**Lab 4**

**Clustering**

1. Write a program to illustrate the k-means clustering algorithm.

K-means clustering is an unsupervised learning algorithm used to partition a dataset into ‘k’ distinct clusters based on feature similarity. The algorithm iteratively assigns data points to clusters and updates the centroids to minimize the variance within each cluster. This program demonstrates the implementation of the k-means clustering algorithm on a sample dataset, which helps in grouping data points based on their inherent similarities.

Program:

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

iris = load\_iris()

X = iris.data

# K-Means clustering

kmeans = KMeans(n\_clusters=3, random\_state=42)

y\_kmeans = kmeans.fit\_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')

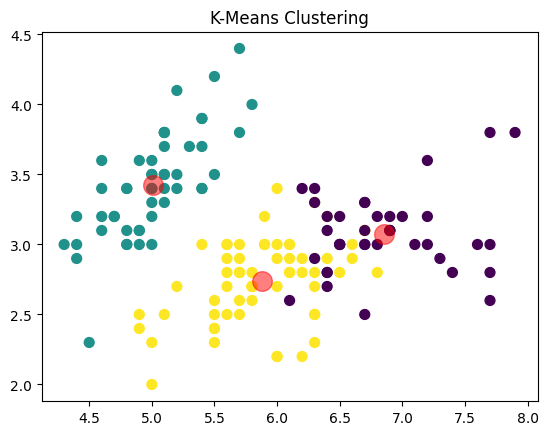
plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='red', s=200, alpha=0.5)

plt.title('K-Means Clustering')

plt.show()

Explanation:

This code applies K-Means clustering to the Iris dataset to group the data into three clusters. The KMeans model from sklearn.cluster is used to perform the clustering, specifying 3 clusters, which corresponds to the three species in the Iris dataset. After fitting the model, it predicts the cluster assignments (y\_kmeans). A scatter plot is then created to visualize the clusters, with different colors representing the predicted cluster labels. The cluster centers are plotted in red, and the size of the points is increased for better visibility. The plot helps to visualize how K-Means has grouped the data based on sepal length and width.

Output:  


1. Write a program to illustrate the mini-batch k-means clustering algorithm.

Mini-batch k-means clustering is a variant of the k-means algorithm designed to improve efficiency by using small, random subsets of data (mini-batches) rather than the entire dataset for each iteration. This approach reduces computation time while still providing effective clustering results. This program illustrates the mini-batch k-means algorithm, offering a faster alternative for clustering large datasets.

Program:

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.cluster import MiniBatchKMeans

iris = load\_iris()

X = iris.data

# Mini-Batch K-Means clustering

minibatch\_kmeans = MiniBatchKMeans(n\_clusters=3, random\_state=42)

y\_minibatch\_kmeans = minibatch\_kmeans.fit\_predict(X)

# Plotting

plt.scatter(X[:, 0], X[:, 1], c=y\_minibatch\_kmeans, s=50, cmap='viridis')

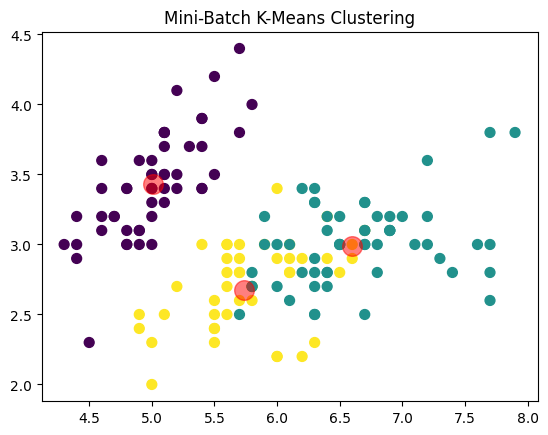
plt.scatter(minibatch\_kmeans.cluster\_centers\_[:, 0], minibatch\_kmeans.cluster\_centers\_[:, 1], c='red', s=200, alpha=0.5)

plt.title('Mini-Batch K-Means Clustering')

plt.show()

Explanation:  
In this code, Mini-Batch K-Means clustering is applied to the Iris dataset. The key difference from traditional K-Means is that Mini-Batch K-Means processes smaller random batches of data at a time, which can improve the efficiency when dealing with large datasets. Here, we are using 3 clusters (n\_clusters=3), which matches the number of species in the Iris dataset. The MiniBatchKMeans algorithm is fitted to the Iris data, and the resulting cluster assignments (y\_minibatch\_kmeans) are used for coloring the data points in the scatter plot. The red points represent the cluster centers. This clustering technique is more efficient for larger datasets but behaves similarly to regular K-Means for small datasets like Iris.

