Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Лабораторна робота №2

«Реалізація алгоритму Джонсона»

Виконав студент 2-го курсу

Групи ІПС21

Вербицький Артем Віталійович

Київ – 2024

**Завдання**:

Алгоритм Джонсона для розріджених графів (включає алгоритми Беллмана-Форда і Дейкстри). В алгоритмі Дейкстри використайте піраміду Фібоначчі. Ваги ребер задаються дійсними числами.

**Теорія**

Алгоритм Джонсона дозволяє знайти найкоротші шляхи між усіма парами вершин зваженого орієнтованого графа. На відміну від алгоритму Дейкстри, працює для графів із ребрами від'ємної ваги, але без циклів з від'ємною вагою. В алгоритмі Джонсона використовують алгоритм Беллмана-Форда та алгоритм Дейкстри втілені у вигляді підпрограм. Ребра зберігають у вигляді переліків суміжних вершин.

Алгоритм Беллмана-Форда – алгоритм пошуку найкоротшого шляху в зваженому графі. Знаходить найкоротші шляхи від однієї вершини графа до всіх інших. На відміну від алгоритму Дейкстри, алгоритм Беллмана-Форда допускає ребра з негативною вагою.

Під час роботи алгоритму Дейкстри всі невідвідані вершини графу зберігаються у Піраміді Фібоначчі, яка з кожним кроком алгоритму зменшуються тобто в кожному кроці алгоритму Дейкстри з неї видаляються відвідані вершини, і так як видалення з піраміди Фібоначчі має середню складність 𝑂(log V) то час роботи алгоритму Дейкстри зменшується з 𝑂(V2) до 𝑂(Vlog V + E) що і зменшує складність самого алгоритму Джонсона.

**Алгоритм**

Алгоритм складається з таких кроків:

1. Спочатку до графу додається нова вершина S, пов'язанa ребрами з нульовою вагою з кожним з інших вузлів. Перевіряємо дані на повтори.
2. Використовуємо алгоритм Беллмана-Форда, починаючи з нової вершини S, для знаходження для кожної вершини a мінімальної ваги h(a) шляху з S в a
   * Створюємо масив dist[V + 1] (V + 1 – кількість вершин включно з S) для збереження відстаней з S до всіх інших вершин. Присвоюємо всім елементам масиву значення нескінченність, окрім dist[0], яку робимо нульовою (відстань з S до S).
   * Другим кроком обчислюються найкоротші відстані. Наступні кроки потрібно виконувати n-1 раз. Для кожного ребра u-v: перевіряємо умову  dist[v] > dist[u] + вага ребра uv, то присвоюємо dist[v] = dist[u] + вага ребра uv.
   * Проходимо цей цикл ще раз. Якщо наведена зверху умова виконується, значить граф містить цикл негативної ваги і алгоритм Джонсона для нього не працює.
3. Повторно зважуємо ребра графа, присвоюючи їм значення w(u, v) + h(u) - h(v). Тепер вони гарантовано не мають від’ємної ваги і ми можемо використовувати алгоритм Дейкстри.
4. Видаляємо з графа вершину S та виконуємо алгоритм Дейкстри:
   * Кожній вершині графа ставимо у відповідність мітку – мінімальну відому відстань від цієї вершини до певної вершини А. Алгоритм працює покроково – на  кожному кроці він «відвідує» одну вершину і намагається зменшувати мітки. Робота алгоритму завершується, коли всі вершини відвідані.
   * Мітка самої вершини А встановлюється рівною 0, мітки інших вершин – нескінченності. Це відображає те, що відстані від А до інших вершин поки невідомі. Всі вершини графа позначаються як невідвідані.
   * Якщо всі вершини відвідані, алгоритм завершується. В іншому випадку, з ще не відвіданих вершин вибирається вершина U, що має мінімальне значення мітки.

**Складність**

Якщо в алгоритмі Дейкстри неспадну чергу з пріоритетами втілено у вигляді піраміди Фібоначчі, то тривалість роботи алгоритму Джонсона дорівнює O(EV+ V2logV).

