**Прізвище:** Новицька

**Ім’я:** Ярина

**Група:** КН-405

**Варіант:** 18

**Кафедра.:** Кафедра Систем

Автоматизованого Проектування

**Дисципліна:** Дискретні моделі в САПР

**Перевірив:** Кривий Р.З.

**GitHub:** [**https://github.com/NovytskaYaryna/Discrete\_models\_CAD**](https://github.com/NovytskaYaryna/Discrete_models_CAD)

**Звіт**

До лабораторної роботи №2

На тему “Алгоритм рішення задачі листоноші”

**Мета роботи:** метою даної лабораторної роботи є вивчення алгоритмів рішення задачі листоноші.

**Короткі теоретичні відомості**

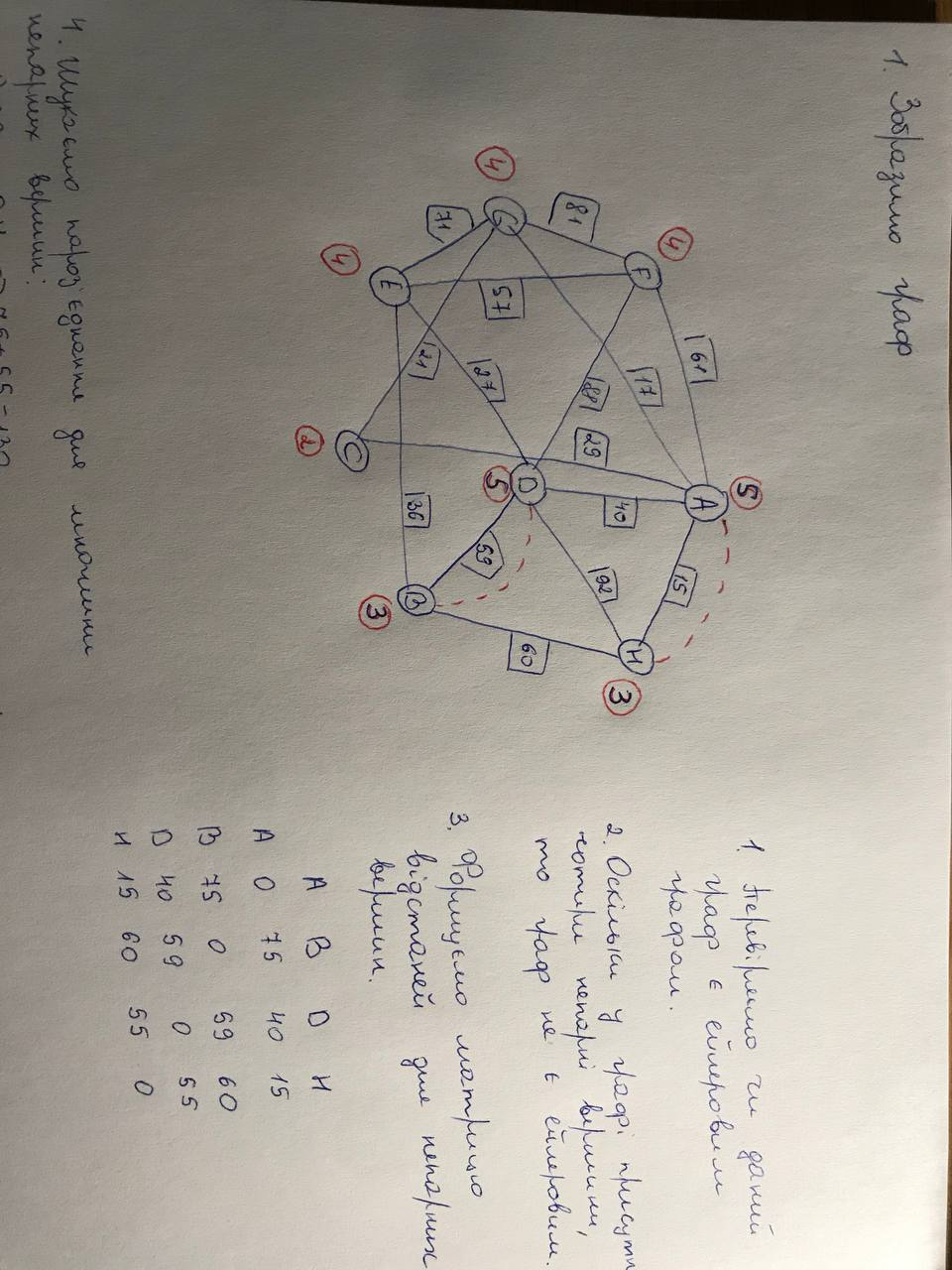
Будь-який листоноша перед тим, як відправитись в дорогу повинен підібрати на пошті листи, що відносяться до його дільниці, потім він повинен рознести їх адресатам, що розмістились вздовж маршрута його проходження, і повернутись на пошту. Кожен листоноша, бажаючи втратити якомога менше сил, хотів би подолати свій маршрут найкоротшим шляхом. Загалом, задача листоноші полягає в тому, щоб пройти всі вулиці маршрута і повернутися в його початкову точку, мінімізуючи при цьому довжину пройденого шляху.

