

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



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

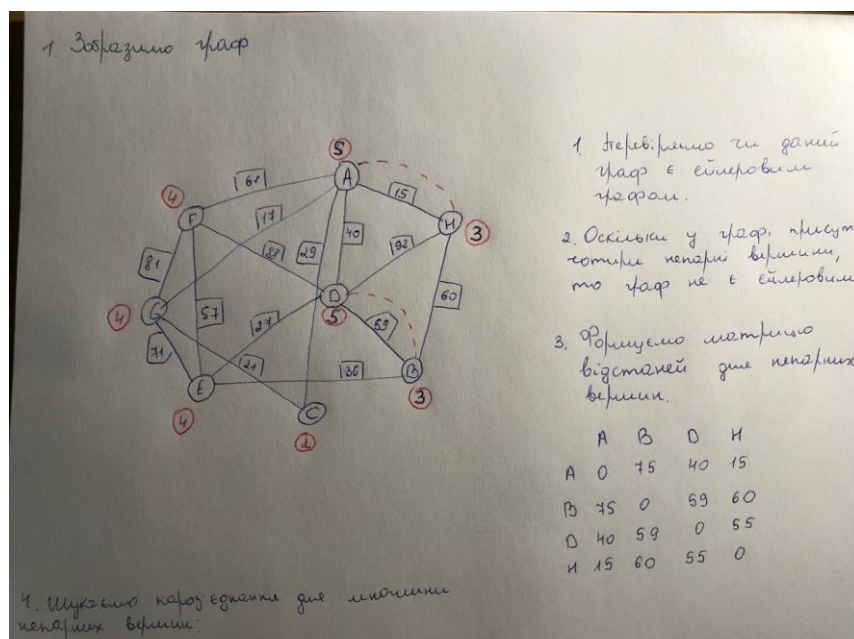
Будь-який листоноша перед тим, як відправитись в дорогу повинен підібрати на пошті листи, що відносяться до його ділянки, потім він повинен рознести їх адресатам, що розмістились вздовж маршрута його проходження, і повернутись на пошту. Кожен листоноша, бажаючи втратити якомога менше сил, хотів би подолати 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 рішення 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:



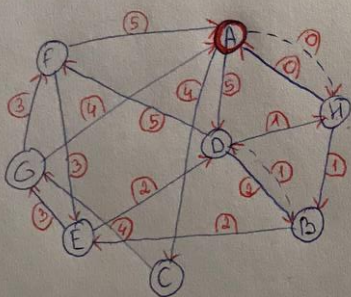
4. Шукаємо на роз'єднання для лінійних неперервних вершин:

$$1) AB : DH \Rightarrow 75 + 55 = 130$$

$$2) DB : AH \Rightarrow 59 + 15 = 74 \text{ - додано дані / ед/а}$$

$$3) AD : BH \Rightarrow 40 + 60 = 100$$

5. Граф став єйлеровим. Печер з алгоритмом Хуана-Тіуї шукаємо єйлеровий цикл.



Точаткова вершина А.
Побудуємо простий цикл.

АНА - базовий індекс

НВДН - 1

ДВЕД - 2

ЕГФЕ - 3

БАСБ - 4

ФАДФ - 5

Об'єднуємо цикли, починаючи з 1-ої точки:

A $\xrightarrow{40}$ D $\xrightarrow{59}$ F $\xrightarrow{61}$ A $\xrightarrow{29}$ C $\xrightarrow{21}$ G $\xrightarrow{17}$ A $\xrightarrow{15}$ H $\xrightarrow{60}$ B $\xrightarrow{59}$ D $\xrightarrow{36}$ E $\xrightarrow{71}$ G $\xrightarrow{81}$ F $\xrightarrow{57}$ E $\xrightarrow{27}$ D $\xrightarrow{92}$ H $\xrightarrow{15}$ A

