Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1

з дисципліни «Вступ до штучного інтелекту»

на тему «Ознайомлення з середовищем Jupyter Notebook»

Виконав:
студент групи ІМ-22
Довженко Антон Андрійович

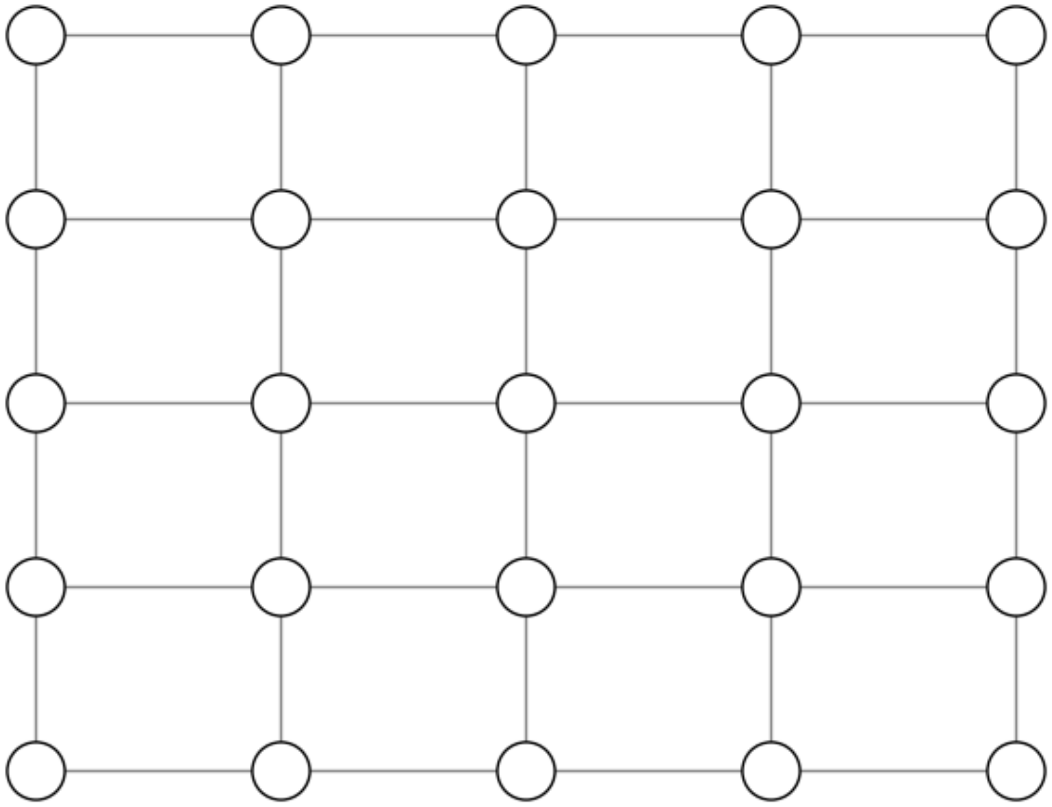
Перевірив:
Ю. П. Кочура

Київ 2024

1. Розробити алгоритм генерації дороги у вигляді графу.

Граф-дорога має бути наступної структури:

Де вершини графу це перехрестя, а ребра графу – дороги між перехрестями. Перехрестя повинні бути розташовані у вигляді квадратної сітки.



Код для відображення першого графу (зі всіма ребрами):

