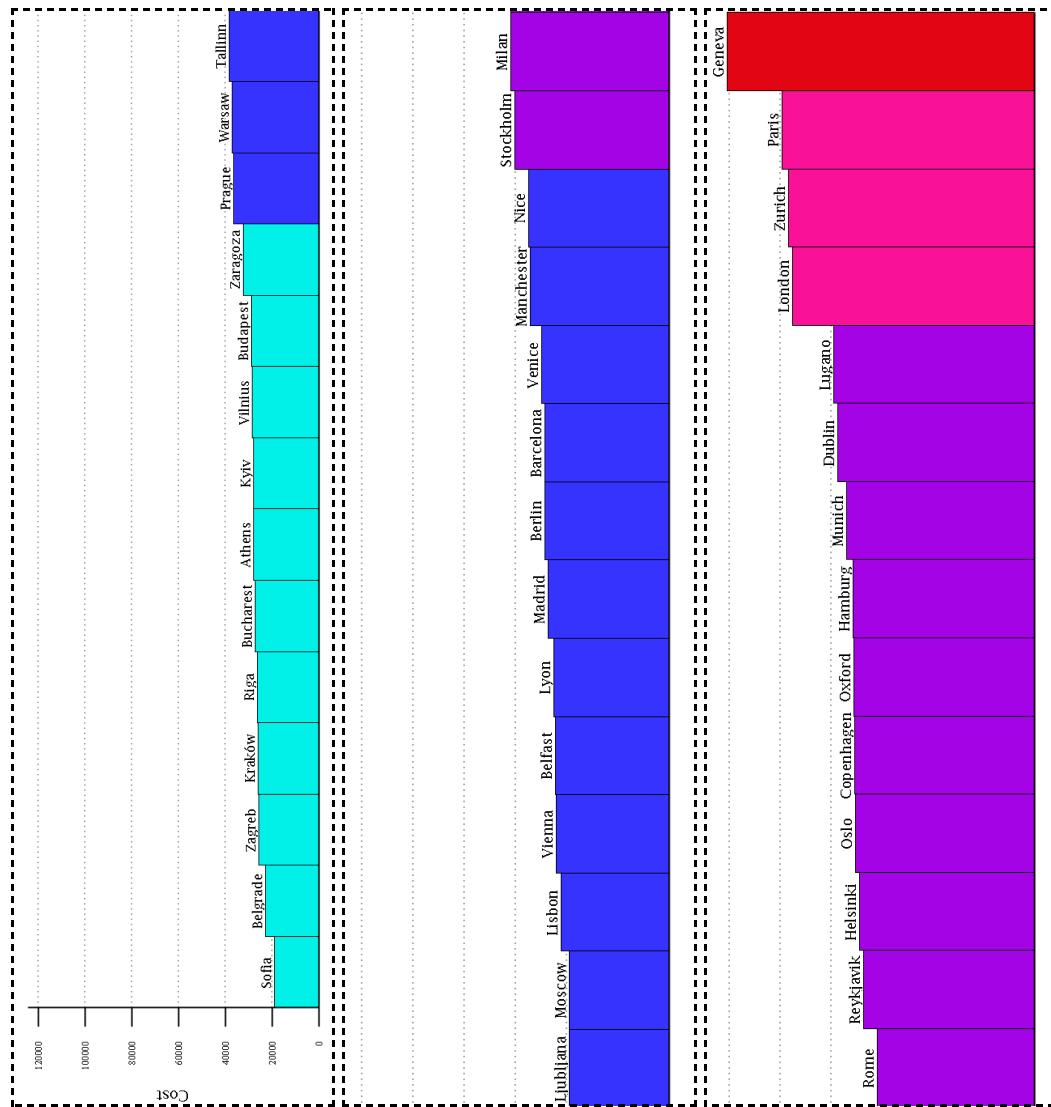


Визначення нових центроїдів



Блакитним позначені старі центроїди (перше наближення), червоним – нові (друге наближення).

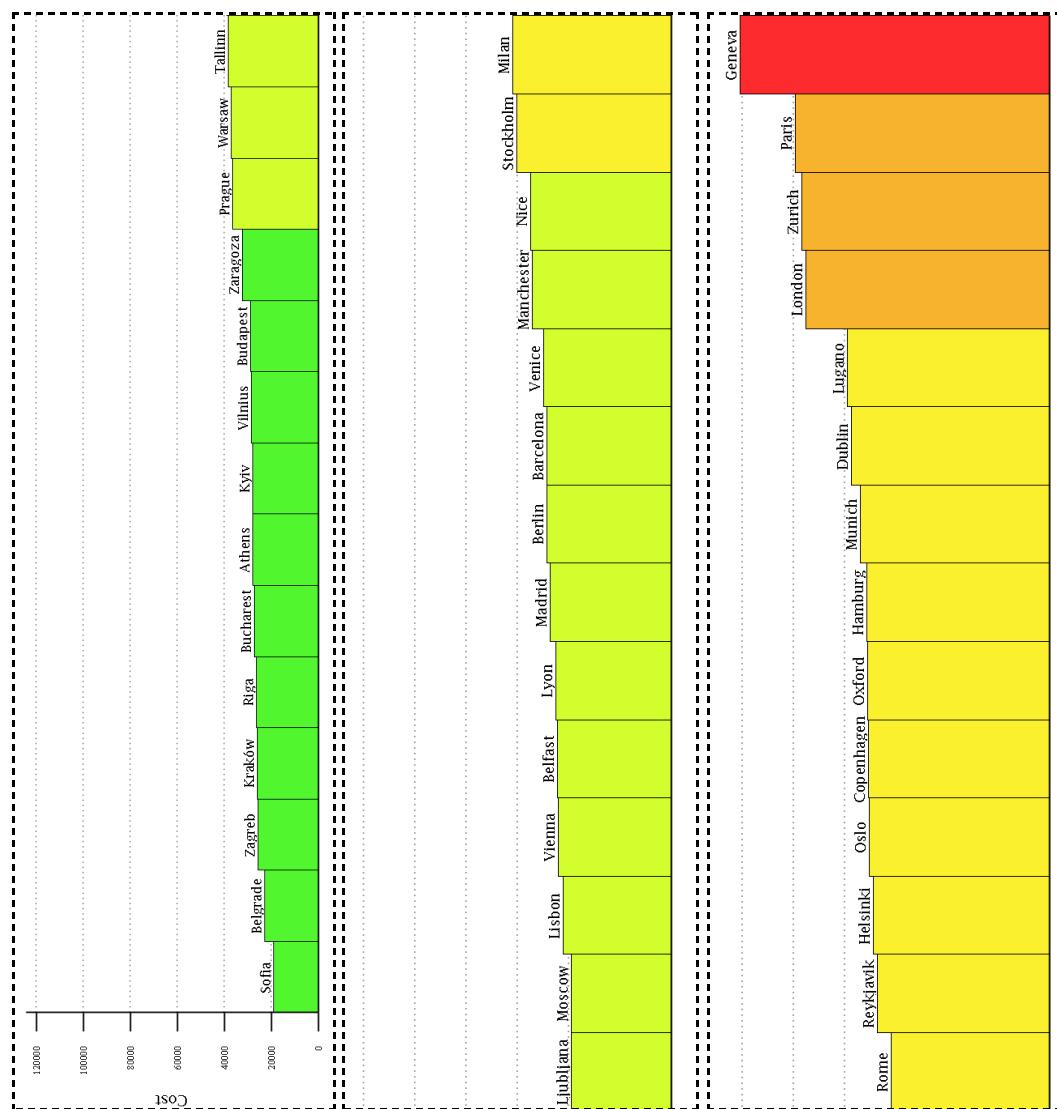
Визначення нових кластерів



Наразі бачимо, що майже нічого не змінилося: лише місто Сарагоса змінило свою приналежність іншому кластеру. Це пов'язано з тим, що обсяг даних не достатньо великий, щоб можна було помітити значні зміни. Якщо зіставити значення нових і старих центроїдів, стане зрозуміло, що їхня різниця майже ніяк не впливає на значення кластерів: знайшлася лише одна така точка X , що при «перенесенні» старого центроїда C_1 у нове місцеположення ця точка X стала ближчою до іншого центроїда C_2 .

Подальший ітераційний процес не приведе ні до яких змін, тому на даному кроці зупиняємо алгоритм.

Результати



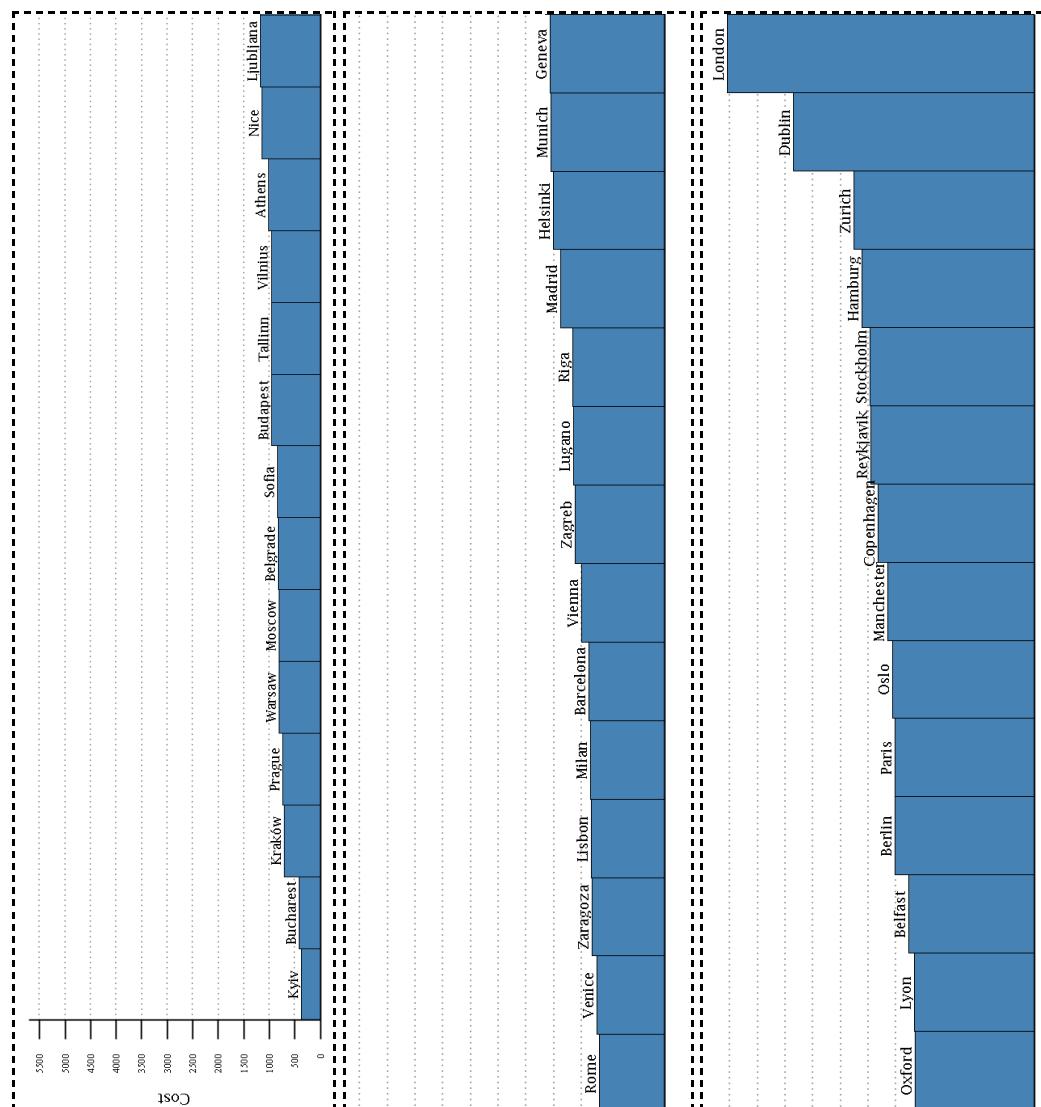
Маємо 5 кластерів, які, починаючи від позначеного зеленим кольором, можна охарактеризувати, відповідно, як «дешева», «дешевше середнього», «середня», «дорожча середньої» та «дорога» оренди (за вартістю). До складу «дешевої» оренди увійшло 11 міст з проміжком цін від 18 906 грн. (м. Софія) до 32 162 грн. (м. Сарагоса). До складу «нижче середнього» було додано 15 міст з проміжком вартості оренди від 36 486 грн. (м. Прага) до 54 874 грн. (м. Ніцца). 12 міст відносяться до «середньої» вартості з проміжком від 60 038 грн. (м. Стокгольм) до 79 023 грн. (м. Лугано). Лондон, Цюрих та Париж виявилися містами із показниками вартості «вище середньої». І лише одне місто, Женева, виявилося у складі «дорогих» зі значенням середньої вартості

оренди житла 120 788 грн. Надалі дані, що описують граничні значення кластерів, будуть відображатись у вигляді таблиці:

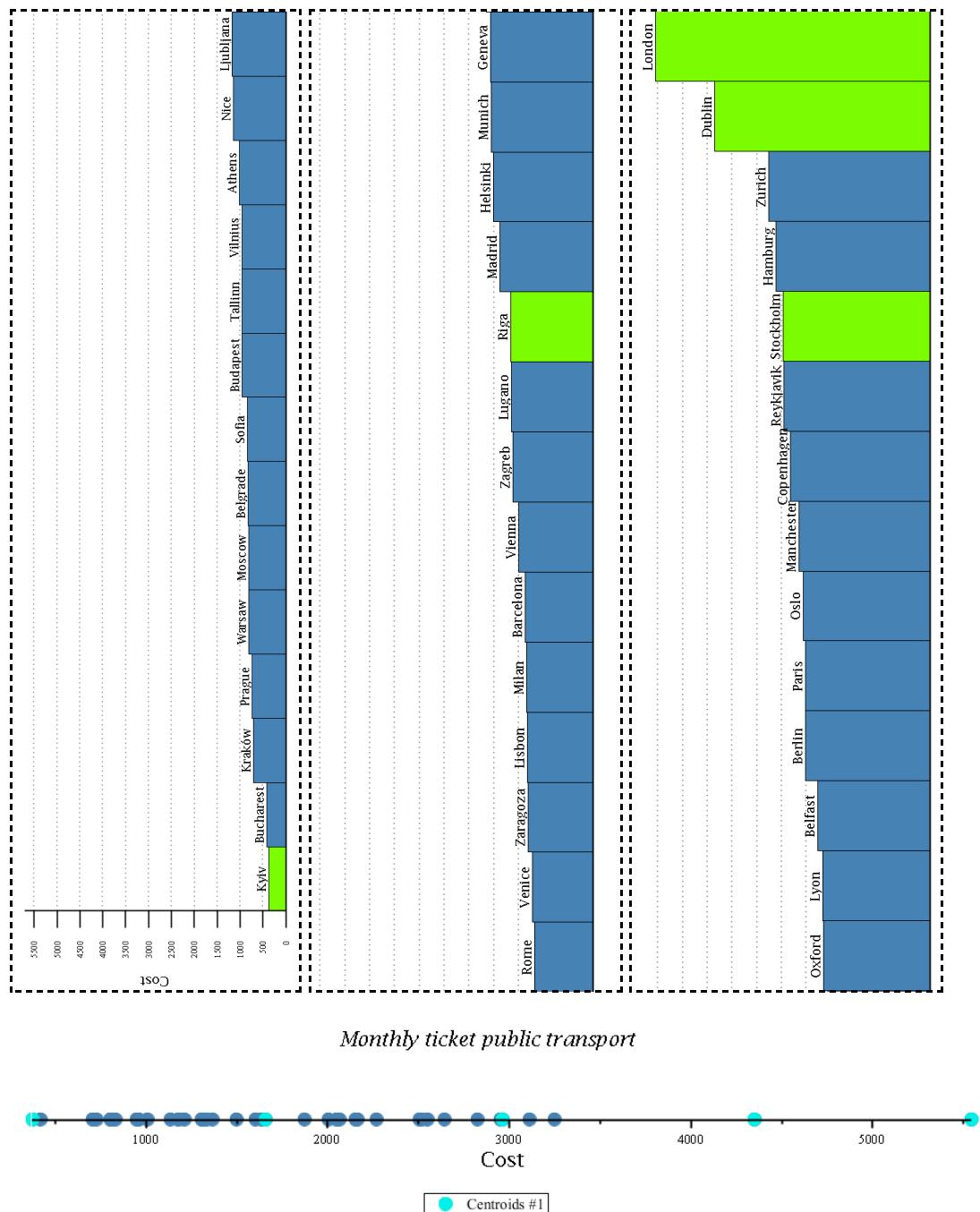
"Cluster #1"	"Points : 11"	"Borders : from Sofia (18905 hr.) to Zaragoza (32162 hr.)"	"Difference : 13257 hr."
"Cluster #2"	"Points : 15"	"Borders : from Prague (36486 hr.) to Nice (54874 hr.)"	"Difference : 18388 hr."
"Cluster #3"	"Points : 12"	"Borders : from Stockholm (60038 hr.) to Lugano (79023 hr.)"	"Difference : 18985 hr."
"Cluster #4"	"Points : 3"	"Borders : from London (95160 hr.) to Paris (99101 hr.)"	"Difference : 3941 hr."
"Cluster #5"	"Points : 1"	"Borders : from Geneva (120788 hr.) to Geneva (120788 hr.)"	"Difference : 0 hr."

2.4.2. Місячний квиток на міський транспорт

Початкові дані, відображені у вигляді гістограм

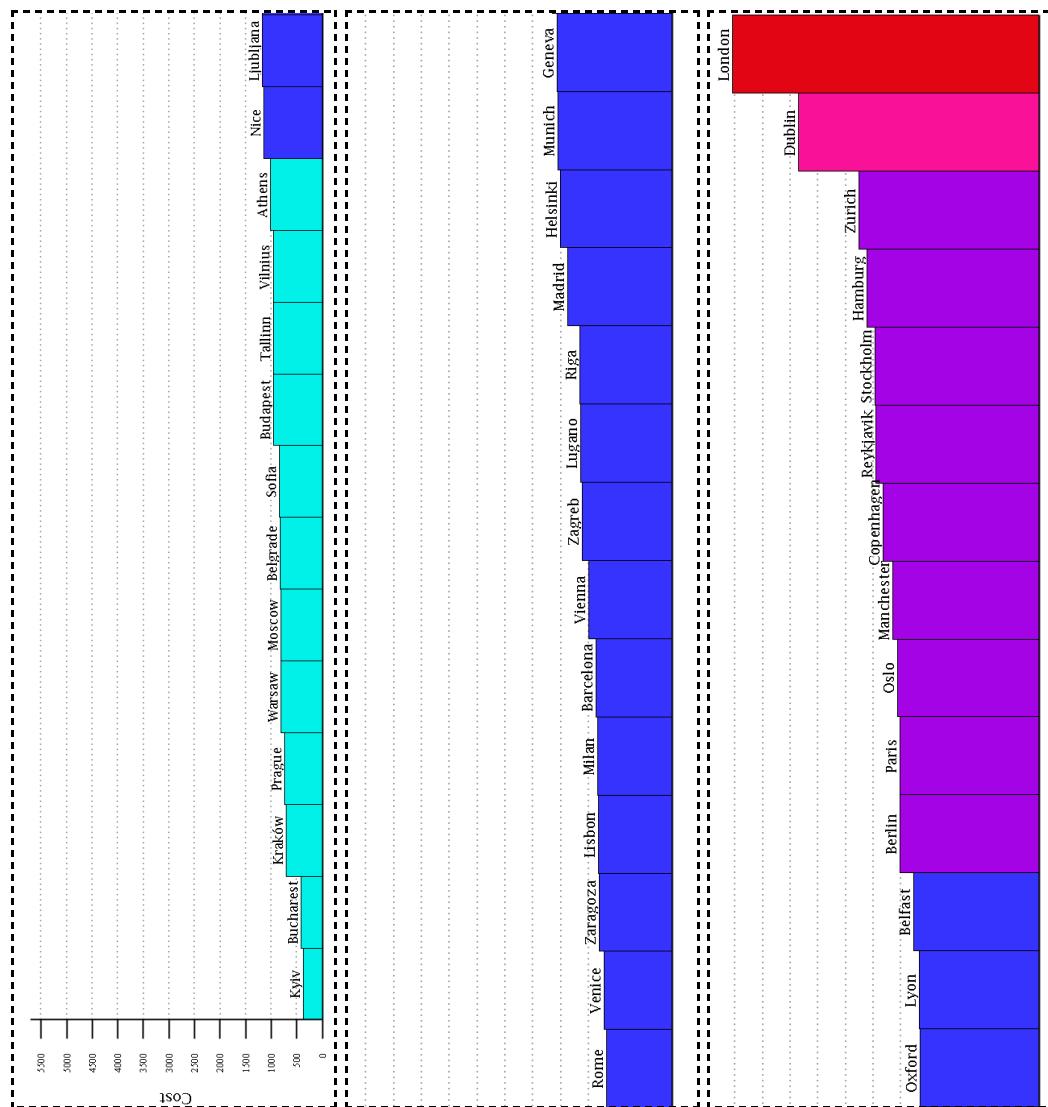


Перше наближення центроїдів

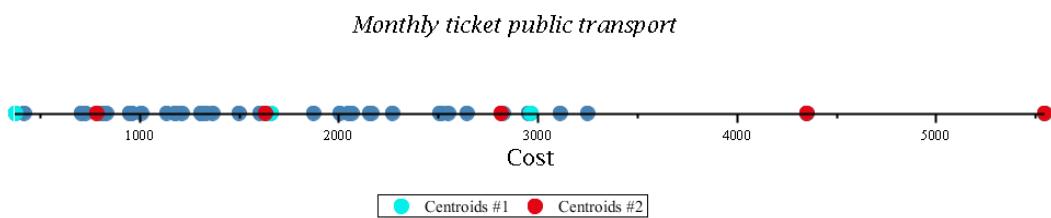


На гістограмі центроїди початкового наближення відмічені зеленим кольором, на числовій прямій – блакитним.

Визначення кластерів (перше наближення)



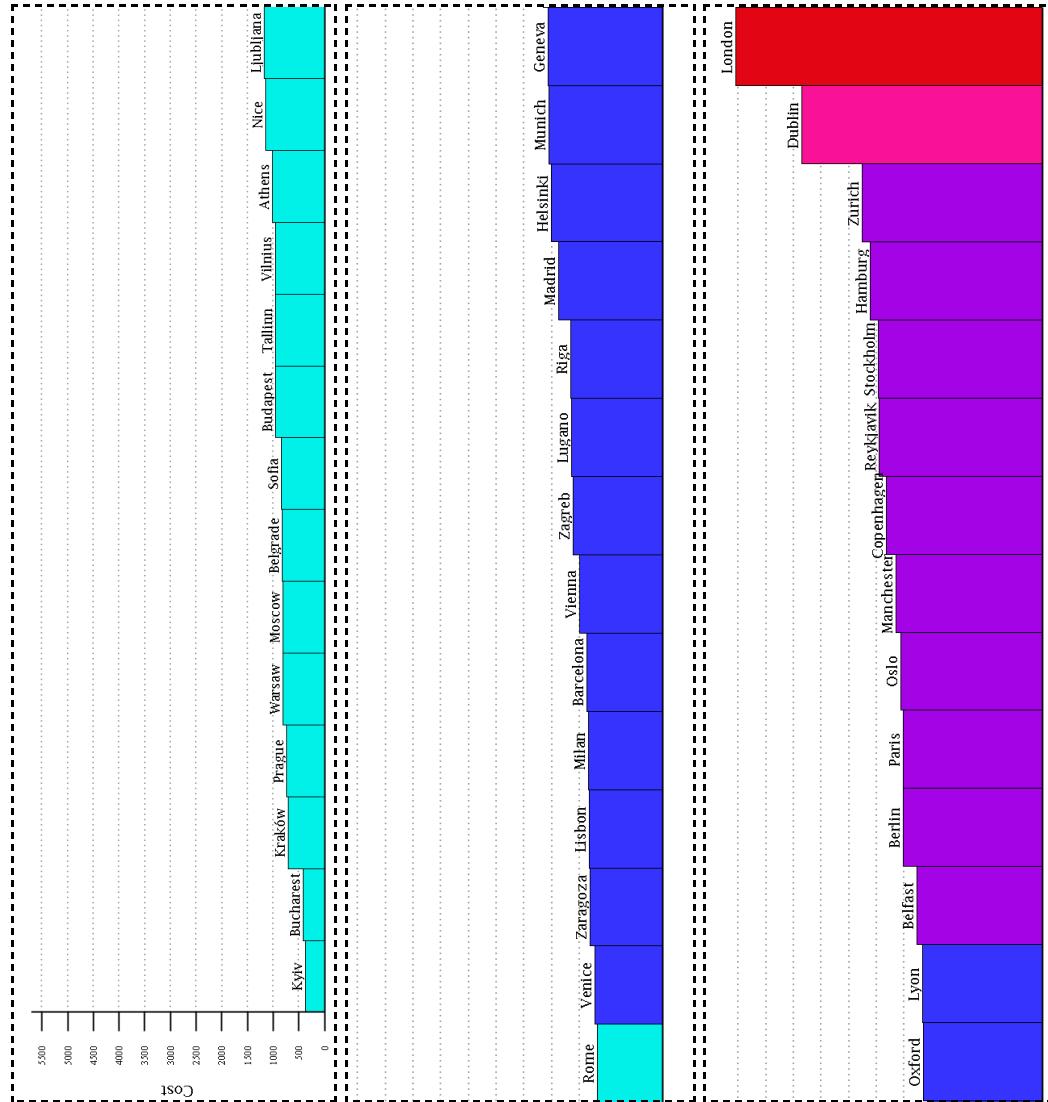
Визначення нових центроїдів



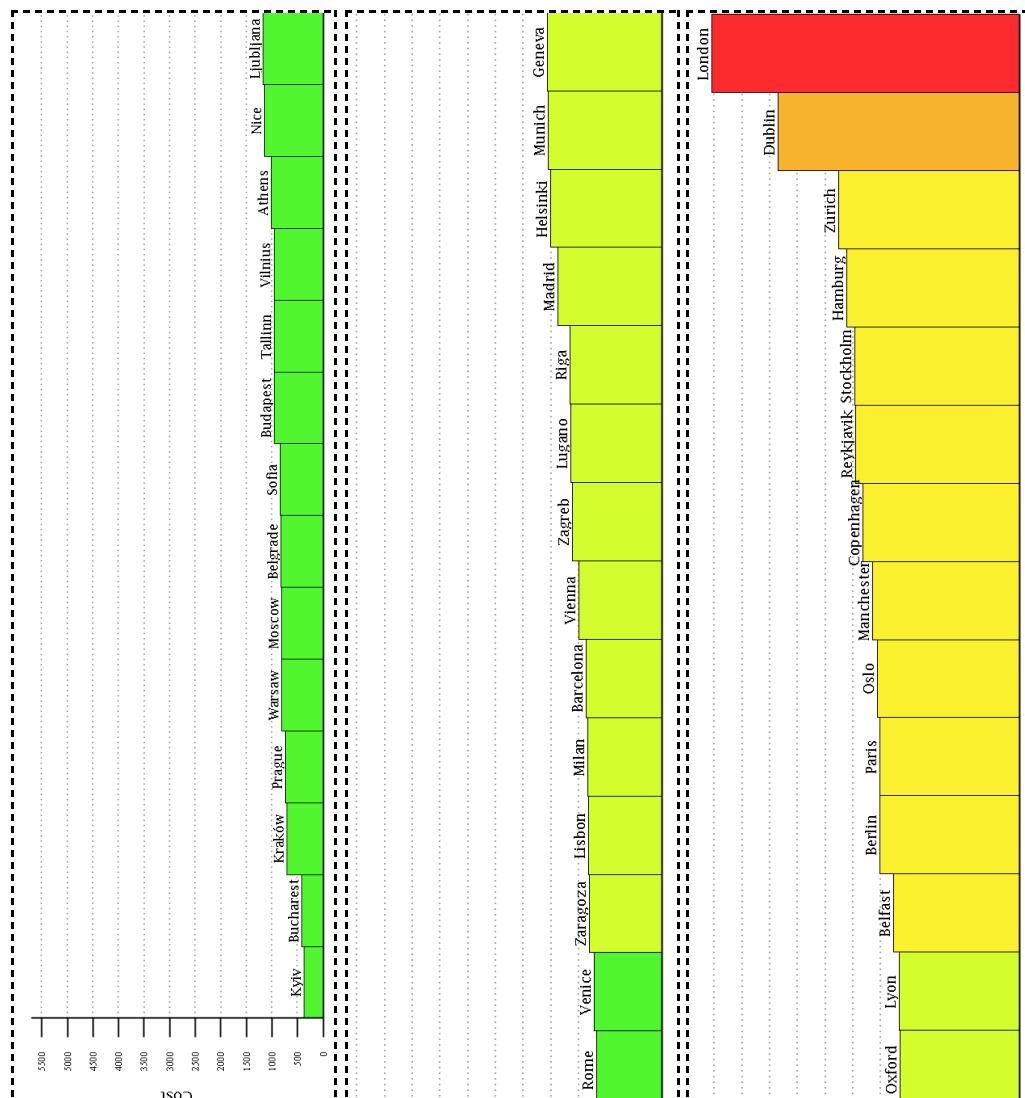
З числової прямої бачимо, що перший та третій центроїди помітно змінили своє місце положення, другий трохи змістився вліво, а четвертий і п'ятий – залишилися на місці. Це можна пояснити тим, що останні два кластери містять лише по одному елементу (можна побачити на гістограмі

першого наближення), які, очевидно, і являють собою центрами мас. Отже, зміщення бути не може.

Визначення нових кластерів (друге наближення)



Результати

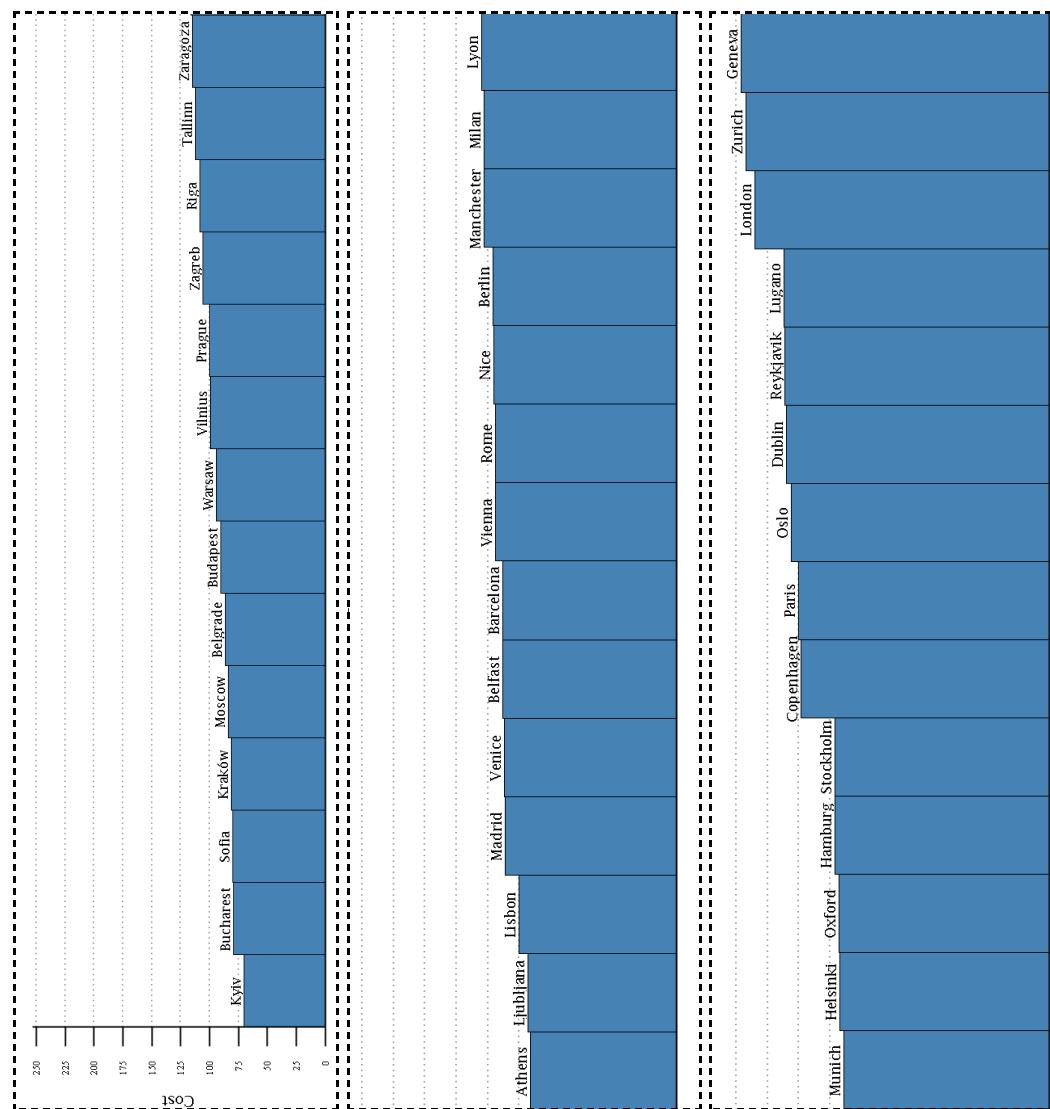


"Cluster #1"	"Points : 16"	"Borders : from Kyiv (376 hr.) to Venice (1211 hr.)"	"Difference : 835 hr."
"Cluster #2"	"Points : 14"	"Borders : from Zaragoza (1309 hr.) to Lyon (2166 hr.)"	"Difference : 857 hr."
"Cluster #3"	"Points : 10"	"Borders : from Belfast (2272 hr.) to Zurich (3251 hr.)"	"Difference : 979 hr."
"Cluster #4"	"Points : 1"	"Borders : from Dublin (4353 hr.) to Dublin (4353 hr.)"	"Difference : 0 hr."
"Cluster #5"	"Points : 1"	"Borders : from London (5546 hr.) to London (5546 hr.)"	"Difference : 0 hr."

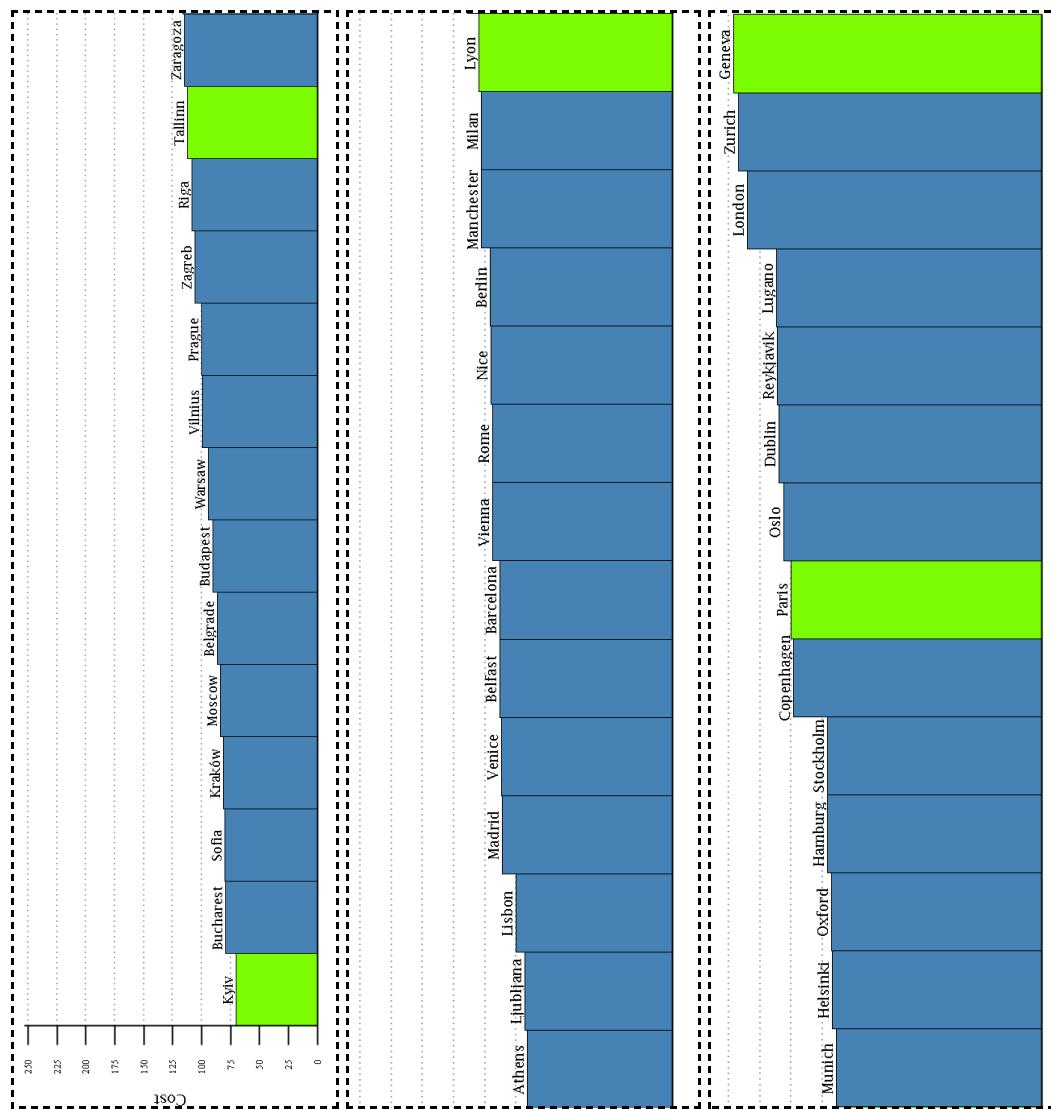
Методу знадобилося лише 3 наближення, щоб дати результат: під час останнього з них місто Венеція було віднесено до «дешевих» (за вартістю проїзду).

2.4.3. Індекс вартості проживання

Початкові дані

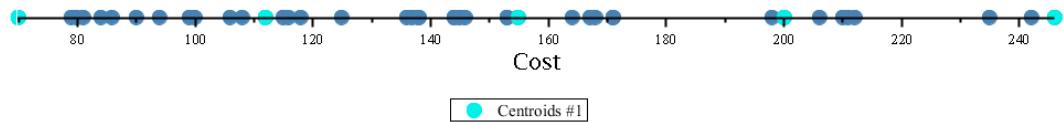


Перше наближення центроїдів

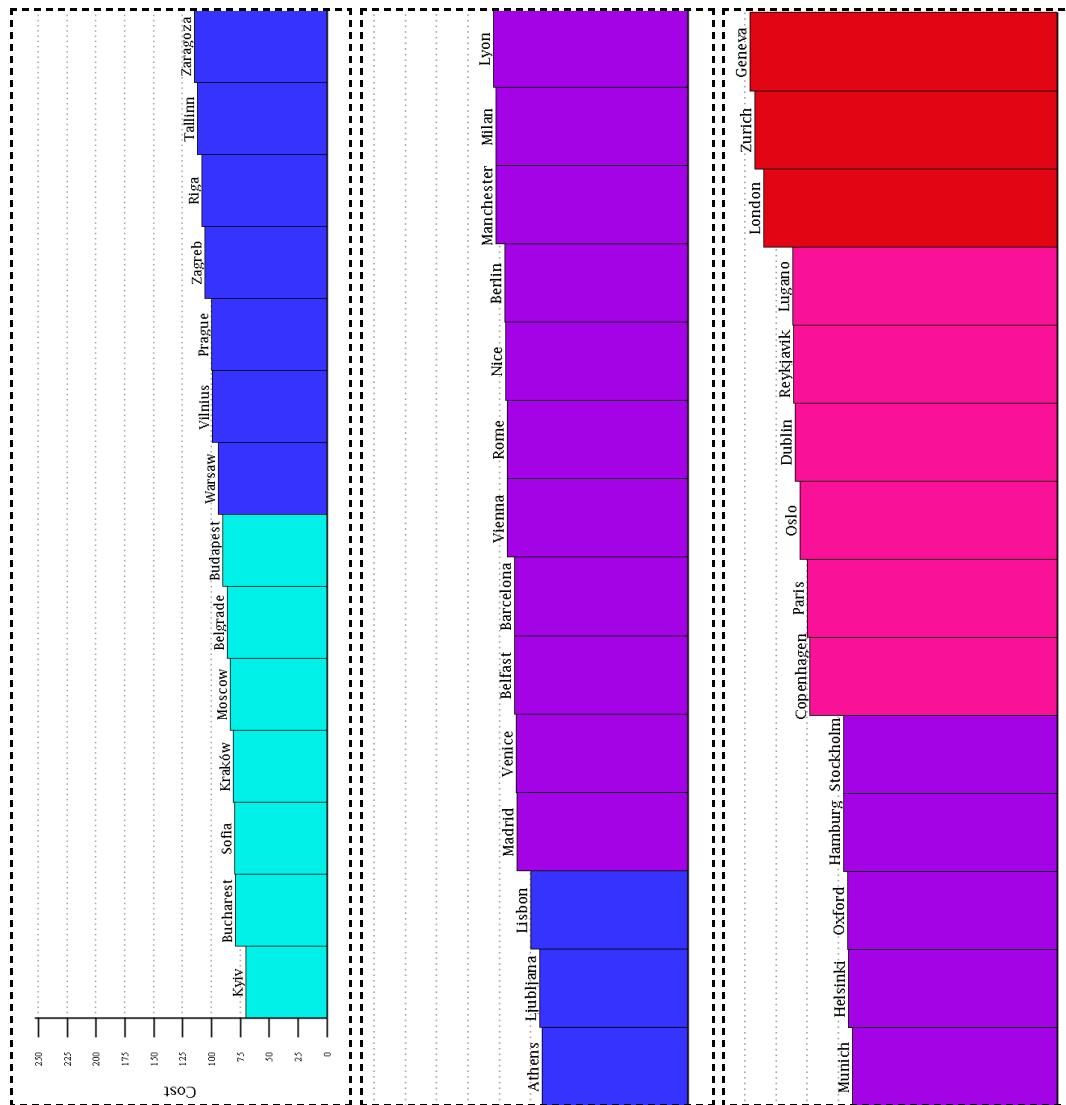


Аналогічно до попереднього прикладу, на гістограмі зеленим кольором зображені міста, що були обрані центроїдами. Далі, на числовій прямій, ті ж самі центроїди відображені блакитним кольором.