**Мова реалізації алгоритму** С++

**Модулі програми:**

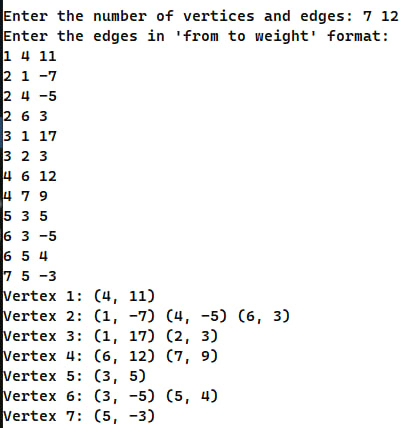
* struct Edge // Структура для зберігання ребер
* class Graph(vector<Edge>& edges, int V) // Клас для здійснення операцій над графом заданого ребрами
* void print\_graph()  // Функція для виведення графа у консоль
* vector<vector<int>> Johnson(int E, vector<vector<int>> & paths) // Алгоритм Джонсона
* vector<int> BellmanFord(int& V, vector<Edge> edges) // Алгоритм Беллмана-Форда
* vector<int> Dijkstra(int src, vector<vector<pair<int, int>>>& adj\_list, int V, vector<vector<int>>& paths) // Алгоритм Дейкстри
* signed main() // Головна функція

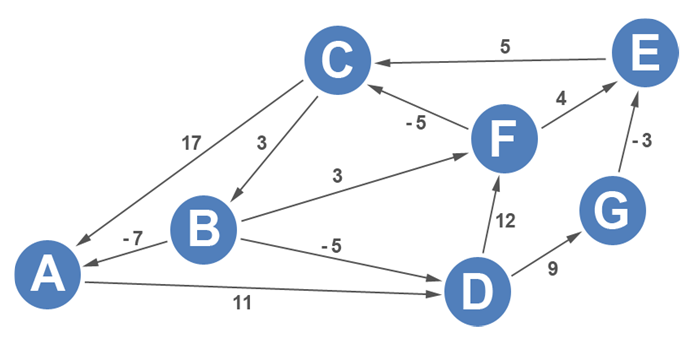
**Інтерфейс користувача**

Ввід і вивід даних здійснюється в консолі.

