УКРАЇНСЬКИЙ КАТОЛИЦЬКИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНИХ НАУК

Комп’ютерні науки

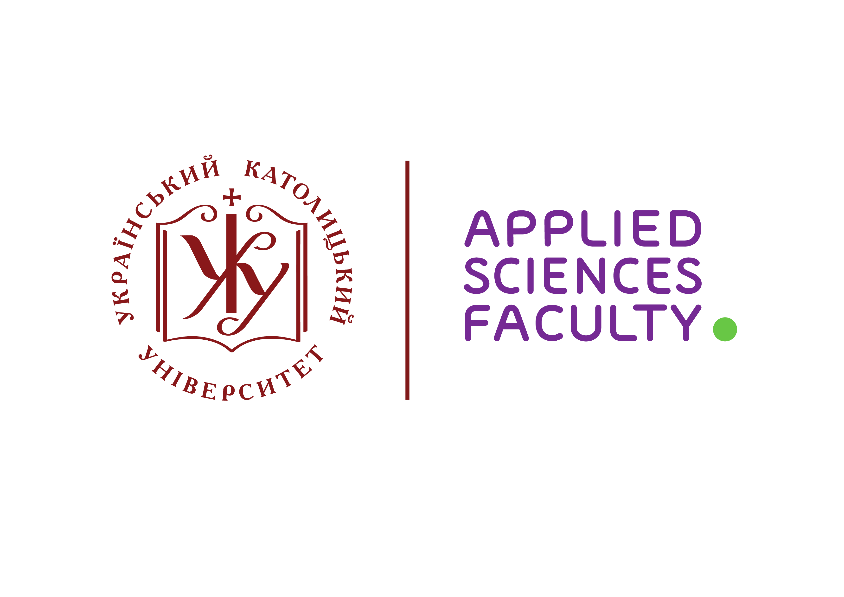
**Пошук найкоротшого шляху у зваженому неорієнтованому зв’язному графі, використовуючи алгоритм Дейкстри**

*Автори:*

*Гапунович Юлія*

*Шувар Софія*

14 червня 2019



1. **Вступ.**

Завдання: Зважений неорієнтований зв’язний граф задано матрицею ваг. Знайти найкоротший шлях від заданої вершини до іншої. Використати алгоритм Дейкстри.

Над проектом працювали: Гапунович Юлія та Шувар Софія

Даний проект складається з трьох файлів:

1. dijkstra\_search.py – файл написаний на мові програмування Python, у якому міститься основний код програми.
2. matrix.txt – текстовий файл, що містить приклад зваженого неорієнтованого зв’язного графа, заданого матрицею ваг.
3. example\_dijkstra\_search.py - файл написаний на мові програмування Python, який ми пропонуємо вам запустити для тестування даної програми(можна протестувати програму на вище запропонованій матриці, а можна на власній, попередньо вказавши назву текстового файлу)

В даній роботі ми використовували знання здобуті під час вивчення курсу Дискретна математика з теми Теорія графів – алгоритм Дейкстри - алгоритм визначення довжини найкоротшого шляху від фіксованої вершини до будь-якої іншої.

1. **Псевдокод.**

Розглянемо модуль dijkstra\_search.py, що містить основний код програми. Цей модуль містить клас Graph, представляє граф зчитаний з текстового файлу, що містить матрицю ваг.

read\_matrix – функція, що зчитує матрицю ваг з файлу, та наповнює словник ваг вагами із матриці у форматі: {'a': {'b': 1, 'c': 5, 'e': 3}, 'b': {'a': 1, 'c': 3, 'd': 8}, 'c': {'a': 5, 'b': 3, 'd': 6, 'e': 2, 'g': 4, 'z': 9}, 'd': {'b': 8, 'c': 6, 'z': 3}, 'e': {'a': 3, 'c': 2, 'g': 6}, 'g': {'c': 4, 'e': 6, 'z': 3}, 'z': {'c': 9, 'd': 3, 'g': 3}}.

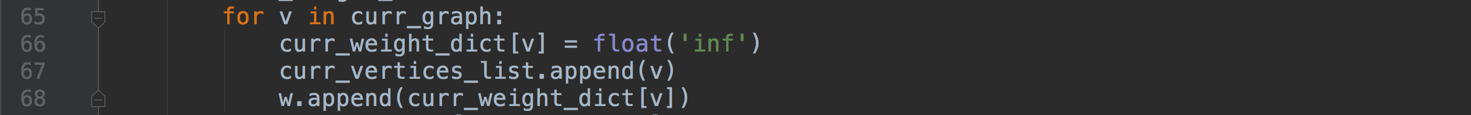
search\_by\_dijkstra – функція, що ілюструє роботу алгоритму Дейкстри:

1)Уводимо наступні змінні, де self.weights – словник ваг, зазначений вище.

A close up of a logo

Description automatically generated

2) Проходимось по усіх вершинах з curr\_graph та запонюємо curr\_weight\_dict – постійні мітки вершин: ключі – вершини матриці, згачення – присвоюємо значення нескінченності. У спфсок curr\_vertices\_list додаємо усі вершини графа, а у w додаємо значення усіх ваг (inf)

3)Присвоюємопочатковій вершині вагу 0. А також створюємо пустий список path\_vertices – список пройдених вершин.

A close up of a logo

Description automatically generated4)Проходимось по списку вершин curr\_vertices\_list, та додаємо до path\_vertices вершину з мінімальною вагою та видаляємо її зі списку curr\_vertices\_list. Для цього косистуємось статичним методом extract.

A picture containing device

Description automatically generatedУ цьому ж циклі проходимось по усіх вершинах суміжних із вершиною з мінімальною вагою і, якщо виконується нерівність:

A close up of a logo

Description automatically generatedПрисвоюємо вершинам нову вагу:

A close up of a logo

Description automatically generated5)Метод extract:

A screenshot of a cell phone

Description automatically generated

6)Функція повертає два значення: словник відсортованих ваг від заданої початкової вершини до усіх іншик та словник усіх пройдених вершин підчас проходження графа.

Також була реалізована функція min\_distanse, що шукає найкоротшу відстань від початкової вершини до заданої, використовуючи вище згадану функцію search\_by\_dijkstra

Та реалізовано метод \_\_str\_\_ для ілюстрації роботи алгоритму у вигляді таблиці, де l – постійні мітки вершини, p – вершини, з яких у дану верщину заходить найкоротший шлях. A picture containing black, photo

Description automatically generated

1. **Висновки.**

В результаті виконання комп’ютерного проекту ми :

* Розробили ADT Graph для зчитування файлу та перетворення його у необхідну для роботи модель графа.
* Розробили необхідний набір методів цього ADT для реалізації алгоритму Дейкстри
* Розробили модуль, що запускатиме на виконання даний алгоритм для введених користувачем даних
* В результаті отримали можливість визначати найкоротший шлях між будь-якими двома заданими вершинами зваженого неорієнтованого графа, а також обраховувати вагу цього марштуту та виводити ці дані в доступній для користувача формі ( таблиця та текстовий опис результату)