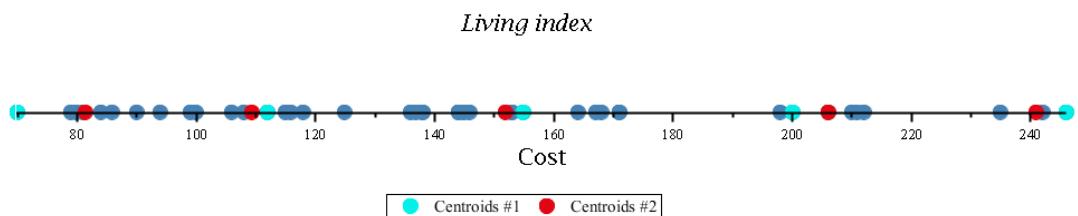
Living index



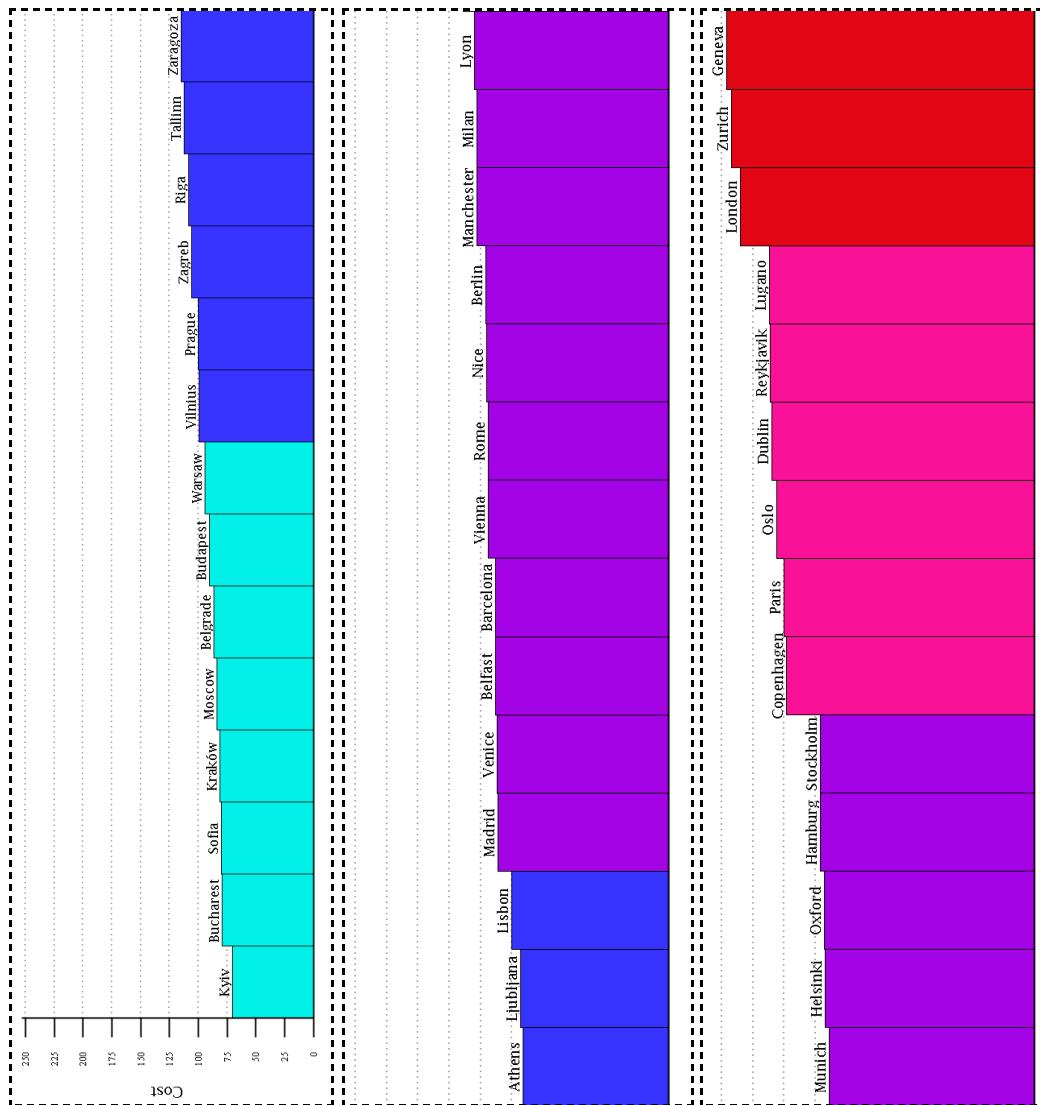
Визначення кластерів (перше наближення)



Визначення нових центроїдів

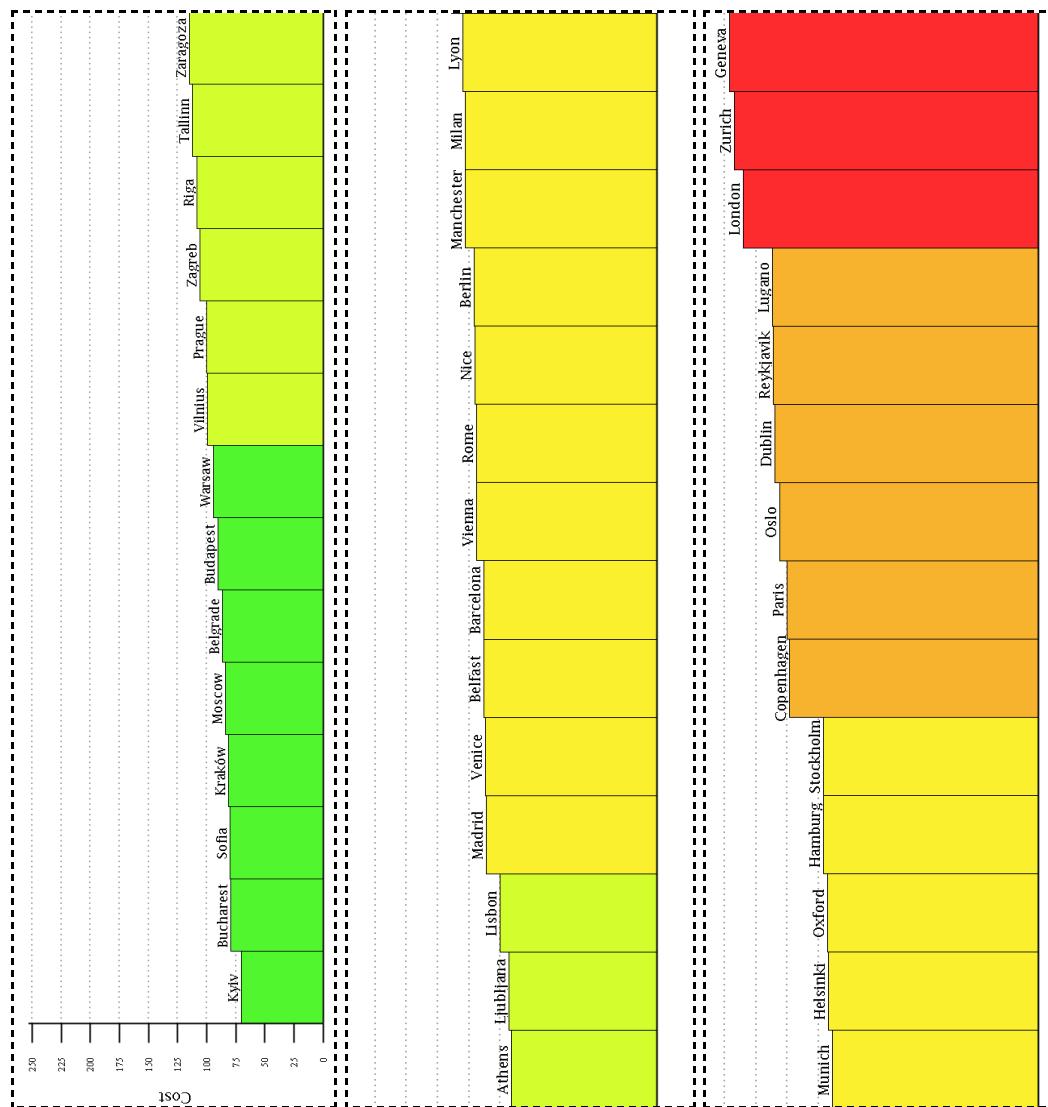


Визначення нових кластерів



І знову, як і в прикладі з орендою, отримані результати майже не відрізняються від першого наближення: лише одне місто, – Варшава, змінило свій кластер із 1-го наближення на інший у 2-му наближенні.

Результати



Маємо:

"Cluster #1"	"Points : 8"	"Borders : from Kyiv (70 hr.) to Warsaw (94 hr.)"	"Difference : 24 hr."
"Cluster #2"	"Points : 9"	"Borders : from Vilnius (99 hr.) to Lisbon (125 hr.)"	"Difference : 26 hr."
"Cluster #3"	"Points : 16"	"Borders : from Madrid (136 hr.) to Stockholm (171 hr.)"	"Difference : 35 hr."
"Cluster #4"	"Points : 6"	"Borders : from Copenhagen (198 hr.) to Lugano (212 hr.)"	"Difference : 14 hr."
"Cluster #5"	"Points : 3"	"Borders : from London (235 hr.) to Geneva (246 hr.)"	"Difference : 11 hr."

2.4.4. Аналіз отриманих результатів

Наразі було проведено кластеризацію методом K-means по трьом критеріям. Відомо, що індекс вартості проживання формується за рахунок певної кількості факторів, серед яких є розглянуті вище (вартість оренди та

місячного квитка на транспорт). Тож давайте розглянемо, наскільки результати за цими факторами відрізняються від результатів за індексом.

Почнемо з «дешевих» за всіма показниками міст: Софія, Белград, Krakів, Бухарест, Київ, Будапешт. Вони займають однакові кластери за всіма розглянутими факторами в результаті роботи методу K-means. Лісабон – єдине місто, що після всіх трьох процедур виявилося по вартості у групі «нижче середнього». До кластеру «середніх» було розподілено лише Стокгольм та Гамбург. Та жодне із розглядуваних міст не займає кластер «вище середнього» за всіма факторами одночасно.

Що ж до інших міст, то результати їх кластеризації не сильно різняться в залежності від обраних факторів. Загреб, Рига та Сарагоса, будучи одними з найдешевших міст для оренди житла, займають кластер «нижче середнього» за двома іншими показниками. Майже аналогічна ситуація з містами Прага, Варшава, Таллін, Любляна, Москва. Міста Відень, Белфаст, Берлін, Манчестер, Мілан, Хельсінкі, Оксфорд та Мюнхен, які «стрибають» між групами «нижче середнього» та «середнім». Рим за одним із показників є «дешевим» містом, але за іншим фактором та індексом належить до «середніх». Дублін та Париж займають перехідне положення між «середніми» та «вище середнього». Лондон та Цюрих є «дорогими» містами за вартістю оренди та загальним індексом, але «вище середнього» та «середнім» за вартістю проїзду, відповідно.

Трохи слабший вплив розглянутих факторів на загальний індекс вартості проживання представляють результати міст, в яких кластеризація за окремими показниками зовсім не співпала із кластеризацією за індексом. Наприклад, Афіни та Вільнюс, незважаючи на перебування у групі «дешевих» за вартостями оренди та квитка, за індексом були долучені до кластеру «нижче середнього». Ліон, Мадрид, Барселона за факторами були віднесені до «нижче середнього», але за індексом до «середніх». Також можна виділити Венецію та Ніццу, які за своїми показниками «стрибають»

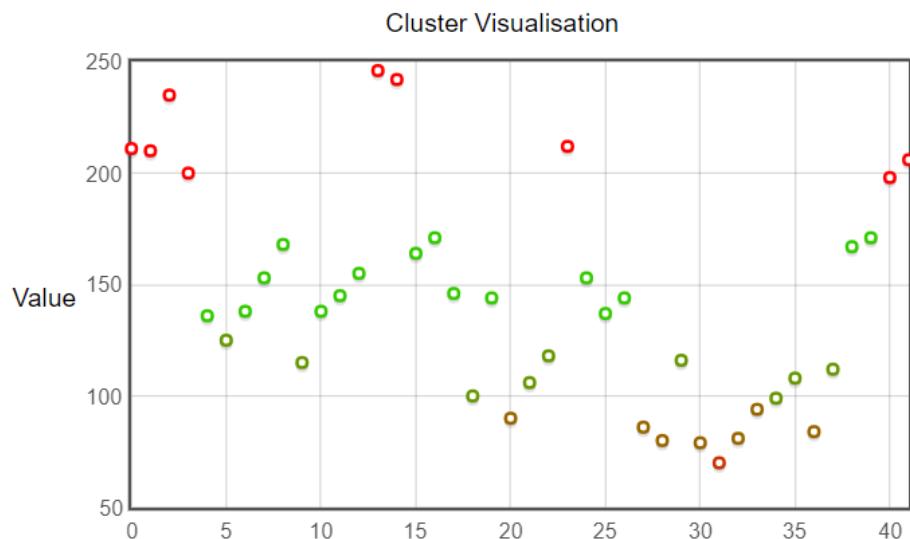
між «дешеві» та «нижче середнього», але за індексом належать до «середніх». Рейк'явік, Осло, Копенгаген та Люгано за показниками є «середніми» або «нижче середнього» містами, але відповідно до загального індексу були класифіковані до групи «вище середнього». Цю різницю відбувається через положення «середні»-«вище середнього» за обраними окремими факторами, але займає кластер «дорогі» за загальним індексом.

Отож, наразі маємо 9 міст, що займають одинакові кластери, незалежно від обраного фактору (серед розглянутих); 21 місто, результати кластеризації за загальним індексом яких співпадали хоча б з одним із результатів роботи методу за факторами. У 12 міст результати кластеризації за індексом не співпадали з жодним результатом роботи алгоритму за розглянутими факторами.

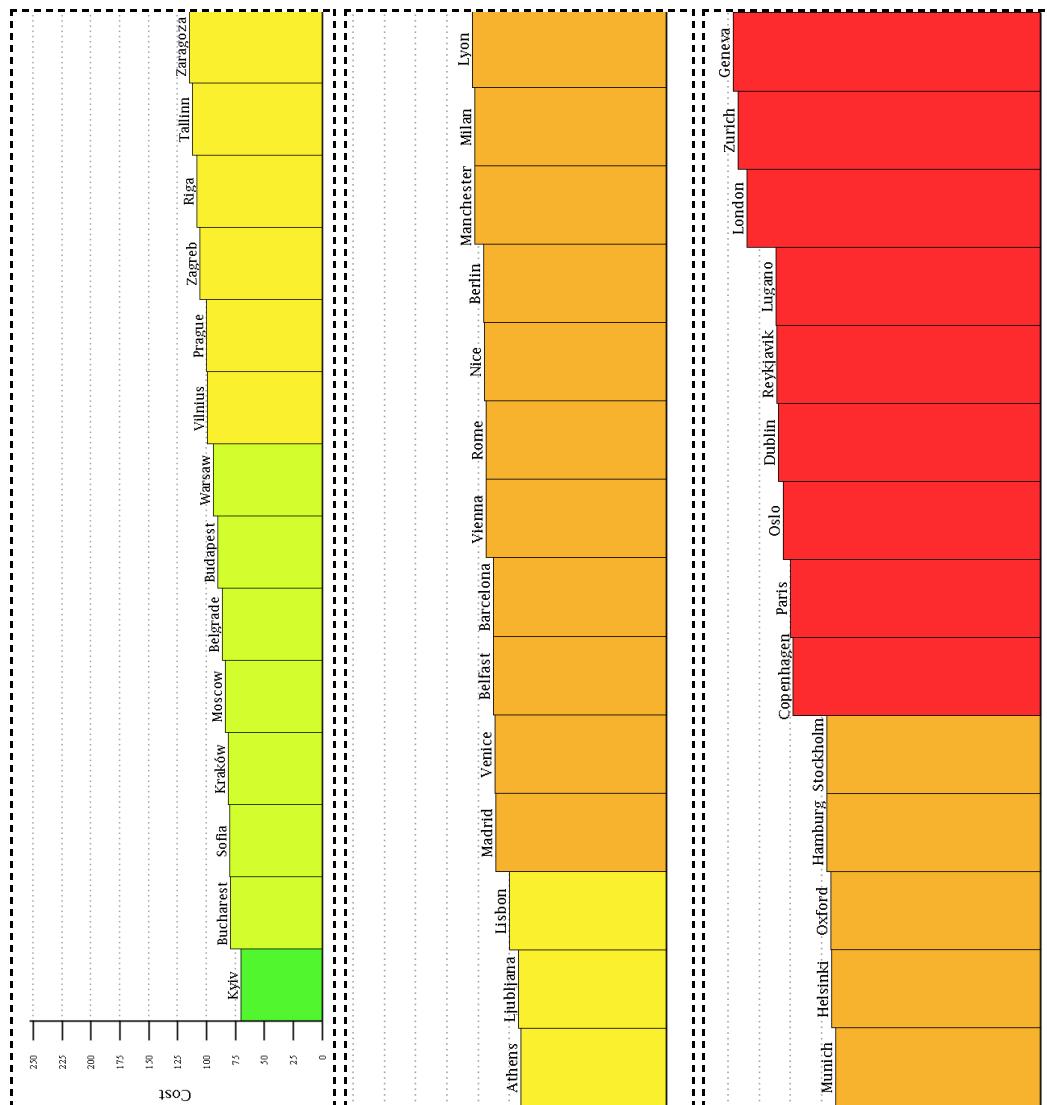
2.4.5. Стороння реалізація алгоритму

Скористаємося стороннім сервісом, щоб подивитись, наскільки будуть відрізнятися отримані дані, на прикладі індексу вартості проживання. Інформацію щодо стороннього сервісу див. [2].

Візуалізація результатів ресурсами сервісу:



Побудуємо гістограму та виведемо інформацію щодо кластерів у Maple:



"Cluster #1"	"Points : 1"	"Borders : from Kyiv (70 hr.) to Kyiv (70 hr.)"	"Difference : 0 hr."
"Cluster #2"	"Points : 7"	"Borders : from Bucharest (79 hr.) to Warsaw (94 hr.)"	"Difference : 15 hr."
"Cluster #3"	"Points : 9"	"Borders : from Vilnius (99 hr.) to Lisbon (125 hr.)"	"Difference : 26 hr."
"Cluster #4"	"Points : 16"	"Borders : from Madrid (136 hr.) to Stockholm (171 hr.)"	"Difference : 35 hr."
"Cluster #5"	"Points : 9"	"Borders : from Copenhagen (198 hr.) to Geneva (246 hr.)"	"Difference : 48 hr."

Бачимо, що результати кластеризації сильно відрізняються від попередніх: тепер у перший кластер «найдешевших міст» увійшов лише Київ. З іншого боку, до «найдорожчих» потрапило аж 9 міст.

Таку різницю між результатами можна пояснити одним з недоліків K-means, а саме значну залежність кластеризації від початкового вибору центройдів. Річ у тому, що алгоритм ресурсу призначений для великої кількості даних, тому в якості початкового наближення центройдів

обираються точки із випадкової підмножини X_0 всього набору точок. Це призвело до того, що точки, які були б більш доцільними початковими центроїдами, не увійшли до X_0 .

РОЗДІЛ III. МЕТОД КЛАСТЕРИЗАЦІЇ BIRCH

3.1. Загальний опис методу

Незважаючи на простоту та прозорість, метод K-means, як і більшість інших методів, стає менш ефективним при збільшенні бази даних. Крім того, багато методів неадекватно ведуть себе у разі, коли даних є занадто багато, щоб поміститися в оперативну пам'ять. Тоді на поміч приходить метод BIRCH.

BIRCH розшифровується як збалансоване ітераційне скорочення та кластеризація з використанням ієрархій (balanced iterative reducing and clustering using hierarchies). Його призначення полягає у кластеризації дуже великих наборів числових даних. Окрім цього, серед його переваг виділяють роботу на обмеженому об'ємі пам'яті, адже кожен крок кластеризації є локальним та здійснюється без перегляду всіх точок даних та існуючих на поточний момент кластерів.

Для ефективної кластеризації алгоритму потрібно лише одне сканування даних (тобто зі збільшенням кількості даних складність збільшується лінійно). А за допомогою однієї або більше додаткових ітерацій можна і далі поліпшувати ефективність. BIRCH також є першим алгоритмом, запропонованим для ефективного управління «шумами» (даними, які не вписуються в загальне уявлення моделі).

Недоліками методу є робота лише з числовими даними та добре відокремлення лише кластерів сферичної форми.

3.2. Математичний опис методу

Введення у термінологію

Нехай у кластері є N точок розмірності d : $\{\bar{X}_i\}$, $i = 1, 2, \dots, N$.

Тоді центроїд \bar{X}_0 , радіус R і діаметр D кластеру визначаються наступним чином:

$$\overrightarrow{X_0} = \frac{\sum_{i=1}^N \overrightarrow{X_i}}{N}$$

$$R = \sqrt{\frac{\sum_{i=1}^N (\overrightarrow{X_i} - \overrightarrow{X_0})^2}{N}}$$

$$D = \sqrt{\frac{\sum_{i=1}^N \sum_{j=1}^N (\overrightarrow{X_i} - \overrightarrow{X_j})^2}{N(N-1)}}$$

Для вимірювання близькості один до одного двох кластерів вводяться наступні метрики:

D_0 – евклідова відстань між центроїдами

D_1 – манхеттенська відстань між центроїдами

Нехай є два кластери, що містять відповідно N_1 та N_2 точок розмірності

$d: \{\overrightarrow{X}_i\}$, де $i = 1, 2, \dots, N_1$ та $\{\overrightarrow{X}_j\}$, де $j = N_1 + 1, N_1 + 2, \dots, N_1 + N_2$. Середня відстань D_2 , середня відстань у кластері D_3 та дисперсія D_4 для двох кластерів визначаються наступним чином:

$$D_2 = \sqrt{\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\overrightarrow{X_i} - \overrightarrow{X_j})^2}{N_1 N_2}}$$

$$D_3 = \sqrt{\frac{\sum_{i=1}^{N_1+N_2} \sum_{j=1}^{N_1+N_2} (\overrightarrow{X_i} - \overrightarrow{X_j})^2}{(N_1 + N_2)(N_1 + N_2 - 1)}}$$

$$D_4 = \sum_{k=1}^{N_1+N_2} \left(\overrightarrow{X_k} - \frac{\sum_{i=1}^{N_1+N_2} \overrightarrow{X_i}}{N_1 + N_2} \right)^2 - \sum_{i=1}^{N_1} \left(\overrightarrow{X_i} - \frac{\sum_{i=1}^{N_1} \overrightarrow{X_i}}{N_1} \right)^2 - \sum_{j=N_1+1}^{N_1+N_2} \left(\overrightarrow{X_j} - \frac{\sum_{i=N_1+1}^{N_1+N_2} \overrightarrow{X_i}}{N_2} \right)^2$$

Таким чином, величини \bar{X}_0 , R та D відносяться до окремих кластерів, а D_0, D_1, D_2, D_3, D_4 – до об'єднання двох кластерів.

Поняття кластерного елементу

Означення. Нехай в кластері містяться N точок розмірності $d: \{\bar{X}_i\}$, де $i = 1, 2, \dots, N$. **Кластерний елемент** (CF) - це трійка чисел, що характеризує інформацію про кластер: $CF = (N, \bar{LS}, SS)$, де N – це кількість елементів вхідних даних, що містяться у кластері; \bar{LS} – лінійна сума вхідних даних $(\sum_{i=1}^N \bar{X}_i)$ та SS – сума квадратів вхідних даних $(\sum_{i=1}^N \bar{X}_i^2)$. Наприклад, CF вектор для точки (3,4) дорівнює (1, (3,4), 25).

Теорема. Нехай $CF_1 = (N_1, \bar{LS}_1, SS_1)$ та $CF_2 = (N_2, \bar{LS}_2, SS_2)$ – CF вектори двох кластерів, що не перетинаються. Тоді при їх об'єднанні утворюється кластер з наступним CF вектором:

$$CF = CF_1 + CF_2 = (N_1 + N_2, \bar{LS}_1 + \bar{LS}_2, SS_1 + SS_2)$$

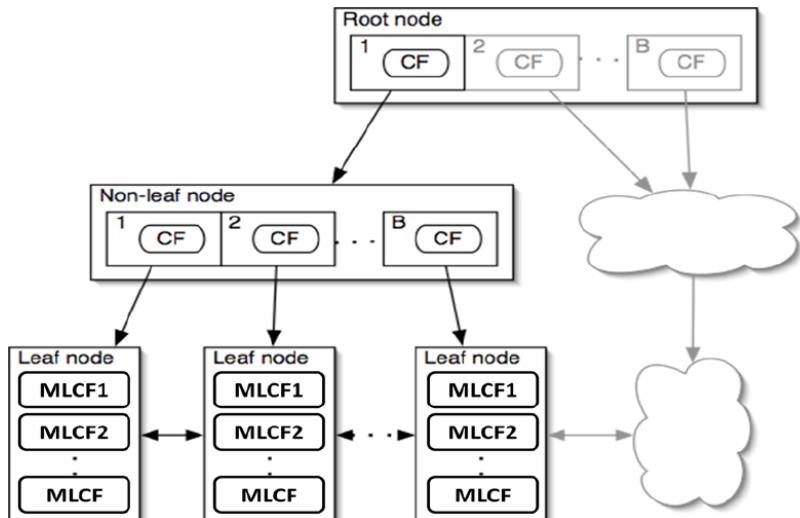
Із зазначених вище визначення та теореми випливає той факт, що при об'єднанні кластерів їх CF вектор може обчислюватися інкрементно й точно. Легко показати, що обчислити величини \bar{X}_0 , R , D , D_0, D_1, D_2, D_3, D_4 , як і звичайні метрики, також не складе труднощів.

Таким чином, в даному випадку інформацію про кластер є не весь набір точок, а лише вектор CF. Зберігання одного тільки вектора може здатися неефективним, але цього цілком достатньо для оперування всіма необхідними метриками, на основі яких приймаються рішення в алгоритмі BIRCH.

Поняття кластерного дерева

Означення. Кластерне дерево (CF Tree) - це виважено збалансоване дерево з двома параметрами: B – коефіцієнт розгалуження, T – порогова величина.

Кожен нелистовий вузол дерева має не більше ніж B входжень вузлів наступної форми: $[CF_i, child_i]$, де $i = 1, \dots, B$, $child$ – вказівник на i -ий дочірній вузол, CF_i – CF вектор відповідного підкластера.



Кожен листовий вузол має посилання на два сусідні вузли для того, щоб зв'язати всі вузли для ефективного сканування. Усі листові вузли повинні задовольняти пороговому обмеженню T , тобто діаметр (радіус) не повинен перевищувати T . Під час додання нових даних CF дерево будується динамічно.

Вставка елементів у дерево

Етап 1. Визначення відповідного листового вузла. Починаючи з кореня, алгоритм рекурсивно спускається по CF дереву, обираючи найближчий дочірній вузол за обраною метрикою D_0, D_1, D_2, D_3 або D_4 .

Етап 2. Перетворення листа. Коли алгоритм доходить до листа, він знаходить найближчий елемент L , а потім перевіряє, чи можна додати у нього

новий без порушення граничної умови. Якщо можна, тоді SF вектор елемента L змінюється з урахуванням нового елемента. Якщо ж не можна, тоді буде створений новий елемент листа. Якщо в листі для цього достатньо місця, то етап завершено, інакше лист повинен розщепитися. Розщеплення вузла проводиться за допомогою вибору за основу пари найбільш віддалених один від одного елементів і перерозподіл решти елементів відповідно до обраного критерію близькості.

Етап 3. Зміна шляху до листового вузла. Після вставки нового елемента у лист ми повинні оновити інформацію SF вектору для кожного нелистового елемента на шляху до листового вузла. При відсутності розщеплення це означає просте додавання SF векторів для відображення факту додавання нового елемента. А от розщеплення листа вже вимагає вставки нового нелистового елемента в батьківський вузол для задання щойно створеного листа. Якщо у «батьків» є місце для цього елемента, то необхідно просто перетворити SF вектори на всіх більш високих рівнях. Інакше, нам також доведеться розщепити батьківський вузол. І так далі до кореня. Якщо розщеплений корінь, то висота дерева збільшується на одиницю.

Головна ідея алгоритму

Головна ідея полягає у багаторазовому виконанні операції вставки нового елемента у дерево, що, у свою чергу, потребує обчислення наступних величин:

- радіуса та діаметра (при спробі вставити новий елемент в один з підклasterів листового вузла дерева);
- відстані між двома елементами (при пошуку найближчого до нового елементу підклasterа у вузлі дерева за однією з розглянутих раніше метрик).

Знаючи характеристики кластерних елементів, можна значно спростити формули необхідних значень:

$$\overrightarrow{X_0} = \frac{\overrightarrow{LS}}{N},$$

$$R = \sqrt{\frac{SS - 2\overrightarrow{X_0}\overrightarrow{LS} + N(\overrightarrow{X_0})^2}{N}},$$

$$D = \sqrt{\frac{2(N \cdot SS - (\overrightarrow{LS})^2)}{N(N - 1)}}$$

$$D_2 = \sqrt{\frac{N_1 \cdot SS_2 + N_2 \cdot SS_1 - 2\overrightarrow{LS}_1\overrightarrow{LS}_2}{N_1 \cdot N_2}}$$

$$D_3 = \sqrt{\frac{2((N_1 + N_2)(SS_1 + SS_2) - (\overrightarrow{LS}_1 + \overrightarrow{LS}_2)^2)}{(N_1 + N_2)(N_1 + N_2 - 1)}}$$

3.3. Структура алгоритму

Фаза 1. Будуємо CF дерево

Головним завданням фази є сканування всіх даних і побудова в пам'яті вихідного CF дерева.

Реалізація фази 1:

- Вставити точку у дерево
 - Знайти шлях (базуючись на значеннях D_0, D_1, D_2, D_3, D_4 між CF векторами дітей у нелистовому вузлі)
 - Модифікувати лист:
 - знайти найближчий листовий елемент (базуючись на значеннях D_0, D_1, D_2, D_3, D_4 CF векторів у листовому вузлі);
 - перевірити, чи можна у нього вставити нову точку.

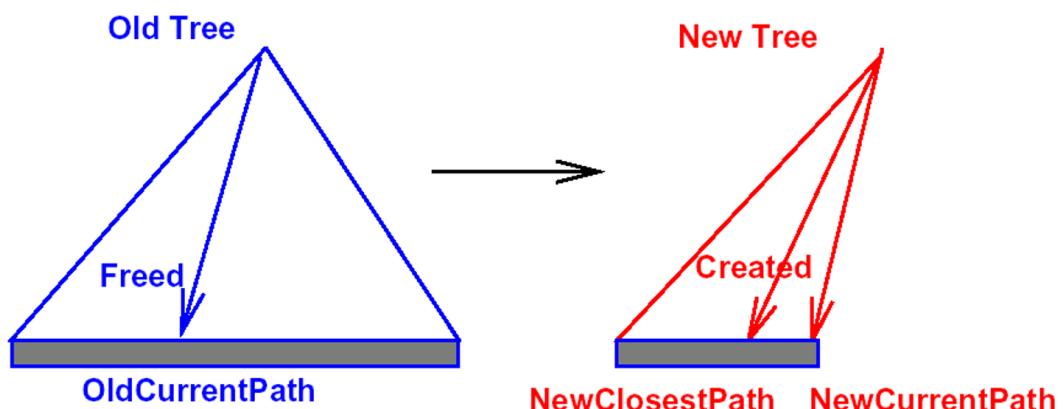
- Модифікувати шлях до листового елементу
- Виконати розбиття, якщо листовий вузол заповнений, внести зміни у батьківські вузли

Фаза 2. Ущільнення CF дерева

Дана фаза є опціональною. Відомо, що існуючі методи глобальної кластеризації, що використовуються в рамках фази 3, мають різні діапазони вхідних значень, при яких їхня реалізація є найбільш ефективною. Тому потенційно існує прогалина між розміром першої фази і вхідними значеннями третьої. Друга фаза якраз є відсутньою ланкою: так само як і перша, вона сканує листові елементи початкового CF дерева для того, щоб побудувати менше дерево, видаляючи ще більше елементів з низькою щільністю та групуючи переповнені підкластери у більші кластери.

Реалізація фази 2:

- Вибрати більший T (поріг)
- Розглянути записи у вузлах листків
- Повторно вставити записи CF у нове дерево:
 - якщо новий шлях стоїть «перед» старим шляхом, перемістити його у новий;
 - якщо новий шлях дорівнює оригінальному, залишити його незмінним.



Фаза 3. Глобальна кластеризація

У цій фазі для листових елементів застосовується будь-який алгоритм глобальної кластеризації. Існуючі алгоритми кластеризації наборів точок можуть також працювати з наборами підкластерів, кожен з яких представлений своїм CF вектором. Знаючи CF вектор, можна обчислити центр мас і надалі замінити всю інформацію про кластер цим значенням (цього достатньо для обчислення більшості необхідних метрик). Після третьої фази ми отримуємо набір кластерів, який відображає основні характеристики даних, що розподіляються.

Реалізація фази 3:

- Розглянути записи CF векторів у листових вузлах
- Використати центроїд як представника кластера
- Виконати традиційну кластеризацію (наприклад, за допомогою K-means)

Фаза 4. Переробка кластерів

Після фази 3 можуть залишитися деякі неточності. Для їх вирішення було запропоновано фазу 4. Дані фаза є опціональною і спричиняє витрати на додаткове сканування даних. Ідея полягає у використанні центроїдів кластерів, отриманих у третій фазі, як основи і перерозподіляє точки вхідних даних у найближчі для них основи для отримання нового списку кластерів. Фаза 4, якщо це необхідно користувачу, може бути розширенна додатковими проходами для впевненості в тому, що результат наближається до оптимального.

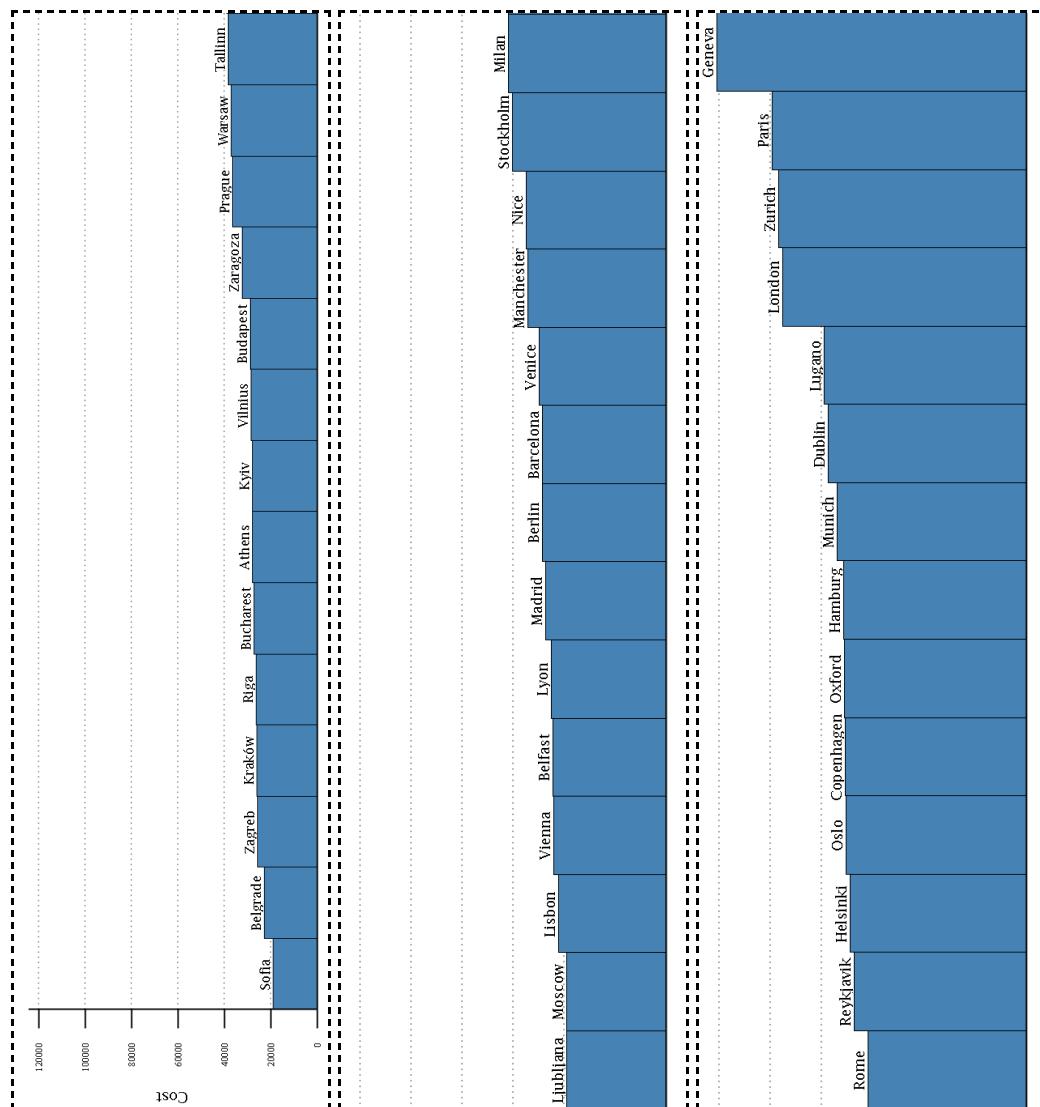
3.4. Робота методу

Робота методу BIRCH буде продемонстрована на розбитті множини міст за величиною місячної оренди 85 m^2 мебльованого житла в дорогому районі, місячною вартістю квитка на міський транспорт та за величиною

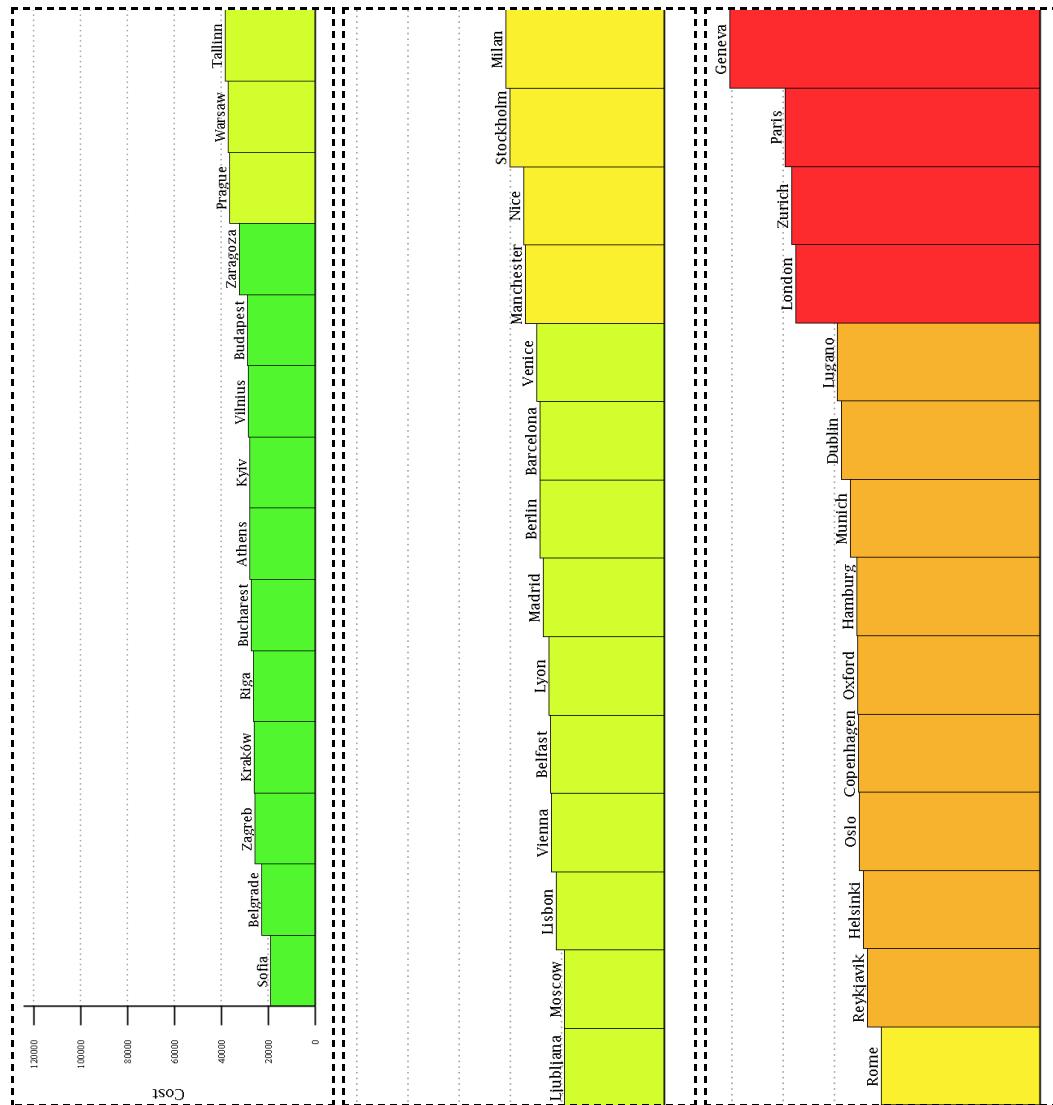
індексу вартості проживання. Алгоритм був реалізований за допомогою мови програмування Python. Обрані дані для вибірки див. у Додатку А.

