МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет імені Михайла Драгоманова

Факультет математики, інформатики та фізики

Кафедра інформаційних технологій та програмування

**Звіт**

з лабораторної роботи №10

«Графи, динамічне програмування і жадібні алгоритми»

з дисципліни «Програмування»

Виконала:

студентка ІІІ курсу групи 31І

Мотовилець М. Т.

Перевірила:

викладач Устименко О.Б.

Оцінка\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_

Київ - 2024

**Зміст**

[Мета роботи 3](#_Toc150170352)

[1 Постановка задачі 4](#_Toc150170353)

[2 Основна частина 5](#_Toc150170354)

[2.1 Опис вхідних та вихідних даних 5](#_Toc150170355)

[2.1 Блок-схема 6](#_Toc150170356)

[Висновки 7](#_Toc150170357)

**Мета роботи**

Мета лабораторної роботи ознайомитись з основними принципами та підходами до вирішення задач за допомогою динамічного програмування та жадібних алгоритмів у контексті графів. Розглянути та порівняти різні підходи до вирішення задач на графах з використанням цих алгоритмів, зосереджуючись на їх застосуванні до таких задач, як знаходження найкоротших шляхів.

**Постановка задачі**

Варіант №14

Задача №1

Написати програму для пошуку найкоротшого шляху між двома заданими вершинами:

Зображення, що містить годинник, ряд, коло

Автоматично згенерований опис

Задача№2

Дана двовимірна матриця розміром 𝑛×𝑚, що складається з 0 і 1. Знайти площу найбільшого прямокутника, який складається тільки з 1.

**Основна частина**

**2.1 Опис вхідних та вихідних даних**

**Задача №1**

**Вхідні дані:**

* Імпорт модулю heapq.
* graph: словник, що представляє граф, де ключі - це вершини, а значення - це словники сусідніх вершин і відповідних ваг ребер.
* start: початкова вершина для пошуку найкоротшого шляху.
* end: кінцева вершина для пошуку найкоротшого шляху.

**Вихідні дані:**

* shortest\_path: список вершин, що представляє найкоротший шлях від start до end.
* dist[end]: відстань найкоротшого шляху між start і end, виведена разом з найкоротшим шляхом у форматі: "Найкоротший шлях між [start] і [end]: [shortest\_path] з відстанню [dist[end]]".

**Задача №2**

**Вхідні дані:**

* matrix: двовимірна матриця розміром 𝑛×𝑚, що складається з 0 і 1. Кожен рядок матриці представляє собою рядок у двоічному форматі, де 1 відповідає наявності одиниць, а 0 - відсутності.

