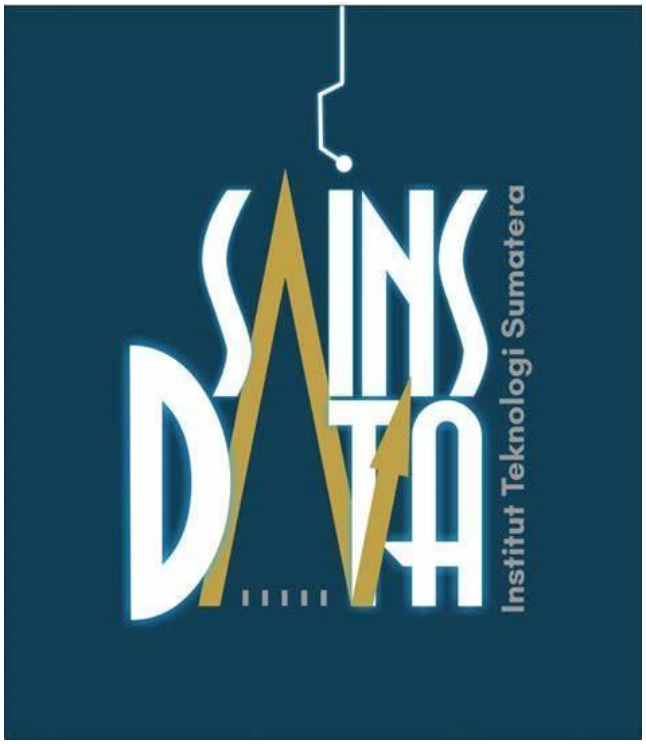
MODUL PRAKTIKUM
MATA KULIAH: DATA MINING
MODUL VI
[**Data Graf I**]



Program Studi Sains Data

Fakultas Sains

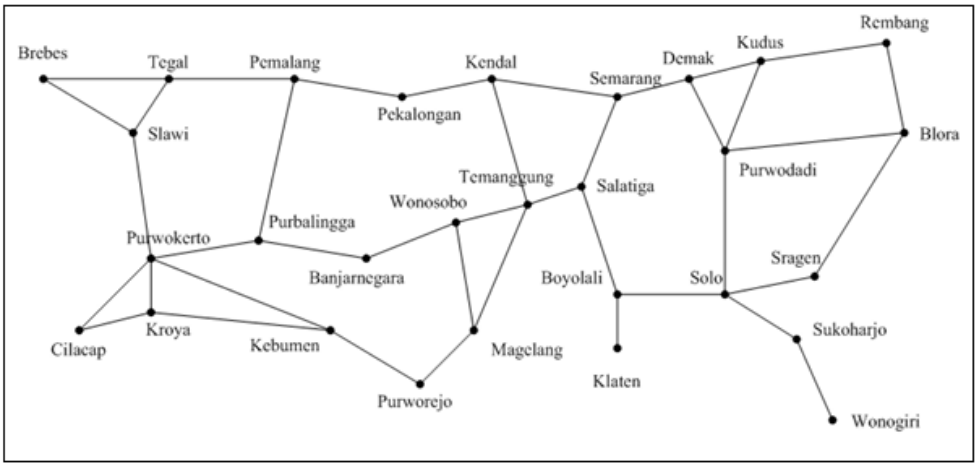
INSTITUT TEKNOLOGI SUMATERA

Tahun Ajaran Ganjil 2024/2025.

1. Tujuan Praktikum
    a. Memahami dan menerapkan konsep-konsep dasar graf.
    b. Mengembangkan keterampilan analisis dan pemecahan masalah yang melibatkan struktur data graf.
2. Tinjauan Pustaka
    a. Konsep Graf
        i. Graf

        Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objekobjek tersebut. Representasi visual dari graf adalah dengan menyatakan objek sebagai noktah, bulatan, atau titik, sedangkan hubungan antara objek dinyatakan dengan garis.