3.4.1. Місячна оренда 85 м² мебльованого житла в дорогому районі

Початкові дані, відображені у вигляді гістограми



Результати після побудови СF дерева та глобальної кластеризації за допомогою агломеративного методу



```

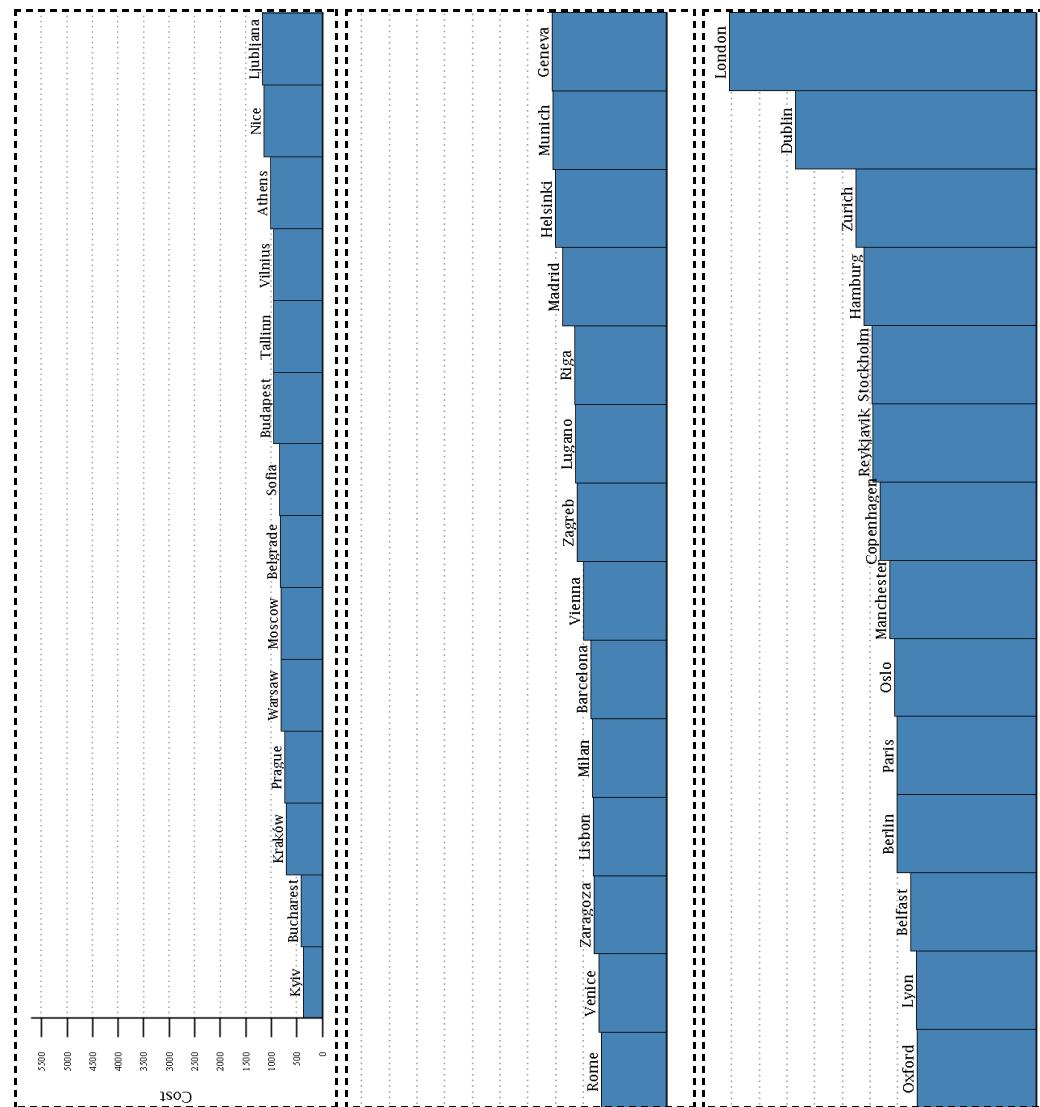
[ "Cluster #1" "Points : 11" "Borders : from Sofia (18905 hr.) to Zaragoza (32162 hr.)" "Difference : 13257 hr."
"Cluster #2" "Points : 13" "Borders : from Prague (36486 hr.) to Venice (49814 hr.)" "Difference : 13328 hr."
"Cluster #3" "Points : 5" "Borders : from Manchester (54266 hr.) to Rome (61858 hr.)" "Difference : 7592 hr."
"Cluster #4" "Points : 9" "Borders : from Reykjavik (67185 hr.) to Lugano (79023 hr.)" "Difference : 11838 hr."
"Cluster #5" "Points : 4" "Borders : from London (95160 hr.) to Geneva (120788 hr.)" "Difference : 25628 hr." ]

```

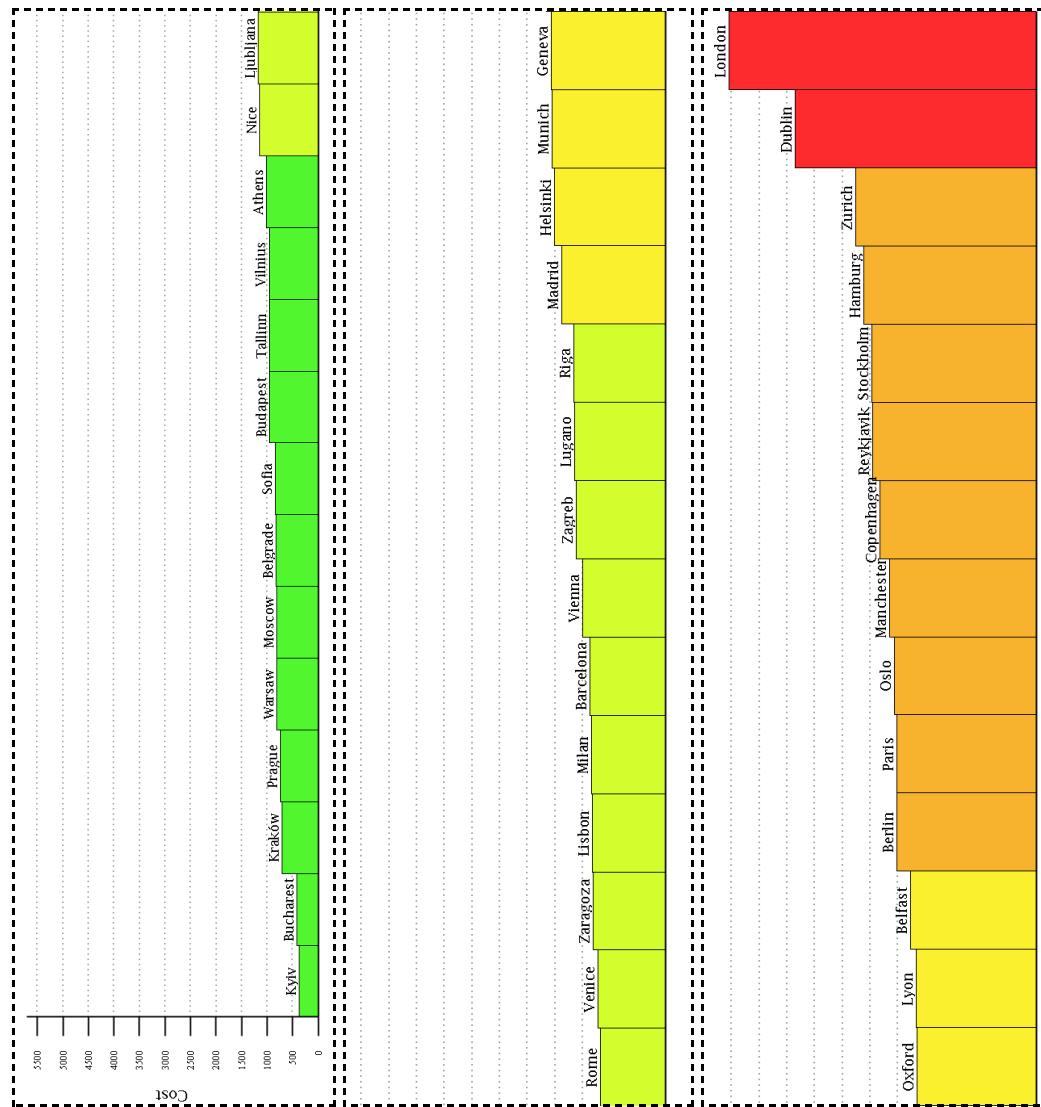
Бачимо, що результати зовсім не відрізняються для першого кластеру, якщо порівнювати з результатами K-means, але далі картина дещо змінюється: BIRCH більш рівномірно розподілив точки по кластерам.

3.4.2. Місячний квиток на міський транспорт

Початкові дані, відображені у вигляді гістограми



Результати методу BIRCH

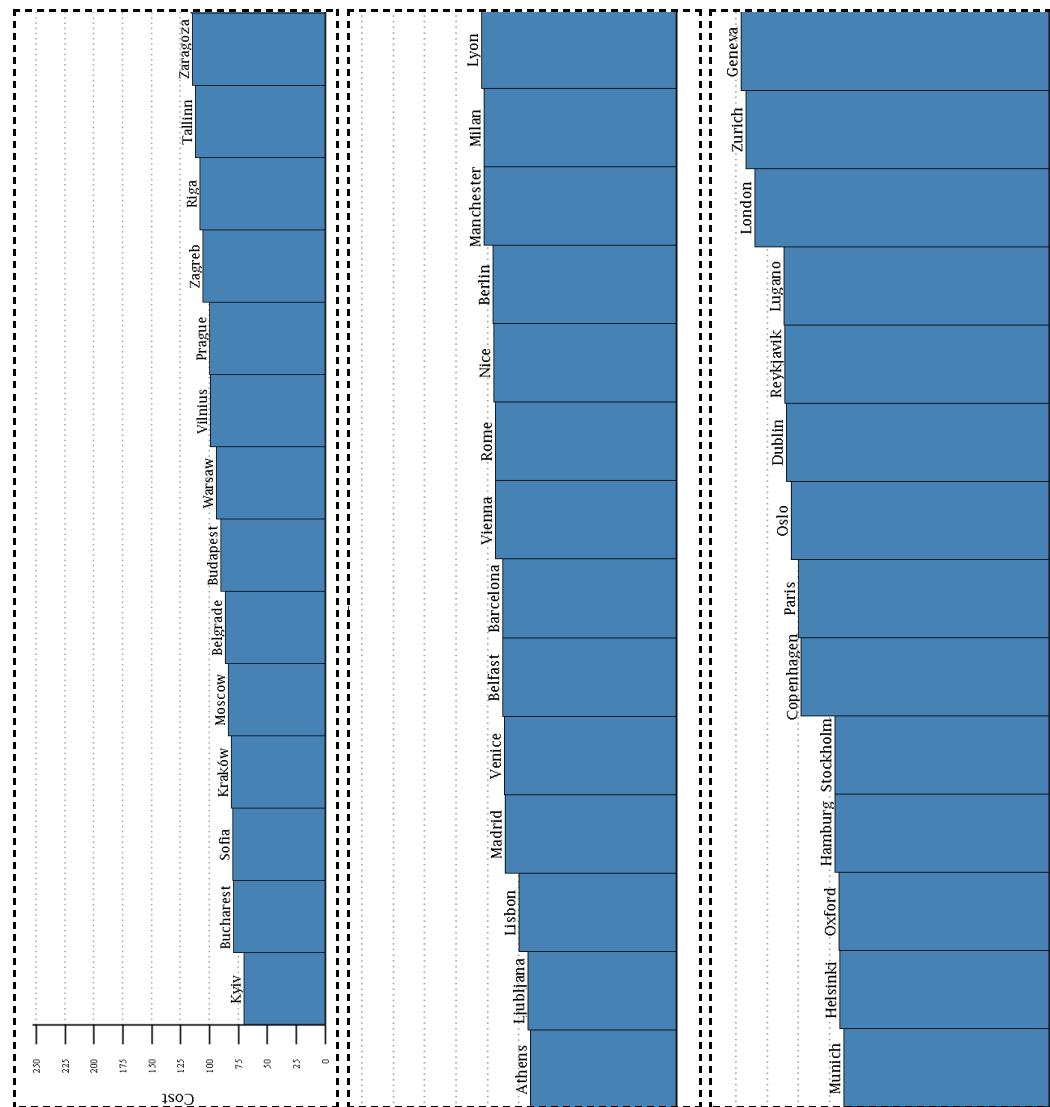


Порівнюючи з результатами аналогічної процедури методом K-means, можна зауважити, що кластеризація методом BIRCH має більш рівномірний розподіл. Наприклад, за тим же фактором K-means побудував два кластера, у яких опинилося лише по одному елементу.

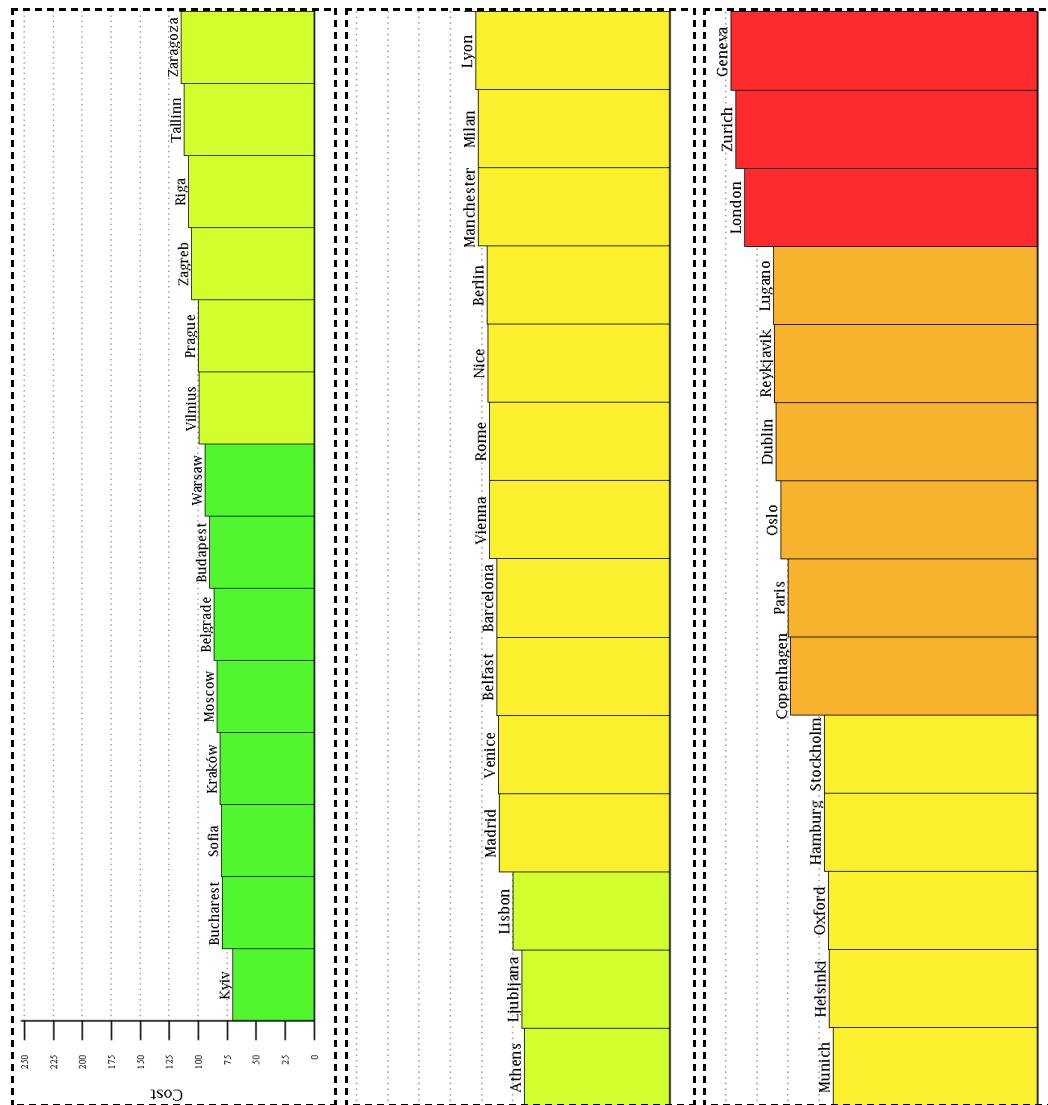
"Cluster #1"	"Points : 12"	"Borders : from Kyiv (376 hr.) to Athens (1011 hr.)"	"Difference : 635 hr."
"Cluster #2"	"Points : 12"	"Borders : from Nice (1138 hr.) to Riga (1658 hr.)"	"Difference : 520 hr."
"Cluster #3"	"Points : 7"	"Borders : from Madrid (1875 hr.) to Belfast (2272 hr.)"	"Difference : 397 hr."
"Cluster #4"	"Points : 9"	"Borders : from Berlin (2508 hr.) to Zurich (3251 hr.)"	"Difference : 743 hr."
"Cluster #5"	"Points : 2"	"Borders : from Dublin (4353 hr.) to London (5546 hr.)"	"Difference : 1193 hr."

3.4.3. Індекс вартості проживання

Початкові дані, відображені у вигляді гістограми



Результати методу BIRCH



"Cluster #1"	"Points : 8"	"Borders : from Kyiv (70 hr.) to Warsaw (94 hr.)"	"Difference : 24 hr."
"Cluster #2"	"Points : 9"	"Borders : from Vilnius (99 hr.) to Lisbon (125 hr.)"	"Difference : 26 hr."
"Cluster #3"	"Points : 16"	"Borders : from Madrid (136 hr.) to Stockholm (171 hr.)"	"Difference : 35 hr."
"Cluster #4"	"Points : 6"	"Borders : from Copenhagen (198 hr.) to Lugano (212 hr.)"	"Difference : 14 hr."
"Cluster #5"	"Points : 3"	"Borders : from London (235 hr.) to Geneva (246 hr.)"	"Difference : 11 hr."

На відміну від результатів дослідження вартості оренди, результати кластеризації за індексом вартості проживання методом BIRCH повністю співпадають з результатами K-means за тим же показником.

РОЗДІЛ IV. МЕТОД СПЕКТРАЛЬНОЇ КЛАСТЕРИЗАЦІЇ

4.1. Загальний опис методу

Розглянуті вище методи є досить зручними, кожен має свої переваги.

Попри все, у K-means та BIRCH є вагомі недоліки:

- не розрізняють кластери, що перетинаються;
- не можуть виокремити множини точок, що утворюють складні форми;

Розглянемо такий приклад.

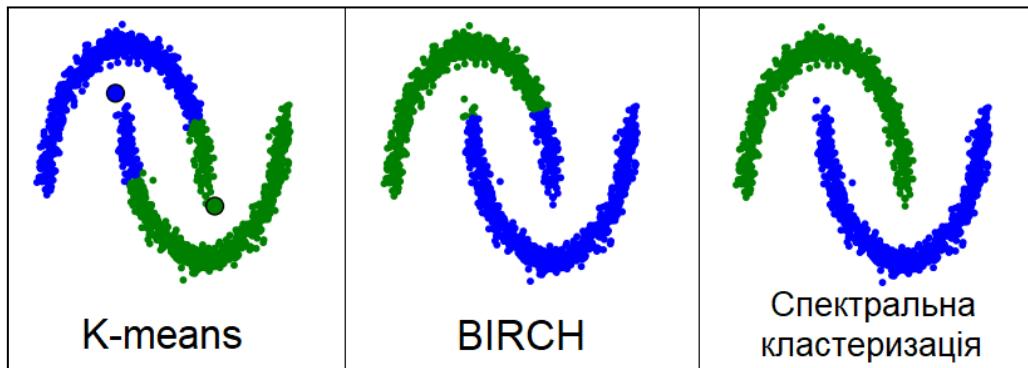


Рисунок 1

На рисунку 1 бачимо, що K-means та BIRCH виконали свою роботу так, як і повинні були. Але, вочевидь, для людини таке групування було б дивним, адже можна чітко виокремити дві групи даних вигляду «підков».

У свою чергу, спектральна кластеризація дає змогу розпізнати певні спільні властивості точок і згрупувати їх так, як цього вимагає здоровий глузд. Такий результат досягається за рахунок побудови графа G , що відображає схожість точок даних, і використання інформації про спектр матриці Кірхгофа цього графа.

4.2. Математичний опис методу

4.2.1. Граф подібності

Нехай дано деякий набір значень x_1, x_2, \dots, x_n та відомі міри подібності $w_{ij} \geq 0$ між кожною парою x_i та x_j . Тоді для того, щоб було зручно поділити задану множину точок за їхньою схожістю, можна представити дані у вигляді *графа подібності* $G = (V, E)$. Кожна вершина v_i

такого графа відповідає певній точці x_i , та, якщо між двома точками даних x_i і x_j міра подібності w_{ij} додатна або перевищує певний поріг, то відповідні їм вершини графа з'єднані ребром із вагою w_{ij} . У такому випадку задача зводиться до пошуку такого розділення графа G , щоб вага ребер між різними групами була відносно низькою, а всередині груп відносно високою.

4.2.2. Введення в термінологію

Нехай $G = (V, E)$ – неорієнтований граф з множиною вершин $V = \{v_1, \dots, v_n\}$. Окрім того, G – зважений, тобто кожне ребро між вершинами графа має невід’ємну вагу w_{ij} . Тоді для нього можна визначити зважену матрицю суміжності вигляду $W = (w_{ij})_{i,j=1, \dots, n}$, назовемо її *матрицею подібності*. Зрозуміло, що $w_{ij} = 0$, коли відповідні вершини не зв’язані, та $w_{ij} = w_{ji}$, бо граф – неорієнтований.

Степінь d_i вершини v_i визначається наступним чином:

$$d_i = \sum_{j=1}^n w_{ij}.$$

Матриця степенів D – це діагональна матриця зі значеннями d_1, \dots, d_n .

Також введемо поняття *індикаторного вектора* $1_A = (f_1, f_2, \dots, f_n)'$ $\in \Re^n$ – це вектор, в якому $f_i = 1$, якщо $v_i \in A$, інакше $f_i = 0$.

4.2.3. Як побудувати граф подібності?

Для того, щоб перетворити множину точок даних x_1, x_2, \dots, x_n зі заданими мірами подібності у граф подібності, існує декілька підходів.

Граф ε -сусідів (англ. the ε -neighborhood graph). Ідея полягає в з’єднанні тих точок, відстані між якими менші за ε . Зазвичай після таких маніпуляцій міри подібності між пов’язаними точками відрізняються не суттєво, і вага

ребер вже не несе важливої інформації для нас. Тому здебільшого такий граф розглядається як незважений.

Граф k -найближчих сусідів (англ. k -nearest neighbor graphs). У цьому випадку ми з'єднуємо вершину v_i з вершиною v_j , якщо остання – серед k найближчих сусідів v_i . Однак, в такому випадку відношення сусідства не є симетричним, і, відповідно, граф G буде орієнтованим. Для того, щоб G став неорієнтованим, існує два підходи:

- 1) поєднати вершини, між якими існує хоча б одне орієнтоване ребро;
- 2) поєднати вершини, які одночасно з'єднані одна з одною.

Останній варіант називається граф взаємних k -найближчих сусідів (англ. mutual k -nearest neighbor graph).

Повністю зв'язаний граф. Це випадок із поєднанням усіх вершин графа, між якими міра подібності є більшою за нуль та ребрами із відповідними значеннями ваги. Але в такому разі завдання побудувати вказаний граф таким чином, щоб можна було виокремити певні локальні групи сусідства, покладається на функцію *подібності* – функцію, яка буде визначати наскільки точки подібні одна до одної.

4.2.4. Матриця Кірхгофа

Основним інструментом, що використовується в спектральній кластеризації, є матриця Кірхгофа, а саме три її варіанти: звичайна ненормалізована, симетрично-нормалізована та нормалізована матриця Кірхгофа випадкового блукання.

Надалі будемо вважати, що граф G є неорієнтованим та зваженим із матрицею подібності W ($w_{ij} = w_{ji} \geq 0$). Okрім цього, коли говорять про «перші k власних векторів», то мають на увазі власні вектори, що відповідають k найменшим власним значенням.

Ненормалізована матриця Кірхгофа

Звичайний випадок матриці Кірхгофа виглядає наступним чином:

$$L = D - W.$$

Розглянемо деякі її властивості, які будуть використані далі.

1. Для будь-якого вектора $f \in \mathbb{R}^n$ виконується

$$f' L f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2.$$

Доведення. За визначенням d_i маємо, що

$$\begin{aligned} f' L f &= f' D f - f' W f = \sum_{i=1}^n d_i f_i^2 - \sum_{i,j=1}^n f_i f_j w_{ij} \\ &= \frac{1}{2} \left(\sum_{i=1}^n d_i f_i^2 - 2 \sum_{i,j=1}^n f_i f_j w_{ij} + \sum_{j=1}^n d_j f_j^2 \right) = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2. \end{aligned}$$

2. L – симетрична та додатно визначена матриця.

Доведення. Симетрія L безпосередньо випливає із симетрії W та D .

Друга частина твердження випливає з пункту 1: $f' L f \geq 0 \forall f \in \mathbb{R}^n$.

3. Найменше власне число матриці дорівнює нулю. Доведення очевидно.
4. Існує n невід'ємних дійсних власних значень $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

Доведення слідує як наслідок з попередніх пунктів.

Твердження 1. *Нехай G – неорієнтований зважений граф, що має невід'ємні значення вагових коефіцієнтів. Тоді кратність k власного числа матриці Кірхгофа графа G , що дорівнює нулю, співпадає з кількістю компонент зв'язності графа G .*

Доведення. Розглянемо випадок, коли $k = 1$. Припустимо, що f – це власний вектор, який відповідає власному числу, що дорівнює нулю. Тоді

$$0 = f' L f = \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2.$$

Оскільки $w_{ij} \geq 0$, то ця сума може дорівнювати нулю лише тоді, коли кожен доданок $w_{ij}(f_i - f_j)^2$ дорівнює нулю. Але, якщо дві вершини v_i та v_j з'єднані ребром, тобто $w_{ij} > 0$, то f_i має дорівнювати f_j . Звідси випливає, що елементи вектора f мають бути однаковими для всіх вершин, зв'язаних певним шляхом, тобто для усього компоненту зв'язності.

Тепер розглянемо випадок, коли $k > 1$. Припустимо, що вершини впорядковані згідно з компонентами зв'язності, до яких належать. У цьому випадку матриця подібності W та, відповідно, матриця Кірхгофа L мають квазідіагональну форму.

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}$$

Кожен блок L_i матриці L є матрицею Кірхгофа, що відповідає i -тій компоненті зв'язності. Тоді дляожної L_i існує нульове власне число кратності 1, та відповідний йому власний вектор має одинакові значення елементів, якщо ті відносяться до i -ї компоненти зв'язності. Звідси випливає, що матриця L має стільки нульових власних чисел, скільки наявно компонентів зв'язності, та відповідні власні вектори можуть слугувати індикаторними векторами для своїх компонентів зв'язності.

Нормалізовані матриці Кірхгофа

В цьому пункті розглянемо обидва варіанти нормалізованих матриць Кірхгофа, адже вони тісно пов'язані один з одним.

Отже, симетрично-нормалізована L_{sym} та нормалізована матриця Кірхгофа випадкового блукання L_{rw} (від англ. *random walk*) мають наступний вигляд:

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W.$$

Опишемо важливі властивості цих матриць.

1. Для будь-якого вектора $f \in \Re^n$ виконується

$$f' L_{\text{sym}} f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2.$$

Доведення аналогічно до пункту 1 для ненормалізованої матриці.

2. λ – власне число L_{rw} з відповідним власним вектором u тоді і тільки

тоді, коли λ – власне число L_{sym} з власним вектором $w = D^{1/2} u$.

Для доведення достатньо розглянути рівняння $L_{\text{sym}} w = \lambda w$,

помножити його на $D^{1/2}$ зліва та зробити заміну $u = D^{-1/2} w$.

3. λ – власне число L_{rw} з власним вектором u тоді і тільки тоді, коли λ та u

вирішують узагальнену проблему власних значень $Lu = \lambda Du$.

Для доведення треба розглянути рівняння $L_{\text{rw}} u = \lambda u$, помножити його

на D зліва та скористатися означенням $L_{\text{rw}} := D^{-1} L$.

4. L_{sym} та L_{rw} мають нульові власні числа та відповідні власні вектори w та $D^{1/2} w$.

5. L_{sym} та L_{rw} – додатно визначені та мають n невід'ємних дійсних власних чисел $0 = \lambda_1 \leq \dots \leq \lambda_n$.

Доведення слідують з пунктів 1 та 2.

Твердження 2. *Нехай G – неорієнтований зважений граф, що має невід'ємні значення вагових коефіцієнтів. Тоді кратність k нульового власного числа матриць L_{sym} та L_{rw} співпадає з кількістю компонент зв'язності графа G .*

Доведення є аналогічним до доведення твердження 1.

4.2.5. Головна ідея алгоритму

Розглянемо тривіальний приклад. Нехай задано умову, в якій кожній із 15 назв міст у відповідність поставлено деяке число:

1	Reykjavik	111
2	Dublin	123
3	London	124
4	Paris	121
5	Madrid	241
6	Barcelona	344
7	Nice	342
8	Lyon	345
9	Geneva	342
10	Zurich	342
11	Budapest	987
12	Zagreb	908
13	Ljubljana	999
14	Lugano	908
15	Milan	967

Використовуючи функцію подібності (в нашему випадку це гаусівська функція схожості), формуємо матрицю подібності. Далі за одним із описаних вище способів будуємо граф G , який пов'язує точки даних за рівнем схожості. В нашему випадку було використано принцип k -найближчих сусідів, де $k = 3$. В результаті отримаємо 3 компоненти зв'язності.

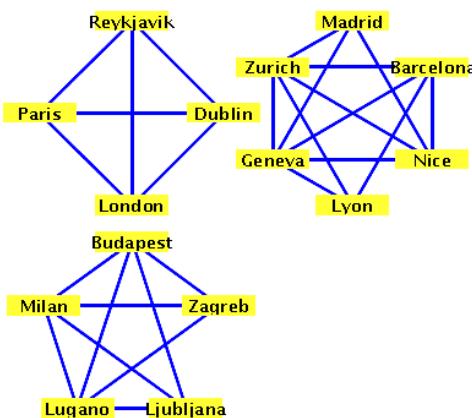


Рисунок 2

На рисунку 2 бачимо, що в результаті побудови графа подібності сформувалися 3 компоненти зв'язності.

Будуємо матрицю Кірхгофа для графа G (в нашему випадку це ненормалізована матриця Кірхгофа), визначаємо її власні числа та відповідні власні вектори. Отримаємо наступне:

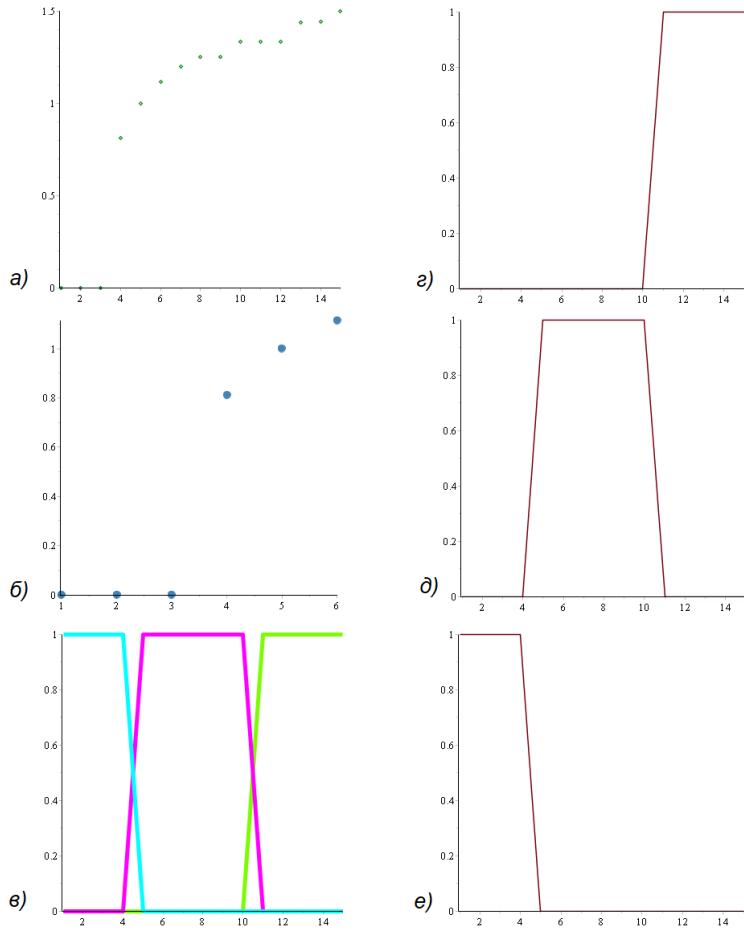
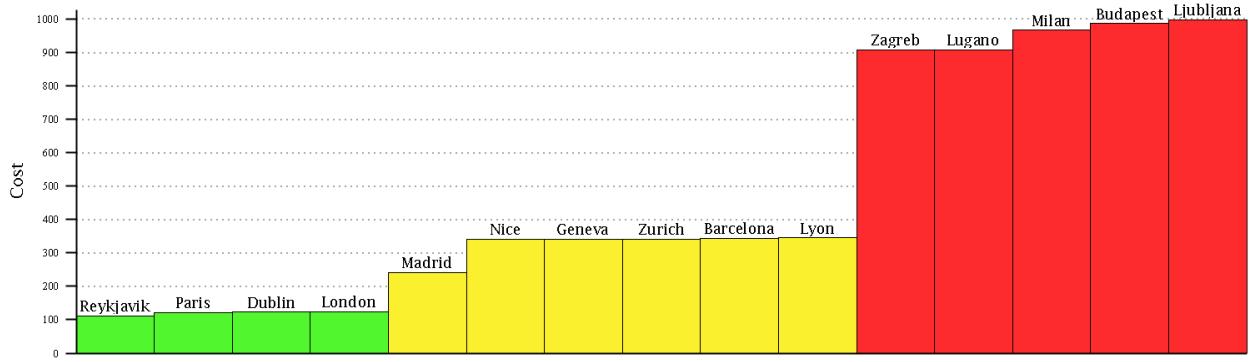


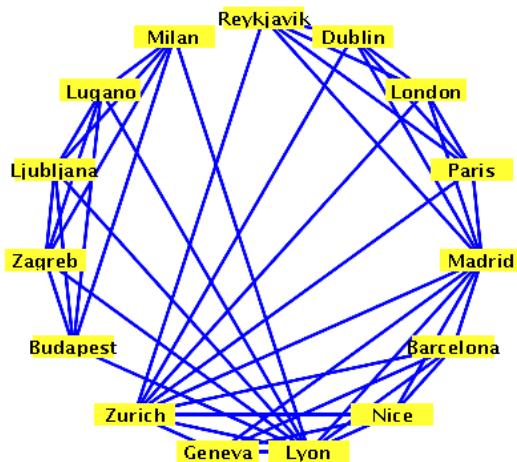
Рисунок 3

На рисунках 3.а) та 3.б) відображені відсортовані за зростанням усі власні числа та перші 6 із них, відповідно. Зауважимо, що кратність нульового власного числа співпадає з кількістю компонентів зв'язності. На наступних рисунках зображені власні вектори, що відповідають нульовому власному числу (усі разом та окремо кожен для наглядності).

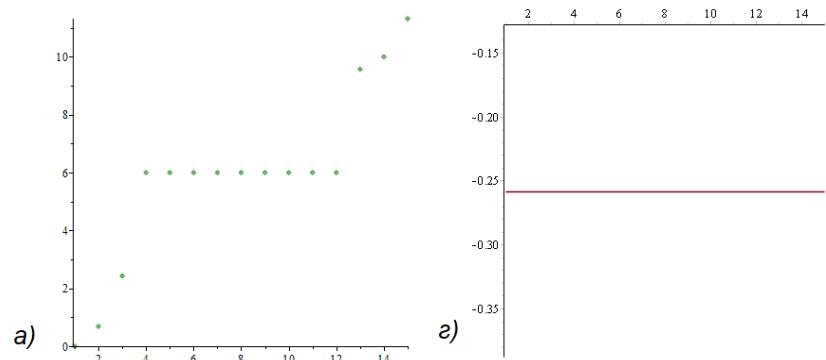
Бачимо, що за отриманими значеннями власних векторів можна чітко виокремити певні множини точок. Отже, використовуючи вже реалізовані методи класифікації (наприклад, K-means) можемо поділити множини початкових точок даних на кластери. Результати роботи методу K-means продемонстровані на рисунку:

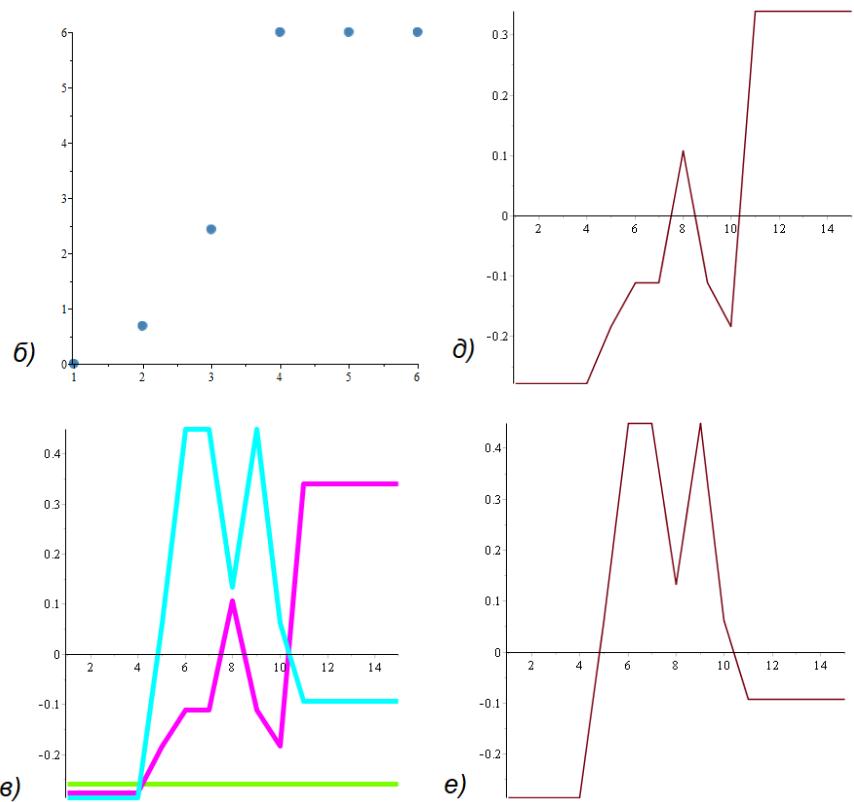


Тепер побудуємо граф G за принципом k -найближчих сусідів, але тепер $k = 5$. В результаті отримаємо одну компоненту зв'язності.



Аналогічно проробляємо всі подальші кроки та отримуємо:



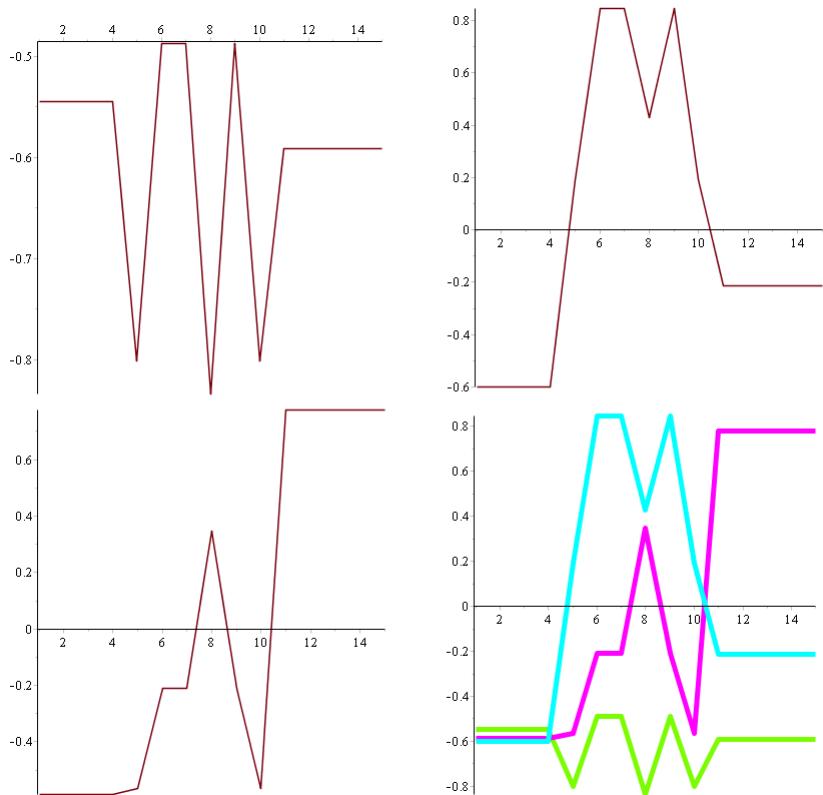


Нульове власне число має кратність 1, бо присутня одна компонента зв'язності, та відповідний їй власний вектор має однакові елементи.