Output:



1. Write a program to illustrate the k-medoids clustering algorithm.

K-medoids clustering is an unsupervised learning algorithm similar to k-means but uses actual data points (medoids) as cluster centers rather than the mean of the points. It is more robust to noise and outliers than k-means, making it a suitable choice for datasets with such issues. This program demonstrates the k-medoids clustering algorithm, which groups data points into clusters based on the similarity of their features to the medoids.

Program:

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn\_extra.cluster import KMedoids

# K-Medoids clustering

kmedoids = KMedoids(n\_clusters=3, random\_state=42)

y\_kmedoids = kmedoids.fit\_predict(X)

# Plotting

plt.scatter(X[:, 0], X[:, 1], c=y\_kmedoids, s=50, cmap='viridis')

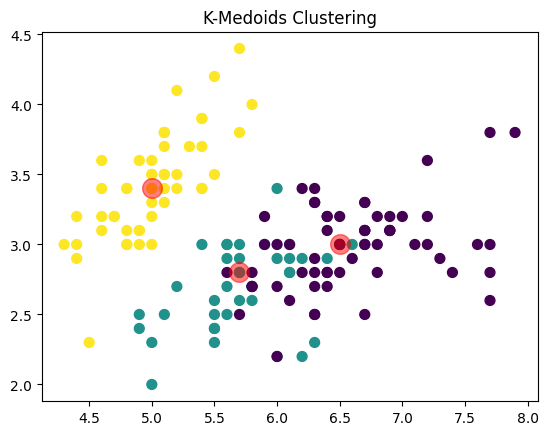
plt.scatter(kmedoids.cluster\_centers\_[:, 0], kmedoids.cluster\_centers\_[:, 1], c='red', s=200, alpha=0.5)

plt.title('K-Medoids Clustering')

plt.show()

Explanation:  
In this code, K-Medoids clustering is applied to the Iris dataset. K-Medoids is similar to K-Means, but instead of using the mean of the points in a cluster to represent the cluster center, K-Medoids selects actual data points (medoids) as the center of each cluster. This makes K-Medoids more robust to noise and outliers compared to K-Means, which uses the mean as the center. The red points in the scatter plot represent the medoids (cluster centers), and the data points are color-coded based on their cluster assignment. The KMedoids class from sklearn\_extra.cluster is used here to perform the clustering with n\_clusters=3, which matches the number of species in the Iris dataset.

Output:



1. Write a program to illustrate the agglomerative clustering algorithm.

Agglomerative clustering is a type of hierarchical clustering that begins by treating each data point as its own cluster and iteratively merges the closest clusters. This approach creates a tree-like structure called a dendrogram, which can be used to determine the number of clusters. This program demonstrates the agglomerative clustering algorithm, providing a visual representation of the clustering process.

Program:

from scipy.cluster.hierarchy import dendrogram, linkage

from matplotlib import pyplot as plot

x = [[i] for i in [12,10,6,8,4,2,1,16]]

Z = linkage (x, 'ward')

figure =plot.figure(figsize=(25,10))

den = dendrogram(Z)

Z=linkage(x,'single')