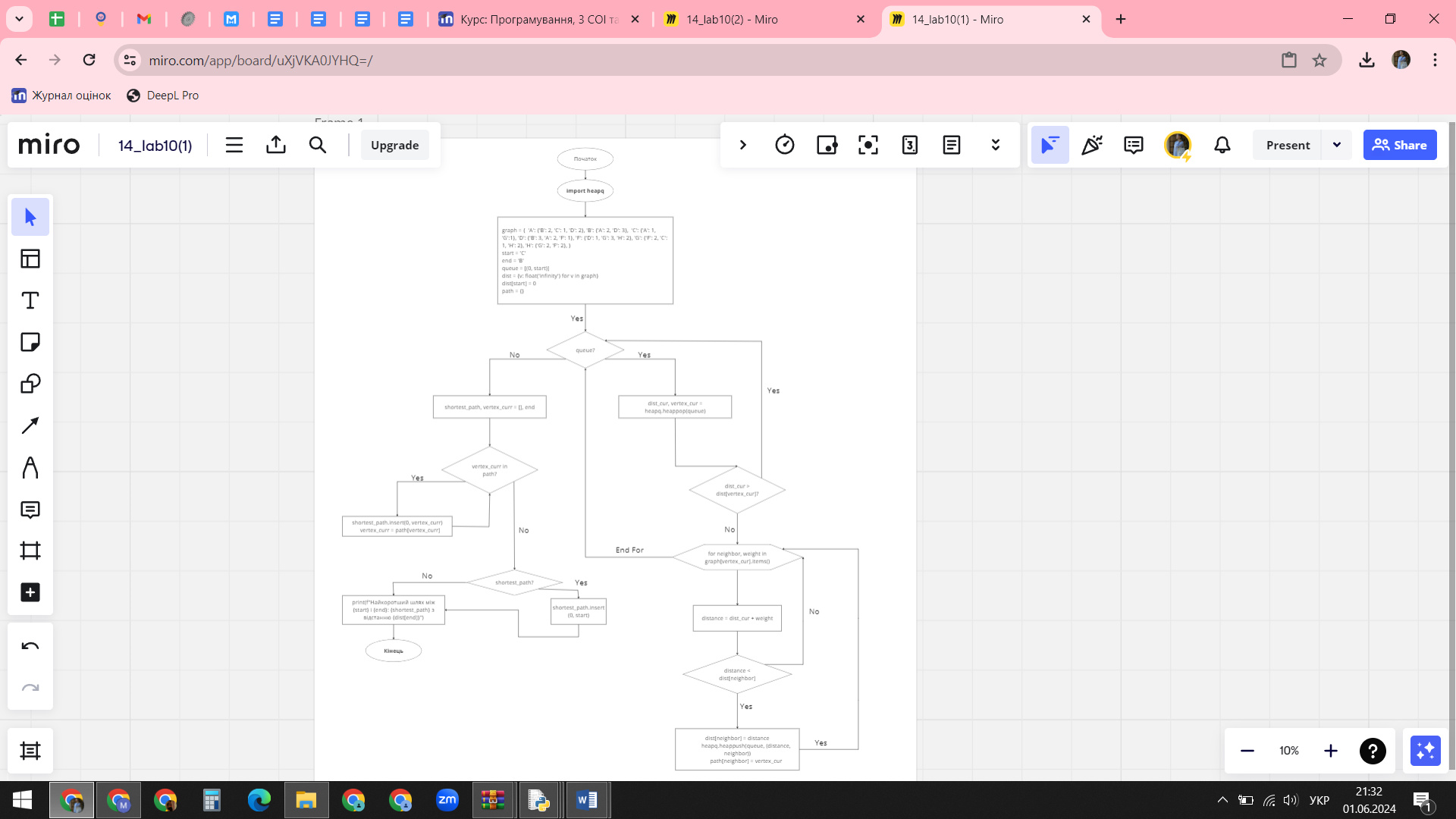
**Вихідні дані:**

* max\_area: площа найбільшого прямокутника, який складається тільки з одиниць у заданій матриці. Це значення виводиться у вигляді повідомлення: "Максимальна площа прямокутника з одиниць: [max\_area]".