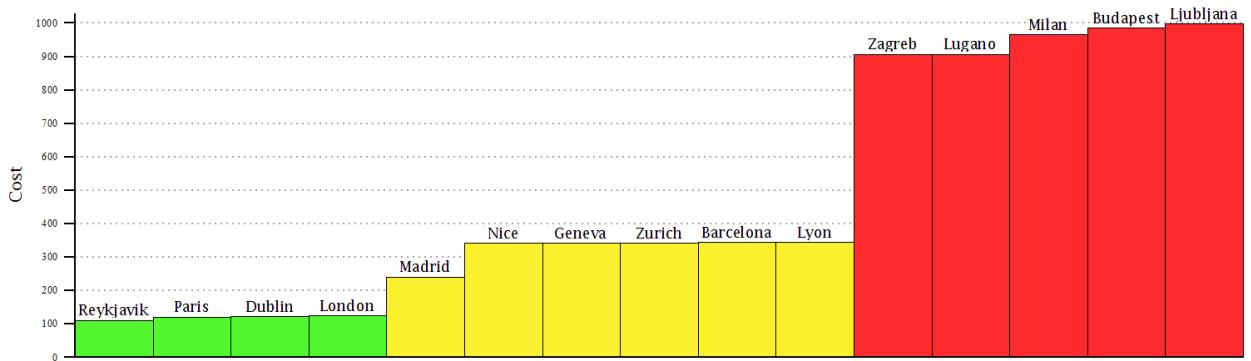
Бачимо, що навіть коли в нас одна компонента зв'язності, за другим і третьим власними векторами можна визначити кластери:



Результати роботи методу K-means, які зображені на рисунку вище, не дуже втішні. Для їхнього покращення використаємо нормалізацію, яка буде описана нижче у п. 4.3.2. Отримаємо наступні графіки, за якими можна отримати більш коректні результати кластеризації:



Після роботи методу K-means над множиною точок після процедури нормалізації отримуємо результат:



4.3. Структура алгоритму

4.3.1 Підготовчі питання

Вище було описано декілька розгалужених шляхів, що приводять до розв'язання задачі спектральної кластеризації. Наразі треба визначитись, якою саме “стежкою” ми підемо.

Функція подібності

Перед тим, як конструювати граф подібності, треба визначитись з функцією, що визначає цю подібність. Зрозуміло, що граф має відображати «сусідство» точок даних настільки, наскільки ці дані «схожі» за своєю оригінальною природою. Найбільш відомою функцією, що може нам послугувати, є гаусівська функція подібності

$$w(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / 2\sigma^2).$$

Граф схожості

За рекомендацією [11] в якості графа схожості я обрав граф k -найближчих сусідів. З ним зручно працювати, в результаті отримаємо розріджену матрицю суміжності (саме суміжності, бо подібність нам вже не потрібна). Окрім цього, такий граф менш вразливий до недостатньо добре обраних параметрів, ніж ε -сусідів (залежить від параметру ε) або повністю зв'язаний (залежить від параметрів функції подібності).

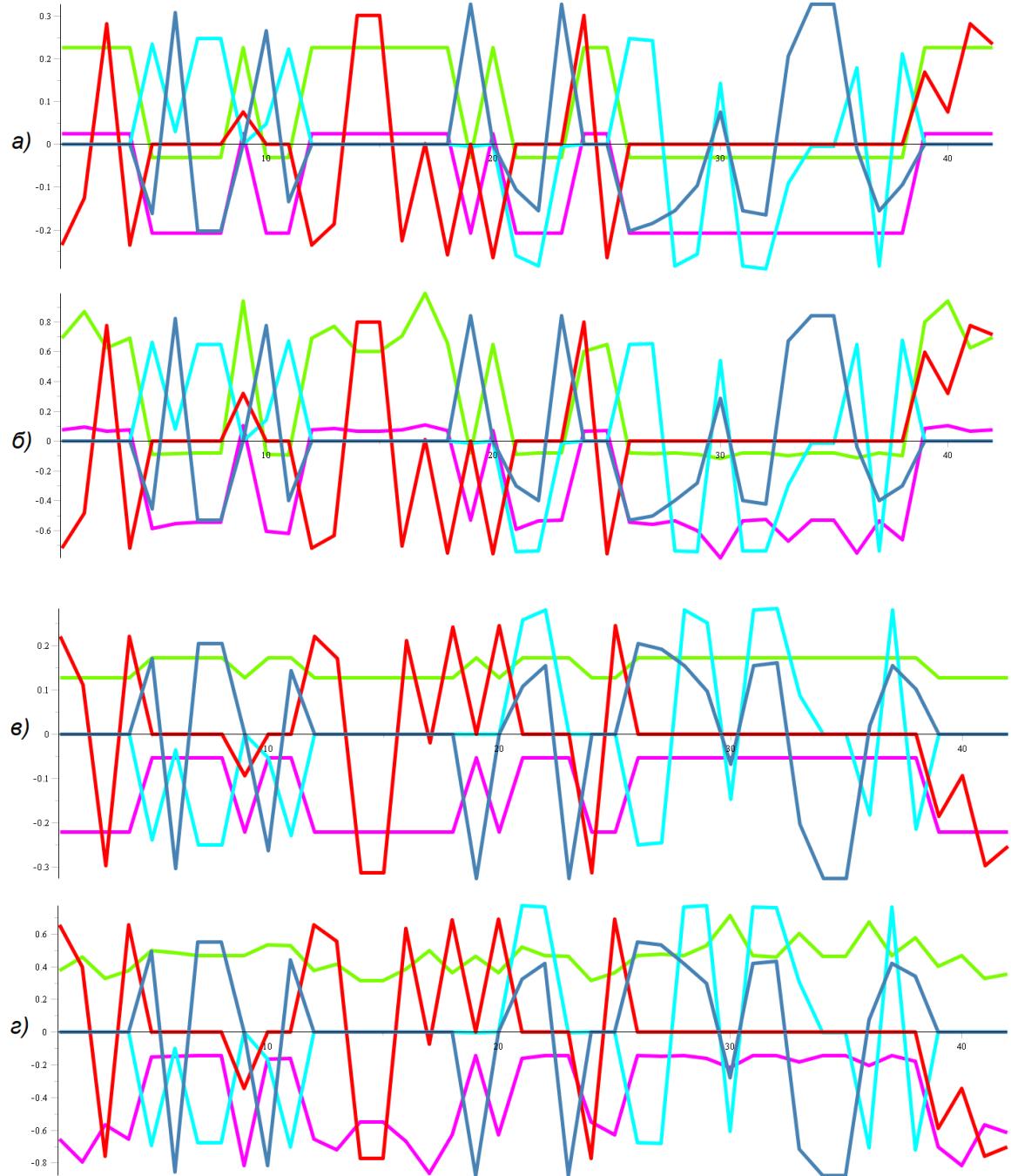
Значення параметру k буде обиратися під час роботи алгоритму таким чином, щоб кількість компонентів зв'язності була значно меншою за кількість шуканих кластерів.

Матриця Кірхгофа

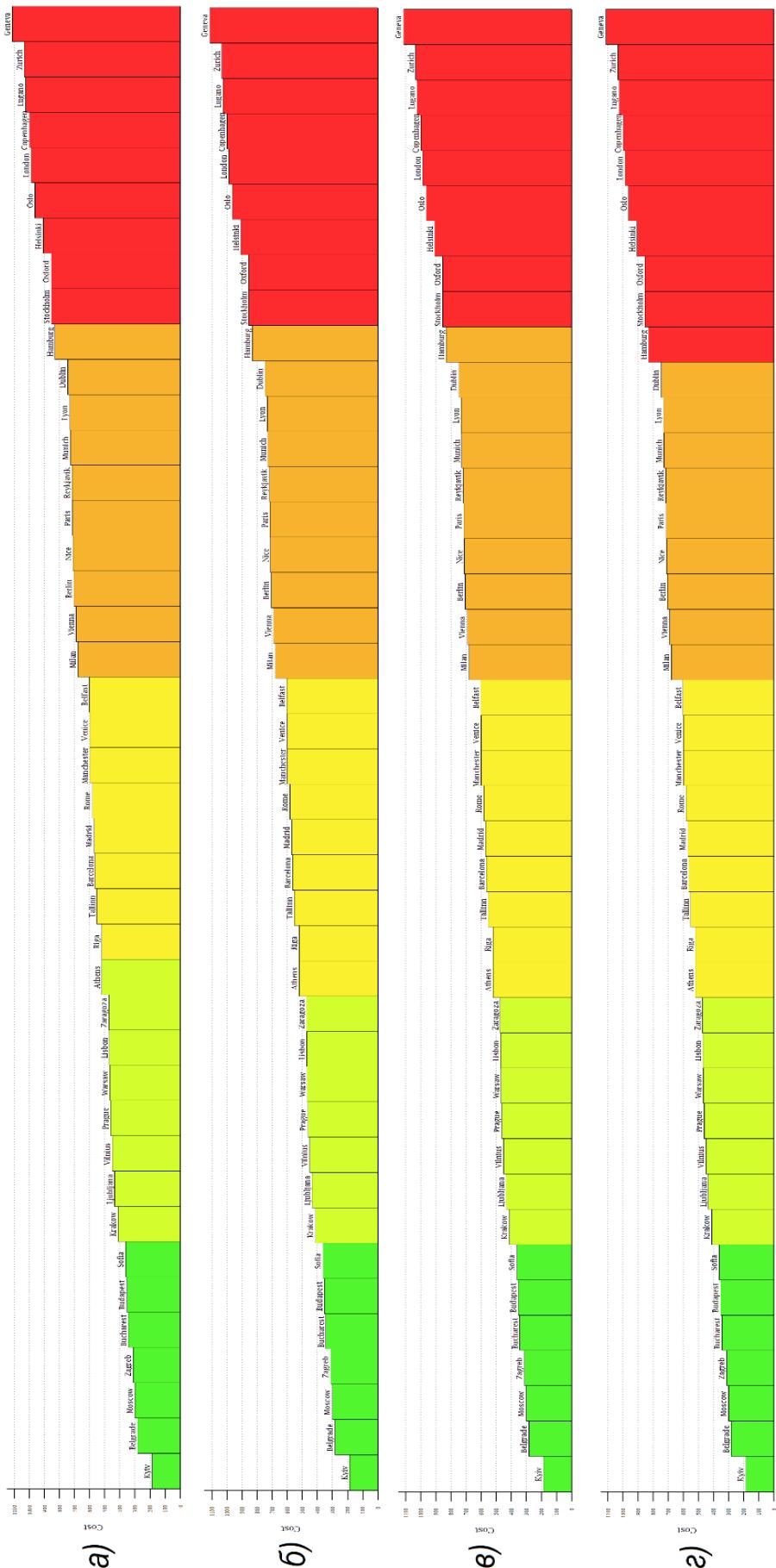
Питання «Яку матрицю Кірхгофа обрати?» не має великого значення, адже набір точок, що розглядається, досить малий. Але, на практиці було визначено, що найкращі результати кластеризації можна отримати, використовуючи матрицю Кірхгофа випадкового блукання L_{rw} . Далі для прикладу наведено роботу спектральної кластеризації з використанням:

- a) ненормалізованої матриці Кірхгофа без нормалізації власних векторів,
- б) ненормалізованої матриці Кірхгофа та нормалізації власних векторів,
- в) нормалізованої матриці Кірхгофа L_{rw} без нормалізації власних векторів,
- г) нормалізованої матриці Кірхгофа L_{rw} та нормалізації власних векторів.

В якості вхідних даних було взято інформацію про середню вартість 2 квитків у кіно. Кількість кластерів рівна 5. Перші 5 власних векторів відображені на наступних рисунках:



Відповідні результати після роботи K-means продемонстровано далі:



Різниця між результатами не достатньо велика, але вона присутня:

- варіант *a*) відрізняється від усіх тим, що місто Афіни, яке за значенням ціни квитка у 522 грн, було угруповано разом з містом Сарагоса (475 грн). Але очевидно, що місто Афіни набагато ближче саме до Ріги (523 грн). У варіантах *b), e)* та *g)* Афіни вже перебуває в третьому кластері, як і має бути.
- варіант *g)*, на відміну від усіх інших, відокремив Гамбург (832 грн) від Дубліна (749 грн), які очевидно менш подібні за своїми значеннями, ніж Гамбург та Стокгольм (856 грн).

a)	"Cluster #1" "Points : 7"	"Borders : from Kyiv (188 hr.) to Sofia (364 hr.)"	"Difference : 176 hr."
	"Cluster #2" "Points : 8"	"Borders : from Krakow (414 hr.) to Athens (522 hr.)"	"Difference : 108 hr."
	"Cluster #3" "Points : 8"	"Borders : from Riga (523 hr.) to Belfast (604 hr.)"	"Difference : 81 hr."
	"Cluster #4" "Points : 10"	"Borders : from Milan (682 hr.) to Hamburg (832 hr.)"	"Difference : 150 hr."
	"Cluster #5" "Points : 9"	"Borders : from Stockholm (856 hr.) to Geneva (1116 hr.)"	"Difference : 260 hr."

b)-e)	"Cluster #1" "Points : 7"	"Borders : from Kyiv (188 hr.) to Sofia (364 hr.)"	"Difference : 176 hr."
	"Cluster #2" "Points : 7"	"Borders : from Krakow (414 hr.) to Zaragoza (475 hr.)"	"Difference : 61 hr."
	"Cluster #3" "Points : 9"	"Borders : from Athens (522 hr.) to Belfast (604 hr.)"	"Difference : 82 hr."
	"Cluster #4" "Points : 10"	"Borders : from Milan (682 hr.) to Hamburg (832 hr.)"	"Difference : 150 hr."
	"Cluster #5" "Points : 9"	"Borders : from Stockholm (856 hr.) to Geneva (1116 hr.)"	"Difference : 260 hr."

e)	"Cluster #1" "Points : 7"	"Borders : from Kyiv (188 hr.) to Sofia (364 hr.)"	"Difference : 176 hr."
	"Cluster #2" "Points : 7"	"Borders : from Krakow (414 hr.) to Zaragoza (475 hr.)"	"Difference : 61 hr."
	"Cluster #3" "Points : 9"	"Borders : from Athens (522 hr.) to Belfast (604 hr.)"	"Difference : 82 hr."
	"Cluster #4" "Points : 9"	"Borders : from Milan (682 hr.) to Dublin (749 hr.)"	"Difference : 67 hr."
	"Cluster #5" "Points : 10"	"Borders : from Hamburg (832 hr.) to Geneva (1116 hr.)"	"Difference : 284 hr."

Отже, бачимо, що найкраще спрацював саме останній варіант, тобто використання нормалізованої матриці Кірхгофа випадкового блукання та нормалізація випадкових векторів дало найбільш коректний результат.

4.3.2. Алгоритм

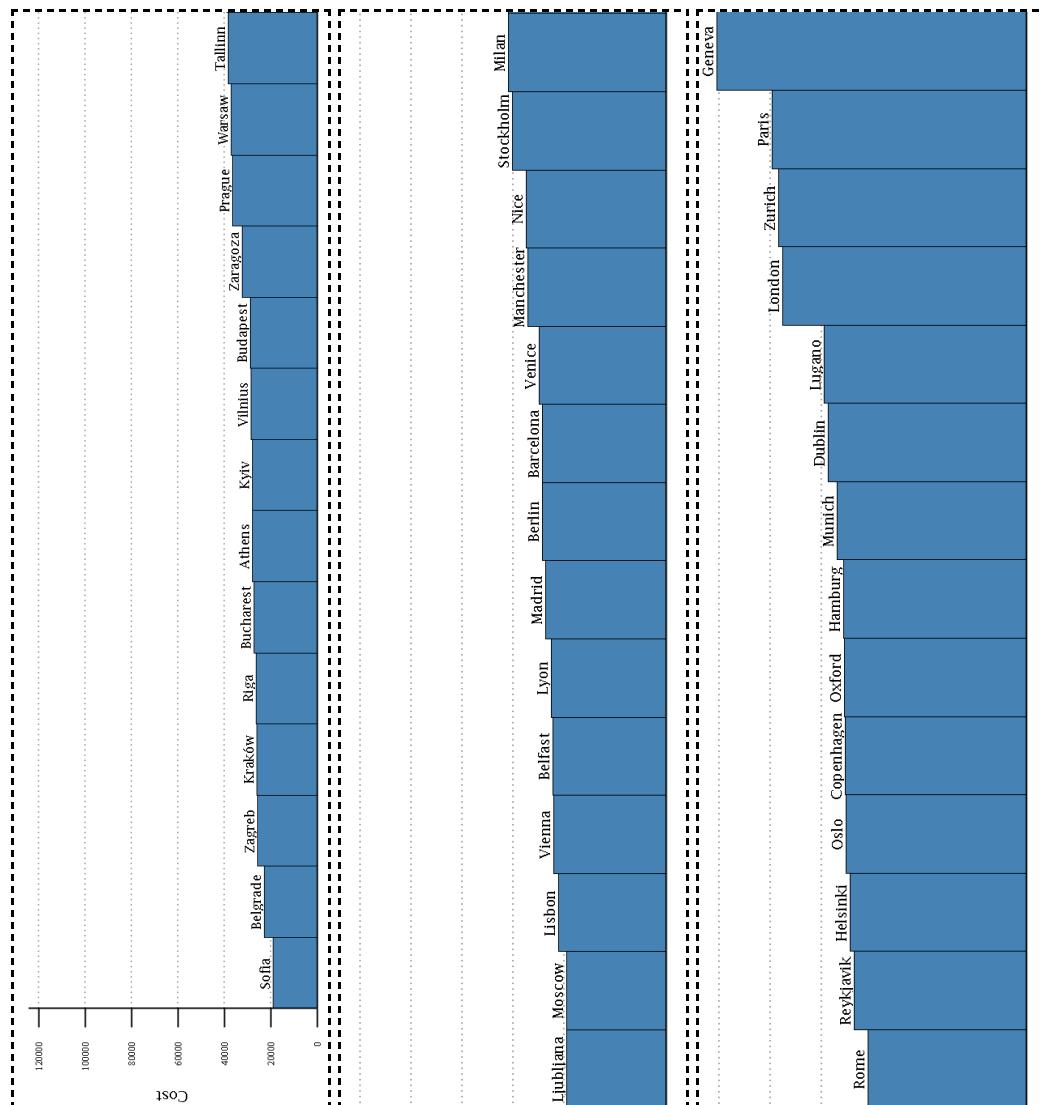
1. Будуємо матрицю подібності, використовуючи гаусівську функцію схожості.
2. Будуємо граф подібності за принципом *k*-найближчих сусідів.

3. Будуємо матрицю Кірхгофа L , L_{sym} або L_{rw} .
4. Визначаємо перші k власних вектора u_1, \dots, u_k матриці Кірхгофа.
5. Формуємо нову множину точок для подальшої кластеризації.
 - a. Нехай $U \in \Re^{n \times k}$ – матриця, колонками якої є вектори u_1, \dots, u_k .
Тоді, для $i = 1, \dots, n$ вектор $y_i \in \Re^k$, що відповідає i -му рядку матриці U , буде координатою відповідної нової точки.
 - b. Нормалізуємо координати нових точок: $u_{ij} = \frac{u_{ij}}{\left(\sum_k (u_{ik})^2\right)^{1/2}}$.
6. Кластеризуємо точки $(y_i)_{i=1..n}$ за допомогою методу K-means.

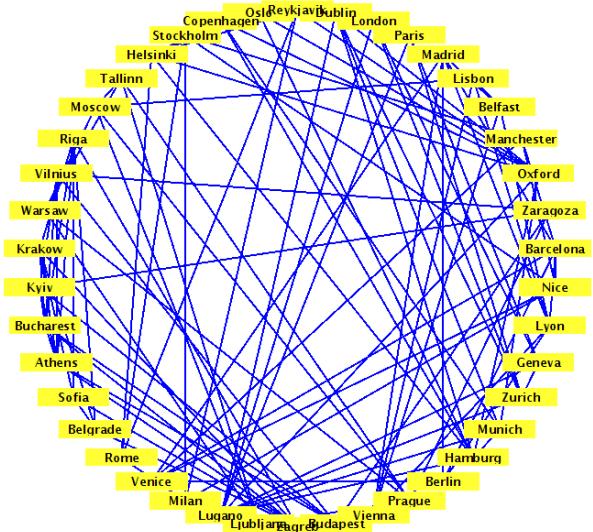
4.4. Робота методу

Робота спектральної кластеризації буде продемонстрована на розбитті множини міст за величиною місячної оренди 85 m^2 мебльованого житла в дорогому районі, місячною вартістю квитка на міський транспорт та за величиною індексу вартості проживання. Алгоритм був реалізований за допомогою програмного забезпечення Maple з підтримкою бібліотек Python. Обрані дані для вибірки див. у Додатку А.

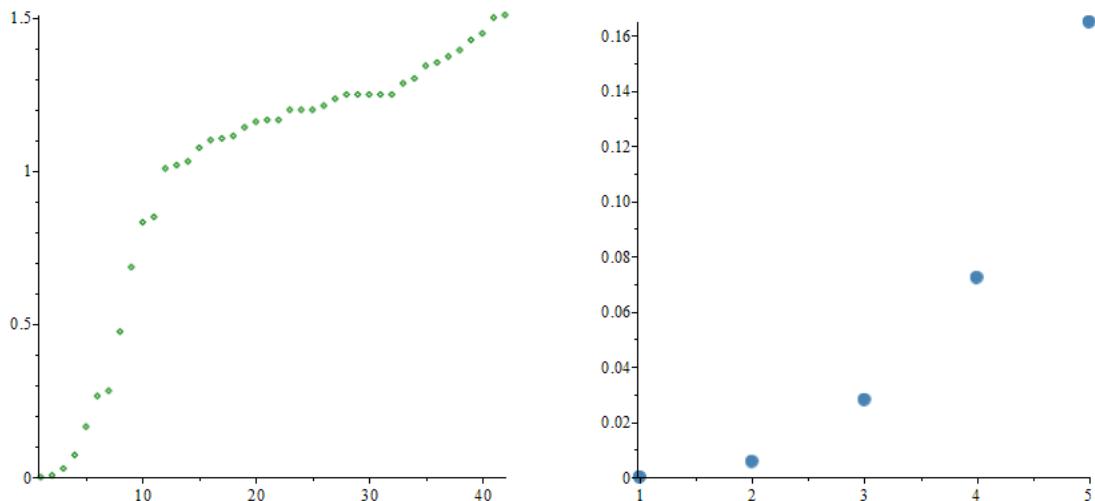
4.4.1. Місячна оренда 85 м² мебльованого житла в дорогому районі Початкові дані, відображені у вигляді гістограми



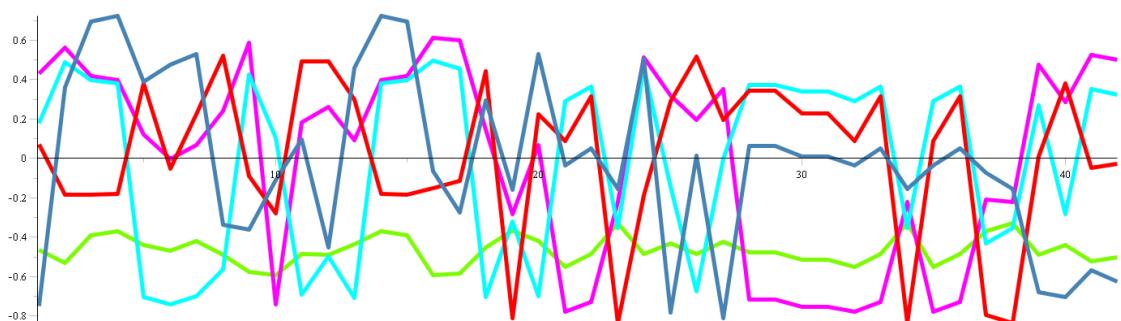
Процес роботи: граф подібності, власні числа та вектори



Граф подібності з параметром $k = 4$

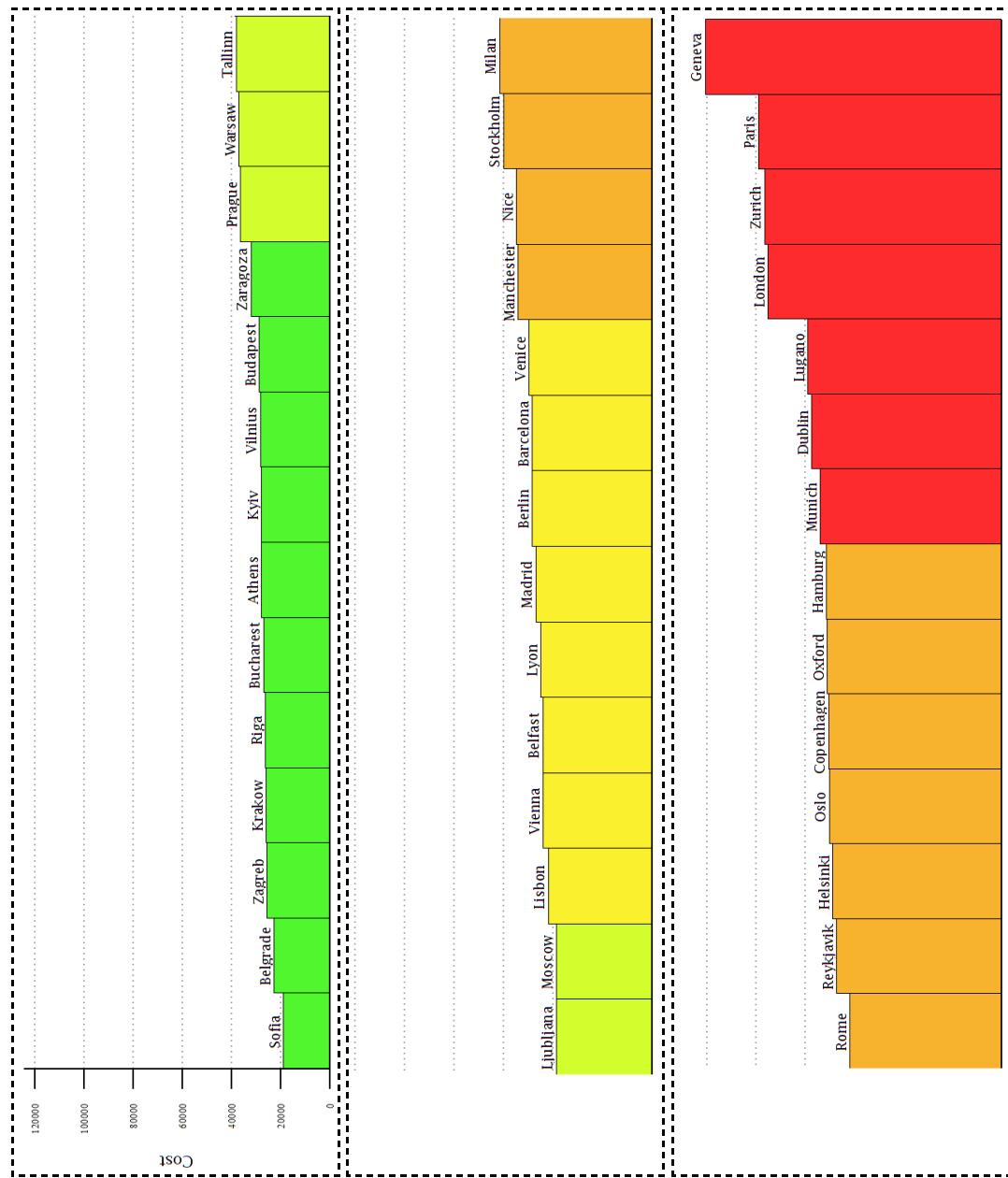


Усі та перші 5 власних значень відповідно



Перші 5 власних векторів

Результати роботи спектральної кластеризації



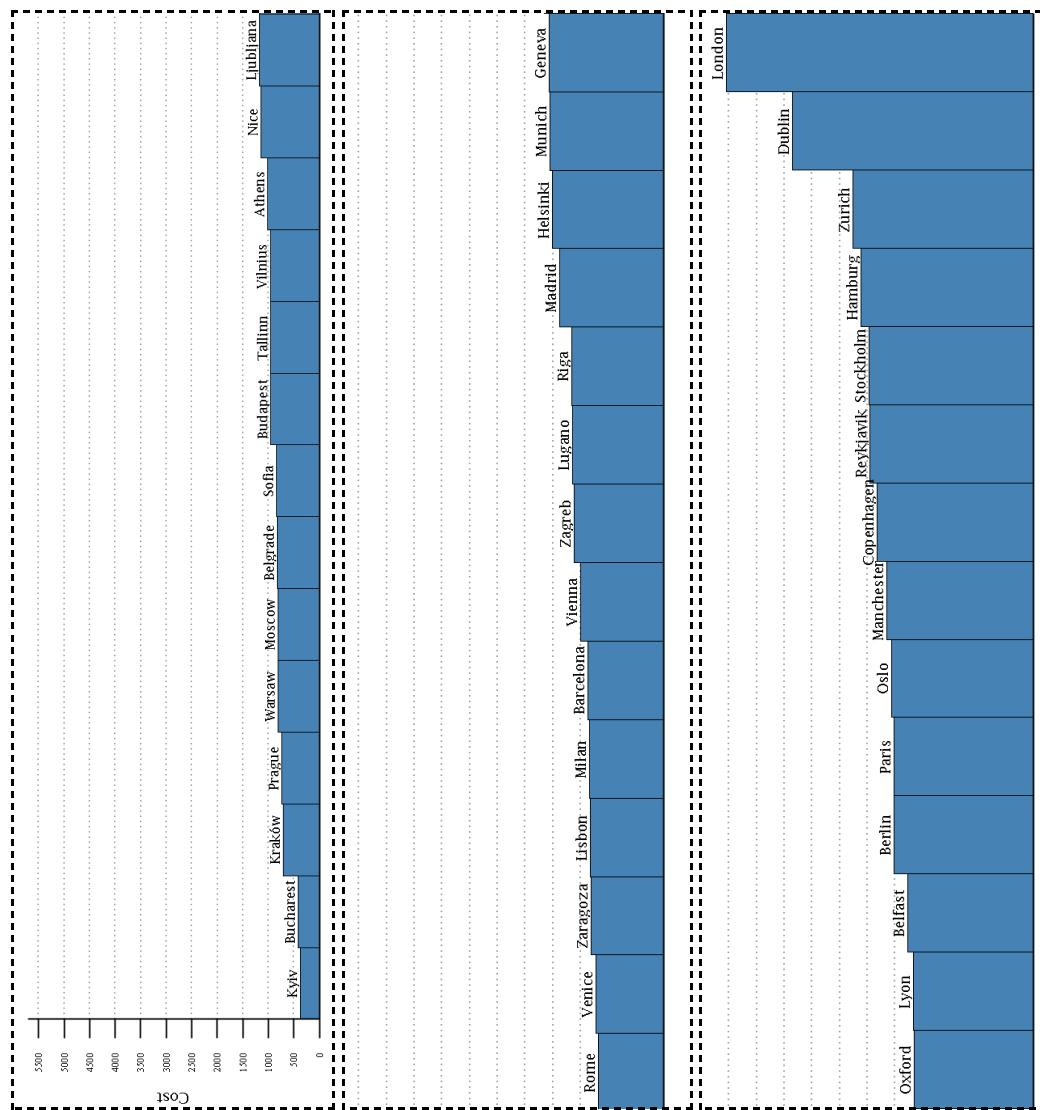
"Cluster #1" "Points : 11"	"Borders : from Sofia (18905 hr.) to Zaragoza (32162 hr.)"	"Difference : 13257 hr."
"Cluster #2" "Points : 5"	"Borders : from Prague (36486 hr.) to Moscow (38868 hr.)"	"Difference : 2382 hr."
"Cluster #3" "Points : 8"	"Borders : from Lisbon (42061 hr.) to Venice (49814 hr.)"	"Difference : 7753 hr."
"Cluster #4" "Points : 11"	"Borders : from Manchester (54266 hr.) to Hamburg (71284 hr.)"	"Difference : 17018 hr."
"Cluster #5" "Points : 7"	"Borders : from Munich (73939 hr.) to Geneva (120788 hr.)"	"Difference : 46849 hr."

До першого кластеру було кластеризовано ті ж міста, що й були наведені в результатах роботи K-means та BIRCH. Але надалі спектральна кластеризація показує більш рівномірний розподіл: другий, третій, четвертий та п'ятий кластери включають у себе 5, 8, 11 та 7 точок відповідно, у той час

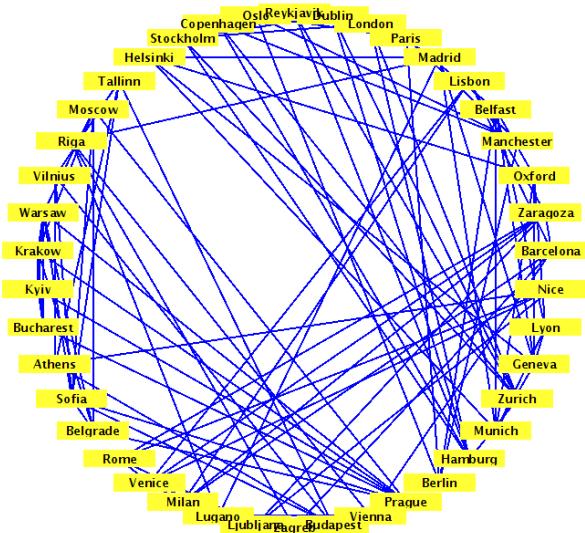
як K-means та BIRCH ті ж точки поділили у співвідношенні 15/12/3/1 та 13/5/9/4 відповідно. На мою думку, такий розподіл пов'язаний з методом побудови графа G: якби було використано метод ε -сусідів або повного пов'язання графу зі збереженням вагових коефіцієнтів, то результати були б більш схожі до презентованих BIRCH-кластеризацією.

4.4.2. Місячний квиток на міський транспорт

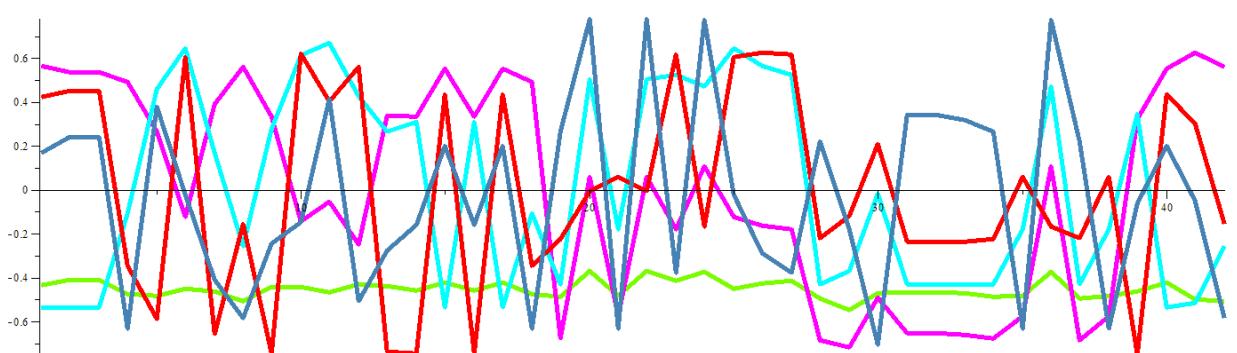
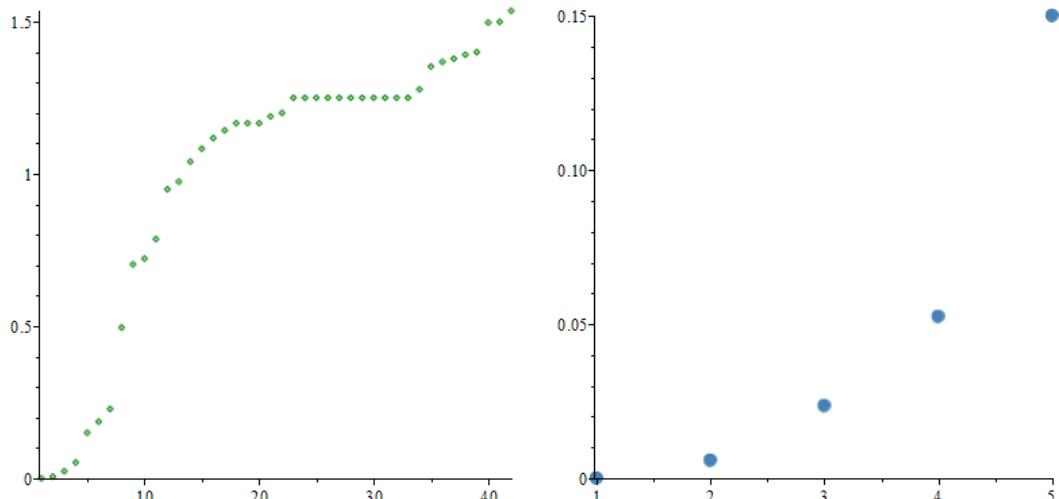
Початкові дані, відображені у вигляді гістограм



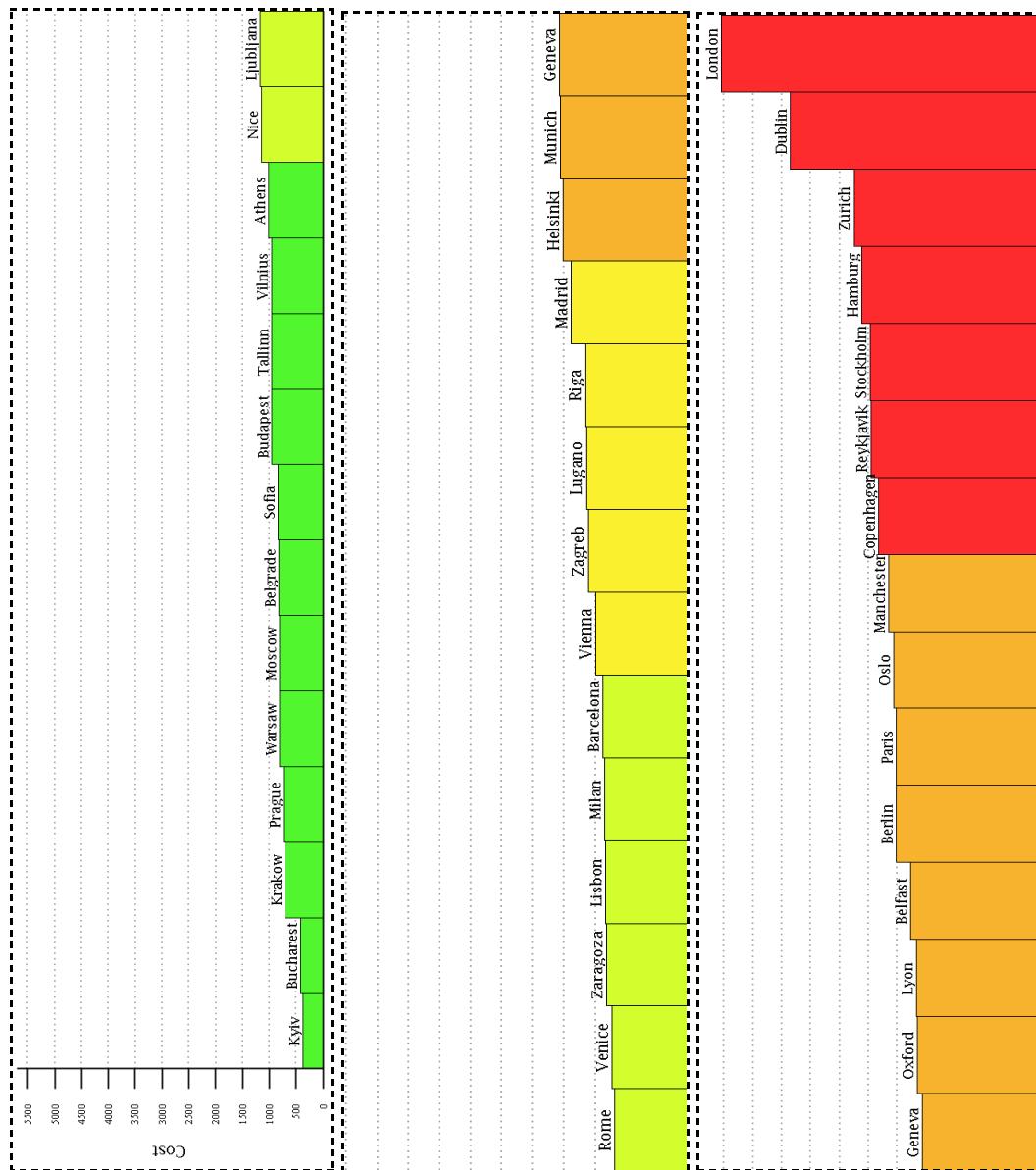
Процес роботи: граф подібності, власні числа та вектори