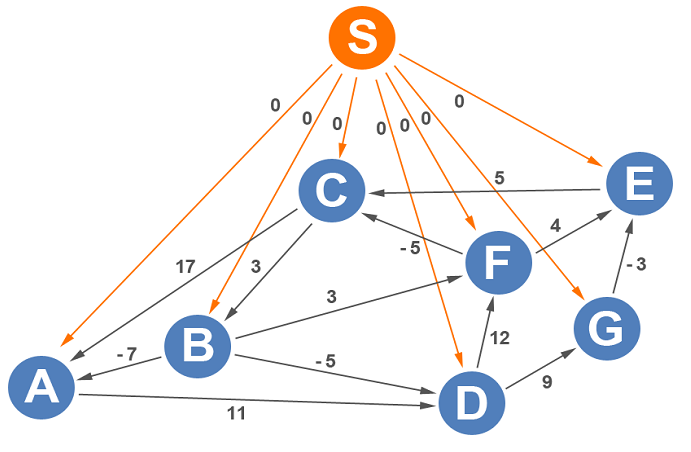
**Тестові приклади**

**Приклад 1**

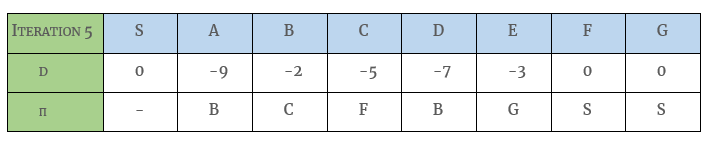


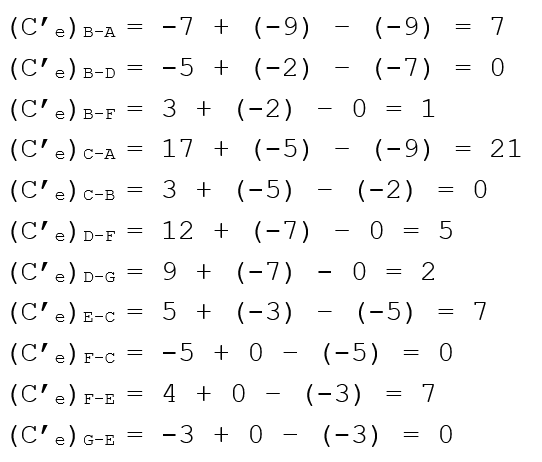


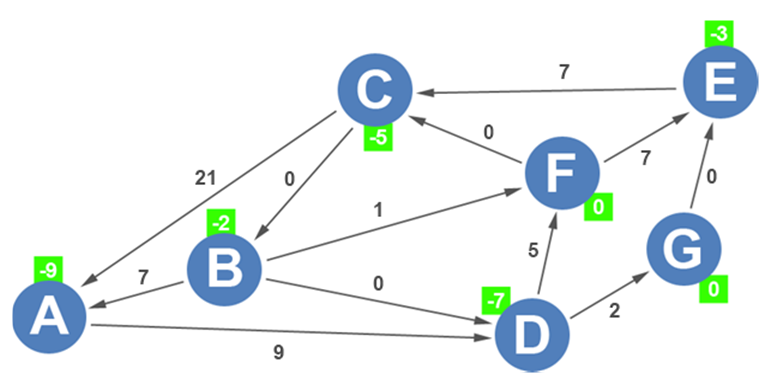
Маємо такий граф, додаємо уявну вершину S



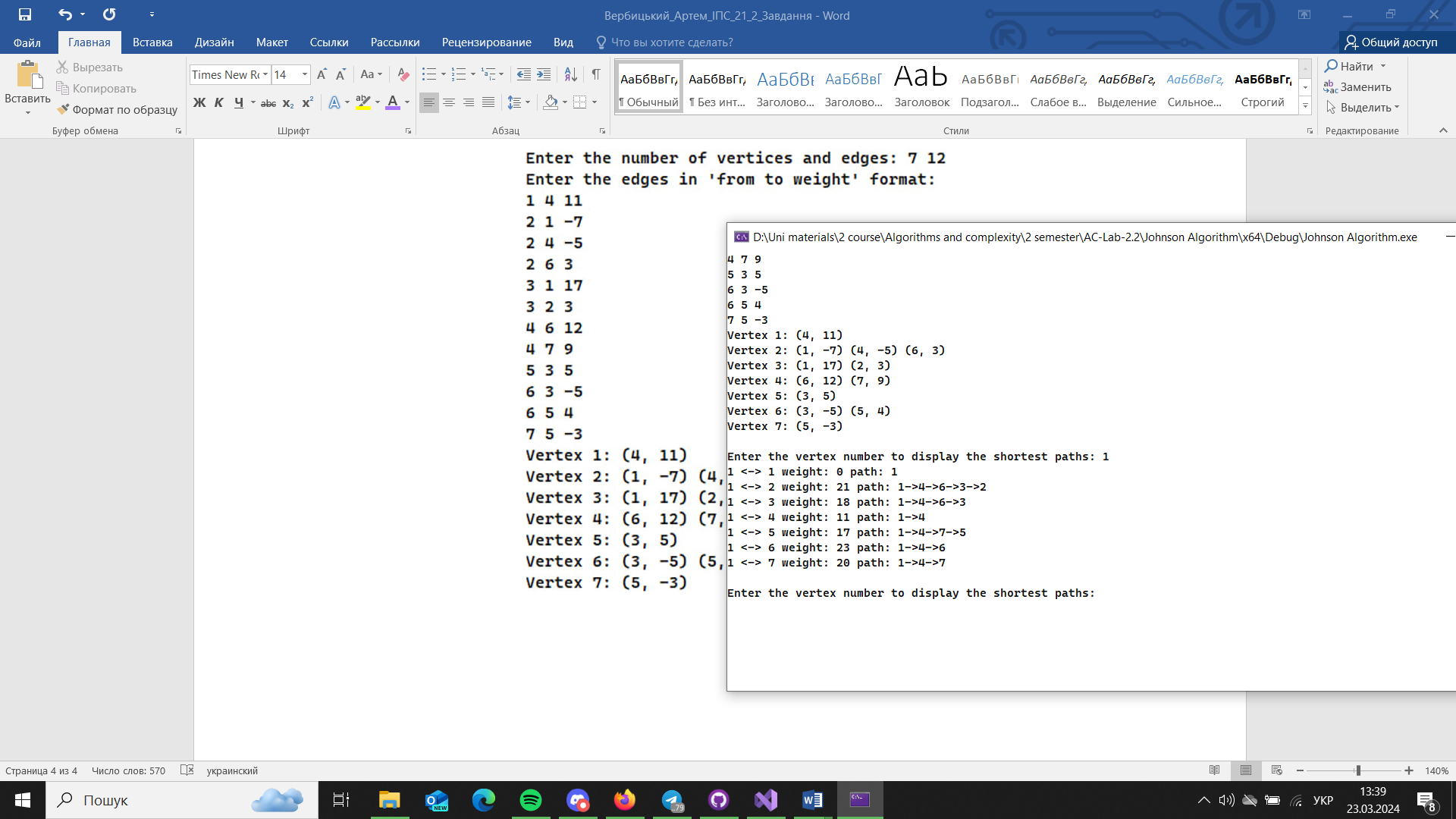
Після проходження алгоритму Беллмана-Форда маємо такі значення в масиві dist