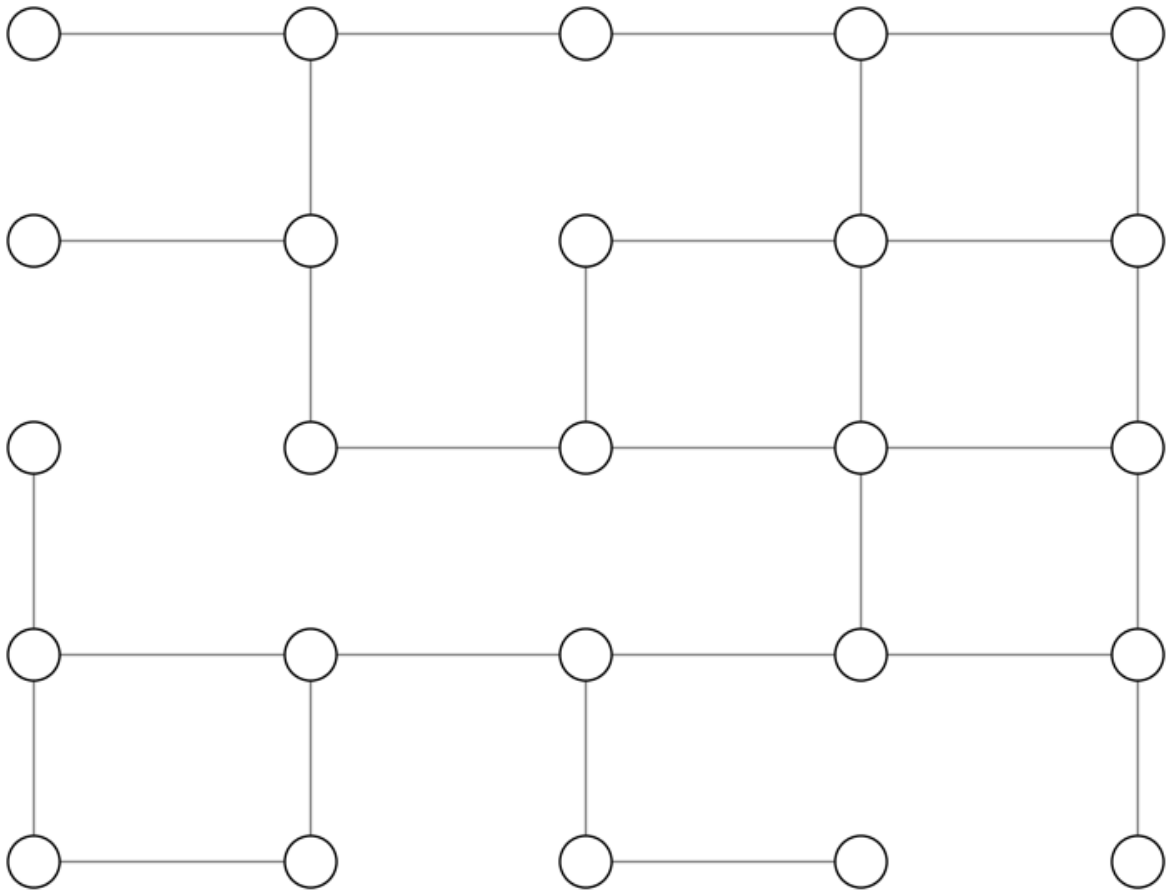
```
import networkx as nx
import matplotlib.pyplot as plt


G = nx.grid_2d_graph(5, 5)


pos = {(x, y): (x, y) for x, y in G.nodes()}


nx.draw(G, pos, with_labels=False, node_size=500, node_color='white',
edgecolors='black', linewidths=1, edge_color='gray')
plt.show()
```

Цей код створює двовимірний сітковий граф розміром 5x5 за допомогою бібліотеки NetworkX і візуалізує його за допомогою Matplotlib. Граф відображається без міток на вузлах, з білими вузлами, чорними контурами та сірими ребрами.

2. Зі згенерованого графу потрібно видалити певну кількість ребер так, щоб результуючий граф залишився повністю зв’язним.

Конфігурованими повинні бути наступні параметри генерації:

- Розмір графу (кількість вершин-перехресть);
- Кількість ребер для видалення;



Код для відображення другого графу (на прикладі графу 5 на 5, з 10 видаленими ребрами):

```
import networkx as nx

import matplotlib.pyplot as plt

import random
```

```
def generate_removable_edges(G):
    all_edges = list(G.edges())

    random.shuffle(all_edges)

    return all_edges

def remove_edges_while_connected(G, edges_to_remove, max_edges_to_remove):
    G_copy = G.copy()

    removed_edges = []

    for edge in edges_to_remove:
        if len(removed_edges) >= max_edges_to_remove:
            break

        G_copy.remove_edge(*edge)

        if not nx.is_connected(G_copy):
            G_copy.add_edge(*edge)
        else:
            removed_edges.append(edge)

    return removed_edges, G_copy

def visualize_graph(G_removed):
    pos = {(x, y): (x, y) for x, y in G_removed.nodes()}

    plt.figure(figsize=(8, 6))

    nx.draw(G_removed, pos, with_labels=False, node_size=500,
node_color='white', edgecolors='black', linewidths=1, edge_color='gray')
```

```
    plt.show()


# graph size
G = nx.grid_2d_graph(5, 5)


# number of edges to remove
max_edges_to_remove = 10


edges_to_remove = generate_removable_edges(G)


removed_edges, G_removed = remove_edges_while_connected(G,
edges_to_remove, max_edges_to_remove)


visualize_graph(G_removed)
```

Цей код імплементує аналіз сіткового графа розміром 5x5, використовуючи бібліотеки NetworkX та Matplotlib. Він спочатку створює список всіх ребер графа і випадковим чином перемішує їх за допомогою функції generate_removable_edges. Далі функція remove_edges_while_connected видаляє ребра, перевіряючи, чи залишається граф зв'язаним після кожного видалення, і зберігає список видалених ребер. Результуючий граф візуалізується за допомогою функції visualize_graph, яка використовує Matplotlib для побудови графу. Основна частина коду створює граф, визначає кількість ребер для видалення, видаляє їх, зберігаючи зв'язність, і відображає результат.

Висновок:

Отже, ця лабораторна робота дозволила мені ознайомитися з середовищем Jupyter Notebook та бібліотеками для аналізу і візуалізації графів, такими як NetworkX і Matplotlib. Я навчився створювати та маніпулювати графами, а також ефективно візуалізувати результати. Практичне застосування цих інструментів допомогло краще зрозуміти як функціонують графи та їх властивості. Візуалізація графів дозволила наочно побачити результати аналізу та краще зрозуміти структуру даних. Цей досвід є корисним для подальшого дослідження і роботи з графами в різних контекстах.