Записана база: 828

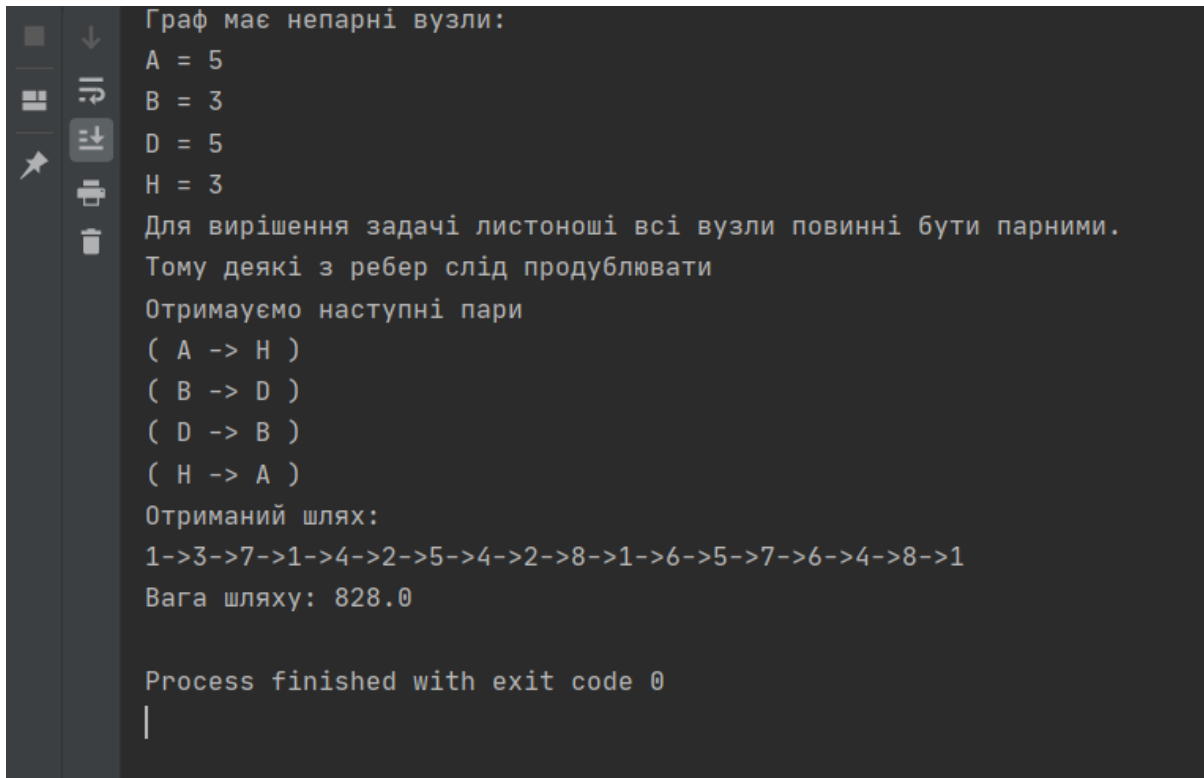
Виконання:

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

	1	2	3	4	5	6	7	8	9	10
1	8									
2	0	0	29	40	0	61	17	15		
3	0	0	0	59	36	0	0	60		
4	29	0	0	0	0	0	21	0		
5	40	59	0	0	27	88	0	92		
6	0	36	0	27	0	57	71	0		
7	61	0	0	88	57	0	81	0		
8	17	0	21	0	71	81	0	0		
9	15	60	0	92	0	0	0	0		
10										

Рис. 1 Файл l2_2.txt

Наступним кроком є запуск програми:



```
Граф має непарні вузли:
A = 5
B = 3
D = 5
H = 3
Для вирішення задачі листиноші всі вузли повинні бути парними.
Тому деякі з ребер слід продублювати
Отримаємо наступні пари
( A -> H )
( B -> D )
( D -> B )
( H -> A )
Отриманий шлях:
1->3->7->1->4->2->5->4->2->8->1->6->5->7->6->4->8->1
Вага шляху: 828.0

Process finished with exit code 0
|
```

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.