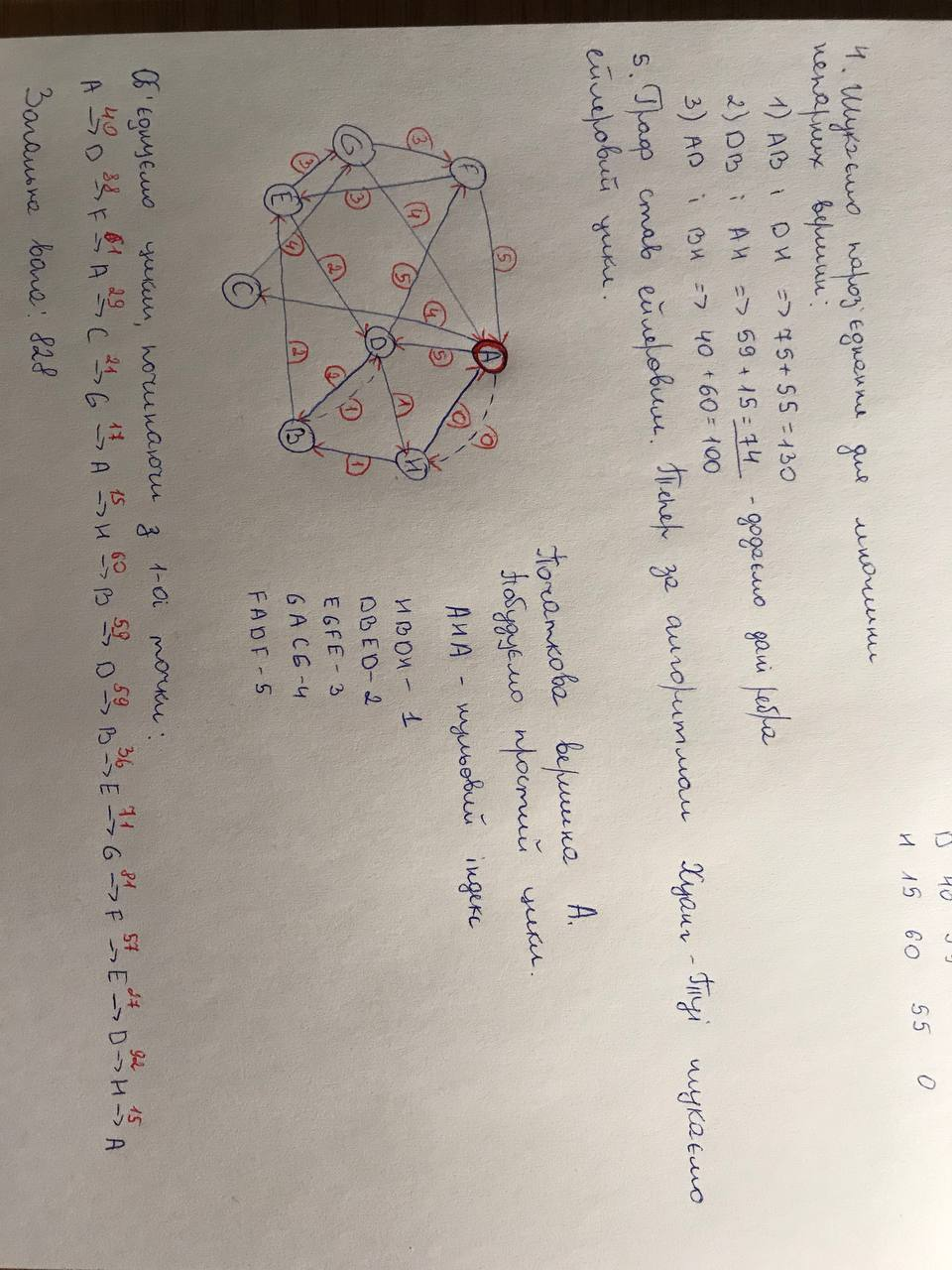
Перша публікація, присвячена рішенню подібної задачі, появилась в одному з китайських журналів, де вона й була названа задачею листоноші. Очевидно, що така задача стоїть не тільки перед листоношею. Наприклад, міліціонер хотів би знати найбільш ефективний шлях патрулювання вулиць свого району, ремонтна бригада зацікавлена у виборі найкоротшого шляху переміщення по всіх дорогах.

Задача листоноші може бути сформульована в термінах теорії графів. Для цього побудуємо граф G = (X , E), в якому кожна дуга відповідає вулиці в маршруті руху листоноші, а кожна вершина - стик двох вулиць. Ця задача являє собою задачу пошуку найкоротшого маршруту, який включає кожне ребро хоча б один раз і закінчується у початковій вершині руху.

**Індивідуальне завдання:** реалізувати алгоритм вирішення задачі листоноші.

Розв\*язання алгоритму китайського листоноші вручну:





**Виконання:**

Матриця суміжності для даної задачі зчитується з зовнішнього файлу формату .txt (Рис. 1)

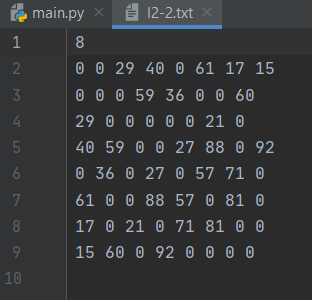


Рис. 1 Файл l2\_2.txt

Наступним кроком є запуск програми:

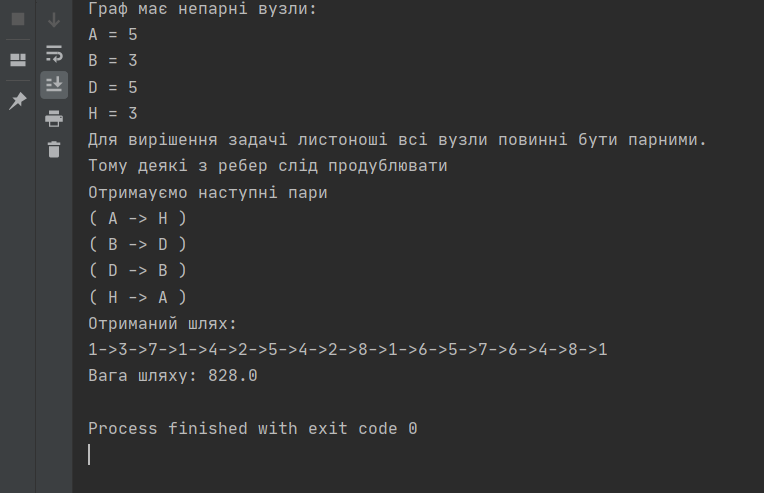


Рис. 2 Запуск програми

**Код пошуку ейлерового циклу:**

def get\_degree(tour):  
 degree = {}  
 for x, y in tour:  
 degree[x] = degree.get(x, 0) + 1  
 degree[y] = degree.get(y, 0) + 1  
 return degree  
  
  
def find\_eulerian\_tour(graph):  
 tour = []  
 deg = get\_degree(graph)  
 for node in deg:  
 if deg.get(node)%2==1:  
 first = node  
 else:  
 first = list(deg.keys())[0]  
   
 node = first  
 tour.append(node)  
 checkpoint=[]  
 while graph:  
 edges = [t for t in graph if t[0] == node or t[1] == node]  
  
 if not edges:  
 tour = checkpoint[-1][0]  
 graph = checkpoint[-1][1]  
 node = checkpoint[-1][2]  
 edges = [t for t in graph if (t[0] == node or t[1] == node) and t != checkpoint[-1][3]]  
 checkpoint.remove(checkpoint[-1])  
   
 path = edges[0]  
 if len(edges) > 1:  
 checkpoint.append([list(tour), list(graph), int(node), tuple(path)])  
   
 if path[0] == node:  
 node = path[1]  
 else:  
 node = path[0]  
   
 tour.append(node)  
 graph.remove(path)  
   
 return tour

**Висновок:** в ході виконання лабораторної роботи було отримано теоретичні знання про методи вирішення задачі листоноші, написано програму, яка реалізує алгоритм розв’язання задачі листоноші та проведено обрахунки вручну.