Граф подібності з параметром $k = 4$



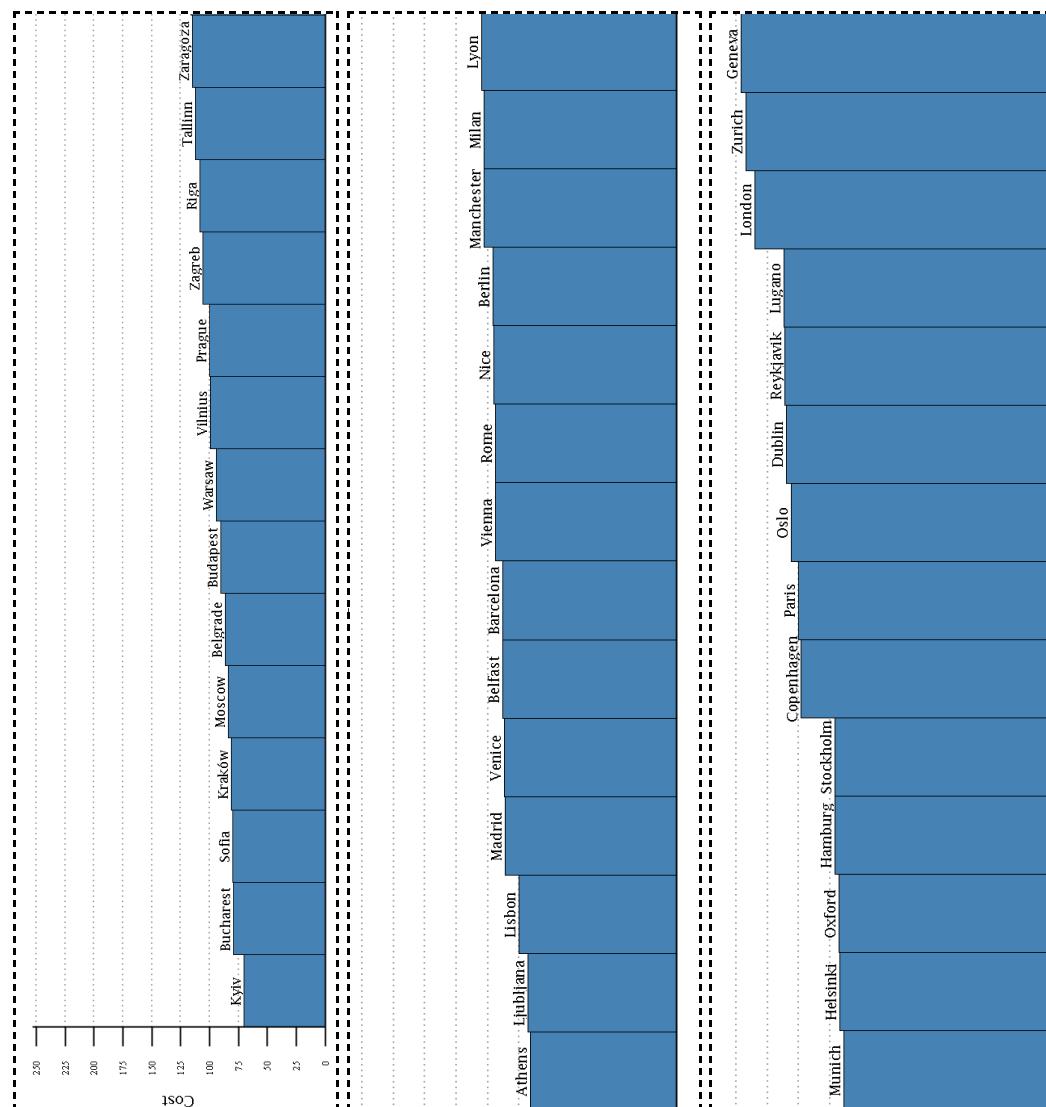
Результати роботи спектральної кластеризації



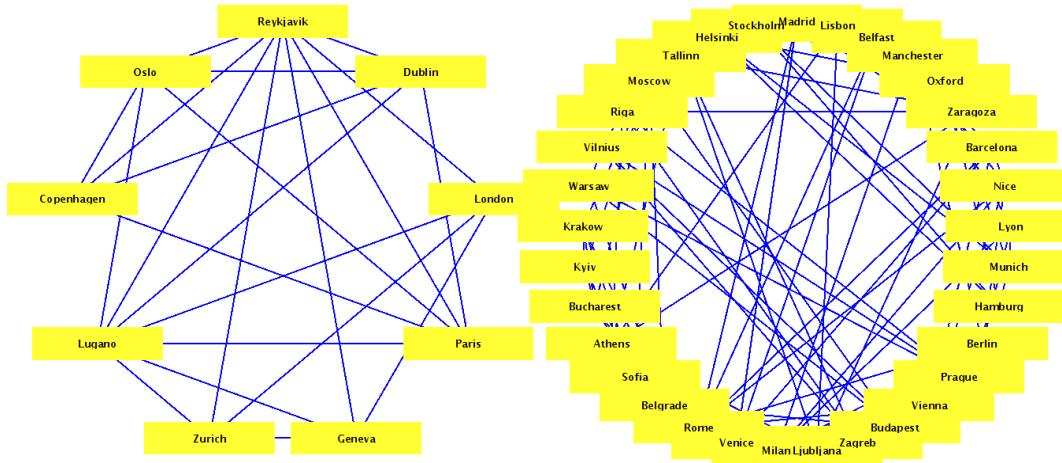
"Cluster #1" "Points : 12"	"Borders : from Kyiv (376 hr.) to Athens (1011 hr.)"	"Difference : 635 hr."
"Cluster #2" "Points : 8"	"Borders : from Nice (1138 hr.) to Barcelona (1368 hr.)"	"Difference : 230 hr."
"Cluster #3" "Points : 5"	"Borders : from Vienna (1500 hr.) to Madrid (1875 hr.)"	"Difference : 375 hr."
"Cluster #4" "Points : 10"	"Borders : from Helsinki (2006 hr.) to Manchester (2644 hr.)"	"Difference : 638 hr."
"Cluster #5" "Points : 7"	"Borders : from Copenhagen (2827 hr.) to London (5546 hr.)"	"Difference : 2719 hr."

Аналогічно до попереднього прикладу, спектральна кластеризація показує більш рівномірний розподіл точок. Це особливо добре можна побачити на останньому кластері: спектральна кластеризація виокремила 7 «дорогих» міст, тоді як K-means та BIRCH – 1 та 2 відповідно.

4.4.3. Індекс вартості проживання Початкові дані, відображені у вигляді гістограми

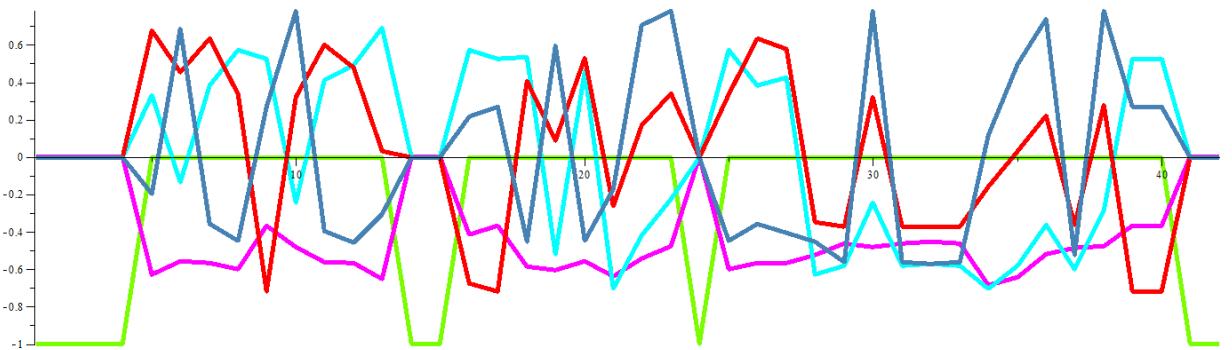


Процес роботи: граф подібності, власні числа та вектори



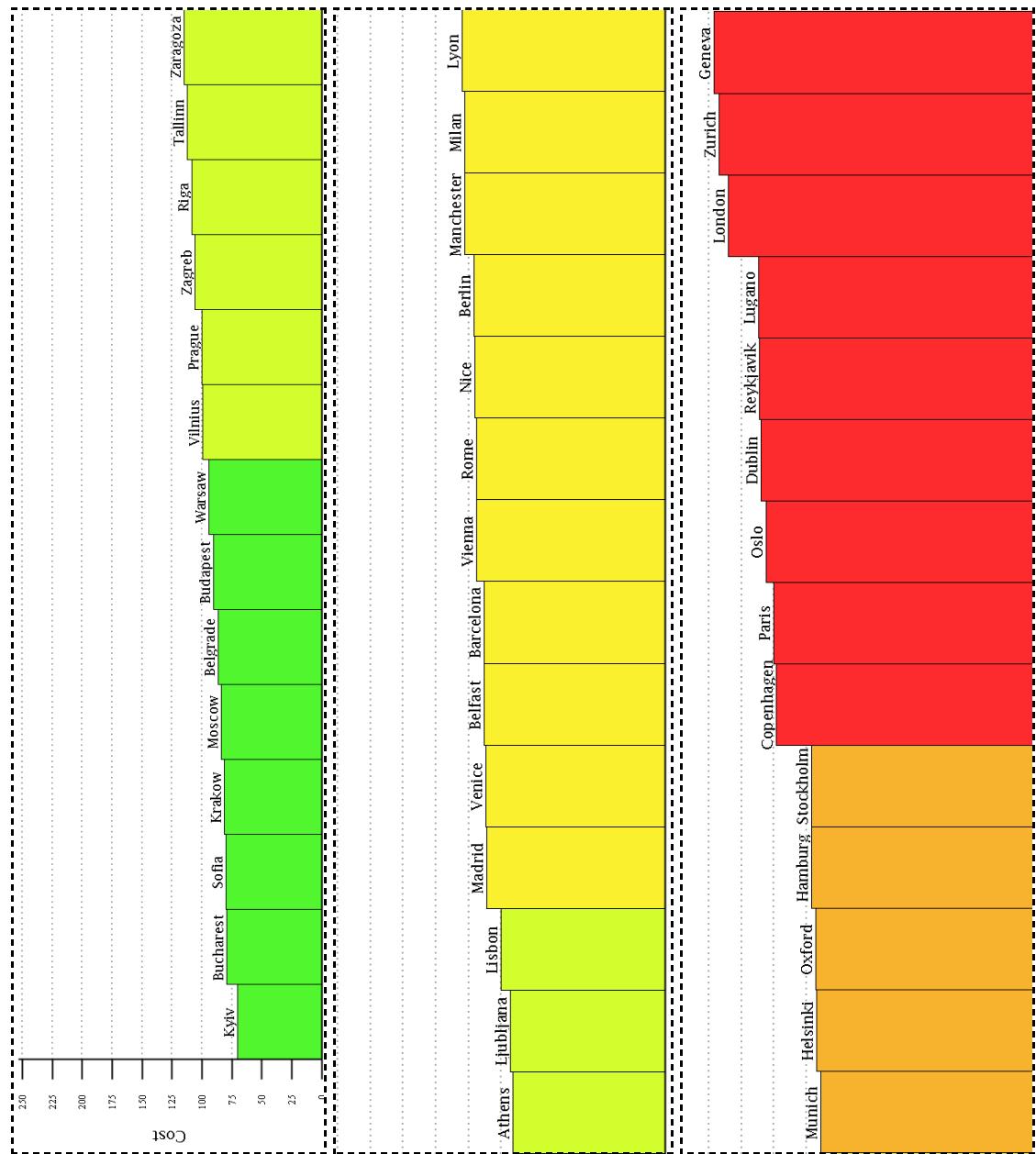
Граф подібності з параметром $k = 4$

Зауважимо, що при $k < 6$ формується більше однієї компоненти зв'язності. Коли ж для кожної вершини ми будемо шукати по 6 найближчих сусідів, то граф буде досить загромаджений. Це призводить до того, що комп'ютер досить погано знаходить власні вектори матриці Кірхгофа, а, отже, результати кластеризації будуть некоректними. Як вже було сказано, рекомендується обирати таке значення параметру k , щоб кількість компонентів зв'язності була меншою за кількість кластерів, тому було обрано $k = 4$.



Перші 5 власних векторів

Результати роботи спектральної кластеризації



"Cluster #1"	"Points : 8"	"Borders : from Kyiv (70 hr.) to Warsaw (94 hr.)"	"Difference : 24 hr."
"Cluster #2"	"Points : 9"	"Borders : from Vilnius (99 hr.) to Lisbon (125 hr.)"	"Difference : 26 hr."
"Cluster #3"	"Points : 11"	"Borders : from Madrid (136 hr.) to Lyon (155 hr.)"	"Difference : 19 hr."
"Cluster #4"	"Points : 5"	"Borders : from Munich (164 hr.) to Stockholm (171 hr.)"	"Difference : 7 hr."
"Cluster #5"	"Points : 9"	"Borders : from Copenhagen (198 hr.) to Geneva (246 hr.)"	"Difference : 48 hr."

Тенденція рівномірного розподілу зберігається і на цьому прикладі: спектральна кластеризація поділила точки на кластери у відношенні 8/9/11/5/9, тоді як K-means та BIRCH – у відношенні 8/9/16/6/3 та 8/9/16/6/3.

ВИСНОВКИ

Під час роботи над кваліфікаційною роботою було реалізовано три алгоритми кластеризації: K-means, BIRCH та спектральну кластеризацію. Їхня робота була протестована на множині даних, яка складається з набору середніх показників вартості певної продукції та загального індексу вартості проживання.

Серед переваг методу K-means варто виділити простоту та прозорість алгоритму. Однак, метод виявився занадто чутливим до «викидів», які можуть спотворювати результати. До недоліків треба ще додати достатньо повільну роботу на великих базах даних та необхідність визначати кількість кластерів завчасно. Також, можемо зазначити, що K-means, на відміну від, наприклад, спектральної кластеризації, не може об'єктивно оцінити кластери, коли дані утворюють складні форми та накладаються один на одного.

Метод кластеризації BIRCH характеризується двохетапністю (а саме, побудовою CF-дерева та глобальною кластеризацією). Завдяки представленню множини точок у вигляді CF-вектору, можна уникнути збереження великої кількості даних, що у свою чергу дозволяє працювати зі значним об'ємом інформації, маючи відносно невеликий об'єм пам'яті. Збалансована структура дерева дозволяє отримувати більш рівномірні результати (на відміну від K-means). Під час роботи з методом BIRCH було помічено лише один недолік: необхідність завчасно підібрати порогову величину. Проте, необхідно зазначити, що BIRCH, як і K-means, добре працює з кластерами опуклої форми, але не може впоратися з більш складними випадками.

Спектральна кластеризація, у свою чергу, найкраще працює з множиною точок, що утворюють складні форми та перетинаються. На жаль, використати цю властивість у кваліфікаційній роботі не вдалося, адже ми працюємо з одновимірними точками даних, а порівняння одразу за двома та більше факторами було залишено "на перспективу". Проте, зазначимо, що

метод дає більш рівномірний розподіл множини точок на кластери, порівнюючи з K-mean та BIRCH. Під час реалізації спектральної кластеризації виникло чимало питань стосовно налаштування методу: функція подібності, вид графу схожості, тип матриці Кірхгофа, метод подальшої кластеризації – усе це може по різному впливати на результат. З одного боку, такий масштаб налагодження дає змогу підкоригувати метод відповідно до вхідних даних. З іншого, спектральна кластеризація вимагає підібрати параметри таким чином, щоб власні вектори добре відображали кластери та щоб комп'ютер мав змогу обчислити ці вектори (для досить нагромадженого графу схожості виникають проблеми).

Окрім порівняння процесу роботи та результатів самих методів, було розглянуто відмінності між кластеризацією за загальним індексом та показниками, за якими цей індекс було визначено. Це дає змогу оцінити вплив показників на приналежність міста до того чи іншого кластеру за загальним індексом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Expatistan [Електронний ресурс] / Режим доступу: <https://www.expatistan.com/>
- [2] K-means clustering calculator [Електронний ресурс] / Режим доступу: <https://scistatcalc.blogspot.com/2014/01/k-means-clustering-calculator.html>
- [3] Дюран Б., Оделл П. Кластерный анализ //М.: статистика. – 1977. – Т. 128.
- [4] Князь Д. В. Методы кластеризации многомерных статистических данных. – 2014.
- [5] Алгоритмы кластеризации данных [Електронний ресурс] / Режим доступу: https://studref.com/698757/informatika/algoritmy_klasterizatsii_dannyh
- [6] Обзор алгоритмов кластеризации данных [Електронний ресурс] / Режим доступу: <https://habr.com/ru/post/.../>
- [7] Understanding K-means Clustering in Machine Learning [Електронний ресурс] / Режим доступу: [https://towardsdatascience.com/understanding-k-means-clustering\[...\]](https://towardsdatascience.com/understanding-k-means-clustering[...]/)
- [8] Bradley P. S., Bennett K. P., Demiriz A. Constrained k-means clustering //Microsoft Research, Redmond. – 2000. – Т. 20. – №. 0. – С. 0.
- [9] BIRCH Clustering Algorithm [Електронний ресурс] / Режим доступу: [https://towardsdatascience.com/machine-learning-birch-clustering-algorithm\[...\]](https://towardsdatascience.com/machine-learning-birch-clustering-algorithm[...]/)
- [10] BIRCH [Електронний ресурс] / Режим доступу: [https://algowiki-project.org/ru/\[...\]/BIRCH](https://algowiki-project.org/ru/[...]/BIRCH)
- [11] A Tutorial on Spectral Clustering [Електронний ресурс] / Режим доступу: [https://people.csail.mit.edu/\[...\]/Luxburg07_tutorial_spectral_clustering.pdf](https://people.csail.mit.edu/[...]/Luxburg07_tutorial_spectral_clustering.pdf)

ДОДАТКИ

Додаток А. Початкові дані

Початкові дані були взяті із відкритого доступу на сайті expatistan.com [1]. Всі ціни переведено за поточним на 10.02.2022 р. курсом у гривні.

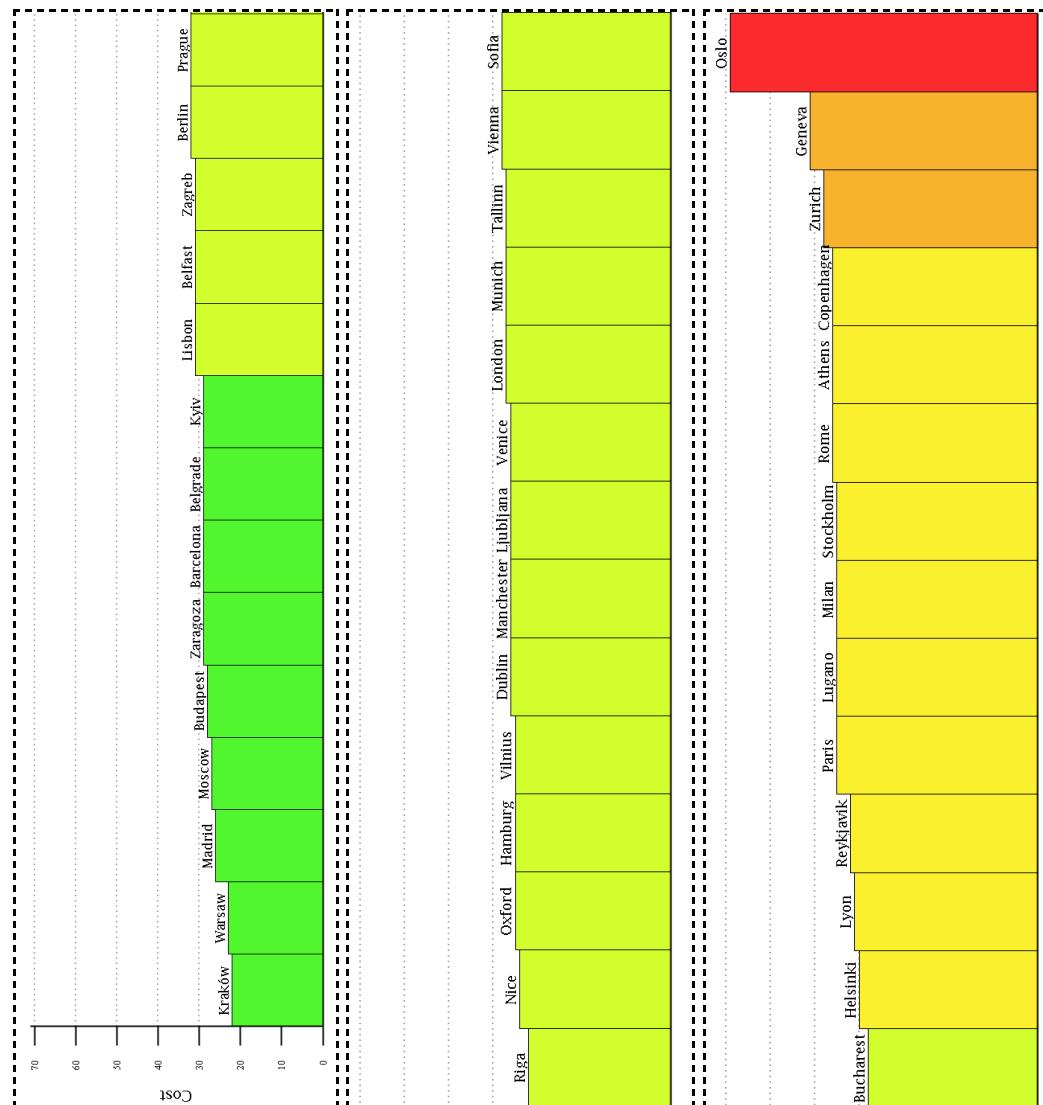
1	City\Criterion	1 liter of whole fat milk	12 eggs, large	30 sqft) furnished apartment	monthly ticket public transp	to private doctor (15 m)	2 tickets to the movies	gym membership in busi	Cost of living index
2	Reykjavík	42	175	67185	2952	1522	720	1638	211
3	Dublin	36	121	77406	4353	1954	749	1698	210
4	London	37	114	95160	5546	4055	990	2011	235
5	Paris	45	157	99101	2520	913	716	1843	200
6	Madrid	26	73	47087	1875	1786	573	1710	136
7	Lisbon	31	75	42061	1313	1591	472	1351	125
8	Belfast	31	89	44190	2272	2521	604	1026	138
9	Manchester	36	91	54266	2644	3908	601	1071	153
10	Oxford	35	105	71009	2153	4019	857	1565	168
11	Zaragoza	29	62	32162	1309	1808	475	1502	115
12	Barcelona	29	93	48562	1368	2390	566	1776	138
13	Nice	34	153	54874	1138	817	712	1722	145
14	Lyon	41	149	44976	2166	832	734	1532	155
15	Geneva	51	239	120788	2067	3805	1116	2723	246
16	Zurich	48	225	96599	3251	3984	1035	3442	242
17	Munich	37	105	73939	2049	1792	730	2218	164
18	Hamburg	35	126	71284	3114	1686	832	1858	171
19	Berlin	32	103	48493	2508	2796	708	1072	146
20	Prague	32	79	36486	729	1286	464	1167	100
21	Vienna	38	203	44088	1500	3995	693	1394	144
22	Budapest	28	58	28798	950	1813	356	1061	90
23	Zagreb	31	78	25677	1604	1281	314	1229	105
24	Ljubljana	36	91	38743	1180	1764	436	1529	118
25	Lugano	45	199	79023	1635	3392	1028	3105	212
26	Milan	45	133	61162	1335	2777	682	2637	153
27	Venice	36	122	49814	1211	3073	602	1297	137
28	Rome	46	105	61858	1180	2701	585	1937	144
29	Belgrade	29	51	22787	818	853	285	924	86
30	Sofia	38	66	18905	832	542	364	862	80
31	Athens	46	120	27781	1011	995	522	1251	116
32	Bucharest	38	94	27038	422	1033	346	1574	79
33	Kyiv	29	33	27892	376	367	188	1324	70
34	Kraków	22	75	25920	707	873	414	823	81
35	Warsaw	23	78	37041	806	1033	468	817	94
36	Vilnius	35	62	28374	962	1083	453	1460	99
37	Riga	32	72	26260	1658	1346	523	1303	108
38	Moscow	27	38	38868	807	678	303	1569	84
39	Tallinn	37	73	38145	958	1453	553	1702	112
40	Helsinki	40	88	68787	2006	2494	908	1870	167
41	Stockholm	45	120	60038	2962	796	856	1553	171
42	Copenhagen	46	158	70527	2827	2474	1001	1169	198
43	Oslo	69	176	70261	2553	2739	964	1536	206

Таблиця відображає (рухаючись по колонкам зліва направо) множину міст та відповідні їм показники середньої вартості молока (1 літр), яєць (12 штук, великих), місячної оренди житла (площею 85 м² у дорогому районі), місячного квитку для проїзду у міському транспорті, короткого візиту до лікаря (15 хвилин), двох квитків у кіно, місячного абонементу до спортивної залі (у діловому районі) та індекс вартості проживання (визначений ресурсами сайту).

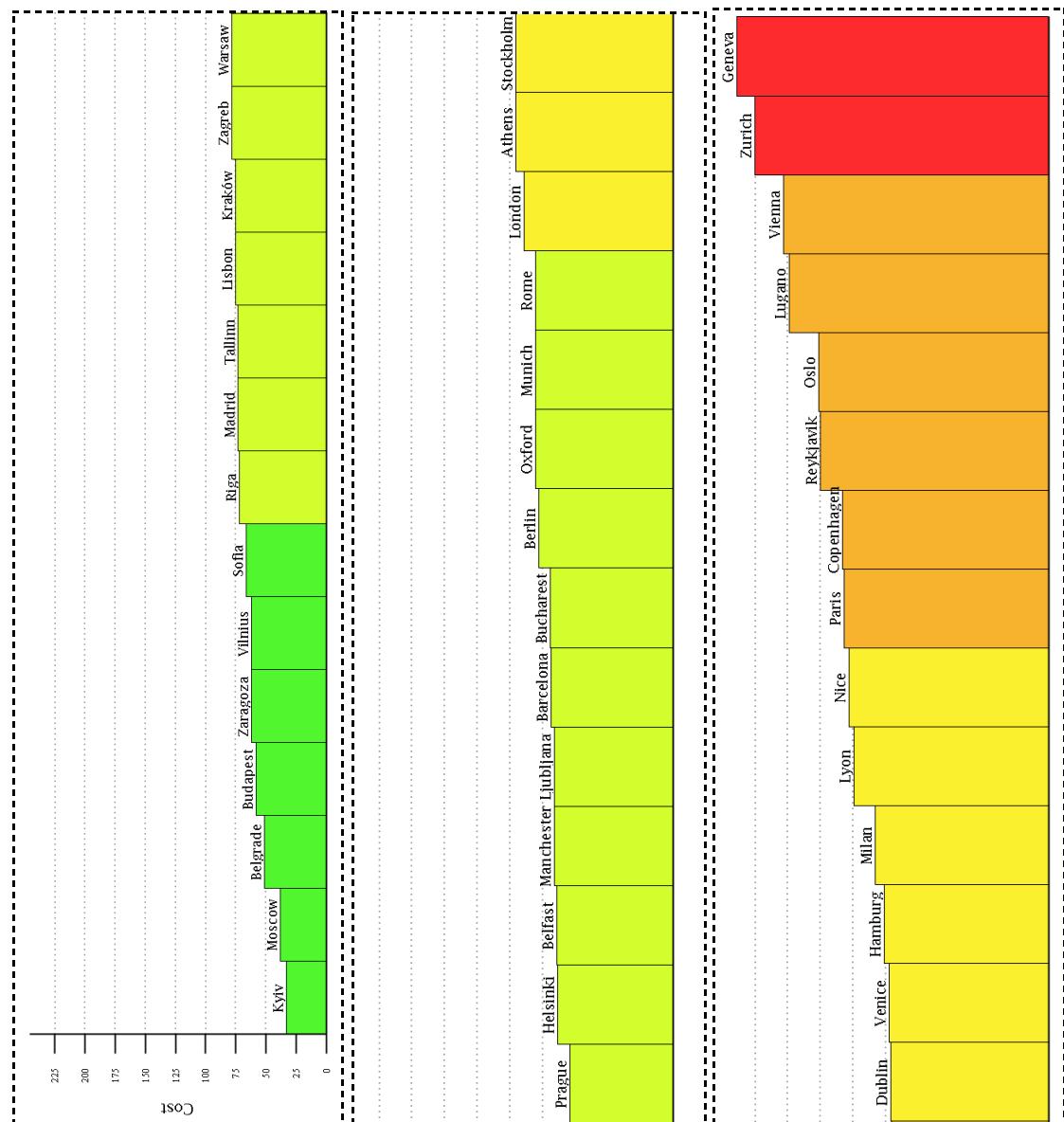
Додаток Б. Отримані дані

Метод K-means (5 кластерів)

Середня вартість молока (1 літр)

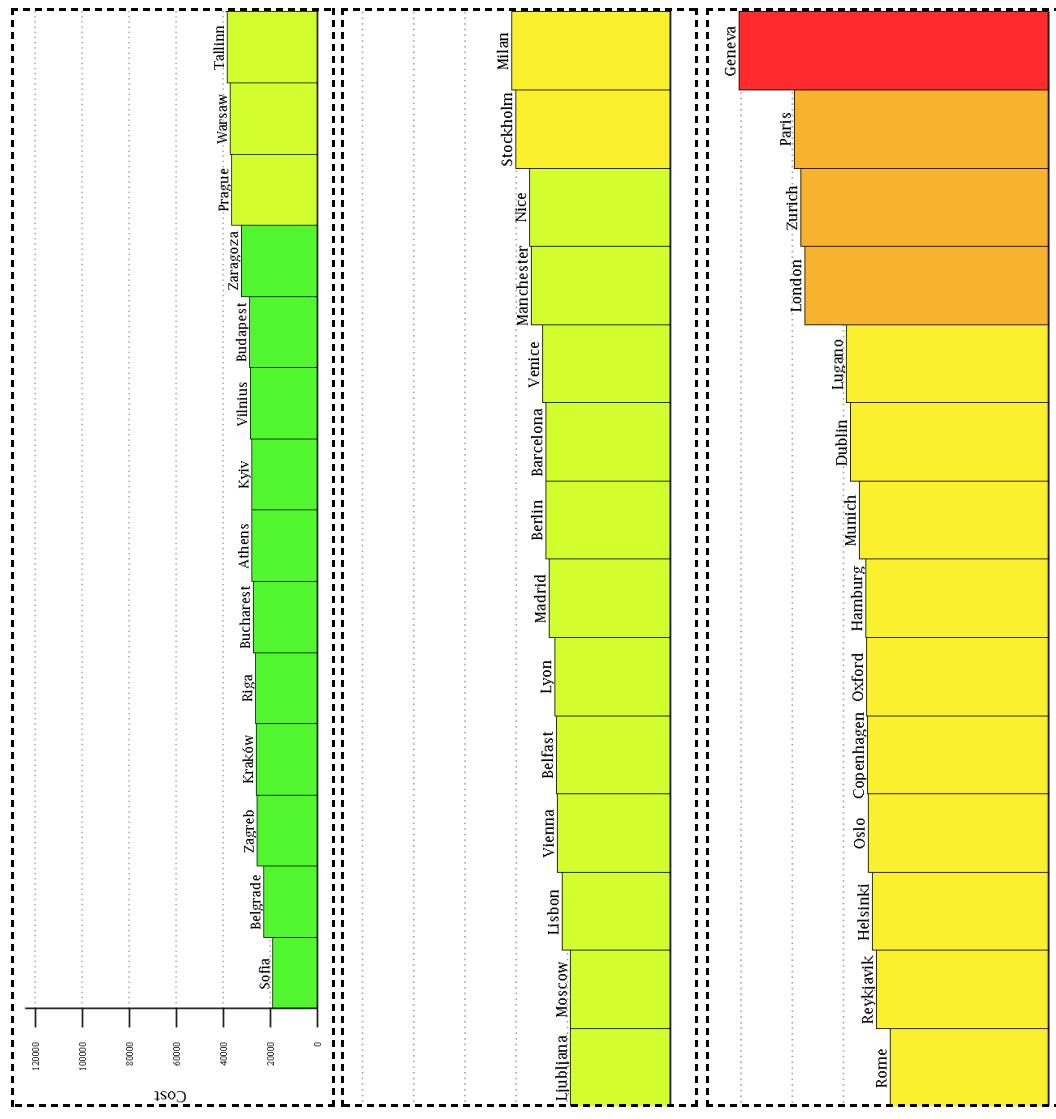


Середня вартість яєць (12 штук, великі)



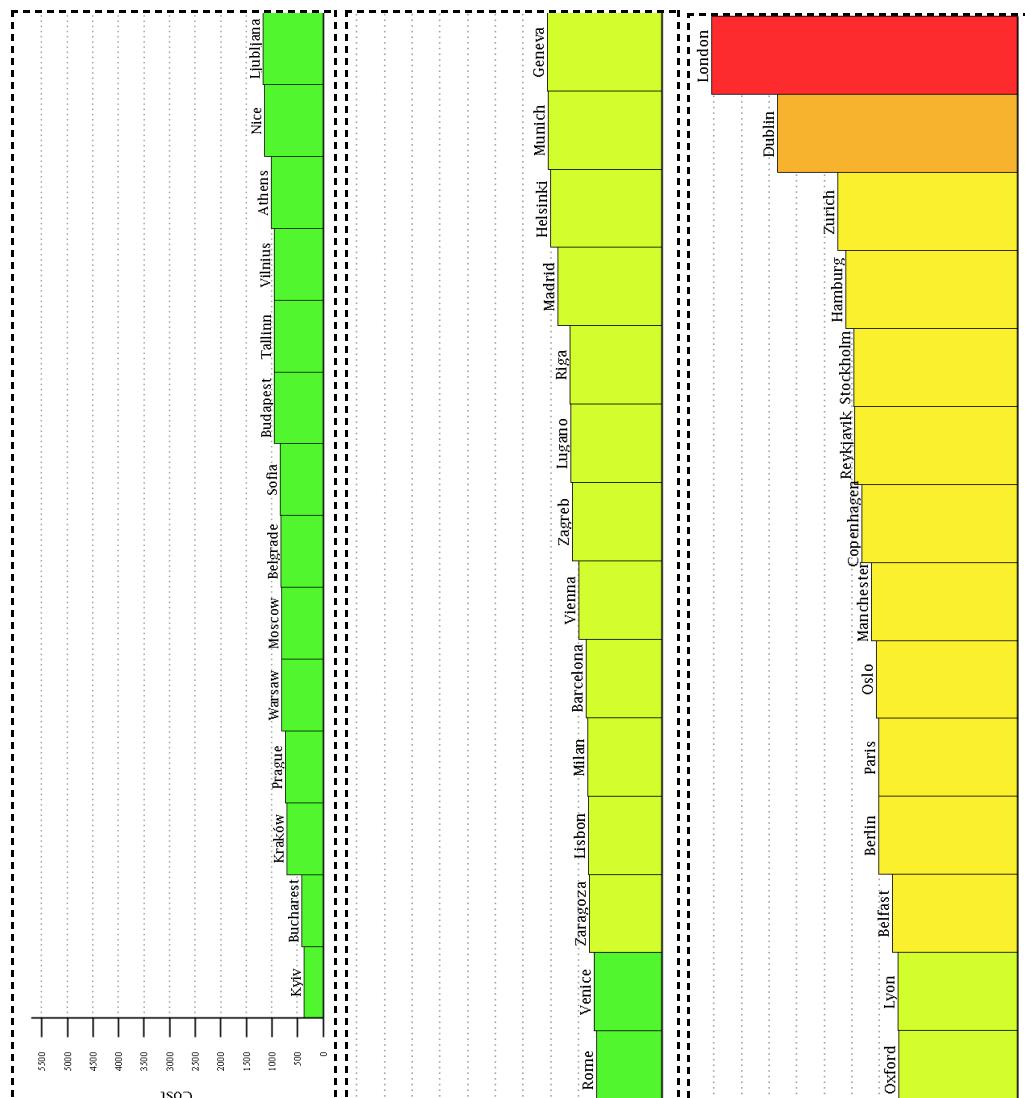
"Cluster #1"	"Points : 7"	"Borders : from Kyiv (33 hr.) to Sofia (66 hr.)"	"Difference : 33 hr."
"Cluster #2"	"Points : 18"	"Borders : from Riga (72 hr.) to Rome (105 hr.)"	"Difference : 33 hr."
"Cluster #3"	"Points : 9"	"Borders : from London (114 hr.) to Nice (153 hr.)"	"Difference : 39 hr."
"Cluster #4"	"Points : 6"	"Borders : from Paris (157 hr.) to Vienna (203 hr.)"	"Difference : 46 hr."
"Cluster #5"	"Points : 2"	"Borders : from Zurich (225 hr.) to Geneva (239 hr.)"	"Difference : 14 hr."

Місячна оренда 85 м² мебльованого житла в дорогому районі



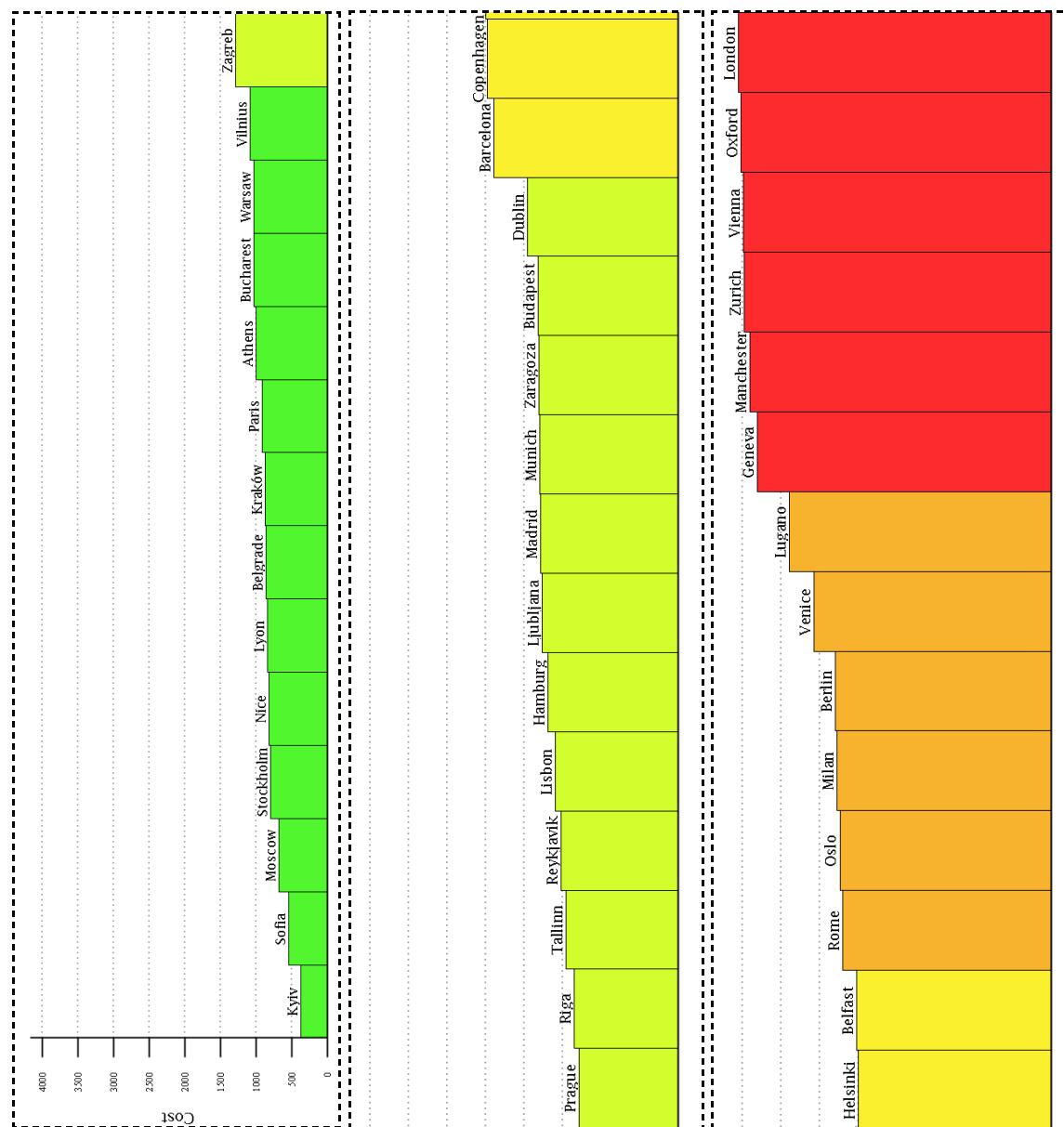
<i>Borders :=</i>	"Sofia"	18905.0	"Zaragoza"	32162.0	"Difference"	13257.0
	"Prague"	36486.0	"Nice"	54874.0	"Difference"	18388.0
	"Stockholm"	60038.0	"Lugano"	79023.0	"Difference"	18985.0
	"London"	95160.0	"Paris"	99101.0	"Difference"	3941.0
	"Geneva"	$1.207880 \cdot 10^5$	"Geneva"	$1.207880 \cdot 10^5$	"Difference"	0.