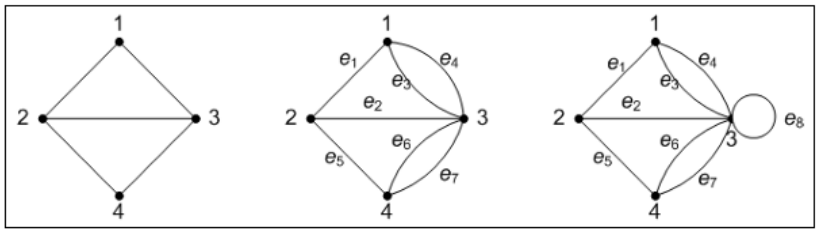
        Definisi Graf adalah representasi visual atau struktural dari data di mana entitas (node atau simpul) dan hubungan antar entitas (edges atau tepi) dimodelkan untuk mencerminkan bagaimana titik-titik data saling terhubung. Graf data sangat berguna untuk memahami hubungan dan pola antar data $G = (V, E)$ yaitu Graf terdiri dari dua hal yaitu V (vertices/nodes), Himpunan dari simpul atau node dalam graf dan E (edges). Dalam graf tak berarah, sisi adalah pasangan tidak terurut dari simpul. Salah satu contoh graf adalah sebagai berikut:



Gambar 1 contoh graf(Peta Jaringan Jalan Raya)

Diatas sudah di jelaskan bahwa graf dapat ditulis dengan menggunakan notasi G = (V, E), dengan V = himpunan tidak-kosong dari simpul-simpul (vertices) = { v1 , v2 , ... , vn }. Sedangkan untuk nilai edges dapat diartikan E = himpunan sisi (edges) yang menghubungkan sepasang simpul = {e1 , e2 , ... , en }.

Simpul pada graf dapat dinomori dengan huruf seperti v, w, ..., dengan bilangan asli 1, 2, 3, ... atau gabungan keduannya. Sedangkan sisi yang menghubngkan simpul vi dengan vj dinyatakan dengan pasangan (vi, vj) atau dengan lambang e1, e2, ... Dengan kata lain, jika e adalah sisi yang menghubungkan simpul vi dengan vj, maka e dapat ditulis sebagai e = (vi, vj). Gambar dibawah ini menunjukkan beberapa contoh graf .



Gambar 2 contoh dari graf

b. *Representasi Graf*

graf harus direpresentasikan di dalam memori. Terdapat beberapa representasi yang mungkin untuk graf. Di sini hanya diberikan tiga macam representasi yang sering digunakan pada graf sederhana, yaitu matriks ketetanggaan, matrik bersisian, dan senarai ketetanggaan.

i. Matriks Ketetanggaan (*Adjancency Matrix*)

graf dapat menggunakan matriks, Salah satunya adalah matriks ketetanggaan. Misalkan G = (V, E) adalah sebuah graf sederhana dimana |V| = n, n > 1. Misalkan simpul dari G adalah v1, v2, … vn. Maka, matriks ketetanggaan A dari G adalah n x n matriks dimana:

$$\mathbf{A}(i,j) = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Matriks ketetanggaan dari sebuah graf mengacu pada keterurutan dari simpul. Sehingga, ada sebanyak n! keterurutan yang berbeda yang terbentuk dari n simpul.

Jika grafiknya berarah, maka matriks ketetanggaan A tidak simetris, karena (vi,vj) ∈ E jelas tidak menyiratkan bahwa (vj,vi) ∈E.

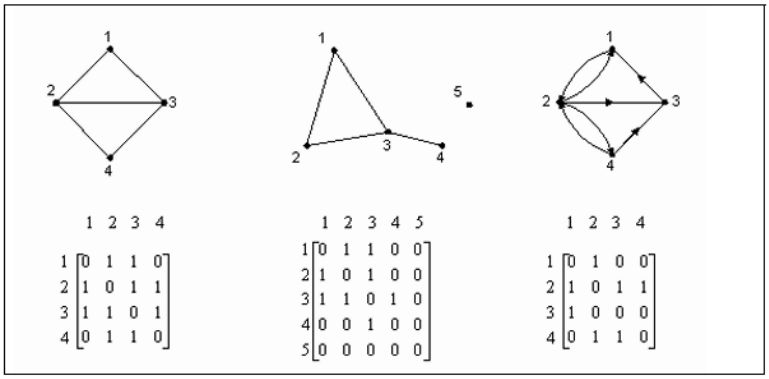
Jika grafiknya terbobot, maka kita memperoleh matriks ketetanggaan terbobot n × n, A, yang didefinisikan sebagai:

$$\mathbf{A}(i,j) = \begin{cases} w_{ij} & \text{if } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

di mana w_{ij} adalah bobot pada sisi $(v_i,v_i) \in E$. Matriks ketetanggaan yang diberi bobot selalu dapat diubah menjadi matriks biner, jika diinginkan, dengan menggunakan beberapa ambang batas (*threshold*)τ pada bobot sisi.