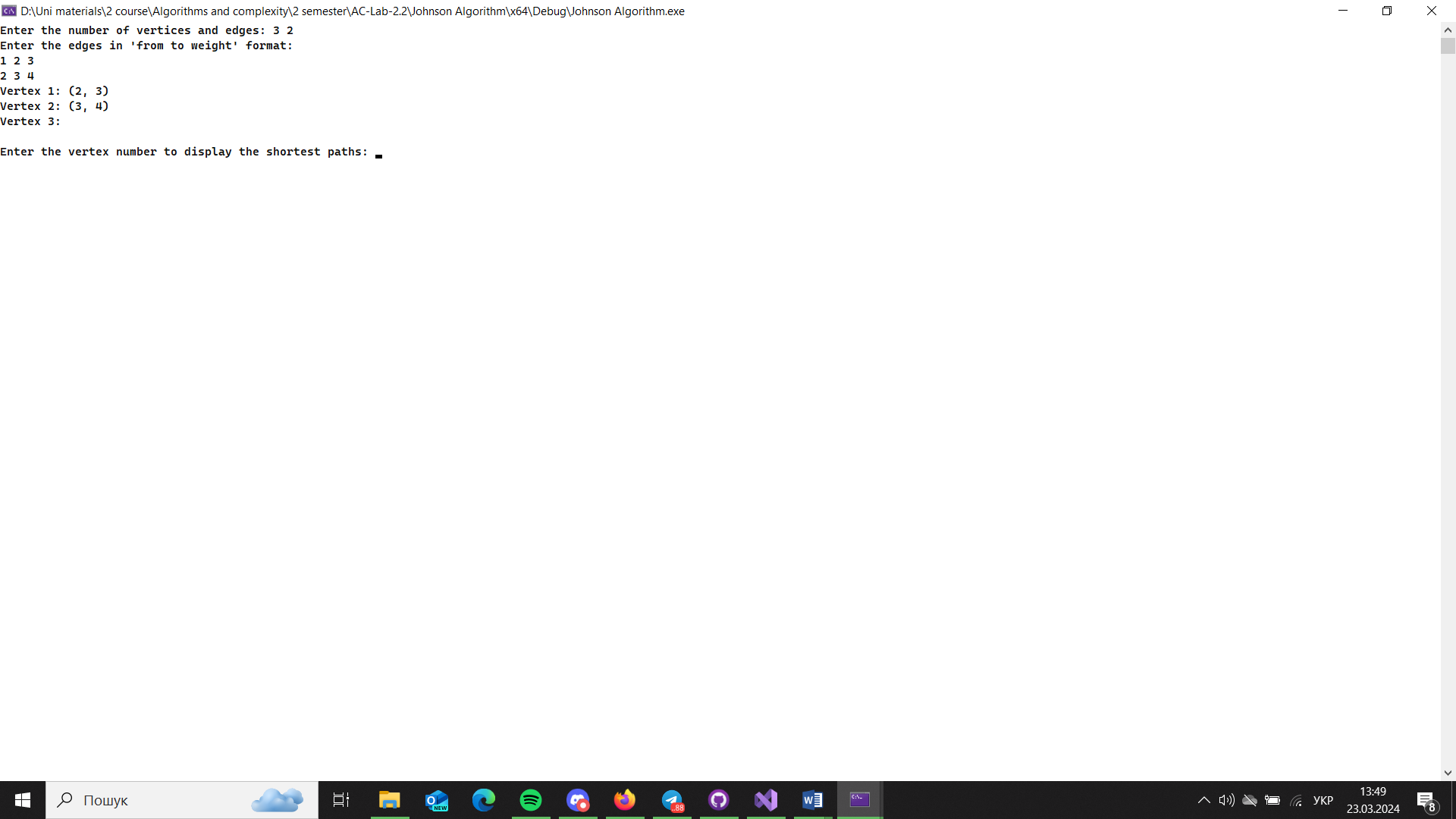


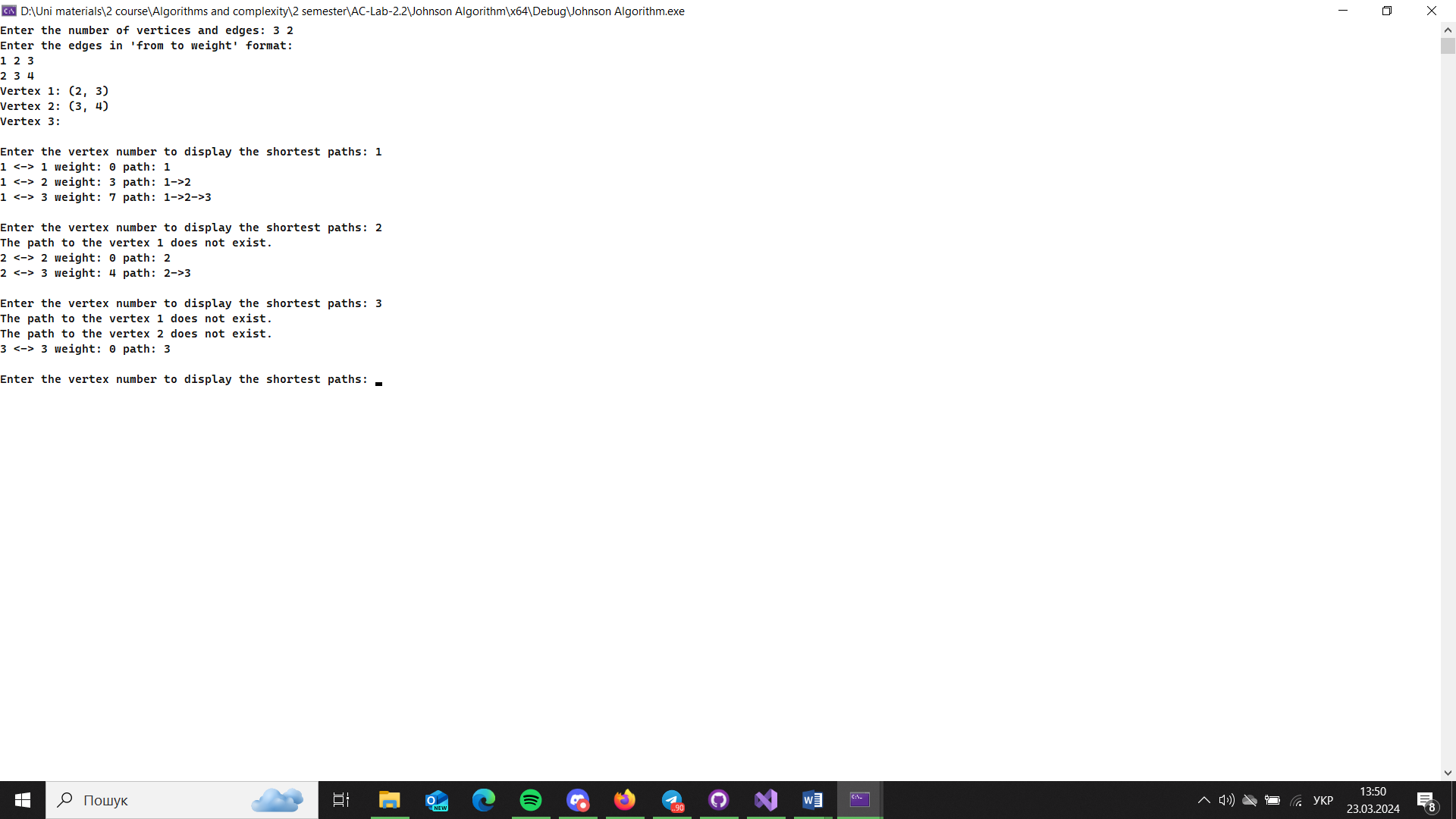
Після проведених процедур виконуємо алгоритм Дейкстри і маємо результати:



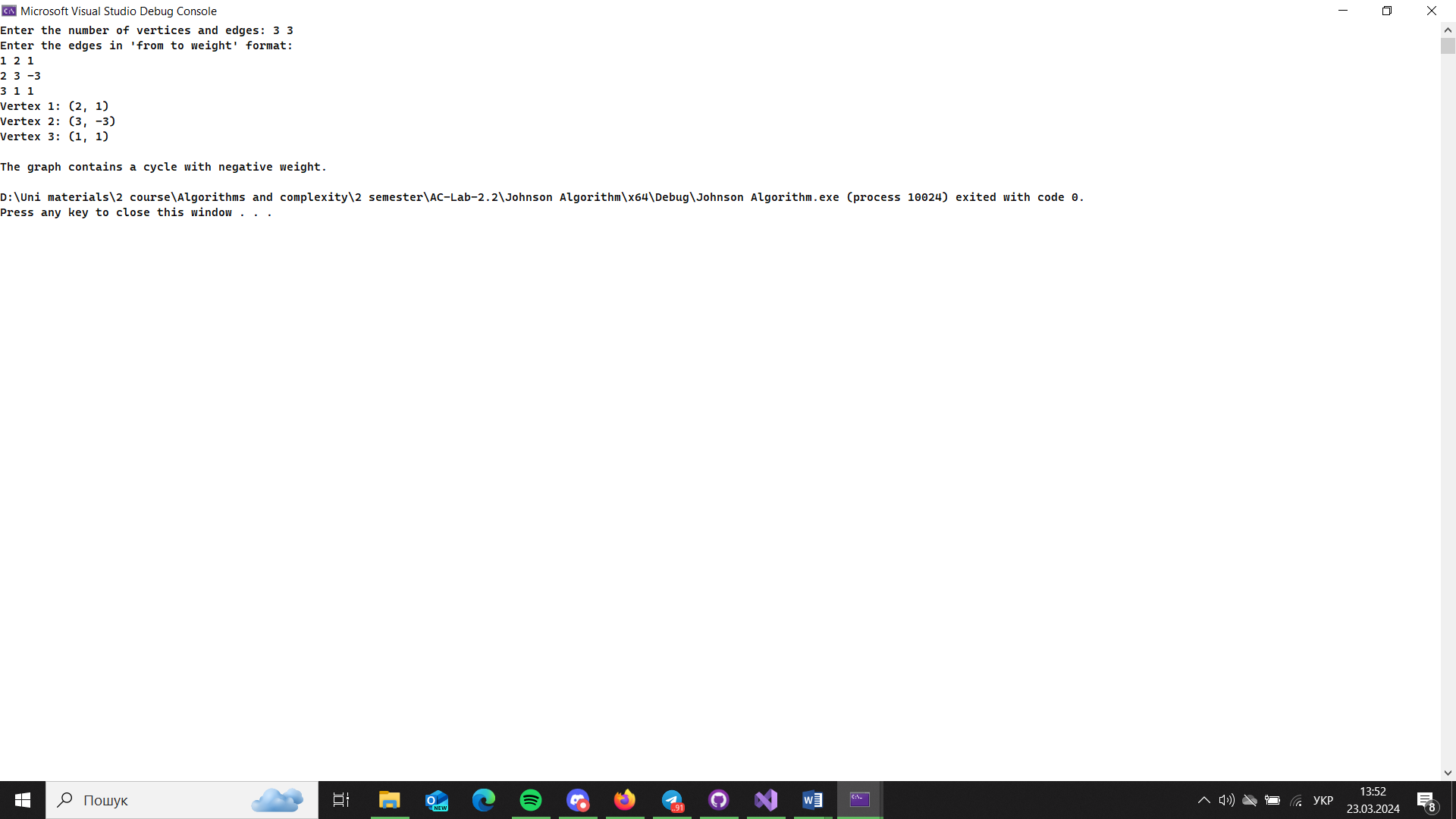
Наведено результати лише для вершини А, для інших алгоритм також все коректно знаходить.

**Приклад 2**





**Приклад 3**



**Висновки**

Так як ми для зберігання невідвіданих вершин використовували купу Фібоначчі, то ми значно зменшили складність алгоритму з тої яка б була при наївній реалізації. При розріджених графах складність алгоритму стає меншою за складність алгоритму Флойда-Уоршала який виконує цю задачу за 𝑂(𝑉2).

**Література**

* <https://gist.github.com/ashleyholman/6793360/ae69b92bad8d5ba6bf0383efcd92a2e6b8760ab0>
* <https://blog.devgenius.io/johnsons-algorithm-visually-explained-188dbd51ce5c>
* <https://blog.devgenius.io/bellman-ford-algorithm-visually-explained-e940b6edb00a>
* <https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B8>