Місячний квиток на міський транспорт



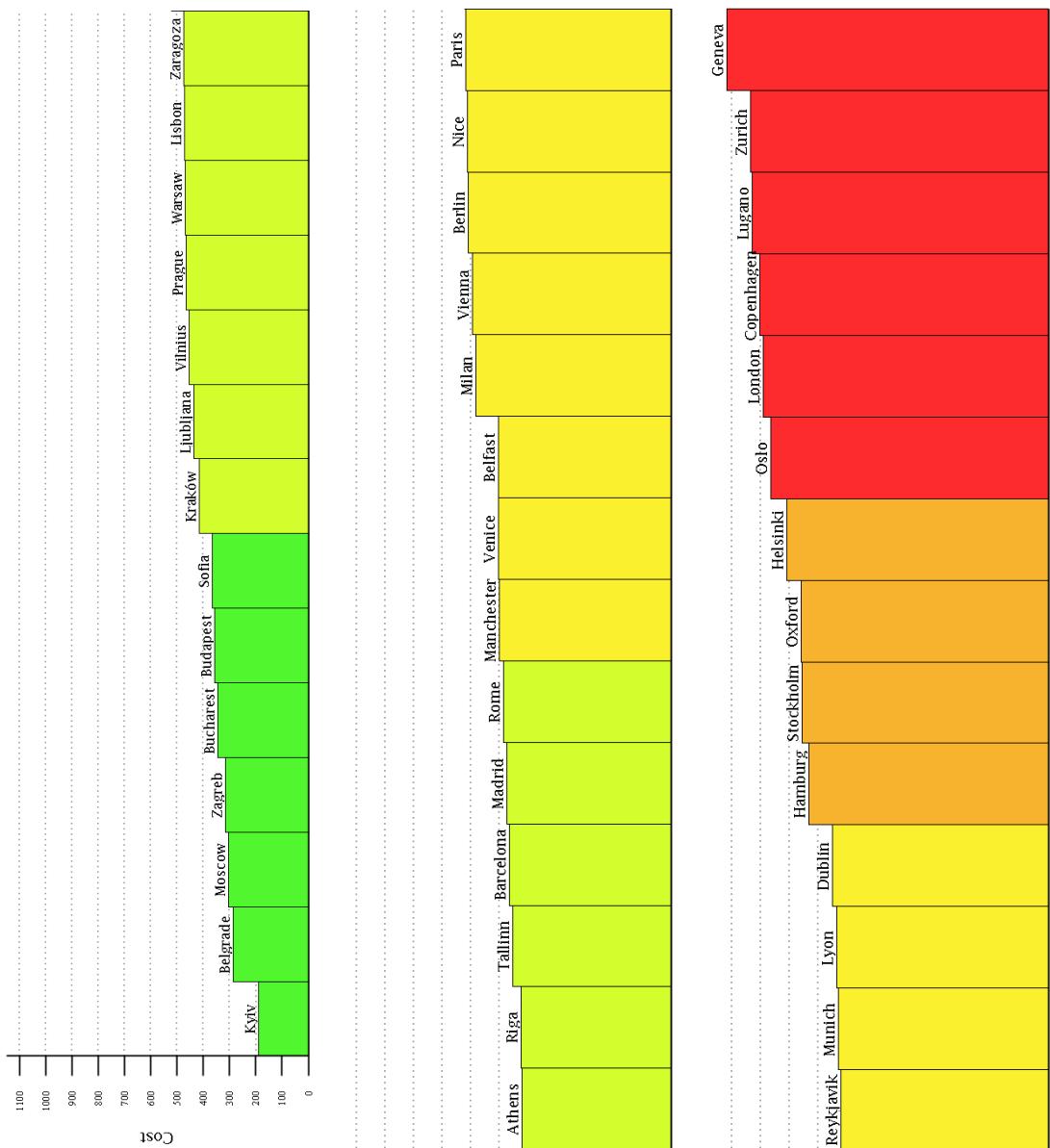
"Cluster #1"	"Points : 16"	"Borders : from Kyiv (376 hr.) to Venice (1211 hr.)"	"Difference : 835 hr."
"Cluster #2"	"Points : 14"	"Borders : from Zaragoza (1309 hr.) to Lyon (2166 hr.)"	"Difference : 857 hr."
"Cluster #3"	"Points : 10"	"Borders : from Belfast (2272 hr.) to Zurich (3251 hr.)"	"Difference : 979 hr."
"Cluster #4"	"Points : 1"	"Borders : from Dublin (4353 hr.) to Dublin (4353 hr.)"	"Difference : 0 hr."
"Cluster #5"	"Points : 1"	"Borders : from London (5546 hr.) to London (5546 hr.)"	"Difference : 0 hr."

Середня вартість короткого візиту до приватного лікаря (15 хв.)



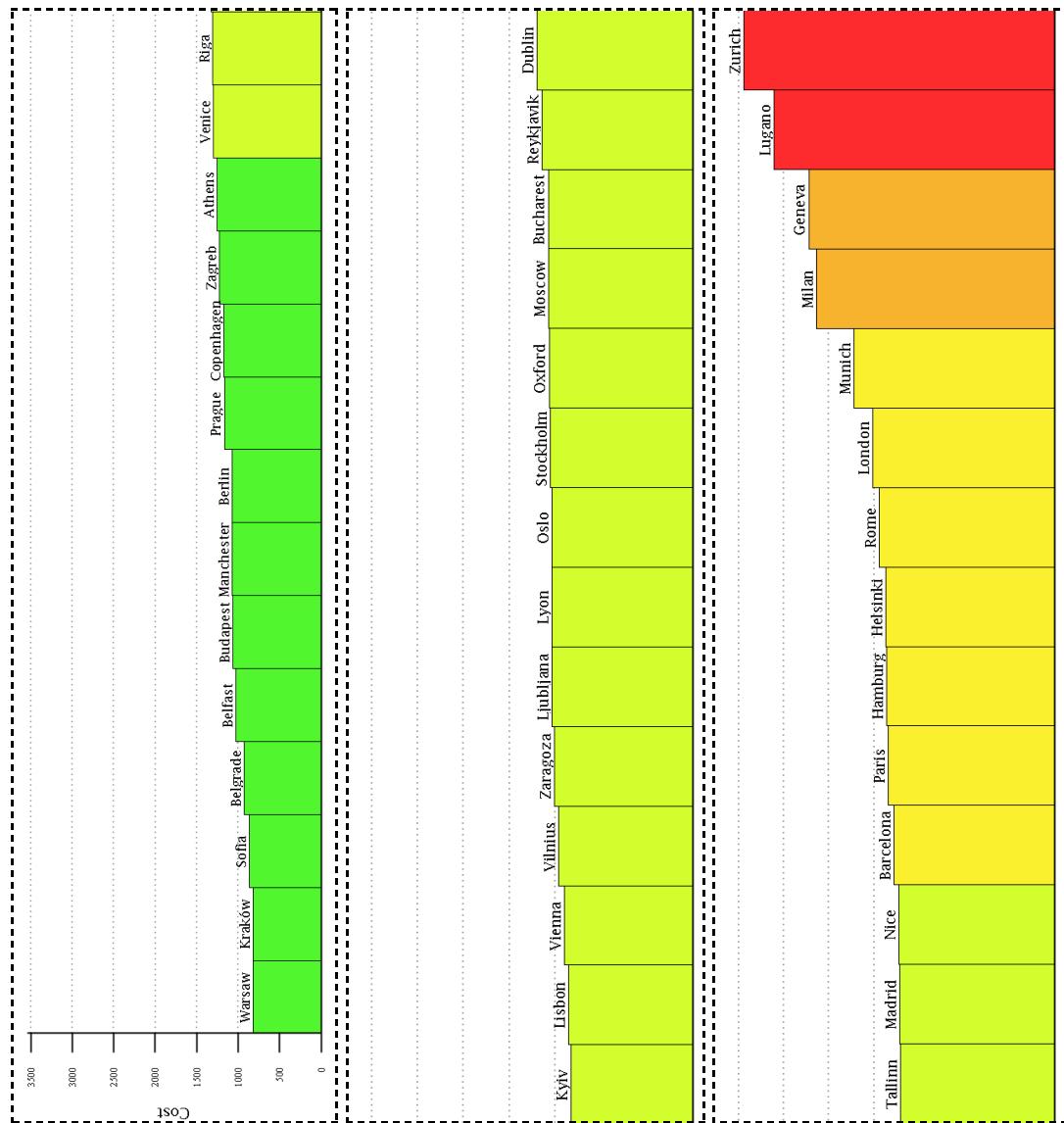
"Cluster #1"	"Points : 13"	"Borders : from Kyiv (367 hr.) to Vilnius (1083 hr.)"	"Difference : 716 hr."
"Cluster #2"	"Points : 13"	"Borders : from Zagreb (1281 hr.) to Dublin (1954 hr.)"	"Difference : 673 hr."
"Cluster #3"	"Points : 4"	"Borders : from Barcelona (2390 hr.) to Belfast (2521 hr.)"	"Difference : 131 hr."
"Cluster #4"	"Points : 6"	"Borders : from Rome (2701 hr.) to Lugano (3392 hr.)"	"Difference : 691 hr."
"Cluster #5"	"Points : 6"	"Borders : from Geneva (3805 hr.) to London (4055 hr.)"	"Difference : 250 hr."

Середня вартість 2 квитків у кіно



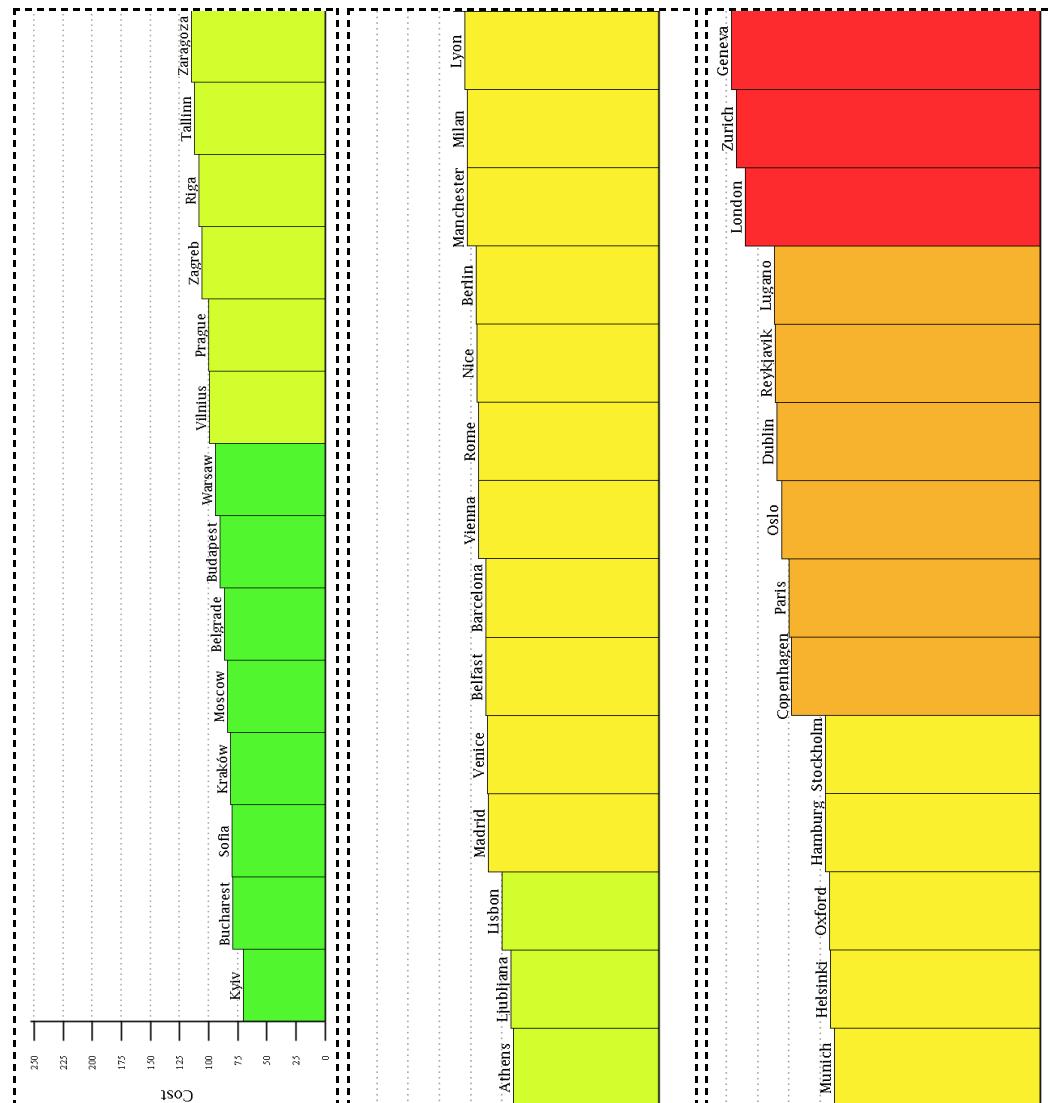
"Cluster #1" "Points : 7"	"Borders : from Kyiv (188 hr.) to Sofia (364 hr.)"	"Difference : 176 hr."
"Cluster #2" "Points : 13"	"Borders : from Kraków (414 hr.) to Rome (585 hr.)"	"Difference : 171 hr."
"Cluster #3" "Points : 12"	"Borders : from Manchester (601 hr.) to Dublin (749 hr.)"	"Difference : 148 hr."
"Cluster #4" "Points : 4"	"Borders : from Hamburg (832 hr.) to Helsinki (908 hr.)"	"Difference : 76 hr."
"Cluster #5" "Points : 6"	"Borders : from Oslo (964 hr.) to Geneva (1116 hr.)"	"Difference : 152 hr."

Місячний абонемент у спортзал у діловому районі



"Cluster #1"	"Points : 12"	"Borders : from Warsaw (817 hr.) to Athens (1251 hr.)"	"Difference : 434 hr."
"Cluster #2"	"Points : 19"	"Borders : from Venice (1297 hr.) to Nice (1722 hr.)"	"Difference : 425 hr."
"Cluster #3"	"Points : 7"	"Borders : from Barcelona (1776 hr.) to Munich (2218 hr.)"	"Difference : 442 hr."
"Cluster #4"	"Points : 2"	"Borders : from Milan (2637 hr.) to Geneva (2723 hr.)"	"Difference : 86 hr."
"Cluster #5"	"Points : 2"	"Borders : from Lugano (3105 hr.) to Zurich (3442 hr.)"	"Difference : 337 hr."

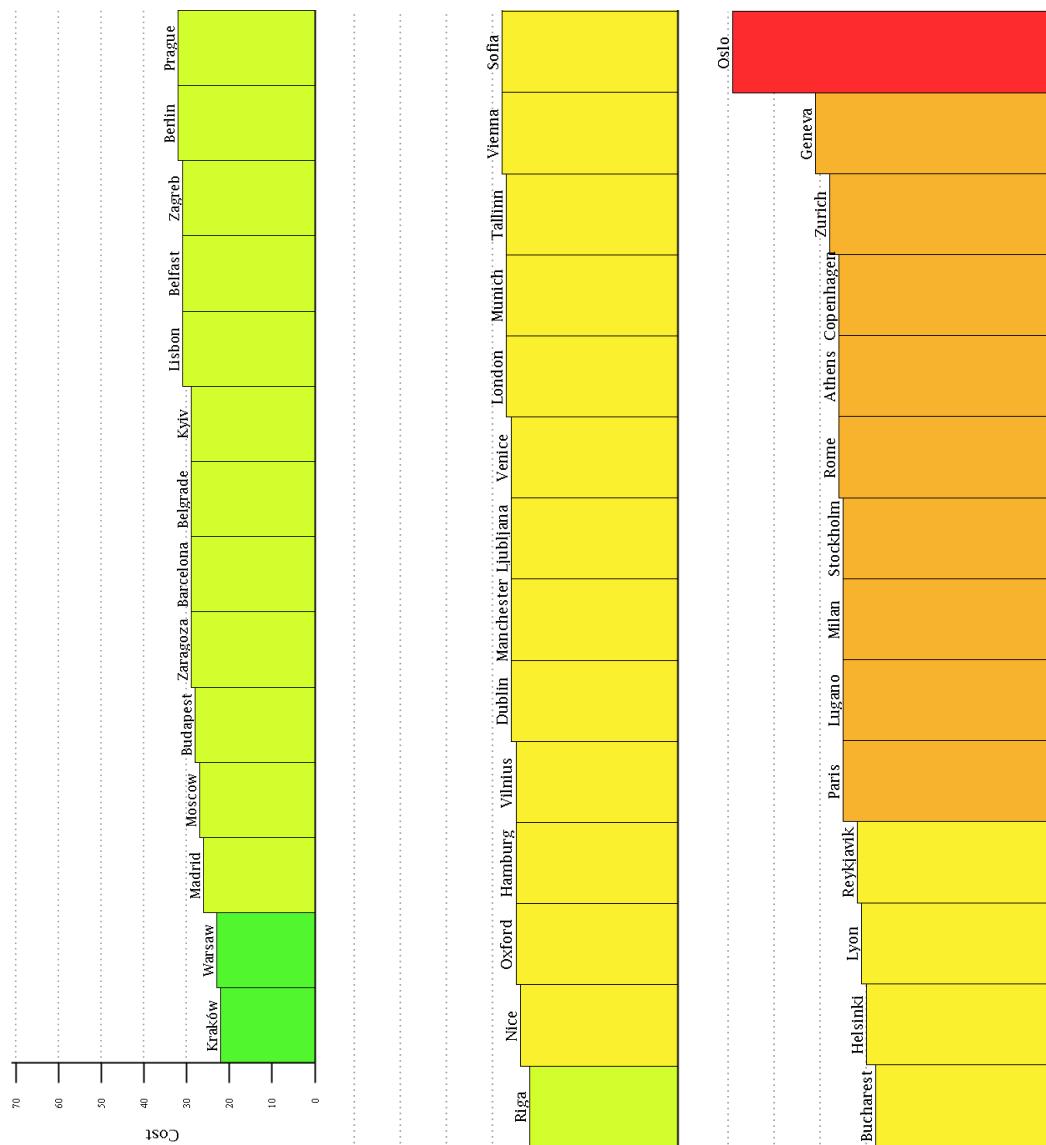
Індекс вартості проживання



"Cluster #1"	"Points : 8"	"Borders : from Kyiv (70 hr.) to Warsaw (94 hr.)"	"Difference : 24 hr."
"Cluster #2"	"Points : 9"	"Borders : from Vilnius (99 hr.) to Lisbon (125 hr.)"	"Difference : 26 hr."
"Cluster #3"	"Points : 16"	"Borders : from Madrid (136 hr.) to Stockholm (171 hr.)"	"Difference : 35 hr."
"Cluster #4"	"Points : 6"	"Borders : from Copenhagen (198 hr.) to Lugano (212 hr.)"	"Difference : 14 hr."
"Cluster #5"	"Points : 3"	"Borders : from London (235 hr.) to Geneva (246 hr.)"	"Difference : 11 hr."

Метод BIRCH (5 кластерів)

Середня вартість молока (1 літр)

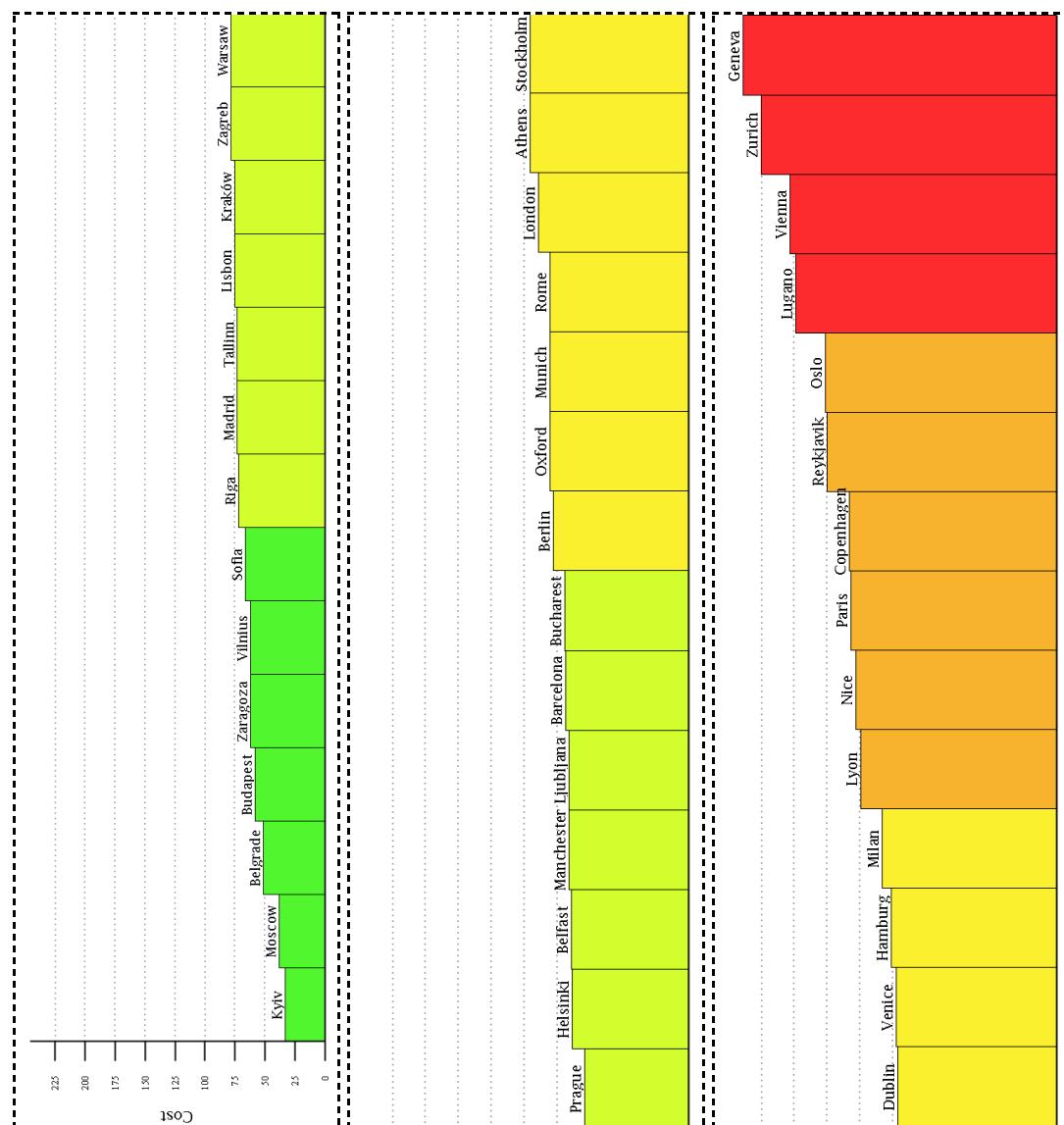


```

[ "Cluster #1" "Points : 2" "Borders : from Kraków (22 hr.) to Warsaw (23 hr.)" "Difference : 1 hr."
"Cluster #2" "Points : 13"    "Borders : from Madrid (26 hr.) to Riga (32 hr.)"    "Difference : 6 hr."
"Cluster #3" "Points : 17"    "Borders : from Nice (34 hr.) to Reykjavik (42 hr.)"    "Difference : 8 hr."
"Cluster #4" "Points : 9"     "Borders : from Paris (45 hr.) to Geneva (51 hr.)"    "Difference : 6 hr."
"Cluster #5" "Points : 1"     "Borders : from Oslo (69 hr.) to Oslo (69 hr.)"    "Difference : 0 hr."
]

```

Середня вартість яєць (12 штук, великі)

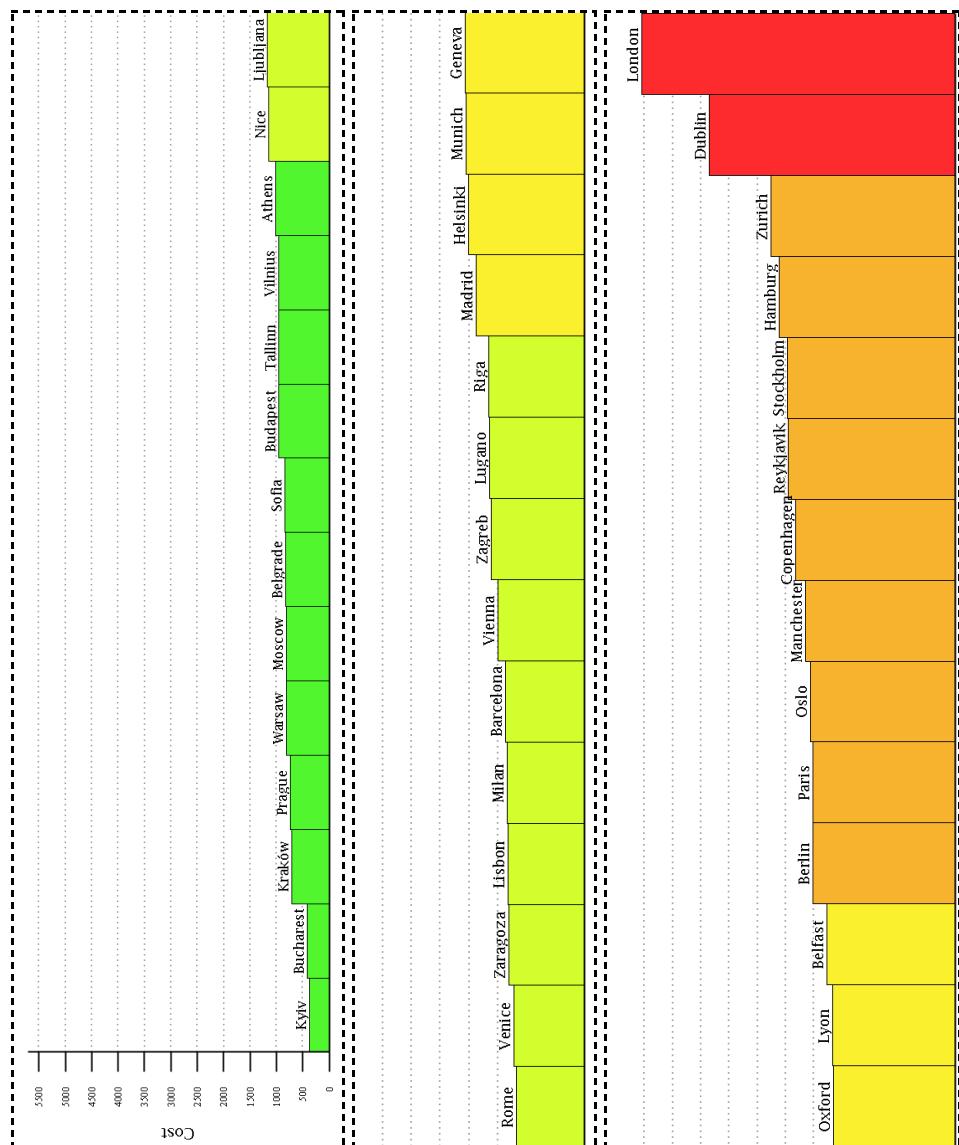


Місячна оренда 85 м² мебльованого житла в дорогому районі



```
[ "Cluster #1" "Points : 11" "Borders : from Sofia (18905 hr.) to Zaragoza (32162 hr.)" "Difference : 13257 hr."
"Cluster #2" "Points : 13" "Borders : from Prague (36486 hr.) to Venice (49814 hr.)" "Difference : 13328 hr."
"Cluster #3" "Points : 5" "Borders : from Manchester (54266 hr.) to Rome (61858 hr.)" "Difference : 7592 hr."
"Cluster #4" "Points : 9" "Borders : from Reykjavik (67185 hr.) to Lugano (79023 hr.)" "Difference : 11838 hr."
"Cluster #5" "Points : 4" "Borders : from London (95160 hr.) to Geneva (120788 hr.)" "Difference : 25628 hr." ]
```

Місячний квиток на міський транспорт



```

["Cluster #1" "Points : 12"    "Borders : from Kyiv (376 hr.) to Athens (1011 hr.)"    "Difference : 635 hr."]
["Cluster #2" "Points : 12"    "Borders : from Nice (1138 hr.) to Riga (1658 hr.)"    "Difference : 520 hr."]
["Cluster #3" "Points : 7"     "Borders : from Madrid (1875 hr.) to Belfast (2272 hr.)"    "Difference : 397 hr."]
["Cluster #4" "Points : 9"     "Borders : from Berlin (2508 hr.) to Zurich (3251 hr.)"    "Difference : 743 hr."]
["Cluster #5" "Points : 2"     "Borders : from Dublin (4353 hr.) to London (5546 hr.)" "Difference : 1193 hr."]

```

Середня вартість короткого візиту до приватного лікаря (15 хв.)

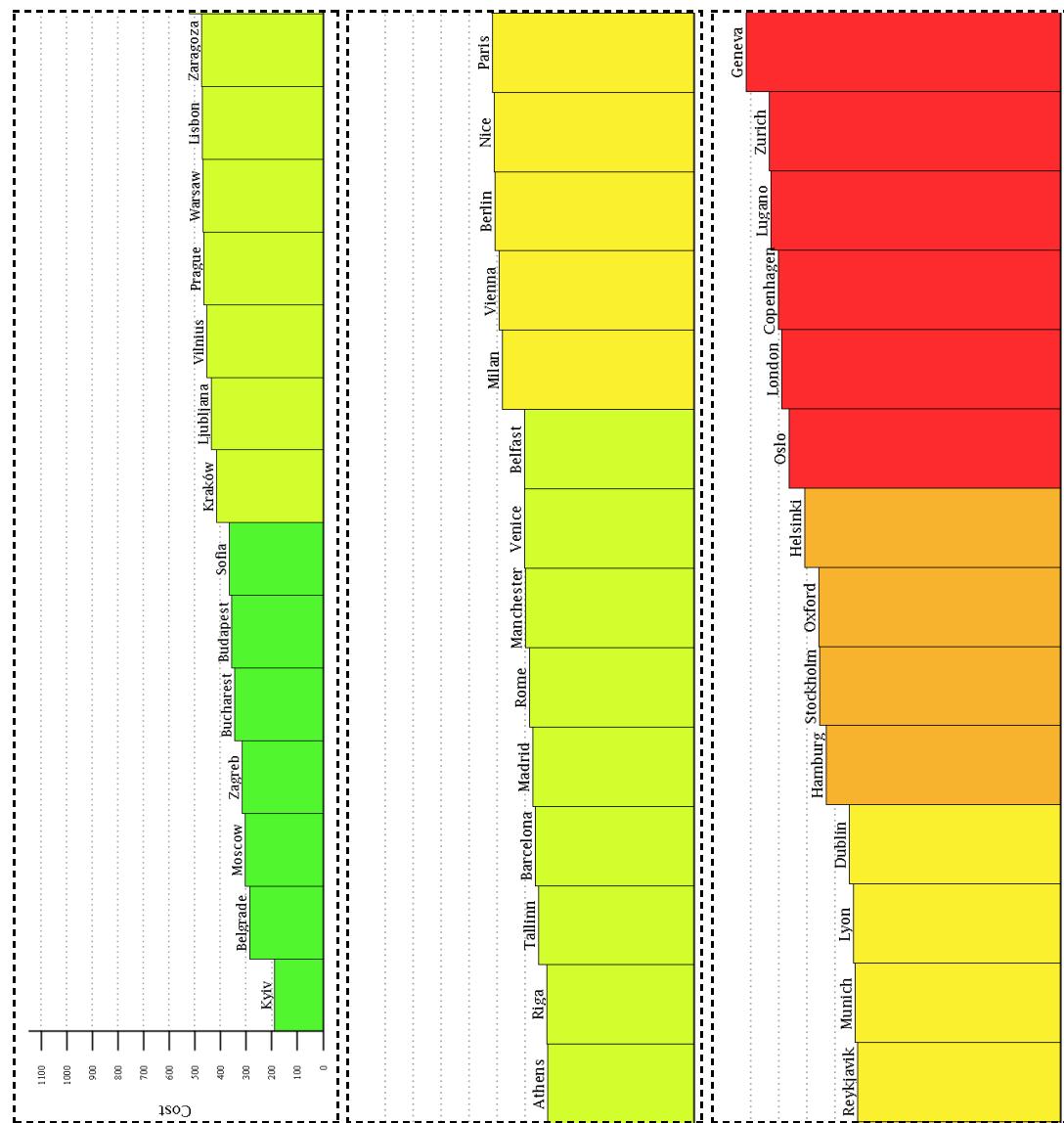


```

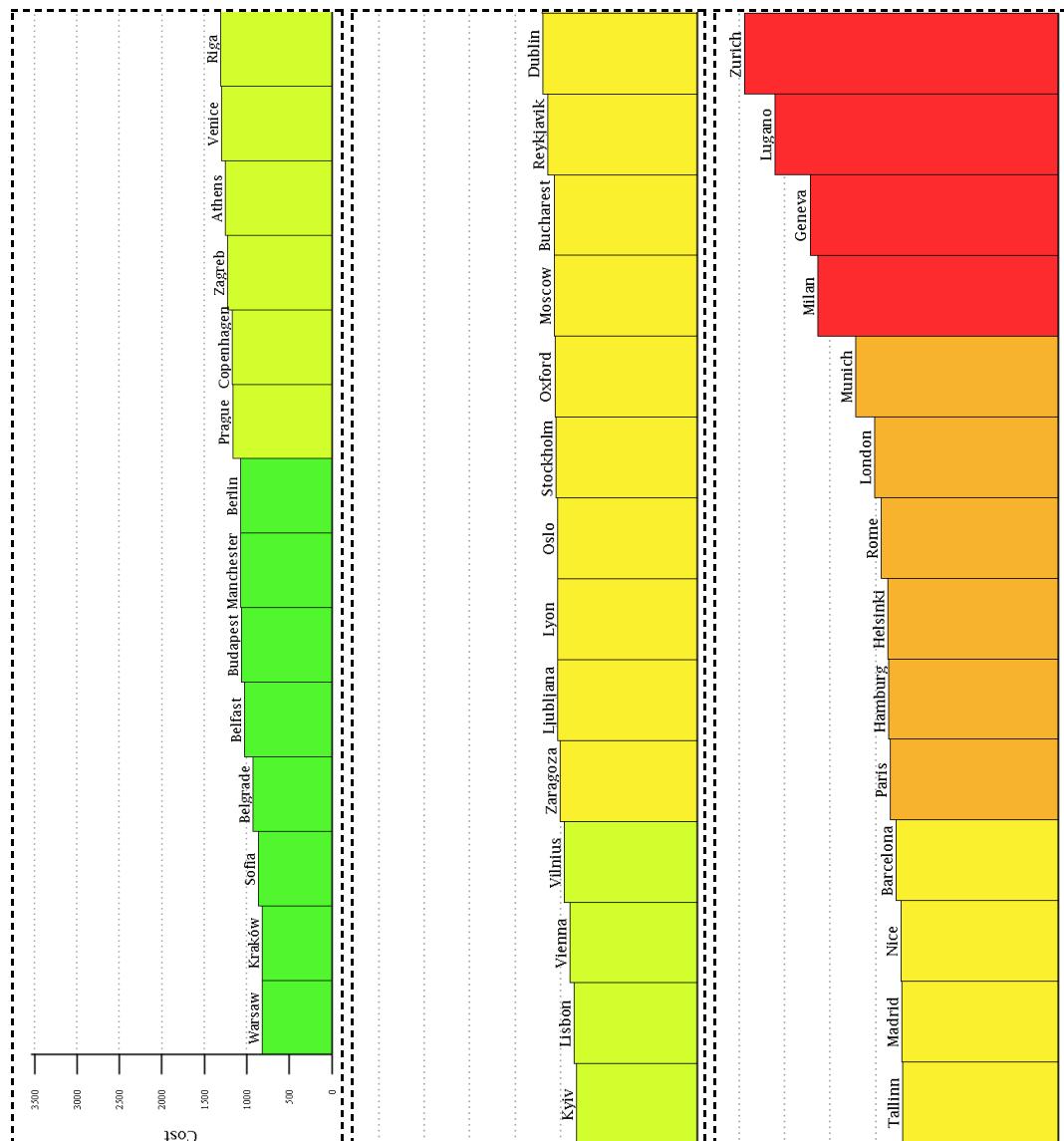
[ "Cluster #1" "Points : 13"    "Borders : from Kyiv (367 hr.) to Vilnius (1083 hr.)"    "Difference : 716 hr."
 "Cluster #2" "Points : 13"    "Borders : from Zagreb (1281 hr.) to Dublin (1954 hr.)"    "Difference : 673 hr."
 "Cluster #3" "Points : 8"     "Borders : from Barcelona (2390 hr.) to Berlin (2796 hr.)"    "Difference : 406 hr."
 "Cluster #4" "Points : 2"     "Borders : from Venice (3073 hr.) to Lugano (3392 hr.)"    "Difference : 319 hr."
 "Cluster #5" "Points : 6"     "Borders : from Geneva (3805 hr.) to London (4055 hr.)"    "Difference : 250 hr." ]

```

Середня вартість 2 квитків у кіно

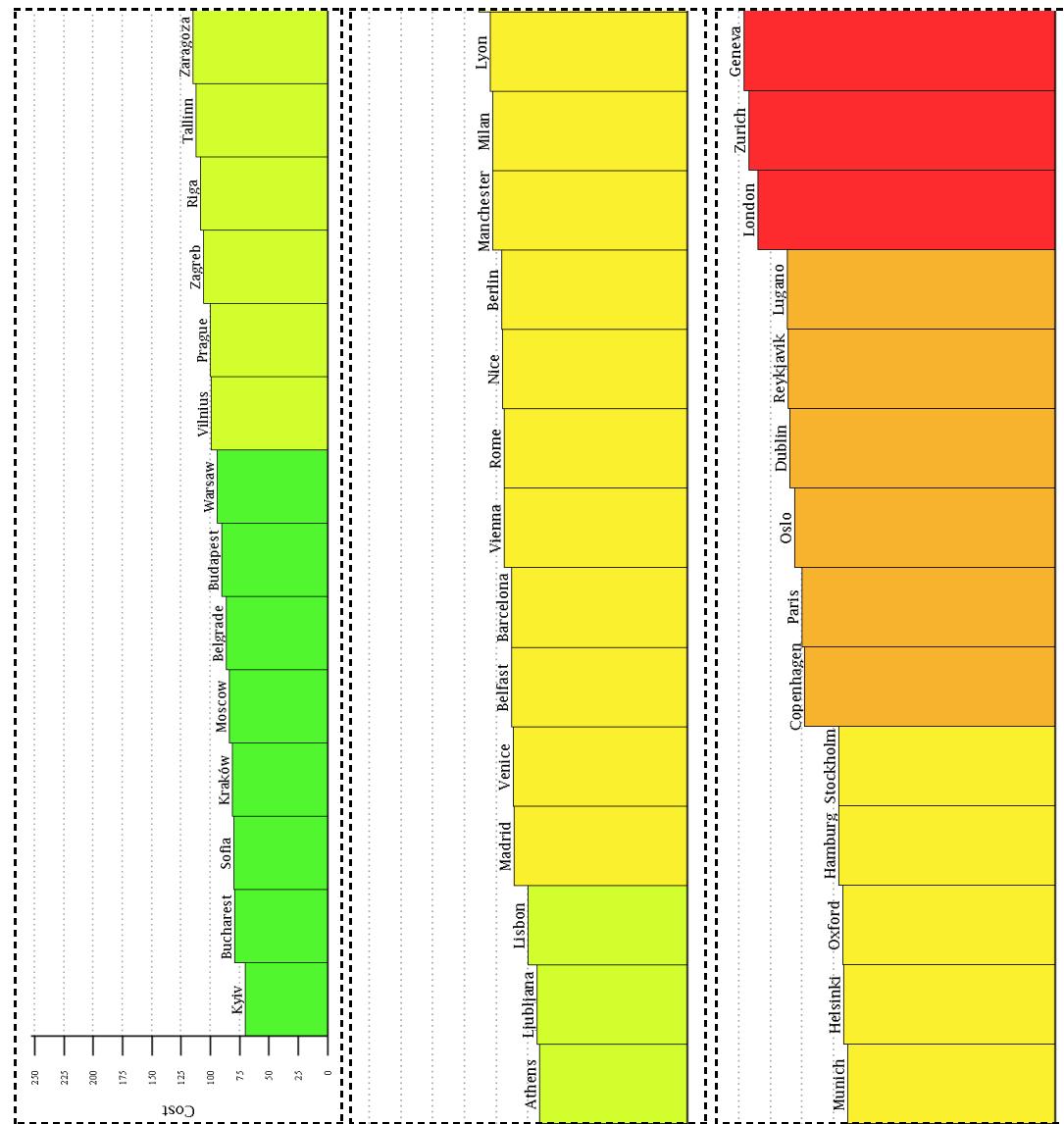


Місячний абонемент у спортзал у діловому районі



"Cluster #1" "Points : 8"	"Borders : from Warsaw (817 hr.) to Berlin (1072 hr.)"	"Difference : 255 hr."
"Cluster #2" "Points : 10"	"Borders : from Prague (1167 hr.) to Vilnius (1460 hr.)"	"Difference : 293 hr."
"Cluster #3" "Points : 14"	"Borders : from Zaragoza (1502 hr.) to Barcelona (1776 hr.)"	"Difference : 274 hr."
"Cluster #4" "Points : 6"	"Borders : from Paris (1843 hr.) to Munich (2218 hr.)"	"Difference : 375 hr."
"Cluster #5" "Points : 4"	"Borders : from Milan (2637 hr.) to Zurich (3442 hr.)"	"Difference : 805 hr."

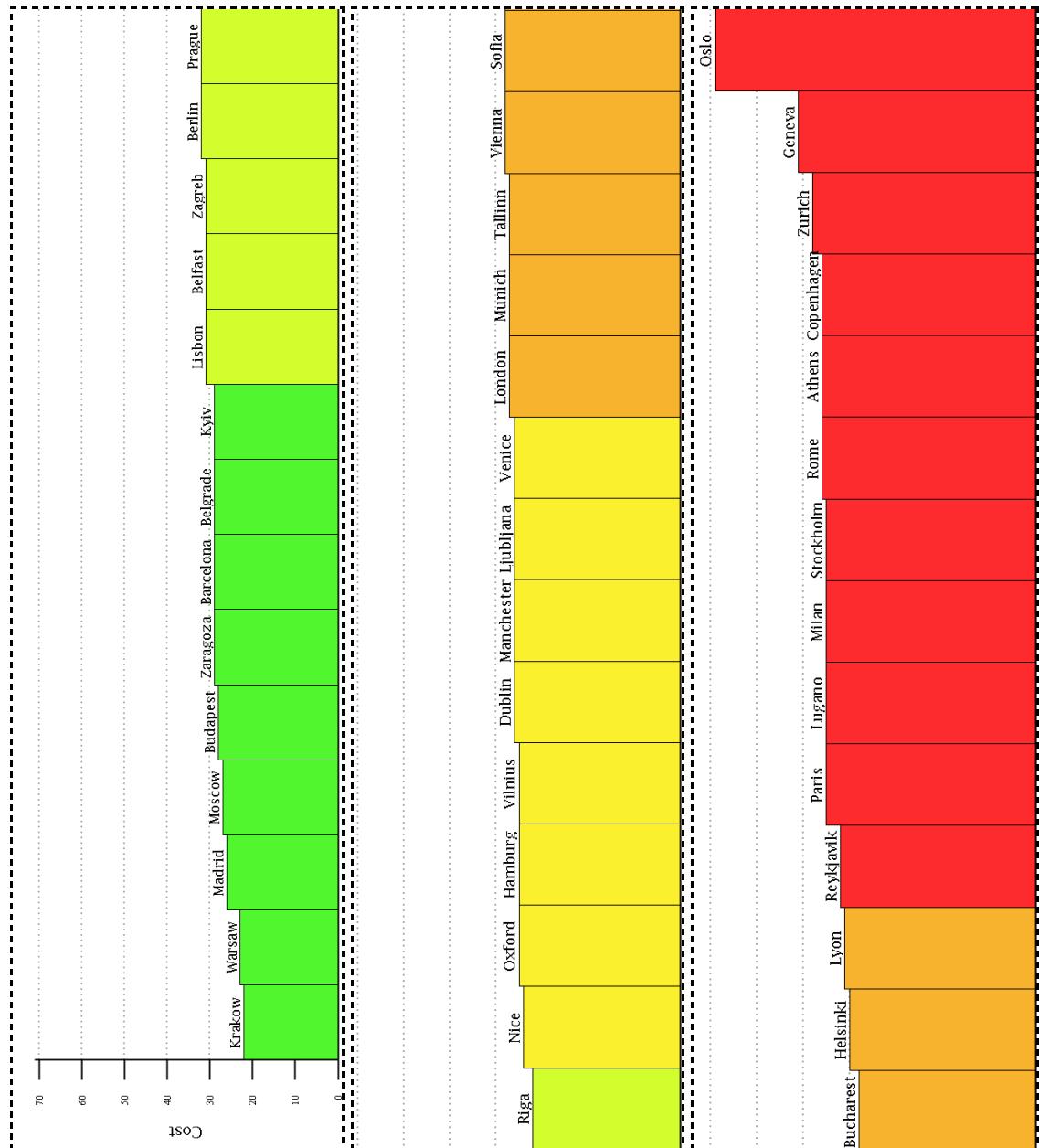
Індекс вартості проживання



"Cluster #1"	"Points : 8"	"Borders : from Kyiv (70 hr.) to Warsaw (94 hr.)"	"Difference : 24 hr."
"Cluster #2"	"Points : 9"	"Borders : from Vilnius (99 hr.) to Lisbon (125 hr.)"	"Difference : 26 hr."
"Cluster #3"	"Points : 16"	"Borders : from Madrid (136 hr.) to Stockholm (171 hr.)"	"Difference : 35 hr."
"Cluster #4"	"Points : 6"	"Borders : from Copenhagen (198 hr.) to Lugano (212 hr.)"	"Difference : 14 hr."
"Cluster #5"	"Points : 3"	"Borders : from London (235 hr.) to Geneva (246 hr.)"	"Difference : 11 hr."

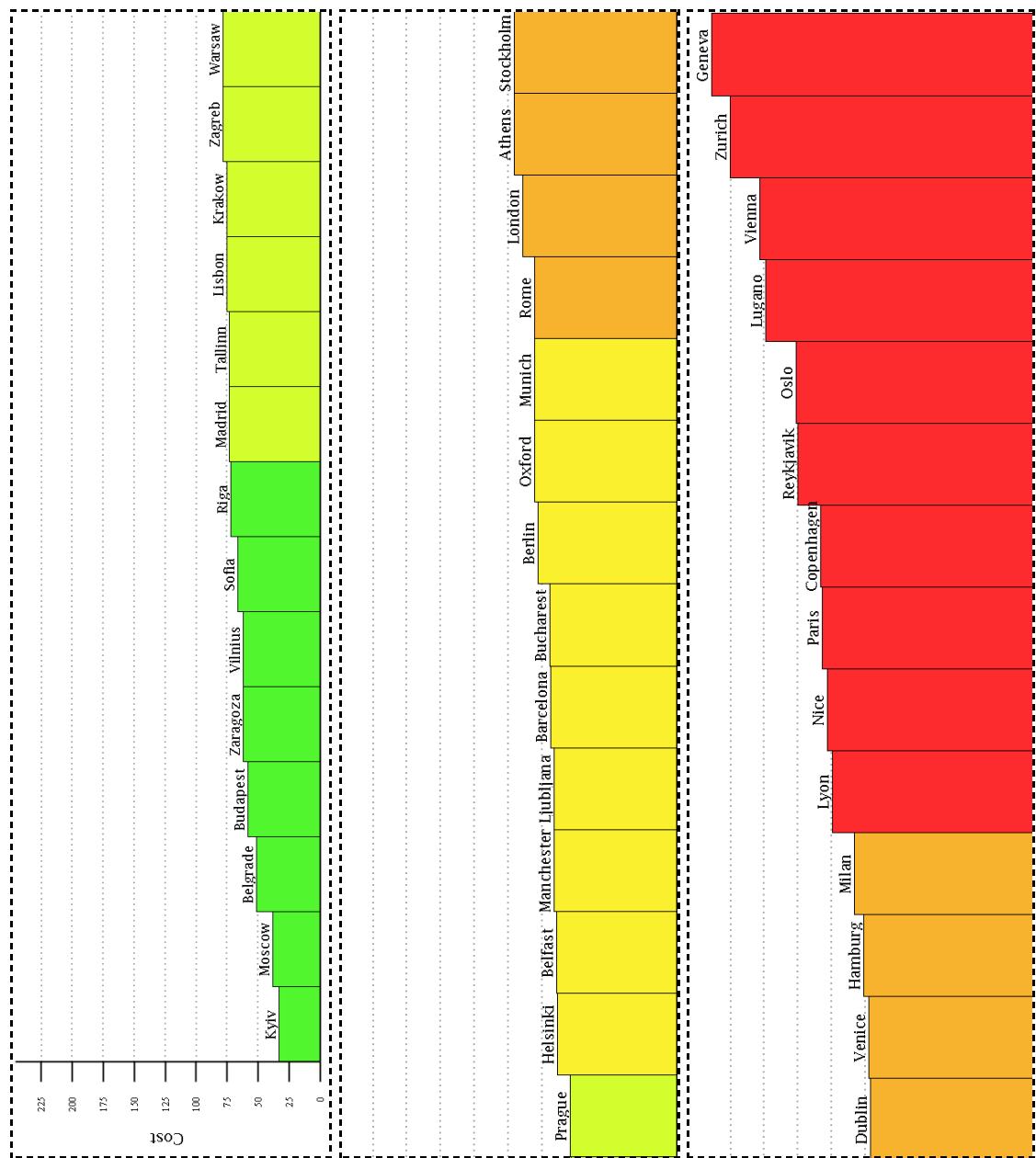
Спектральна кластеризація (5 кластерів)

Середня вартість молока (1 літр)



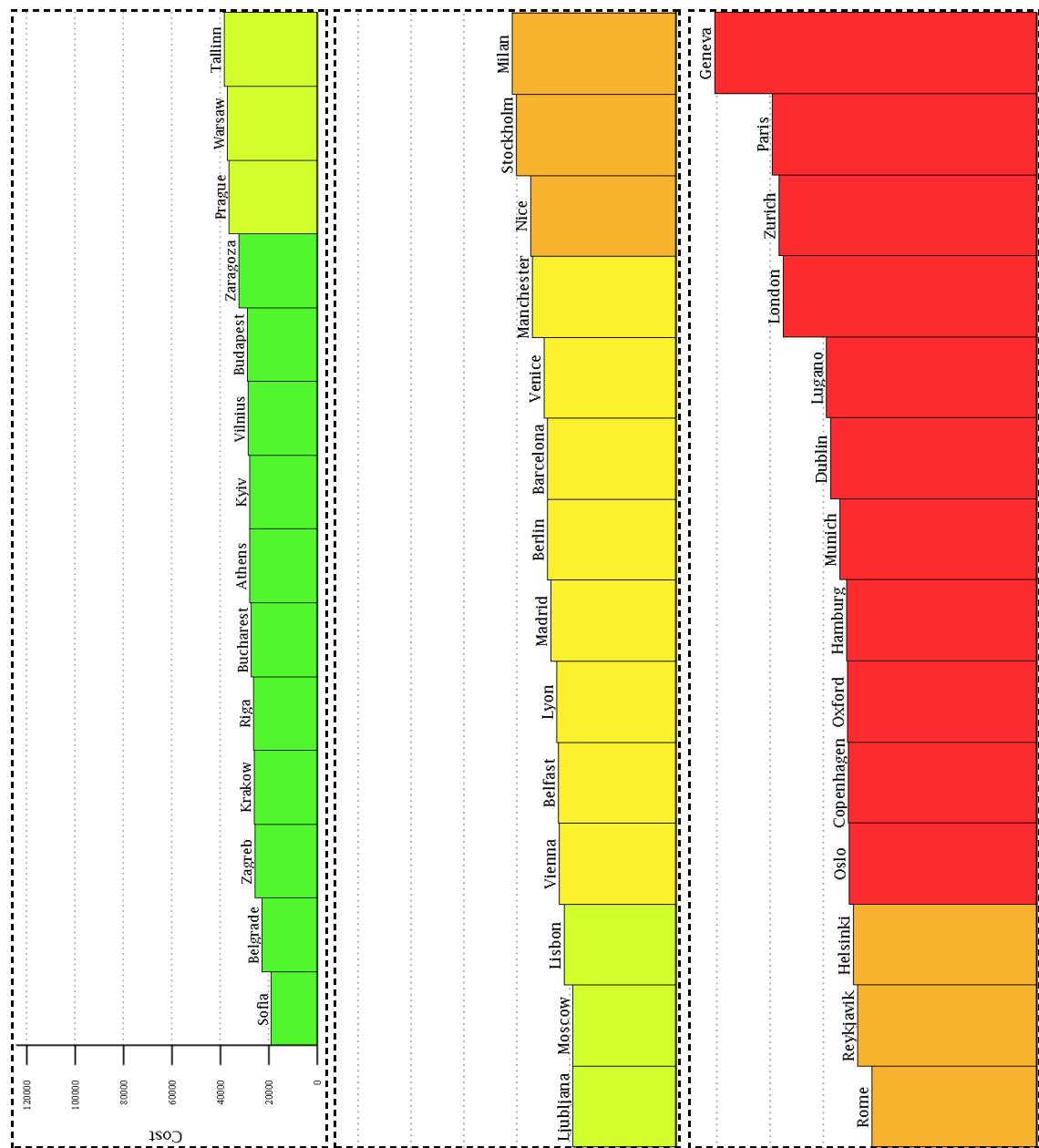
"Cluster #1"	"Points : 9"	"Borders : from Krakow (22 hr.) to Kyiv (29 hr.)"	"Difference : 7 hr."
"Cluster #2"	"Points : 6"	"Borders : from Lisbon (31 hr.) to Riga (32 hr.)"	"Difference : 1 hr."
"Cluster #3"	"Points : 8"	"Borders : from Nice (34 hr.) to Venice (36 hr.)"	"Difference : 2 hr."
"Cluster #4"	"Points : 8"	"Borders : from London (37 hr.) to Lyon (41 hr.)"	"Difference : 4 hr."
"Cluster #5"	"Points : 11"	"Borders : from Reykjavik (42 hr.) to Oslo (69 hr.)"	"Difference : 27 hr."

Середня вартість яєць (12 штук, великі)



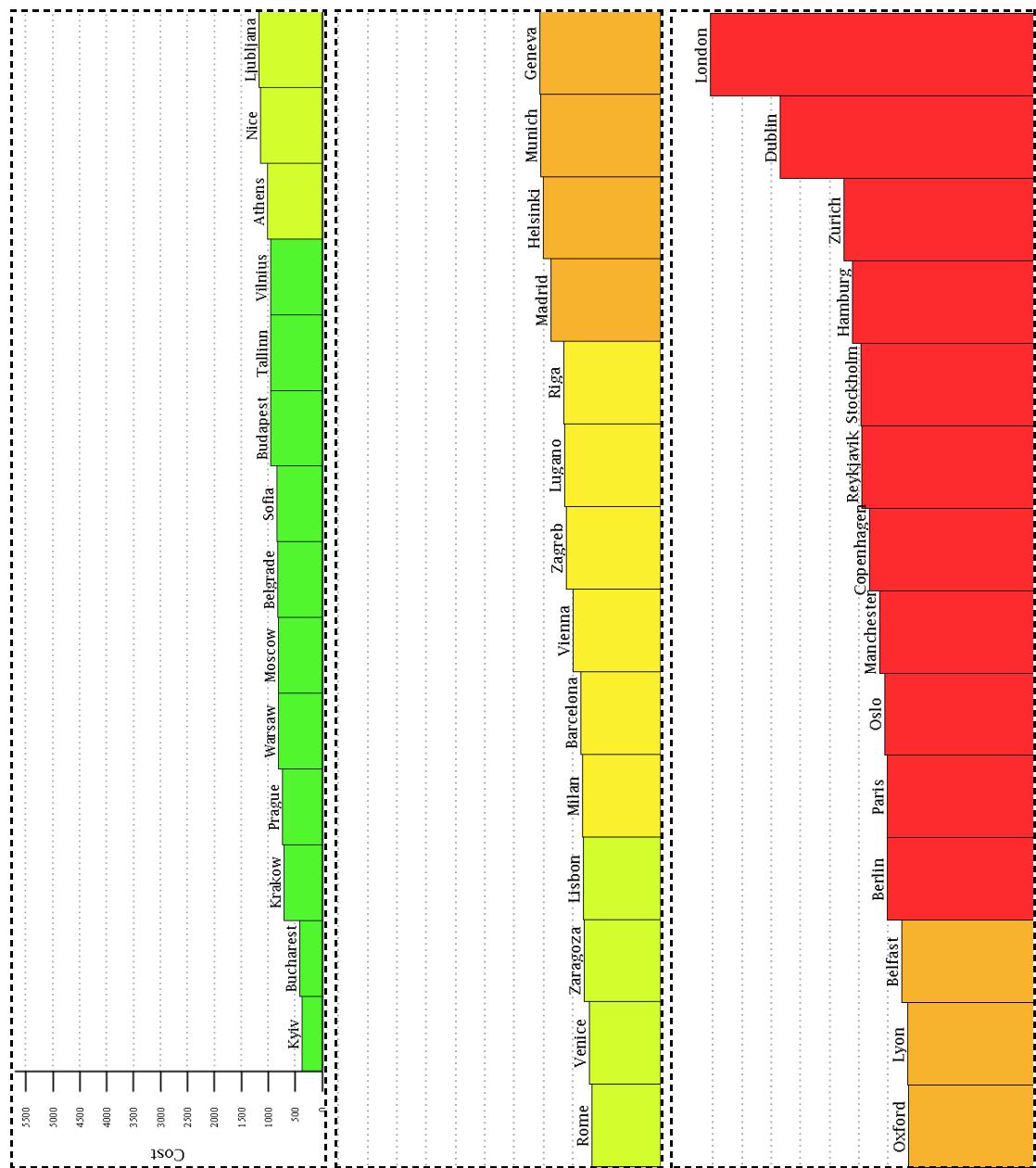
"Cluster #1"	"Points : 8"	"Borders : from Kyiv (33 hr.) to Riga (72 hr.)"	"Difference : 39 hr."
"Cluster #2"	"Points : 7"	"Borders : from Madrid (73 hr.) to Prague (79 hr.)"	"Difference : 6 hr."
"Cluster #3"	"Points : 9"	"Borders : from Helsinki (88 hr.) to Munich (105 hr.)"	"Difference : 17 hr."
"Cluster #4"	"Points : 8"	"Borders : from Rome (105 hr.) to Milan (133 hr.)"	"Difference : 28 hr."
"Cluster #5"	"Points : 10"	"Borders : from Lyon (149 hr.) to Geneva (239 hr.)"	"Difference : 90 hr."

Місячна оренда 85 м² мебльованого житла в дорогому районі



"Cluster #1"	"Points : 11"	"Borders : from Sofia (18905 hr.) to Zaragoza (32162 hr.)"	"Difference : 13257 hr."
"Cluster #2"	"Points : 6"	"Borders : from Prague (36486 hr.) to Lisbon (42061 hr.)"	"Difference : 5575 hr."
"Cluster #3"	"Points : 8"	"Borders : from Vienna (44088 hr.) to Manchester (54266 hr.)"	"Difference : 10178 hr."
"Cluster #4"	"Points : 6"	"Borders : from Nice (54874 hr.) to Helsinki (68787 hr.)"	"Difference : 13913 hr."
"Cluster #5"	"Points : 11"	"Borders : from Oslo (70261 hr.) to Geneva (120788 hr.)"	"Difference : 50527 hr."

Місячний квиток на міський транспорт



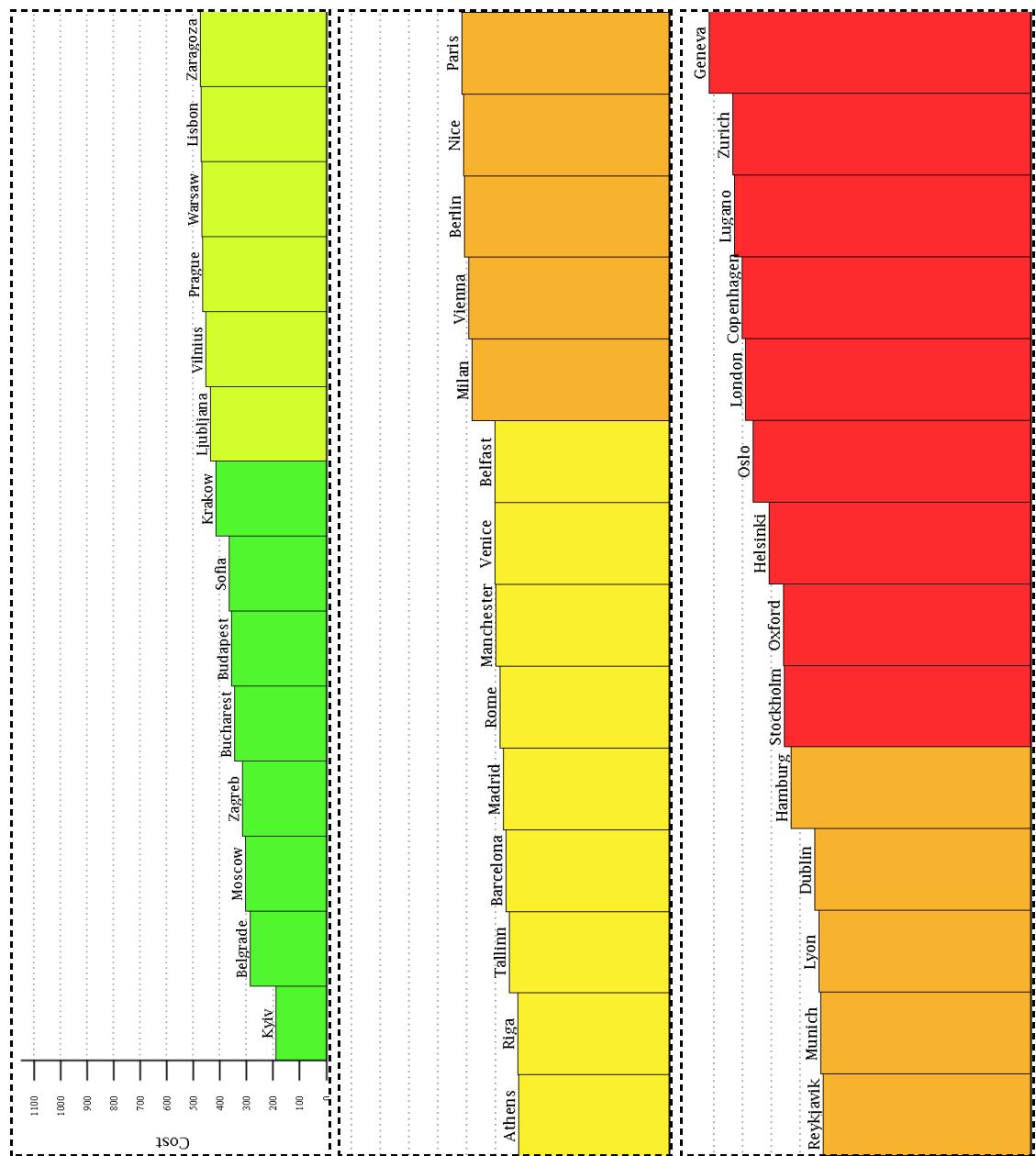
"Cluster #1"	"Points : 11"	"Borders : from Kyiv (376 hr.) to Vilnius (962 hr.)"	"Difference : 586 hr."
"Cluster #2"	"Points : 7"	"Borders : from Athens (1011 hr.) to Lisbon (1313 hr.)"	"Difference : 302 hr."
"Cluster #3"	"Points : 6"	"Borders : from Milan (1335 hr.) to Riga (1658 hr.)"	"Difference : 323 hr."
"Cluster #4"	"Points : 7"	"Borders : from Madrid (1875 hr.) to Belfast (2272 hr.)"	"Difference : 397 hr."
"Cluster #5"	"Points : 11"	"Borders : from Berlin (2508 hr.) to London (5546 hr.)"	"Difference : 3038 hr."

Середня вартість короткого візиту до приватного лікаря (15 хв.)

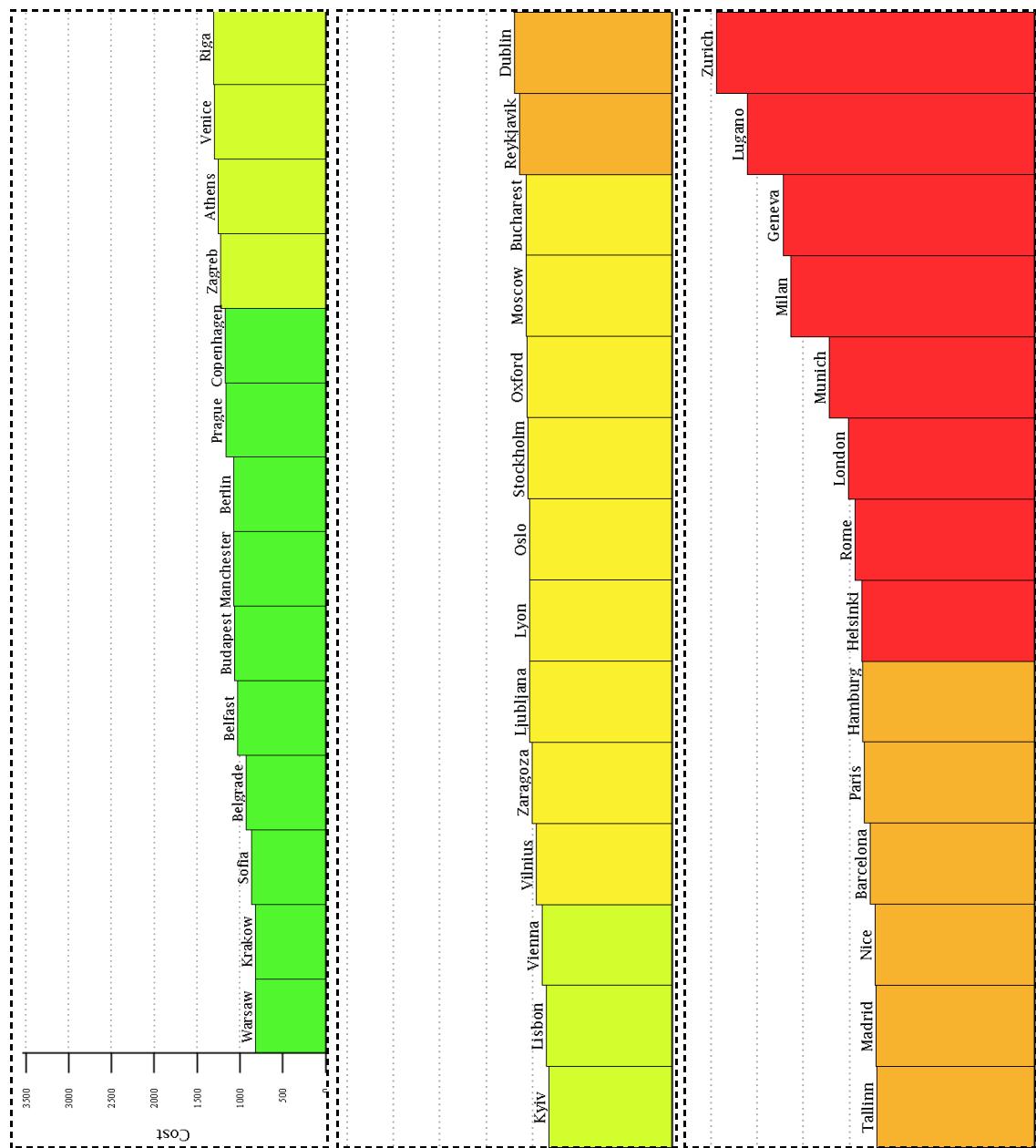


"Cluster #1"	"Points : 11"	"Borders : from Kyiv (367 hr.) to Bucharest (1033 hr.)"	"Difference : 666 hr."
"Cluster #2"	"Points : 7"	"Borders : from Warsaw (1033 hr.) to Reykjavik (1522 hr.)"	"Difference : 489 hr."
"Cluster #3"	"Points : 8"	"Borders : from Lisbon (1591 hr.) to Dublin (1954 hr.)"	"Difference : 363 hr."
"Cluster #4"	"Points : 9"	"Borders : from Barcelona (2390 hr.) to Venice (3073 hr.)"	"Difference : 683 hr."
"Cluster #5"	"Points : 7"	"Borders : from Lugano (3392 hr.) to London (4055 hr.)"	"Difference : 663 hr."

Середня вартість 2 квитків у кіно

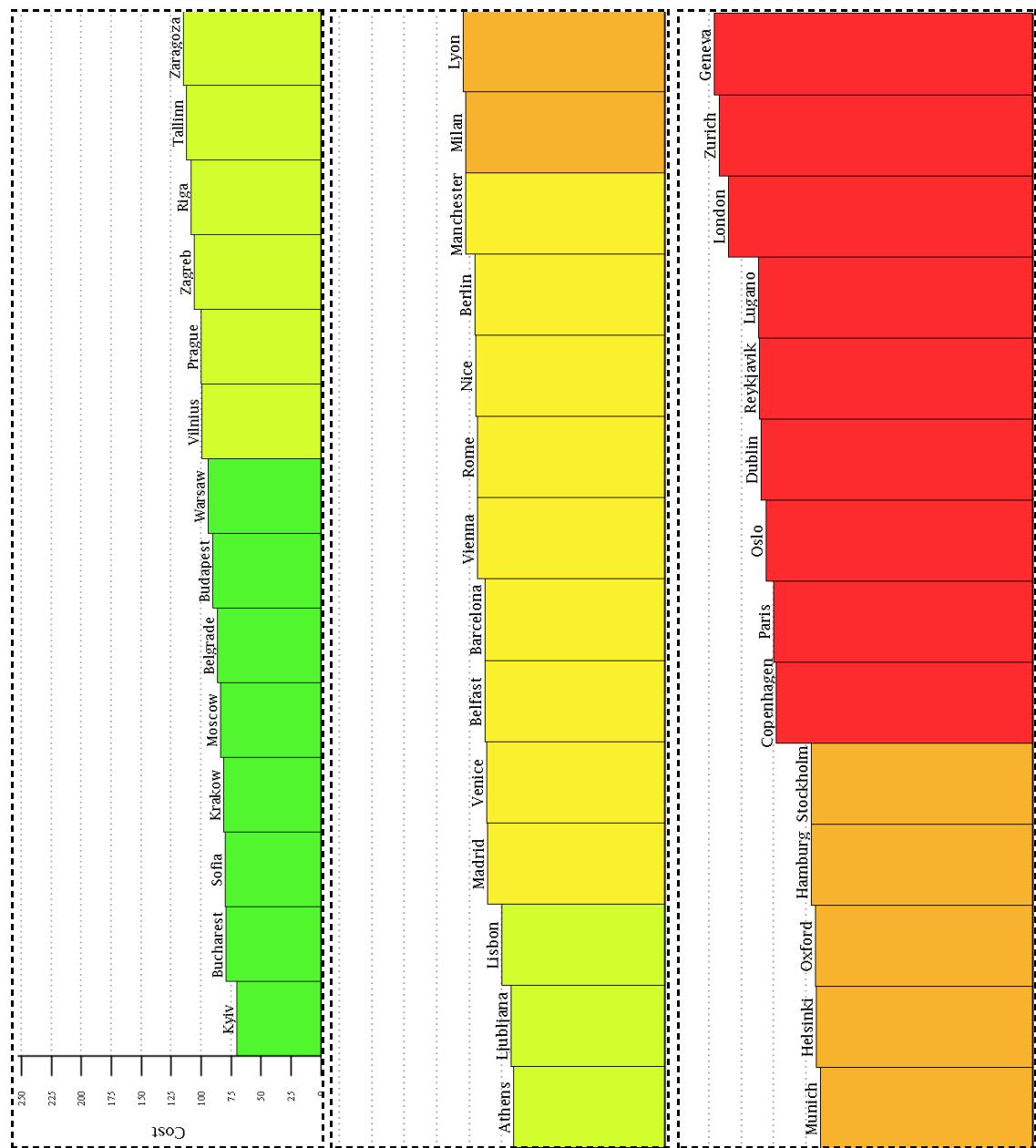


Місячний абонемент у спортзал у діловому районі



"Cluster #1"	"Points : 10"	"Borders : from Warsaw (817 hr.) to Copenhagen (1169 hr.)"	"Difference : 352 hr."
"Cluster #2"	"Points : 7"	"Borders : from Zagreb (1229 hr.) to Vienna (1394 hr.)"	"Difference : 165 hr."
"Cluster #3"	"Points : 9"	"Borders : from Vilnius (1460 hr.) to Bucharest (1574 hr.)"	"Difference : 114 hr."
"Cluster #4"	"Points : 8"	"Borders : from Reykjavik (1638 hr.) to Hamburg (1858 hr.)"	"Difference : 220 hr."
"Cluster #5"	"Points : 8"	"Borders : from Helsinki (1870 hr.) to Zurich (3442 hr.)"	"Difference : 1572 hr."

Індекс вартості проживання



"Cluster #1"	"Points : 8"	"Borders : from Kyiv (70 hr.) to Warsaw (94 hr.)"	"Difference : 24 hr."
"Cluster #2"	"Points : 9"	"Borders : from Vilnius (99 hr.) to Lisbon (125 hr.)"	"Difference : 26 hr."
"Cluster #3"	"Points : 9"	"Borders : from Madrid (136 hr.) to Manchester (153 hr.)"	"Difference : 17 hr."
"Cluster #4"	"Points : 7"	"Borders : from Milan (153 hr.) to Stockholm (171 hr.)"	"Difference : 18 hr."
"Cluster #5"	"Points : 9"	"Borders : from Copenhagen (198 hr.) to Geneva (246 hr.)"	"Difference : 48 hr."

Додаток В. Код реалізації алгоритмів

K-means

Преамбула

```
> restart;
with(LinearAlgebra):
> read "Coursework IOData.mpl": # Пакет містить засоби для імпортування та відображення даних
read "Coursework Subsidiary.mpl": # Пакет з додатковим контентом
```

Початкові дії

Підвантаємо таблицю Excel
 > Points := ImportTabular("Tickets");

(1)

$$\text{Points} := \begin{bmatrix} & 42 \times 2 \text{ Matrix} \\ & \text{Data Type: anything} \\ & \text{Storage: rectangular} \\ & \text{Order: Fortran_order} \end{bmatrix}$$

Обробляємо початкові дані

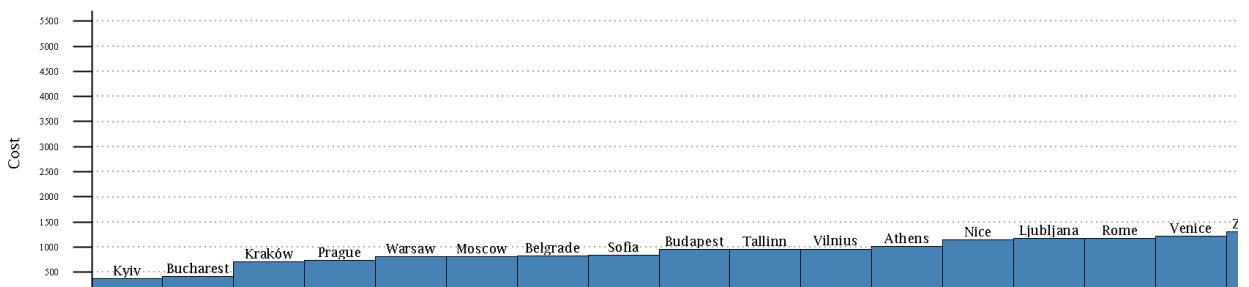
```
> centroidsNumber := 5; # Кількість центроїдів можна змінювати
\xi := 0.001; # Точність ітераційного процесу
portrayCaption := "": # Caption for pictures

Centroids := Matrix(centroidsNumber, 2):
pointsNumber := rowdim(Points);
centroidsNumber := 5
\xi := 0.001
pointsNumber := 42
```

(2)

Відообразимо початкові дані

```
> PortrayHistogram(SortMatrixByColumn(Points, 2), portrayCaption);
```



Вибір початкового наближення центроїдів

Два перших центроїди: шукаємо дві найвіддаленіші точки множини

```
> getDistance := (point1, point2) -> abs(point1[2] - point2[2]):
maxDistance := 0:
for i from 1 to pointsNumber - 1 do
    for j from i + 1 to pointsNumber do
        newDistance := getDistance(Points[i], Points[j]):
        if (maxDistance < newDistance) then maxDistance := newDistance : pIndex1 := i : pIndex2 := j : end if:
    end do:
end do:
Centroids[1] := Points[pIndex1];
Centroids[2] := Points[pIndex2];
Centroids1 := [ "London" 5546.0 ]
Centroids2 := [ "Kyiv" 376.0 ]
```

(3)

Усі інші центроїди

```
> for k from 3 to centroidsNumber do
    maxMinDistance := 0:
    for i from 1 to pointsNumber do
        minDistance := getDistance(Points[i], Centroids[1]):
        for j from 2 to k - 1 do
            newDistance := getDistance(Points[i], Centroids[j]):
            if (newDistance < minDistance) then minDistance := newDistance : end if:
        end do:
```

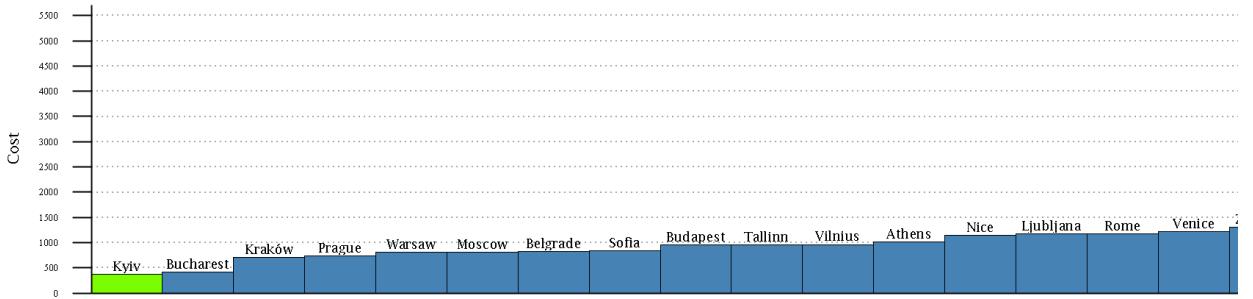
```

    if (minDistance > maxMinDistance) then maxMinDistance := minDistance : pIndex := i: end if;
end do;
Centroids[k] := Points[pIndex];
end do;

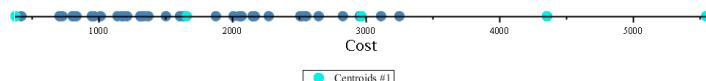
```

Виведемо отримані результати

```
> PortrayHistogramByCentroids(SortMatrixByColumn(Points, 2), Centroids, portrayCaption);
```



```
> PortrayPointsByCentroids(Points, [Centroids], portrayCaption);
```



Визначення кластерів (перше наближення)

Створюємо масив ClustersDistribution, який для кожної точки зберігає номер відповідного їй кластеру

```
> ClustersDistribution := Array(1 .. pointsNumber) :
```

Створюємо масив ClustersSizes, який для кожного кластеру зберігає його розмір

```
> ClustersSizes := Array(1 .. centroidsNumber) :
```

Для кожної точки визначаєм кластер та заповнюємо масив ClustersDistribution

```

> for i from 1 to pointsNumber do
    minDistance := getDistance(Points[i], Centroids[1]);
    cluster := 1;
    for j from 2 to centroidsNumber do
        newDistance := getDistance(Points[i], Centroids[j]);
        if (newDistance < minDistance) then minDistance := newDistance: cluster := j: end if;
    end do;
    ClustersDistribution[i] := cluster;
    ClustersSizes[cluster] := (ClustersSizes[cluster] + 1);
end do;

```

Створюємо набір масивів Clusters, в якому будуть зберігатися поділені по кластерам точки

```
> Clusters := [seq(Matrix(ClustersSizes[i], 2), i = 1 .. centroidsNumber)];
```

$$\text{Clusters} := \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \right], \left[\begin{array}{cc} 12 & 2 \\ \text{Data Type: anything} & \\ \text{Storage: rectangular} & \\ \text{Order: Fortran_order} & \end{array} \right], \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array} \right], \left[\begin{array}{cc} 19 & 2 \\ \text{Data Type: anything} & \\ \text{Storage: rectangular} & \\ \text{Order: Fortran_order} & \end{array} \right], \left[\begin{array}{cc} 0 & 0 \end{array} \right]$$
(4)

Створюємо масив ClustersFilling, в якому зберігається кількість точок, що наразі знаходиться у відповідному кластері

```
> ClustersFilling := Array(1 .. centroidsNumber);
```

$$\text{ClustersFilling} := [0 \ 0 \ 0 \ 0 \ 0]$$
(5)

Заповнюємо Clusters

```

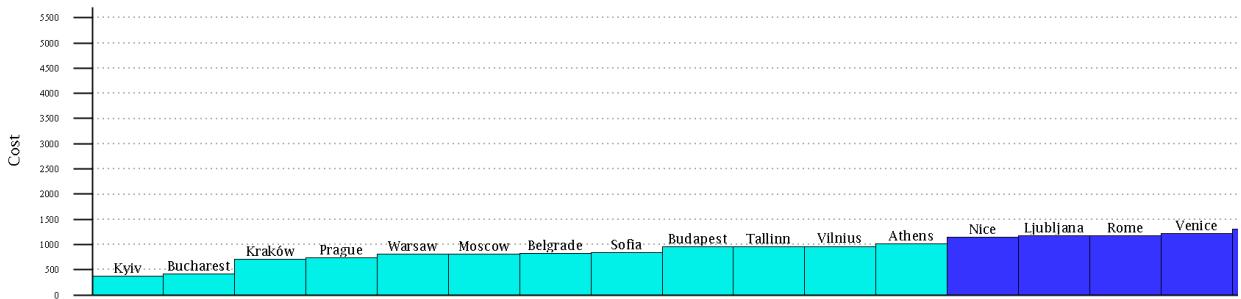
> for i from 1 to pointsNumber do
    Clusters[ClustersDistribution[i]][(ClustersFilling[ClustersDistribution[i]] + 1)] := Points[i];
    ClustersFilling[ClustersDistribution[i]] := ClustersFilling[ClustersDistribution[i]] + 1;
end do;