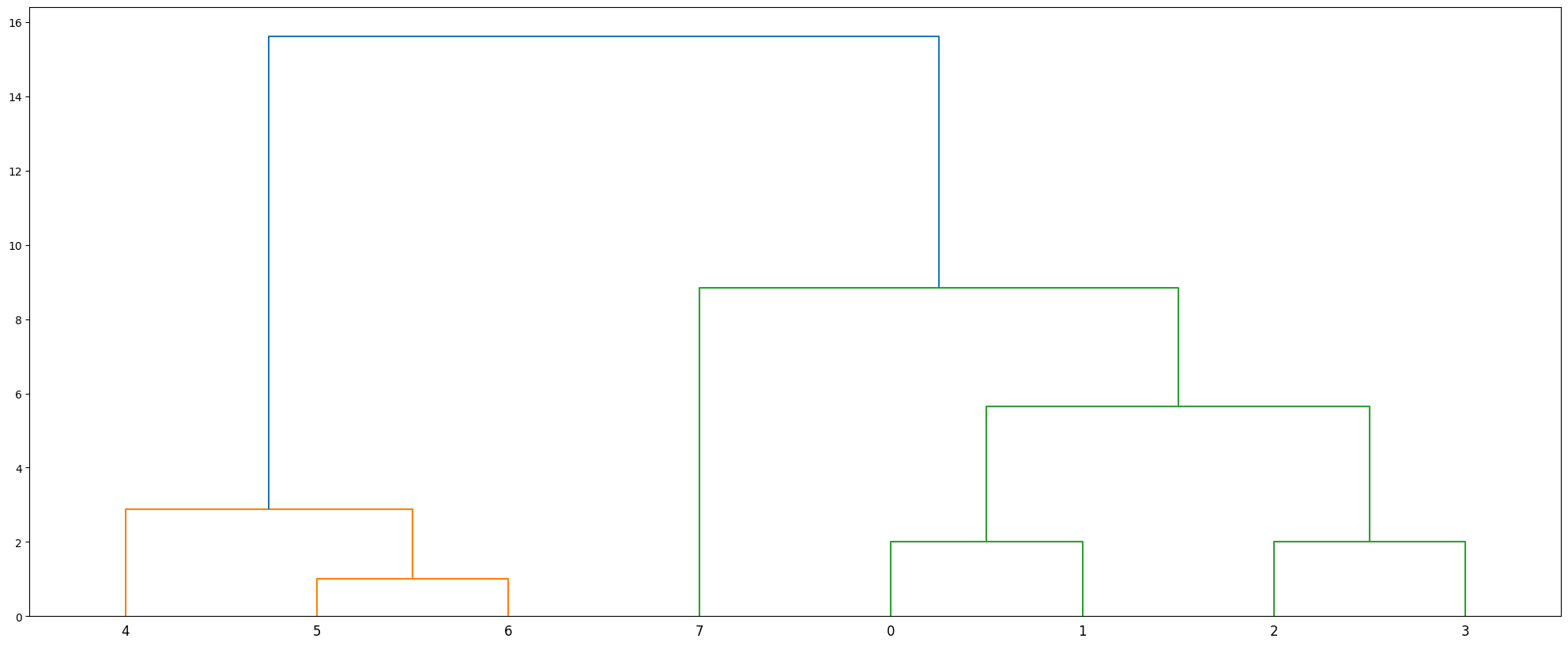
figure = plot.figure(figsize=(25,10))