$$\mathbf{A}(i,j) = \begin{cases} 1 & \text{if } w_{ij} \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

Berikut contoh graf dengan matriks ketegangan/ *Adjancency Matrix*



Gambar 3 graf dengan Adjancency Matrix

Matriks ketetanggaan merupakan graf sederhana yang simetris, yaitu a_{ij} = aji. Hal ini disebabkan oleh kedua-duanya adalah 1 ketika v_i dan v_j mempunyai sisi, dan adalah 0 jika tidak ada sisi diantaranya, dinamakan juga matriks nol-satu, sedangkan untuk graf berarah, mtriks ketetanggaanya belum tentu simetri (akan berupa simtri jika berupa graf berarah lengkap). Selain itu, graf sederhana tidak mempunyai gelang, sehingga diagonal utamanya selalu 0 karena aii, i = 1, 2, 3, …, n adalah 0. Sayangnya, matrks ketetanggaan nol-satu tidak dapat digunakan untuk merepresentasikan graf yang mempunyai sisi ganda. Untuk mengatasinya, maka elemen aij pada matiks ketetanggaan sama dengan jumlah yang berasosiasi dengan (v_i, v_j). Matriks ketetanggaannya bukan lagi matriks nol-satu Untuk graf semu, gelang pada simpul vi dinyatakan dengan nilai satu pada posisi (i, j) di matriks ketetanggaannya. Keuntungan representasi dengan matriks adalah, elemen matriksnya dapat diakses langsung melalui indeks. Selain itu, kita dapat mengetahui secara langsung apakah simpul i dan simpul j bertetangga.

ii. Graf dari Data Matrix

Banyak himpunan data yang tidak berbentuk grafik dapat diubah menjadi satu. Misalkan D adalah himpunan data yang terdiri dari n titik

xi∈R dalam ruang berdimensi d. Kita dapat mendefinisikan grafik berbobot G = (V, E), di mana terdapat simpul untuk setiap titik di D, dan terdapat sisi di antara setiap pasangan titik, dengan bobot:

$$w_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j) \quad \text{.....(a)}$$

di mana sim(x_i,x_j) menyatakan kesamaan antara titik x_i dan x_j. Misalnya, kesamaan dapat didefinisikan sebagai berbanding terbalik dengan jarak Euclidean antara titik-titik melalui transformasi.

$$w_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right\} \quad \text{.....(b)}$$

di mana σ adalah parameter sebaran (setara dengan deviasi standar dalam fungsi kepadatan normal). Transformasi ini membatasi fungsi kesamaan sim() agar berada dalam rentang [0, 1]. Kita kemudian dapat memilih ambang batas τ yang sesuai dan mengubah matriks ketetanggaan tertimbang menjadi matriks biner melalui Persamaan (a).

c. Atribut Topologi graf

i. Degree(derajat)

Kita telah mendefinisikan derajat sebuah simpul melalui jumlah tetangganya. Definisi yang lebih umum yang berlaku bahkan ketika grafik diberi bobot adalah sebagai berikut:

$$d_i = \sum_j \mathbf{A}(i, j)$$

Derajat jelas merupakan atribut lokal dari setiap node. Salah satu atribut global yang paling sederhana adalah derajat rata-rata:

$$\mu_d = \frac{\sum_i d_i}{n}$$

Definisi sebelumnya dapat dengan mudah digeneralisasi untuk grafik berarah (berbobot). Misalnya, kita dapat memperoleh derajat dan derajat keluar dengan mengambil penjumlahan dari sisi masuk dan keluar, sebagai berikut.

$$id(v_i) = \sum_j \mathbf{A}(j, i)$$

$$od(v_i) = \sum_j \mathbf{A}(i, j)$$

Rata-rata indeks dan rata-rata derajat keluar dapat diperoleh dengan cara yang sama.

ii. Average Path L ength(Panjang Jalur Rata-rata)

Panjang jalur rata-rata, juga disebut panjang jalur karakteristik, dari grafik terhubung diberikan sebagai berikut:

$$\mu_L = \frac{\sum_i \sum_{j>i} d(v_i, v_j)}{\binom{n}{2}} = \frac{2}{n(n-1)} \sum_i \sum_{j>i} d(v_i, v_j)$$

di mana n adalah jumlah simpul dalam grafik, dan $d(v_i,v_j)$ adalah jarak antara v_i dan v_j. Untuk grafik berarah, rata-rata adalah semua pasangan simpul yang diurutkan:

$$\mu_L = \frac{1}{n(n-1)} \sum_i \sum_j d(v_i, v_j)$$

Untuk grafik terputus, rata-rata hanya diambil dari pasangan simpul yang terhubung.

iii. Eccentricity

Eksentrisitas suatu simpul v_i adalah jarak maksimum dari v_i ke simpul lain dalam grafik.

$$e(v_i) = \max_j \{ d(v_i, v_j) \}$$

Jika grafiknya terputus, eksentrisitas hanya dihitung pada pasangan titik sudut dengan jarak terhingga, yaitu hanya empat titik sudut yang dihubungkan oleh suatu lintasan.

iv. Radius and Diameter

Jari-jari grafik terhubung, dilambangkan r(G), adalah eksentrisitas minimum dari setiap simpul dalam grafik:

$$r(G) = \min_i \{ e(v_i) \} = \min_i \{ \max_j \{ d(v_i, v_j) \} \}$$

Diameter, dilambangkan d(G), adalah eksentrisitas maksimum dari setiap titik sudut pada grafik.

$$d(G) = \max_i \{ e(v_i) \} = \max_{i,j} \{ d(v_i, v_j) \}$$

Untuk grafik terputus, diameter adalah e c sentrisitas maksimum atas semua komponen grafik yang terhubung. Diameter grafik G sensitif terhadap outlier.

3. Prosedur Praktikum

a. Konsep graf

Untuk memahami graf kita dapat menijau studi kasus dibawah ini:

Graf memiliki $|V| = 8$ simpul dan $|E| = 11$ sisi. Karena $(v_1, v_5) \in E$, kita mengatakan bahwa v_1 dan v_5 adalah tetangga. Derajat dari v_1 adalah $d(v_1) = d_1 = 4$. Urutan derajat dari graf ini adalah:

$$(4, 4, 4, 3, 2, 2, 2, 1)$$

dan distribusi frekuensi derajatnya adalah:

$$(N_0, N_1, N_2, N_3, N_4) = (0, 1, 3, 1, 3)$$

Kita memiliki $N_0 = 0$ karena tidak ada simpul yang terisolasi, dan $N_4 = 3$ karena ada tiga simpul, yaitu v_1, v_4, dan v_5, yang memiliki derajat $k = 4$; angka lainnya diperoleh dengan cara yang sama. Distribusi derajat diberikan sebagai:

$$(f(0), f(1), f(2), f(3), f(4)) = (0, 0.125, 0.375, 0.125, 0.375)$$

Urutan simpul $(v_3, v_1, v_2, v_5, v_1, v_2, v_6)$ adalah jalur dengan panjang 6 antara v_3 dan v_6. Kita dapat melihat bahwa simpul v_1 dan v_2 telah dikunjungi lebih dari satu kali.