**2.2 Блок-схема**

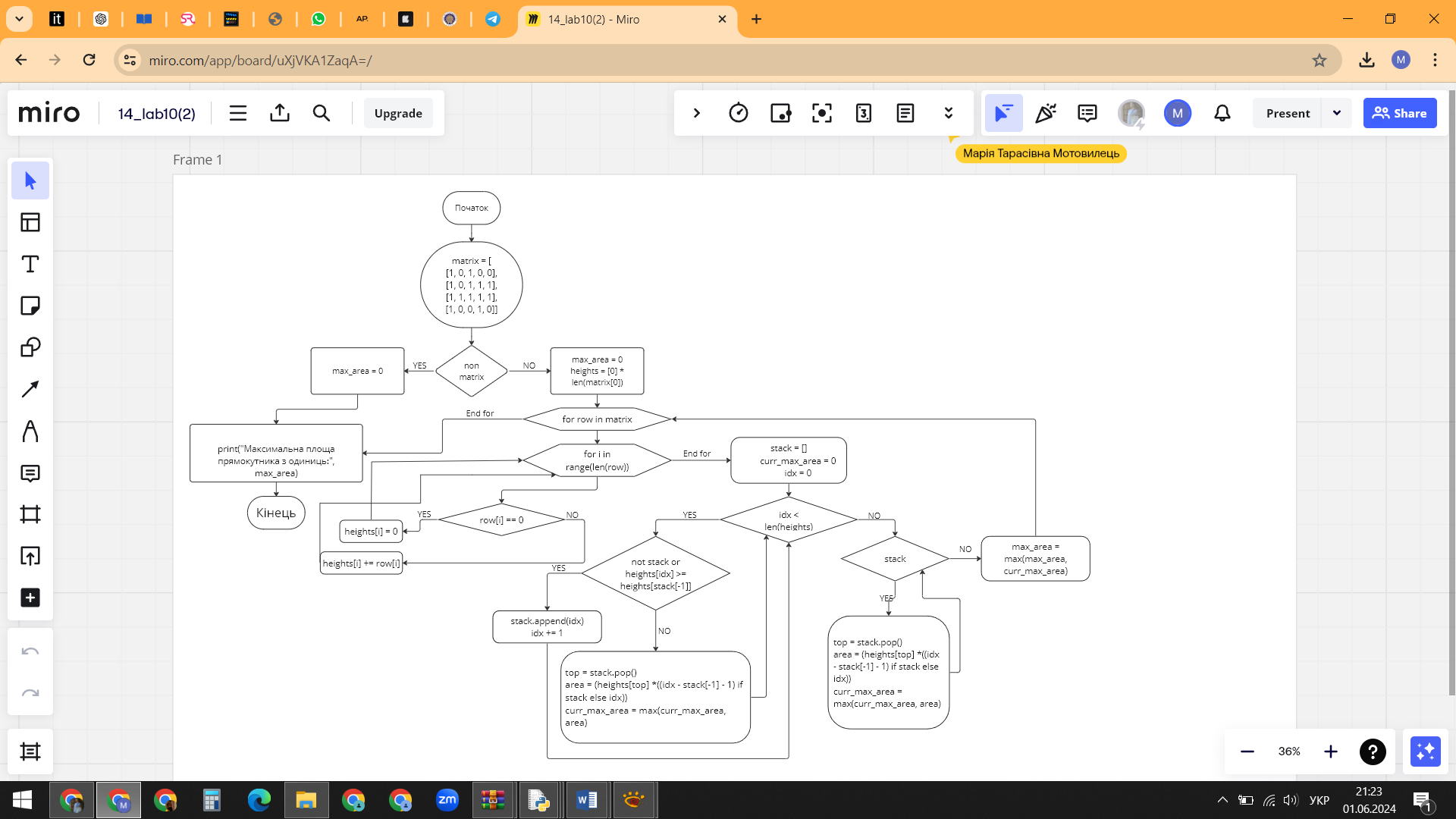
**Задача № 1**

**<https://miro.com/welcomeonboard/Q0FtZ3FoM1EyTmU2MW9MT0lDbkRJelhOaHVocEpmWFJrdXQ3SWlTMjNiaEJyODVXR3JnNHBlcjdoQmpKdFlTTHwzNDU4NzY0NTQ1NDc4OTEzMjUyfDI=?share_link_id=939977199395>**



**Задача №2**

[**https://miro.com/welcomeonboard/YWtDTzhVTzlWOEo1SUNVbTNIaHA2c2UyRWtubDVtT0hOZjcyOFN2OTFBWHBKVnBpRjNvM0dyUktTVmRsajVCN3wzNDU4NzY0NTQ1NDc4OTEzMjUyfDI=?share\_link\_id=758608197358**](https://miro.com/welcomeonboard/YWtDTzhVTzlWOEo1SUNVbTNIaHA2c2UyRWtubDVtT0hOZjcyOFN2OTFBWHBKVnBpRjNvM0dyUktTVmRsajVCN3wzNDU4NzY0NTQ1NDc4OTEzMjUyfDI=?share_link_id=758608197358)



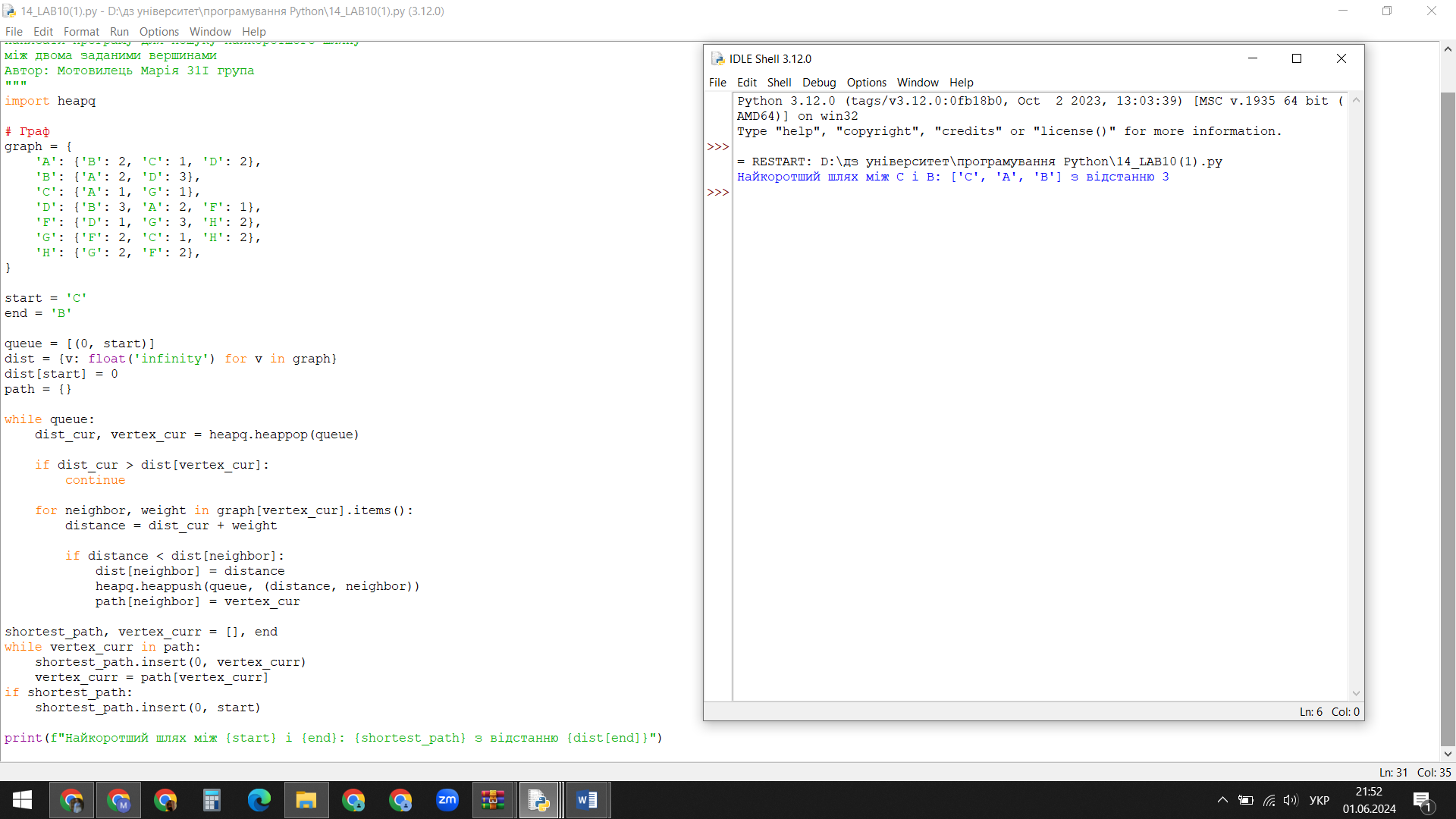
**Висновки**

У цій лабораторній роботі досліджено жадібні алгоритми, графи та динамічне програмування, які є ключовими концепціями алгоритмів. Жадібні алгоритми обирають на кожному кроці найкращий доступний варіант, прагнучи до оптимального розв'язку, і часто застосовуються для оптимізаційних задач. Проте вони не завжди гарантують оптимальний результат і повинні використовуватися лише за умови доведення їхньої ефективності. Графи моделюють ситуації з пов'язаними об'єктами, знаходячи застосування в транспортних, соціальних та комп'ютерних мережах. Алгоритми на графах вирішують задачі пошуку шляхів, компонентів зв'язності, циклів тощо. Динамічне програмування розв'язує проблеми, що розбиваються на підзадачі, комбінуючи їхні оптимальні рішення. Цей метод ефективний для задач оптимізації, таких як пошук найкоротшого шляху та розбиття послідовності.

**Додатки(лістинги програми)**

**Посилання на репозиторій: <https://github.com/mariamotovylets/lab10_Motovylets>**

**Задача №1**



import heapq

# Граф

graph = {

'A': {'B': 2, 'C': 1, 'D': 2},

'B': {'A': 2, 'D': 3},

'C': {'A': 1, 'G': 1},

'D': {'B': 3, 'A': 2, 'F': 1},

'F': {'D': 1, 'G': 3, 'H': 2},

'G': {'F': 2, 'C': 1, 'H': 2},

'H': {'G': 2, 'F': 2},

}

start = 'C'

end = 'B'

queue = [(0, start)]

dist = {v: float('infinity') for v in graph}

dist[start] = 0

path = {}

while queue:

dist\_cur, vertex\_cur = heapq.heappop(queue)

if dist\_cur > dist[vertex\_cur]:

continue

for neighbor, weight in graph[vertex\_cur].items():

distance = dist\_cur + weight

if distance < dist[neighbor]:

dist[neighbor] = distance

heapq.heappush(queue, (distance, neighbor))

path[neighbor] = vertex\_cur

shortest\_path, vertex\_curr = [], end

while vertex\_curr in path:

shortest\_path.insert(0, vertex\_curr)

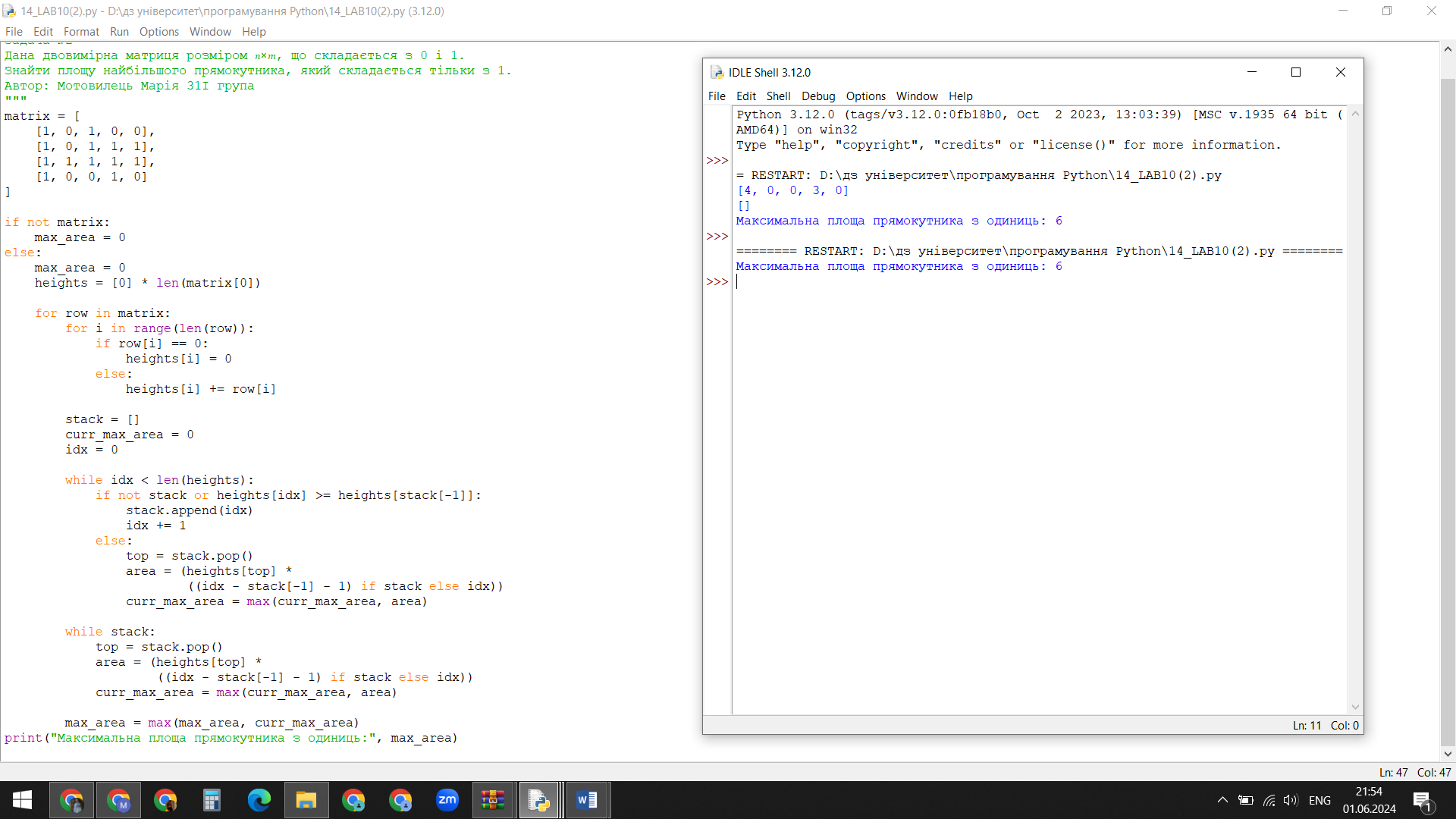
vertex\_curr = path[vertex\_curr]

if shortest\_path:

shortest\_path.insert(0, start)

print(f"Найкоротший шлях між {start} і {end}: {shortest\_path} з відстанню {dist[end]}")

**Задача №2**



matrix = [

[1, 0, 1, 0, 0],

[1, 0, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 0, 0, 1, 0]

]

if not matrix:

max\_area = 0

else:

max\_area = 0

heights = [0] \* len(matrix[0])

for row in matrix:

for i in range(len(row)):

if row[i] == 0:

heights[i] = 0

else:

heights[i] += row[i]

stack = []

curr\_max\_area = 0

idx = 0

while idx < len(heights):

if not stack or heights[idx] >= heights[stack[-1]]:

stack.append(idx)

idx += 1

else:

top = stack.pop()

area = (heights[top] \*

((idx - stack[-1] - 1) if stack else idx))

curr\_max\_area = max(curr\_max\_area, area)

while stack:

top = stack.pop()

area = (heights[top] \*

((idx - stack[-1] - 1) if stack else idx))

curr\_max\_area = max(curr\_max\_area, area)

max\_area = max(max\_area, curr\_max\_area)

print("Максимальна площа прямокутника з одиниць:", max\_area)