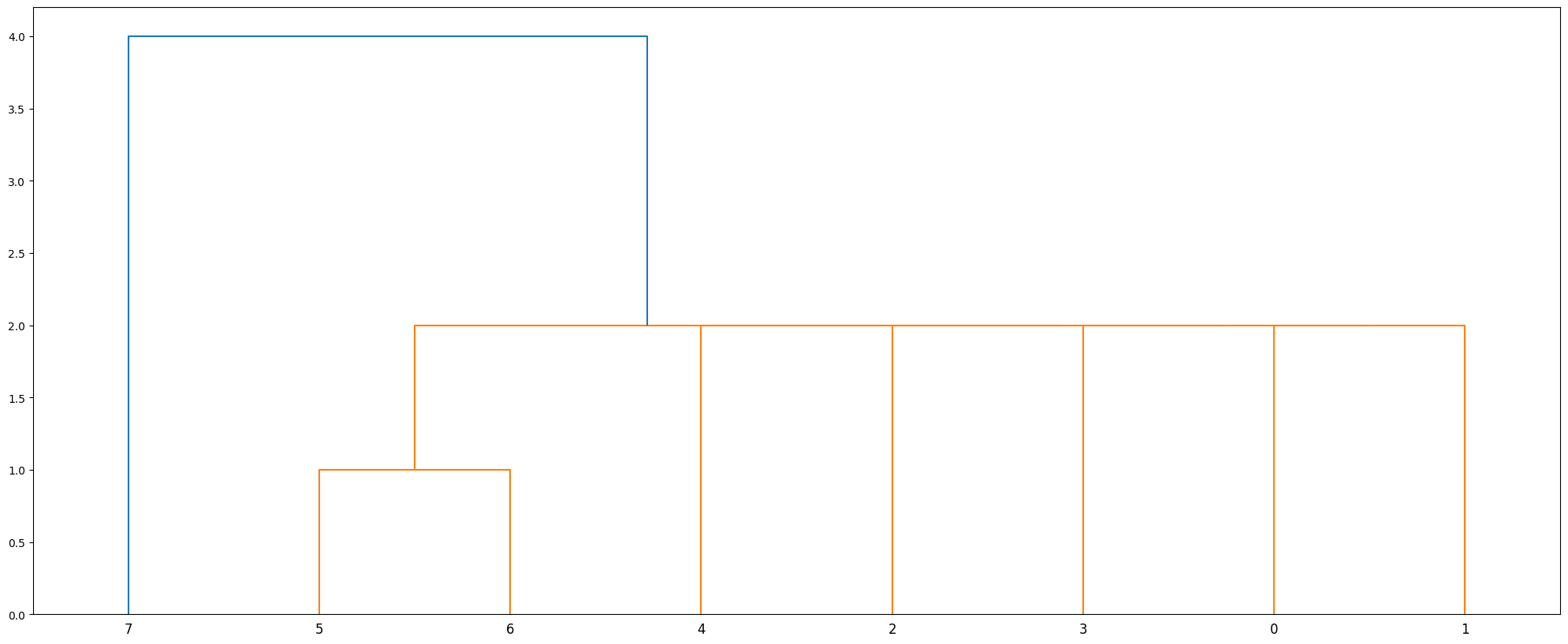
den = dendrogram(Z)

plot.show()

Explanation:  
In this code, Hierarchical Clustering is applied to a small dataset containing the values [12, 10, 6, 8, 4, 2, 1, 16]. The hierarchical clustering process builds a tree of clusters, also known as a dendrogram, which helps visualize how data points are grouped at each level of the hierarchy. Two different linkage methods are used: Ward linkage and Single linkage. The Ward linkage minimizes the variance within clusters, while Single linkage measures the shortest distance between clusters. The dendrograms are plotted for both methods to compare how the clusters are formed differently depending on the linkage criterion. The resulting figures, with large sizes of (25,10), allow for clear visualization of the hierarchical structure of the data.

Output:





1. Write a program to illustrate the divisive clustering algorithm.

Divisive clustering is a top-down approach to hierarchical clustering, where the dataset starts as a single cluster and is recursively split into smaller clusters. The process continues until each data point is in its own cluster or a predefined stopping criterion is met. This program demonstrates the divisive clustering algorithm, which allows for the exploration of how data can be progressively divided into distinct groups.

Program:

from scipy.cluster.hierarchy import dendrogram, linkage

from matplotlib import pyplot as plot

import numpy as np

x = [[i] for i in [12, 10, 6, 8, 4, 2, 1, 16]]

clusters = [x]

while len(clusters) < 3:

largest\_cluster = max(clusters, key=len)

cluster1 = largest\_cluster[:len(largest\_cluster)//2]

cluster2 = largest\_cluster[len(largest\_cluster)//2:]

clusters.remove(largest\_cluster)

clusters.extend([cluster1, cluster2])

Z = linkage(x, 'ward')