Source Code:

```python
import networkx as nx
import matplotlib.pyplot as plt
from collections import Counter
# Membuat graf
G = nx.Graph()
# Menambahkan simpul dan sisi (edges)
edges = [(1, 2), (1, 5), (1, 3), (1, 4), (2, 5), (2, 6),
 (3, 4), (3, 6), (4, 5), (5, 6), (6, 7)]
G.add_edges_from(edges)
# Menghitung urutan derajat (degree sequence)
degree_sequence = sorted([d for n, d in G.degree()], reverse=True)
print("Urutan derajat:", degree_sequence)
# Menghitung distribusi frekuensi derajat
degree_count = Counter(degree_sequence)
```

```python
N_k = [degree_count.get(i, 0) for i in range(max_degree + 1)]
print("Distribusi frekuensi derajat (N_k):", N_k)
# Menghitung distribusi derajat (f(k))
n = len(G.nodes())
f_k = [N_k[i] / n for i in range(max_degree + 1)]
print("Distribusi derajat (f(k)):", f_k)
# Menggambar graf
nx.draw(G, with_labels=True, node_color='lightblue', node_size=2000,
        font_size=12, font_color='black', edge_color='gray')
plt.title("Graf dari studi kasus")
plt.show()
```

Graphs from Data Matrix

Database yang akan digunakan adalah data iris dapat dilihat dari gambar dibawah ini :

Table 1.1. Extract from the Iris dataset

	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Class X_5
$\mathbf{x}_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$\mathbf{x}_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$\mathbf{x}_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$\mathbf{x}_4$	4.6	3.2	1.4	0.2	Iris-setosa
$\mathbf{x}_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$\mathbf{x}_6$	4.7	3.2	1.3	0.2	Iris-setosa
$\mathbf{x}_7$	6.5	3.0	5.8	2.2	Iris-virginica
$\mathbf{x}_8$	5.8	2.7	5.1	1.9	Iris-virginica
⋮	⋮	⋮	⋮	⋮	⋮
$\mathbf{x}_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$\mathbf{x}_{150}$	5.1	3.4	1.5	0.2	Iris-setosa

Studi kasus 2 : dengen menggunkan database iris buatlah Kemiripan (similarity)berpasangan antara pasangan titik yang berbeda dengan $\sigma = \frac{1}{\sqrt{2}}$ (kita tidak memperbolehkan adanya lingkaran, untuk menjaga kesederhanaan grafik). Rata-rata kemiripan antara titik adalah 0.197, dengan standar deviasi 0.290. Matriks adjacency biner menggunakan ambang batas τ=0.777, yang menghasilkan sisi antara titik-titik yang memiliki kemiripan lebih tinggi dari dua standar deviasi dari rata-rata. Grafik yang dihasilkan memiliki 150 node dan 753 sisi.

Source code python:

kita memanggil dataset iris menggunakan Sklearn dan menggambil data sebnyal 150 data dengan random.

```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
import networkx as nx
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
# Load and shuffle iris data
data = load_iris()
iris_features, iris_target = shuffle(data.data, data.target, random_state=10000)
# Use all 150 samples
iris_features = iris_features[:150]
```

Definisikan perumusan dari graf:

```python
# Define sigma and calculate similarity matrix using the given formula
sigma = 1 / np.sqrt(2)
similarity_matrix = np.exp(-np.square(np.linalg.norm(iris_features[:, np.newaxis]
                                                     - iris_features, axis=2)) / (2 * sigma**2))
# Calculate mean and standard deviation of similarities
mean_similarity = np.mean(similarity_matrix)#mean
std_similarity = np.std(similarity_matrix) #standard deviation of similarities
# Set threshold to mean + 2 * std deviation
threshold = mean_similarity + 2 * std_similarity
```

```python
# Create a graph using NetworkX
graph = nx.Graph()
# Add nodes to the graph
for i in range(len(iris_features)):
    graph.add_node(i, label=data.target_names[iris_target[i]])
# Add edges based on similarity threshold
for i in range(len(iris_features)):
    for j in range(i + 1, len(iris_features)):
        if similarity_matrix[i, j] >= threshold:
            graph.add_edge(i, j, weight=similarity_matrix[i, j])
# Draw the graph with different symbols for each species
pos = nx.spring_layout(graph, seed=150)  # Position nodes using spring layout
plt.figure(figsize=(8, 5))
```

```
# Define markers and colors for each species
markers = {'setosa': 's', 'versicolor': 'o', 'virginica': '^'}
colors = {'setosa': 'green', 'versicolor': 'blue', 'virginica': 'coral'}
node_shapes = {species: [] for species in markers.keys()}
# Assign nodes to their respective shapes and colors based on species
for i in range(len(iris_features)):
    species = data.target_names[iris_target[i]]
    node_shapes[species].append(i)
# Draw nodes with different shapes and transparent colors for each species
for species, nodes in node_shapes.items():
    nx.draw_networkx_nodes(graph, pos, nodelist=nodes, node_shape=markers[species],
                           node_color=colors[species], alpha=0.77, node_size= 5, label=species)
```

```
# Draw edges and labels
nx.draw_networkx_edges(graph, pos, edge_color='grey', alpha=0.3)
nx.draw_networkx_labels(graph, pos, labels={i: '' for i in range(len(iris_features))}, font_size=10)
plt.legend(scatterpoints=1)
plt.title("Iris Similarity Graph")
plt.show()
```

b. Topological Attributes

dalam graf ada beberapa attribut yang digunakan seperti mencari radius dan diameter connected graph serta derajat dari graf.

Studi kasus untuk eksentrisitas, jari-jari,diameter: jika anda lakukan analisis graf menggunakan dataset **Iris** dengan beberapa langkah. Anda akan memanfaatkan jarak Euclidean untuk membangun graf, menghitung properti graf seperti eksentrisitas, jari-jari, dan diameter. Kita akan meng

Source code python :

```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
# 1. Load the Iris dataset and prepare it as a DataFrame
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

```
# 2. Create Graph using Euclidean distance
def create_graph(data):
    G = nx.Graph()
    for i in range(len(data)):
        for j in range(i + 1, len(data)):
            distance = np.linalg.norm(data.iloc[i] - data.iloc[j])
            G.add_edge(i, j, weight=distance)
    return G
# Create the graph from the data
G = create_graph(data)
```

```
# 3. Compute Eccentricity
eccentricity = nx.eccentricity(G)
print("Eccentricity of each node:", eccentricity)
# 4. Compute Radius
radius = min(eccentricity.values())
print("Radius of the graph:", radius)
# 5. Compute Diameter
diameter = nx.diameter(G)
print("Diameter of the graph:", diameter)
```

Studi kasus untuk derjat(degree): buatlah distribusi derajat (*degree distribution*) dari data iris dengan menggunakan historgram.

Studi kasus untuk degree distribution:

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
# 1. Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
# 2. Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```python
# 3. Compute pairwise similarities using Euclidean distance
def pairwise_similarity(X):
    n_samples = X.shape[0]
    similarity_matrix = np.zeros((n_samples, n_samples))
    for i in range(n_samples):
        for j in range(i+1, n_samples):
            distance = np.linalg.norm(X[i] - X[j])
            similarity_matrix[i, j] = np.exp(-distance)  # Similarity function
            similarity_matrix[j, i] = similarity_matrix[i, j]  # Ensure symmetry
    return similarity_matrix
```

```python
# Compute similarity matrix
similarity_matrix = pairwise_similarity(X_scaled)
# 4. Compute mean and standard deviation of similarities
mean_similarity = np.mean(similarity_matrix)
std_similarity = np.std(similarity_matrix)
threshold = mean_similarity + 2 * std_similarity
# 5. Create binary adjacency matrix based on the threshold
adjacency_matrix = (similarity_matrix > threshold).astype(int)
# Ensure the adjacency matrix is symmetric
adjacency_matrix = np.maximum(adjacency_matrix, adjacency_matrix.T)
# 6. Build the graph from the adjacency matrix
G = nx.DiGraph(adjacency_matrix)
# 7. Calculate the degree of each node
degrees = [G.degree(n) for n in G.nodes()]
```

```python
# 8. Visualize the degree distribution using a histogram
plt.figure(figsize=(8, 6))
plt.hist(degrees, bins=range(min(degrees), max(degrees) + 2),
         align='left', rwidth=0.8, color='lightgrey', edgecolor='black')
plt.title("Degree Distribution of the Iris Graph")
plt.xlabel("Degree")
plt.ylabel("Frequency")
# Add frequency numbers on top of each bar
for i, count in enumerate(np.bincount(degrees)):
    if count > 0:
        plt.text(i, count, str(count), ha='center', va='bottom')
plt.show()
```