```

```
    end do:
```

Відображення результатів початкового наближення

```
> PortrayHistogramByClusters(SortClusters(Clusters), "CMR", portrayCaption);
```



Визначимо нові центроїди

Створюємо контейнер для збереження значень нових центроїдів

```
> NewCentroids := Matrix(centroidsNumber, 2) :
```

Знаходимо нові центроїди

```
> for i from 1 to centroidsNumber do
    pointsSum := 0 : # Тимчасова змінна, в якій буде зберігатися сума точок, що належать одному кластеру
```

```
    for j from 1 to ClustersSizes[i] do
        pointsSum := pointsSum + Clusters[i][j, 2] :
```

```
end do:
```

Суму значень усіх точок одного кластеру ділимо на загальну кількість точок кластеру

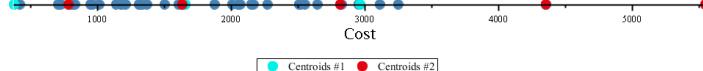
```
NewCentroids[i, 1] := "NewCentroids" :
```

```
NewCentroids[i, 2] :=  $\frac{\text{pointsSum}}{\text{ClustersSizes}[i]}$  :
```

```
end do:
```

Подивимось, як це буде виглядати тепер

```
> PortrayPointsByCentroids(Points, [Centroids, NewCentroids], portrayCaption);
```



Оновлюємо вміст кластерів

Анульовуємо масив ClusterSizes, який для кожного кластеру зберігає його розмір

```
> ClustersSizes := Array(1 .. centroidsNumber) :
```

Для кожної точки визначаємо кластер та заповнюємо масив ClusterDistribution

```
> for i from 1 to pointsNumber do
```

```
    minDistance := getDistance(Points[i], NewCentroids[1]) :
```

```
    cluster := 1 :
```

```
    for j from 2 to centroidsNumber do
```

```
        newDistance := getDistance(Points[i], NewCentroids[j]) :
```

```
        if (newDistance < minDistance) then minDistance := newDistance : cluster := j : end if:
```

```
    end do:
```

```
    ClusterDistribution[i] := cluster :
```

```
    ClustersSizes[cluster] := (ClustersSizes[cluster] + 1) :
```

```
end do:
```

Оновлюємо набір масивів Clusters, в якому будуть зберігатися поділені по кластерам точки

```
> Clusters := [seq(Matrix(ClustersSizes[i], 2), i = 1 .. centroidsNumber)] :
```

Анульовуємо масив ClustersFilling, в якому зберігається кількість точок, що наразі знаходиться у відповідному кластері

```
> ClustersFilling := Array(1 .. centroidsNumber) :
```

Заповнюємо Clusters

```
> for i from 1 to pointsNumber do
```

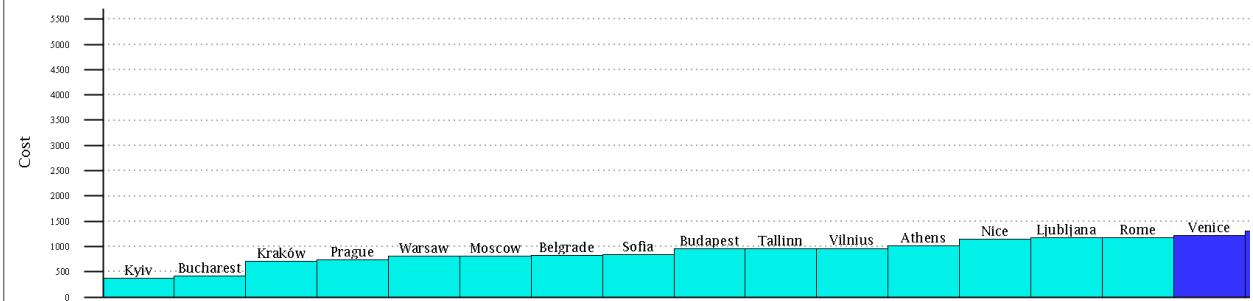
```
    Clusters[ClusterDistribution[i]][(ClustersFilling[ClusterDistribution[i]] + 1)] := Points[i] :
```

```
    ClustersFilling[ClusterDistribution[i]] := ClustersFilling[ClusterDistribution[i]] + 1 :
```

```
end do:
```

Подивимось на результати

> *PortrayHistogramByClusters(SortClusters(Clusters), "CMR", portrayCaption);*



Продовжуємо ітераційно оновлювати центроїди та вміст кластерів, поки не виконається умова збіжності

```
> while (( ( VectorNorm( Vector([ seq(NewCentroids[i, 2], i = 1 .. centroidsNumber) ]) ) - Vector([ seq(Centroids[i, 2], i = 1 .. centroidsNumber) ]) ) / VectorNorm( Vector([ seq(Centroids[i, 2], i = 1 .. centroidsNumber) ]) ) ) > ξ ) do

    # Тепер нові значення центроїдів переходять у статус старих
    for i from 1 to centroidsNumber do
        Centroids[i] := NewCentroids[i]:
    end do:

    # Знаходимо нові центроїди
    for k from 1 to centroidsNumber do
        pointsSum := 0: # Тимчасова змінна, в якій буде зберігатися сума точок, що належать одному кластеру
        for i from 1 to ClustersSizes[k] do
            pointsSum := pointsSum + Clusters[k][i, 2]:
        end do:

        # Суму координат усіх точок одного кластеру ділимо на загальну кількість точок кластеру
        NewCentroids[k, 2] := pointsSum / ClustersSizes[k]:
    end do:

    # Анульуємо масив ClustersSizes, який для кожного кластеру зберігає його розмір
    ClustersSizes := Array(1 .. centroidsNumber):

    # Для кожної точки визначаем кластер та заповнюємо масив ClusterDistribution
    for i from 1 to pointsNumber do
        minDistance := getDistance(Points[i], NewCentroids[1]):
        cluster := 1:
        for j from 2 to centroidsNumber do
            newDistance := getDistance(Points[i], NewCentroids[j]):
            if (newDistance < minDistance) then minDistance := newDistance: cluster := j: end if:
        end do:
        ClusterDistribution[i] := cluster:

        ClustersSizes[cluster] := (ClustersSizes[cluster] + 1):
    end do:

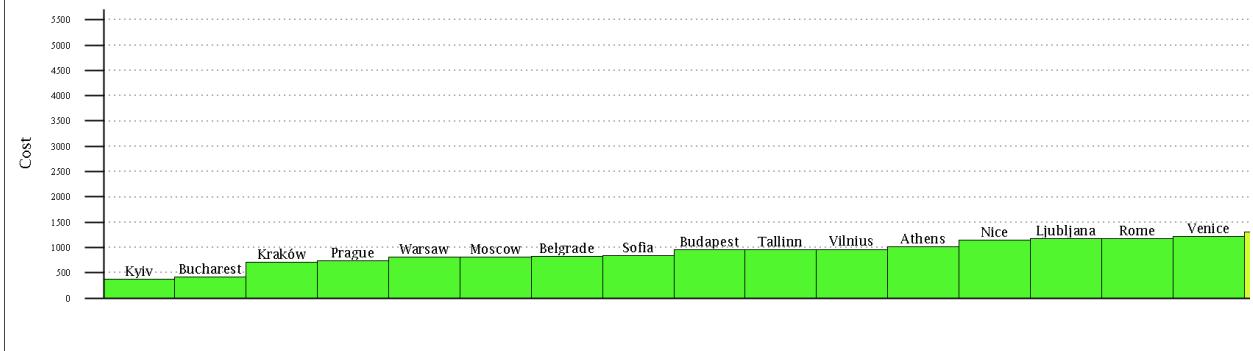
    # Оновлюємо набір масивів Clusters, в якому будуть зберігатися поділені по кластерам точки
    Clusters := [ seq(Matrix(ClustersSizes[i], 2), i = 1 .. centroidsNumber) ]:
    # Анульуємо масив ClustersFilling, в якому зберігається кількість точок, що наразі знаходиться у відповідному кластері
    ClustersFilling := Array(1 .. centroidsNumber):

    # Заповнюємо Clusters
    for i from 1 to pointsNumber do
        Clusters[ClusterDistribution[i]][(ClustersFilling[ClusterDistribution[i]] + 1)] := Points[i]:
        ClustersFilling[ClusterDistribution[i]] := ClustersFilling[ClusterDistribution[i]] + 1:
    end do:
end do:
```

Результатами

Виведено відсортовану гістограму

```
> PortrayHistogramByClusters(SortClusters(Clusters), "GYR", portrayCaption);
```



Відообразимо граничні значення кластерів та різницю між ними

```
> Matrix(GetInfo(Clusters));
```

"Cluster #1" "Points : 16" "Borders : from Kyiv (376 hr.) to Venice (1211 hr.)" "Difference : 835 hr."
"Cluster #2" "Points : 14" "Borders : from Zaragoza (1309 hr.) to Lyon (2166 hr.)" "Difference : 857 hr."
"Cluster #3" "Points : 10" "Borders : from Belfast (2272 hr.) to Zurich (3251 hr.)" "Difference : 979 hr."
"Cluster #4" "Points : 1" "Borders : from Dublin (4353 hr.) to Dublin (4353 hr.)" "Difference : 0 hr."
"Cluster #5" "Points : 1" "Borders : from London (5546 hr.) to London (5546 hr.)" "Difference : 0 hr."

(6)

Spectral clustering

Преамбула

```
> restart;
with(LinearAlgebra):
with(GraphTheory):
> read "IOData.mpl":
read "Subsidiary.mpl":
```

Початкові дані

```
> clustersNumber := 5:
gaussianSimilarity := (x, y, σ) → exp(- $\frac{(x - y)^2}{2 \sigma^2}$ ):
portrayCaption := "":
```

Будуємо матрицю подібності

Підганяємо матрицю з Excel

```
> DataPoints := ImportTabular("Living index");
```

DataPoints :=	42×2 Matrix
	Data Type: anything
	Storage: rectangular
	Order: Fortran_order

(1)

Ініціалізуємо матрицю подібності та заповнюємо її

```
> pointsNumber := rovdim(DataPoints):
SimilarityMatrix := Matrix(pointsNumber + 1, pointsNumber + 1):
# Заповнюємо значеннями всієї складності
for i from 1 to pointsNumber do
    for j from 1 to pointsNumber do
        if (i ≠ j) then SimilarityMatrix[i, j] := gaussianSimilarity(currentPoint, DataPoints(j, 2), 1): end if:
    end do:
end do:
# Додаємо індекси точок, щоб потім можна їх відрізнити
for i from 1 to pointsNumber do
    SimilarityMatrix[i, pointsNumber + 1] := DataPoints[i, 1]:
    SimilarityMatrix[pointsNumber + 1, i] := DataPoints[i, 1]:
end do:
SimilarityMatrix :
```

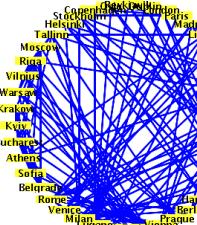
Будуємо граф подібності

Функція формує інформацію для побудови графа k -найближчих сусідів
 Приймає: SimilarityMatrix -- матрицю подібності, k -- кількість сусідів
 Повертає: V -- лист вершин, E -- множину ребер

```
> KNearest := proc(SimilarityMatrix :: Matrix, k :: integer)
local Adjacency, i, V, E;
# Перший елемент -- парівноважливий, усі інші --  $k$ -сусіди
Adjacency := Matrix(pointsNumber, k + 1) :
Adjacency[.., 1] := DataPoints[.., 1];
for i from 1 to pointsNumber do
Adjacency[i, 2..k + 1] := SortMatrixByRow(SimilarityMatrix, i)[pointsNumber + 1, pointsNumber - k + 1 .. pointsNumber];
end do;
V := convert(SimilarityMatrix[pointsNumber + 1, 1 .. pointsNumber], list) :
E := {seq(seq({Adjacency[i, 1], Adjacency[i, j]}, j = 2 .. k + 1), i = 1 .. pointsNumber)} :
return V, E;
end proc;
```

Будуємо граф подібності G

```
> G := Graph(KNearest(SimilarityMatrix, 6)) :
DrawGraph(G)
```



Будуємо матрицю Кірхгофа

Будуємо матрицю суміжності W графа схожості

```
> W := AdjacencyMatrix(G) :
```

Будуємо матрицю степенів D

```
> Deg := Matrix(pointsNumber, pointsNumber) :
for i from 1 to pointsNumber do
Deg[i, i] := Degree(G, DataPoints[i, 1]) ;
end do;
```

Будуємо матрицю Кірхгофа $L = D - W$

```
> L := Deg - W;
```

Нормалізована матриця Кірхгофа випадкового блокання

```
> DegInv := LinearAlgebra[MatrixInverse](Deg) :
Lrw := Multiply(DegInv, L) :
L := Lrw ;
```

Знаходимо перші k власних векторів

Визначаємо власні числа та вектори

```
> # EVValues, EVectors := evalf(Eigenvectors(L)) :
```

Експортуємо матрицю Кірхгофа для знаходження власних векторів

```
> # Адаптуємо дані до таблиці Python
ExcelL := Matrix(pointsNumber + 1, pointsNumber) :
ExcelL[1..] := DataPoints[.., 1] :
ExcelL[2..pointsNumber + 1..] := L :
# Експортуємо в Excel
ExcelTools:-Export(ExcelL, "Laplacian.xlsx") :
```

Імпортуюмо власні числа та вектори

```
> EVValues := Matrix(ExcelTools:-Import("D:\Diploma\Eigenvalues.xlsx")) :
EVectors := Matrix(ExcelTools:-Import("D:\Diploma\Eigenvectors.xlsx")) :
```

Імпортуюмо власні числа та вектори

```
> EVValues := Matrix(ExcelTools:-Import("D:\Diploma\Eigenvalues.xlsx")) :
EVectors := Matrix(ExcelTools:-Import("D:\Diploma\Eigenvectors.xlsx")) :
```

Просортируємо вектори за зростанням відповідних власних чисел

```
> # Порядкуємо числа та вектори
Eigenspace := Matrix(pointsNumber + 1, pointsNumber) :
Eigenspace[1..-2..] := evalf(EVectors) :
# Eigenspace[ pointsNumber + 1..] := evalf(EVValues) :
Eigenspace[ pointsNumber + 1..] := EVValues[.., 1] :
# Сортируємо за власними числами
Eigenspace := SortMatrixByRow(Eigenspace, pointsNumber + 1) :
```

Формуємо точки для класифікації за допомогою K -means

```
> pointsDimension := clustersNumber :
Points := Matrix(pointsNumber, pointsDimension + 1) :
Points[.., pointsDimension + 1] := DataPoints[.., 1] : # Прив'язуємо назви точок
Points[.., 1 .. pointsDimension] := Eigenspace[1 .. pointsNumber, 1 .. pointsDimension] :
Points;
```

Browse Matrix						
Table		Image		Options		
1	2	3	4	5	6	
1	451826... .576191... .66566... .67482... .17223... "Reykjav...					
2	451826... .576191... .66566... .67482... .17223... "Dublin"					
3	433662... .562703... .67524... .77224... .18269... "London"					
4	518189... .613176... .58873... .21785... .09168... "Paris"					
5	452189... .079904... .513884... .070993... .72112... "Madrid"					
6	512709... .29474... .191749... .756114... .20439... "Lisbon"					
7	420033... .119645... .522018... .-0.2226... .-73229... "Belfast"					
8	584370... .243255... .707243... .-18147... .-25731... "Manche...					
9	.368707... .222002... .398861... .-20055... .811813... "Oxford"					
10	.416984... .-38670... .-0.3736... .811468... .134489... "Zarago...					
11	.433838... .130185... .540420... .-0.3717... .-70809... "Barcelo...					
12	.519475... .182470... .632889... .-0.9411... .53566... "Nice"					
13	.520194... .272124... .543736... .-23133... .553334... "Lyon"					
14	.433662... .562703... .-67524... .0.77224... .-18269... "Geneva"					
15	.433662... .562703... .-67524... .0.77224... .18269... "Zurich"					

42 x 6 Matrix
Data Type: anything
Storage: rectangular
Order: Fortran_order

Insert Export Done

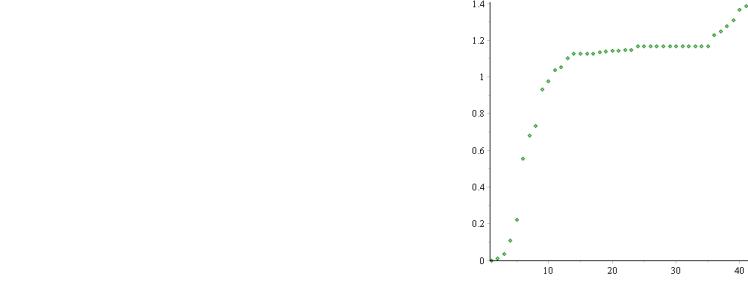
(2)

```

> # Нормалізуємо рядки
for i from 1 to pointsNumber do
  rowNorm := evalf(Norm(Points[i, 1 ..pointsDimension], 2)) :
  for j from 1 to pointsDimension do
    Points[i,j] := Points[i,j] / rowNorm ;
  end do;
end do;

- Відобразимо власні числа на графіку
> plot([seq]( [i, Eigenspace[pointsNumber + 1, i]], i = 1 ..pointsNumber), style=point, color="Green")

```



```

> plot([seq]( [i, Eigenspace[pointsNumber + 1, i]], i = 1 ..clustersNumber), symbol=solidcircle, symbolsize=20, style=point, color="SteelBlue")

```

Відобразимо перші к власні вектори на графіку

```

> for j from 1 to 3 do
  plot([seq]( [i, Points[i, j]], i = 1 ..pointsNumber))
end do;

```

```
> d1 := plot([seq]( [i, Points[i, 1]], i = 1 ..pointsNumber), color="LawnGreen", thickness=5) :
```

```
d2 := plot([seq]( [i, Points[i, 2]], i = 1 ..pointsNumber), color="Fuchsia", thickness=5) :
```

```
d3 := plot([seq]( [i, Points[i, 3]], i = 1 ..pointsNumber), color="Cyan", thickness=5) :
```

```
d4 := plot([seq]( [i, Points[i, 4]], i = 1 ..pointsNumber), color="Red", thickness=5) :
```

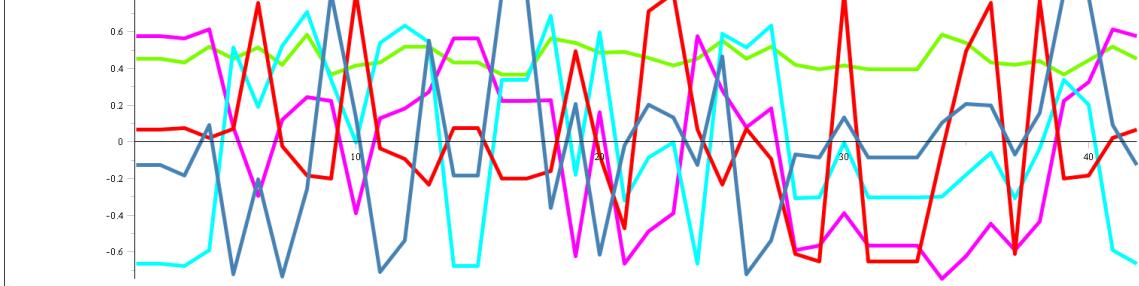
```
d5 := plot([seq]( [i, Points[i, 5]], i = 1 ..pointsNumber), color="SteelBlue", thickness=5) :
```

```
#d6:=plot([seq]( [i, Points[i, 6]], i = 1 ..pointsNumber), color="Green", thickness=5) :
```

```
#d7:=plot([seq]( [i, Points[i, 7]], i = 1 ..pointsNumber), color="Green", thickness=5) :
```

```
> display( {d1, d2, d3, d4, d5} );

```



Кластеризуємо отримані точки за допомогою K-менс

Початкові дані

```
> ξ := 0.001 : # Точність ітераційного процесу
```

Функція для визначення відстані в n-вимірному просторі

Приймає : координати точок + назва

Повертає : відстань між точками

```

> GetDistance := proc(Point1 :: Vector, Point2 :: Vector)
  local i, sum ;
  sum := 0 ;
  for i from 1 to Dimension(Point1) - 1 do
    sum := sum + (Point1[i] - Point2[i])^2 ;
  end do;
  return evalf(sqrt(sum)) ;
end proc;

```

Перши два центроїди : іщуємо дві найвіддаленіші точки мноожини

```

> maxDistance := 0 :
for i from 1 to pointsNumber - 1 do
    for j from i + 1 to pointsNumber do
        newDistance := GetDistance(Points[i..], Points[j..]) :
        if (maxDistance < newDistance) then maxDistance := newDistance : pIndex1 := i : pIndex2 := j : end if:
    end do:
end do:
Centroids := Matrix(clustersNumber, pointsDimension + 1) :
Centroids[1..] := Points[pIndex1..];
Centroids[2..] := Points[pIndex2..];
Centroids1, (.)..(.) := [ 0.4521897119 0.07990455649 0.5138842390 0.07099338964 -0.7211259627 "Madrid" ]
Centroids2, (.)..(.) := [ 0.4438963640 0.3258840885 0.2028497472 -0.1839202106 0.7885308585 "Stockholm" ]
```

(3)

Далі все аналогічно до коду реалізації K-means.

Input/Output Data

Preamble

```

> restart ##### DELETE
with(linalg):
with(plots):
with(plottools):
```

Import Excel data

Accepts criterion name, one of: "Milk", "Eggs", "Rent", "Tickets", "Doctor", "Movie", "Gym", "Living index"
>Returns Matrix [$N \times (City, Cost)$]

```

> ImportTabular :=proc(Criterion :: string)
    return Matrix(ExcelTools:-Import("D:\\Study\\Coursework\\Data.xlsx", Criterion)) :
end proc:
```

Print histogram

Accepts : Matrix [pointsNumber×2], Caption name
> Prints histogram

```

> PortrayHistogram :=proc(Data :: Matrix, portrayCaption :: string)
    local pointsNumber, rectWidth, rise, maxHeight, i, DispText, DispRect :
    pointsNumber := rowdim(Data) :
    rectWidth := 6 :
    maxHeight := 0 :
    for i from 1 to pointsNumber do

        DispRect[i] := display(rectangle([ (i - 1) · rectWidth, 0 ], [ i · rectWidth, Data[i, 2] ], thickness = 0, color = "SteelBlue") ) :
        if (maxHeight < Data[i, 2]) then maxHeight := Data[i, 2] : end if:
    end do:
    rise :=  $\frac{\maxHeight}{33}$  :
    for i from 1 to pointsNumber do
        DispText[i] := textplot( $\left(\frac{(2i-1)}{2} \cdot rectWidth, Data[i, 2] + rise, Data[i, 1]\right)$ , font = [Times, 14]) :
    end do:
    display({seq(DispText[i], i = 1 .. pointsNumber), seq(DispRect[i], i = 1 .. pointsNumber)}, 'view' = [0 .. rectWidth · pointsNumber, 0 .. maxHeight + rise], axis[1] =
        = [thickness = 2], axis[2] = [thickness = 2, gridlines = [10, linestyle = dot]], tickmarks = [[], default], labels = ["", "Cost"], labelfirections = [horizontal,
        vertical], labelfont = [Times, 16], size = [100 · pointsNumber, 500], title = portrayCaption, titlefont = [Times, italic, 16]) :
end proc:
```

Accepts : Matrix [pointsNumber×2], Matrix [centroidsNumber×2], Caption name
> Prints histogram with highlighted centroids

```

> PortrayHistogramByCentroids :=proc(Data :: Matrix, Centroids :: Matrix, portrayCaption :: string)
    local pointsNumber, rectWidth, maxHeight, i, j, DispText, DispRect, isCentroid, rise :
    pointsNumber := rowdim(Data) :
    rectWidth := 6 :
    maxHeight := 0 :
    for i from 1 to pointsNumber do
        isCentroid := false :
        for j from 1 to rowdim(Centroids) while (isCentroid=false) do
            if (Data[i, 1] = Centroids[j, 1]) then isCentroid := true end if:
        end do:
        if (isCentroid = true) then DispRect[i] := display(rectangle([ (i - 1) · rectWidth, 0 ], [ i · rectWidth, Data[i, 2] ], thickness = 0, color = "LawnGreen") ) :
        else DispRect[i] := display(rectangle([ (i - 1) · rectWidth, 0 ], [ i · rectWidth, Data[i, 2] ], thickness = 0, color = "SteelBlue") ) : end if:
```

```

    else DispRect[i] := display(rectangle( [ (i - 1) · rectWidth, 0 ], [ i · rectWidth, Data[i, 2] ], thickness = 0, color = "SteelBlue" )) : end if.
    if (maxHeight < Data[i, 2]) then maxHeight := Data[i, 2] : end if.
end do:
rise :=  $\frac{\text{maxHeight}}{33}$  :
for i from 1 to pointsNumber do
    DispText[i] := textplot( [  $\frac{(2i - 1)}{2} \cdot \text{rectWidth}$ , Data[i, 2] + rise, Data[i, 1] ], font = [ Times, 14 ] ) :
end do:
display( { seq(DispText[i], i = 1 .. pointsNumber), seq(DispRect[i], i = 1 .. pointsNumber) }, 'view' = [ 0 .. rectWidth · pointsNumber, 0 .. maxHeight + rise ], axis[1]
    = [ thickness = 2 ], axis[2] = [ thickness = 2, gridlines = [ 10, linestyle = dot ], tickmarks = [ [ ], default ], labels = [ "", "Cost" ], labeldirections = [ horizontal,
        vertical ], labelfont = [ Times, 16 ], size = [ 100 · pointsNumber, 500 ], title = portrayCaption, titlefont = [ Times, italic, 16 ] ] :
end proc;

Accepts : List [centroidsNumber × pointsNumber[i] × 2], Palette name, Caption name
Prints histogram with highlighted clusters
> PortrayHistogramByClusters := proc(Data :: list, palette :: string, portrayCaption :: string)
local clustersNumber, rectWidth, maxHeight, i, k, j, DispText, DispRect, pointsNumber, rise :
clustersNumber := nops(Data) :
rectWidth := 6 :
maxHeight := 0 :
i := 0 : # Current point index
for k from 1 to clustersNumber do
    for j from 1 to rowdim(Data[k]) do
        i := i + 1 : # Next point
        DispRect[i] := display(rectangle( [ (i - 1) · rectWidth, 0 ], [ i · rectWidth, Data[k][j, 2] ], thickness = 0, color = ColorTools :-
Color(SmartColor(clustersNumber, k, palette)) ) :
        if (maxHeight < Data[k][j, 2]) then maxHeight := Data[k][j, 2] : end if.

    end do:
end do:
rise :=  $\frac{\text{maxHeight}}{33}$  :
i := 0 :
for k from 1 to clustersNumber do
    for j from 1 to rowdim(Data[k]) do
        i := i + 1 : # Next point
        DispText[i] := textplot( [  $\frac{(2i - 1)}{2} \cdot \text{rectWidth}$ , Data[k][j, 2] + rise, Data[k][j, 1] ], font = [ Times, 14 ] ) :
    end do:
end do:
pointsNumber := i :
display( { seq(DispText[i], i = 1 .. pointsNumber), seq(DispRect[i], i = 1 .. pointsNumber) }, 'view' = [ 0 .. rectWidth · pointsNumber, 0 .. maxHeight + rise ], axis[1]
    = [ thickness = 2 ], axis[2] = [ thickness = 2, gridlines = [ 10, linestyle = dot ], tickmarks = [ [ ], default ], labels = [ "", "Cost" ], labeldirections = [ horizontal,
        vertical ], labelfont = [ Times, 16 ], size = [ 100 · pointsNumber, 500 ], title = portrayCaption, titlefont = [ Times, italic, 16 ] ] :
end proc;

Accepts : Matrix pointsNumber × (City name, Cost, Cluster index), Clusters number, Color palette, Caption of portray
Prints histogram due to site results
> PortrayResults := proc(Data :: Matrix, clustersNumber :: integer, palette :: string, portrayCaption :: string)
local rectWidth, maxHeight, i, DispRect, rise, DispText, pointsNumber :
pointsNumber := rowdim(Data) :
rectWidth := 6 :
maxHeight := 0 :
for i from 1 to pointsNumber do
    DispRect[i] := display(rectangle( [ (i - 1) · rectWidth, 0 ], [ i · rectWidth, Data[i, 2] ], thickness = 0, color = ColorTools :-
Color(SmartColor(clustersNumber,
    trunc(Data[i, 3]), palette)) ) :
    if (maxHeight < Data[i, 2]) then maxHeight := Data[i, 2] : end if.

end do:
rise :=  $\frac{\text{maxHeight}}{33}$  :
for i from 1 to pointsNumber do
    DispText[i] := textplot( [  $\frac{(2i - 1)}{2} \cdot \text{rectWidth}$ , Data[i, 2] + rise, Data[i, 1] ], font = [ Times, 14 ] ) :
end do:
display( { seq(DispText[i], i = 1 .. pointsNumber), seq(DispRect[i], i = 1 .. pointsNumber) }, 'view' = [ 0 .. rectWidth · pointsNumber, 0 .. maxHeight + rise ], axis[1]
    = [ thickness = 2 ], axis[2] = [ thickness = 2, gridlines = [ 10, linestyle = dot ], tickmarks = [ [ ], default ], labels = [ "", "Cost" ], labeldirections = [ horizontal,
        vertical ], labelfont = [ Times, 16 ], size = [ 100 · pointsNumber, 500 ], title = portrayCaption, titlefont = [ Times, italic, 16 ] ] :
end proc;

```

Print points

```

Accepts : Matrix [pointsNumber × 2], List of centroids sets, Caption name
Prints points
> PortrayPointsByCentroids := proc(Data :: Matrix, CentroidsSets :: list, portrayCaption :: string)
local symbolSize, pointsNumber, centroidsSetsNumber, dispPoints, k, centroidsNumber, DispCentroids :
symbolSize := 18 :
pointsNumber := rowdim(Data) :
centroidsSetsNumber := nops(CentroidsSets) :
dispPoints := pointplot( [ seq(Data[i, 2], i = 1 .. pointsNumber) ], [ seq(0, i = 1 .. pointsNumber) ], symbol = solidcircle, symbolsize = symbolSize, color
    = "SteelBlue" ) :
for k from 1 to centroidsSetsNumber do
    centroidsNumber := rowdim(CentroidsSets[k]) :

```

```

DispCentroids[ k ] := pointplot( [ seq( CentroidsSets[ k ][ i, 2 ], i = 1 ..centroidsNumber ) ], [ seq( 0, i = 1 ..centroidsNumber ) ], symbol = solidcircle, symbolsize = symbolSize, color = ColorTools:-Color( SmartColor( centroidsSetsNumber, k, "CMR" ) ), legend = cat( "Centroids #", k ) ) :
end do:
display( { dispPoints, seq( DispCentroids[ i ], i = 1 ..centroidsSetsNumber ) }, axis[ 1 ] = [ thickness = 2 ], axis[ 2 ] = [ color = white ], tickmarks = [ default, [ ] ], labels = [ "Cost", "" ], labeldirections = [ horizontal, vertical ], labelfont = [ Times, 16 ], size = [ 1000, 200 ], title = portrayCaption, titlefont = [ Times, italic, 16 ] ) :
end proc:

```

Subsidiary functions

Sorting

Accepts : Matrix, Key-column
 Returns new matrix sorted by 2nd column (by value)

```

> SortMatrixByColumn :=proc(Data :: Matrix, columnNumber)
    return Data[ sort( Data[ .., columnNumber ], 'output'='permutation' ) ]:
end proc:
```

Accepts : list of clusters sets
 Returns new list of clusters sets sorted by values

```

> SortClusters :=proc(OriginalData :: list)
    local clustersNumber, k, Data := copy(OriginalData):
    clustersNumber := nops( Data ):
    for k from 1 to clustersNumber do
        Data[ k ] := SortMatrixByColumn( Data[ k ], 2 ): # Сортування точок всередині кластерів
    end do:
    # Сортування кластерів за значеннями їх перших елементів
    Data := Data[ sort( [ seq( Data[ k ][ 1, 2 ], k = 1 ..clustersNumber ) ], 'output'='permutation' ) ]:
    return Data:
end proc:
```

Color palettes

Accepts : Number of separators, Number of current separator, Palette name
 Palettes : RYG -- red-yellow-green
 CMR -- cyan-magenta-red
 Returns list of RGB code

```

> SmartColor :=proc(separatorsNumber :: integer, currentSeparator :: integer, palette :: string)
    local f, g, h, currentX:
    if ( palette = "RYG" ) then
        f := x → -0.5363 x3 + 4.8141 x2 - 11.6974 x + 254.3007:
        g := x → 0.2811 x3 - 8.0577 x2 + 72.8745 x + 42.8951:
        h := x → 46:
    elif ( palette = "GYR" ) then
        f := x → 0.5363 x3 - 11.3339 x2 + 76.8955 x + 81.9930:
        g := x → -0.2811 x3 + 0.3747 x2 + 3.9553 x + 246.9510:
        h := x → 46:
    elif ( palette = "CMR" ) then
        f := x → -0.8815 x3 + 11.2040 x2 - 1.3994 x + 1.3636:
        g := x → -0.8912 x3 + 18.1550 x2 - 115.9631 x + 241.7483:
        h := x → -0.0278 x3 - 3.6696 x2 + 18.2009 x + 233.2517:
    else
        f := x → 0.1418 x3 + 5.2669 x2 - 66.9192 x + 254.6224:
        g := x → 0.4038 x3 - 2.6987 x2 - 35.9079 x + 255.4825:
        h := x → -1.3543 x3 + 16.0396 x2 - 31.0979 x + 66.2657:
    end if:
    if ( separatorsNumber > 1 ) then currentX :=  $\frac{10}{separatorsNumber - 1} \cdot (currentSeparator - 1)$  : else currentX := 0 : end if:
    return [ trunc( f( currentX ) ), trunc( g( currentX ) ), trunc( h( currentX ) ) ]:
end proc:
```

Other

Accepts : Matrix pointsNumber×(City, Cost index, Cluster number), rise (increaser)

Increases values of clusters numbers by rise value

```

> IncreaseClustersNumbers :=proc(Data :: Matrix, rise :: integer)
    local i:
    for i from 1 to rowdim( Data ) do
        Data[ i, 3 ] := trunc( Data[ i, 3 ] + rise ):
    end do:
end proc:
```

Accepts : Matrix pointsNumber×(City, Cost index, Cluster number)

Returns list of matrices

```

> GetReadableForm :=proc(Data :: Matrix, centroidsNumber :: integer)
    local pointsNumber, ClustersSizes, i, Clusters, ClustersFilling :
    pointsNumber := rowdim( Data ):
    # Визначення кількості точок у кожному класері
    ClustersSizes := Array( 1 .. centroidsNumber ):
    for i from 1 to pointsNumber do
        ClustersSizes[ Data[ i, 3 ] ] := ClustersSizes[ Data[ i, 3 ] ] + 1:
    end do:
    # Створюємо набір масивів Clusters, в якому будуть зберігатися поділені по кластерам точки

```

```

# Створюємо набір масивів Clusters, в якому будуть зберігатися поділені по кластерам точки
Clusters := [seq(Matrix(ClustersSizes[i], 2), i=1..centroidsNumber)]:
# Створюємо масив ClustersFilling, в якому зберігається кількість точок, що наразі знаходиться у відповідному кластері
ClustersFilling := Array(1 .. centroidsNumber):
# Заповнюємо Clusters
for i from 1 to pointsNumber do
    Clusters[Data[i, 3]][(ClustersFilling[Data[i, 3]] + 1)] := Array([Data[i, 1], Data[i, 2]]):
    ClustersFilling[Data[i, 3]] := ClustersFilling[Data[i, 3]] + 1:
end do:
return Clusters:
end proc:

Accepts : list of clusters sets
Returns list of cluster info : number of points, boundary values, difference between borders
> GetInfo := proc(Clusters :: list)
local Data, i:
Data := SortClusters(Clusters):
for i from 1 to nops(Data) do
    Data[i] := [cat("Cluster #", i), cat("Points : ", rowdim(Data[i])), cat("Borders : from ", Data[i][1, 1], "(", trunc(Data[i][1, 2]), ") hr.) to ",
    Data[i][rowdim(Data[i]), 1], " (", trunc(Data[i][rowdim(Data[i]), 2]), ") hr.)", cat("Difference : ", trunc(Data[i][rowdim(Data[i]), 2] - Data[i][1,
    2]), " hr.")]:
end do:
return Data:
end proc:

```



Дата перевірки:
13.06.2022 18:26:37 EEST

Дата звіту:
13.06.2022 18:27:01 EEST

Назва документа: Пишко Андрій Олександрович

Кількість сторінок: 70 Кількість слів: 6876 Кількість символів: 48398 Розмір файлу: 2.92 MB ID файлу: 1011437236

5.83% Схожість

Найбільша схожість: 1.77% з джерелом з Бібліотеки (ID файлу: 8081435)

0% Цитат

0% Вилучень

Експертна оцінка роботи науковим керівником:

Основна частина роботи є оригінальною. Використані матеріали супроводжуються посиланнями на джерела.

Науковий керівник:

A handwritten signature in black ink, appearing to read 'Zatula D.B.'

Затула Д.В.

Оператор:

A handwritten signature in black ink, appearing to read 'Onoцький B.B.'

Оноцький В.В.

Відгук на кваліфікаційну роботу бакалавра на тему:
«Аналіз вартості проживання у великих містах світу із використанням методів
кластеризації»

студента 4-го курсу факультету комп’ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Пишка Андрія Олександровича

Кластерний аналіз займає одне з центральних місць серед методів аналізу даних і є сукупністю підходів, методів і алгоритмів, призначених для знаходження деякого розбиття досліджуваної сукупності об’єктів на підмножини схожих між собою об’єктів.

У дипломній роботі Пишка А. О. було описано і використано методи кластерного аналізу для розбиття множини міст на групи по вартості проживання в них, а саме: досліджено ефективність методів K-means, BIRCH та SpectralClustering на вибірці із 42 європейських міст. В якості критеріїв для кластеризації було обрано вартості молока, пачки яєць, оренди житлового приміщення, місячного квитка на транспорт, візиту до лікаря, квитків на сеанс кіно, абонементу в спортивний зал та загальний індекс вартості проживання. Методи спрацювали схожим чином, проте мали певні відмінності, які були також описані в роботі.

Дослідження були реалізовані за допомогою програмного забезпечення Maple, а також мови програмування Python. Значну роль у підготовці даних відіграло програмне забезпечення Excel, для аналізу даних було застосовано пакети LinearAlgebra та GraphTheory, а також бібліотеку numpy. В цілому програмну частину роботи проведено в Maple.

Вважаю, що кваліфікаційна робота студента Пишка А. О. цілком відповідає вимогам, які висуваються до бакалаврських робіт, і заслуговує оцінки «відмінно», а її автор заслуговує на присвоєння кваліфікації бакалавра.

Асистент кафедри обчислювальної математики
факультету комп’ютерних наук та кібернетики
Київського національного університету
імені Тараса Шевченка,
кандидат фізико-математичних наук

Д. В. Затула

Рецензія на кваліфікаційну роботу бакалавра на тему:
«Аналіз вартості проживання у великих містах світу із використанням методів
кластеризації»

студента 4-го курсу факультету комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Пишка Андрія Олександровича

Метою рецензованої роботи є дослідження застосування методів кластеризації для розбиття множини міст на групи за даними вартості проживання в них.

Автор розглянув такі методи кластеризації: K-means, BIRCH та SpectralClustering. В якості вибірки було обрано значення вартості різних показників у 42 містах світу. Було продемонстровано роботу методів та описано різницю між результатами кластеризації. Як виявилося, метод K-means дуже чутливий до «викидів», які сптворюють результати. BIRCH, який добре працює з великою кількістю даних, показав кращі результати. Найбільш рівномірно розподілену кластеризацію надав SpectralClustering. Крім того, було розглянуто різницю між варіаціями останнього методу, їхній вплив на результат.

Дослідження були реалізовані у середовищах Maple та Jupyter Notebook.

За змістом роботи можна зробити зауваження. Немає математичного обґрунтування, чому в роботі спектральної кластеризації власні вектори добре визначають кластери та чому нормалізація отриманих точок значно покращує результат. Також було б непогано пояснити, чому при використанні принципу k -найближчих сусідів результатуюча кластеризація є більш рівномірною. Але не вважаю це зауваження таким, що зменшує загальну позитивну оцінку роботи.

Вважаю, що кваліфікаційна робота студента Пишка А. О. цілком відповідає вимогам, які висуваються до бакалаврських робіт, і заслуговує оцінки «відмінно», а її автор заслуговує на присвоєння кваліфікації бакалавра.

Асистент кафедри системного аналізу та прийняття рішень
факультету комп'ютерних наук та кібернетики
Київського національного університету
імені Тараса Шевченка,
кандидат фізико-математичних наук



Ю. М. Шевчук