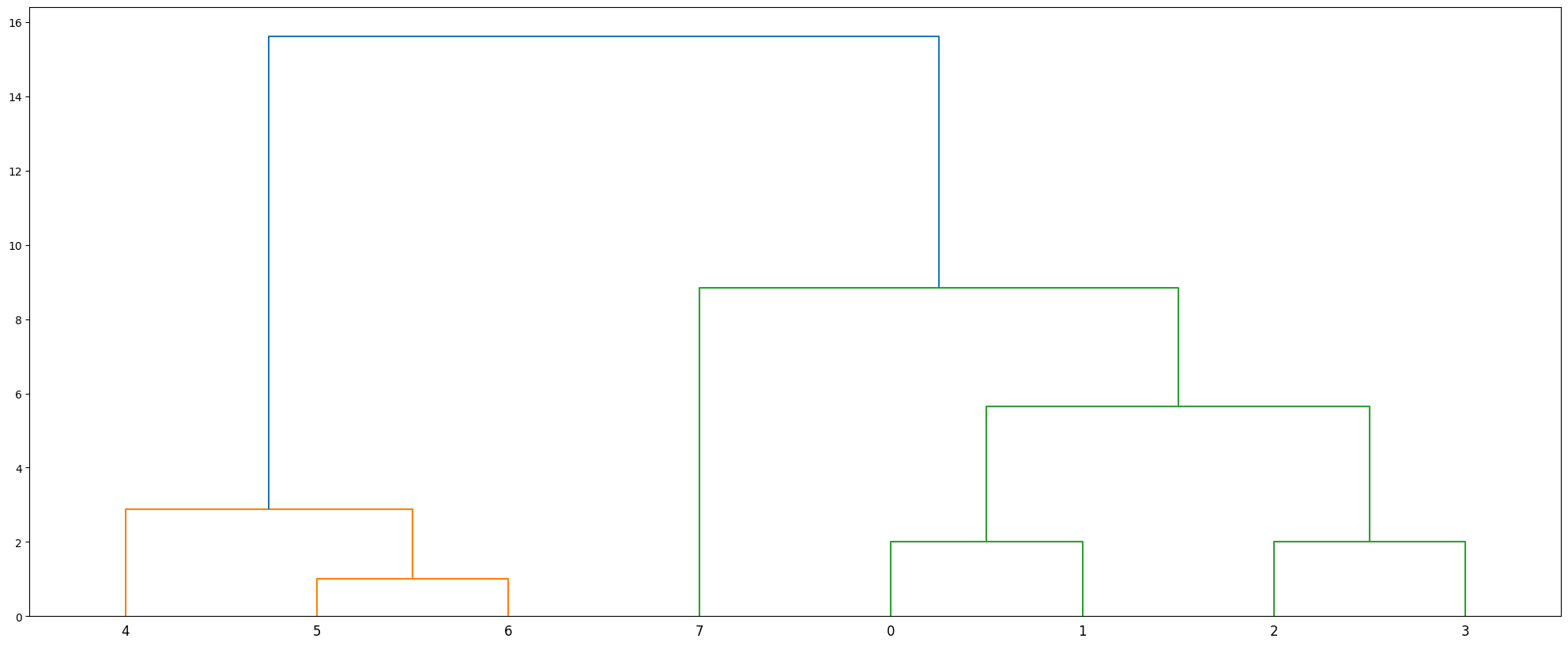
figure = plot.figure(figsize=(25,10))

den = dendrogram(Z)

plot.show()

Explanation:  
In this code, hierarchical clustering is performed on a list of data points x. Initially, the data is manually split into clusters, and the splitting continues until there are three clusters. After this, the linkage() function is used to compute the linkage matrix for hierarchical clustering based on the original data x. The dendrogram() function is then used to generate and display a dendrogram that visualizes the clustering process. This process creates a tree-like diagram showing how the data points are grouped hierarchically based on their distances.

Output:



1. Write a program to illustrate the DB-scan clustering algorithm.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups data points based on their density. It identifies regions of high density and separates them from regions of low density, while also marking outliers as noise. This program illustrates the DBSCAN clustering algorithm, which is especially effective for identifying clusters of arbitrary shapes and handling noise in the data.

Program:

from sklearn.cluster import DBSCAN

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data

# DBSCAN clustering

dbscan = DBSCAN(eps=0.5, min\_samples=5)

y\_dbscan = dbscan.fit\_predict(X)

# Plotting

plt.scatter(X[:, 0], X[:, 1], c=y\_dbscan, s=50, cmap='viridis')

plt.title('DBSCAN Clustering')

plt.show()

Explanation:  
In this code, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is used for clustering the Iris dataset. The algorithm is initialized with a eps value of 0.5 (the maximum distance between two samples for them to be considered as in the same neighborhood) and a min\_samples value of 5 (the number of samples in a neighborhood for a point to be considered a core point). The fit\_predict() function applies DBSCAN to the data and returns the cluster labels for each data point. The resulting clusters are visualized with a scatter plot, where the points are colored based on their assigned cluster. This method is especially useful for identifying noise points, as DBSCAN assigns outliers to a cluster label of -1.

Output:

