**Assignment 2. Implemetation of DP Matchingstudent information**

Student ID: m5281054

Name: WANG Binghao

**Required Libraries:**

import numpy as np

import matplotlib.pyplot as plt

**Run the Program:**

Python: Python3.12

I have put the database file into database/data\_a.txt and database/data\_b.txt

This file also includes all data sets. Make sure you open this folder and install the required libraries. Click Run.

**About Code Explanation:**

**Matching Distance Calculation :**

This part calculates the matching distance between two datasets A and B. It utilizes dynamic programming, initializing a 2D array dp and then filling this array to compute the matching distance.

**Backtrack to Find Corresponding Points :**

After calculating the matching distance, this part implements a backtracking algorithm to find the corresponding points between dataset A and dataset B. It iterates through the dp array values, starting from the bottom-right corner and moving towards the top-left corner.

**Read Data Sets and Initialize Plot :**

This section of the code reads the contents of data sets A and B and stores them as lists data\_a and data\_b, respectively.

**Plot Original Image :**

Using the Matplotlib library, it plots two subplots. The first subplot displays the original datasets A and B.

**Adjust B's Horizontal Coordinates:**

This part of the code adjusts the horizontal coordinates of dataset B to align with corresponding points from dataset A. By iterating through the found corresponding points, it replaces the coordinates of dataset B with the corresponding values from dataset A.

**Plot Modified Image:**

It plots the modified datasets A and B along with the marked corresponding points.

**Code :**

import numpy as np

import matplotlib.pyplot as plt

def matching\_distance(A, B):

    m = len(A)

    n = len(B)

    # 初始化距离矩阵

    dp = np.zeros((m + 1, n + 1))

    # 初始化第一行和第一列

    for i in range(1, m + 1):

        dp[i][0] = dp[i - 1][0] + abs(A[i - 1] - B[0])

    for j in range(1, n + 1):

        dp[0][j] = dp[0][j - 1] + abs(A[0] - B[j - 1])

    # 计算其余位置的距离

    for i in range(1, m + 1):

        for j in range(1, n + 1):

            dp[i][j] = abs(A[i - 1] - B[j - 1]) + min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1])

    return dp

def backtrack\_matching(dp, A, B):

    m, n = dp.shape

    i = m - 1

    j = n - 1

    matching\_points = []

    while i > 0 and j > 0:

        matching\_points.append((i - 1, j - 1))  # 将A和B之间的对应点加入列表中

        if dp[i - 1][j - 1] <= dp[i - 1][j] and dp[i - 1][j - 1] <= dp[i][j - 1]:

            i -= 1

            j -= 1

        elif dp[i - 1][j] <= dp[i - 1][j - 1] and dp[i - 1][j] <= dp[i][j - 1]:

            i -= 1

        else:

            j -= 1

    return matching\_points[::-1]  # 反转列表，使其按照A的顺序排列

# 读取数据集 A

data\_a\_path = r"database/data\_a.txt"

with open(data\_a\_path, 'r') as file:

    data\_a = [float(line.strip()) for line in file.readlines()]

# 读取数据集 B

data\_b\_path = r"database/data\_b.txt"

with open(data\_b\_path, 'r') as file:

    data\_b = [float(line.strip()) for line in file.readlines()]

# 计算匹配距离

dp = matching\_distance(data\_a, data\_b)

# 回溯找到对应点

matching\_points = backtrack\_matching(dp, data\_a, data\_b)

# 绘制图像

fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# 绘制原始图像

axs[0].plot(data\_a, color='red', label='A')

axs[0].plot(data\_b, color='blue', label='B')

# 标记对应点

# for i, j in matching\_points:

#     axs[0].plot([i, j], [data\_a[i], data\_b[j]], color='green')

axs[0].legend()

axs[0].set\_title('Original')

# 更改 B 的水平坐标

new\_data\_b = [None] \* len(data\_a)

for i, j in matching\_points:

    new\_data\_b[i] = data\_b[j]

# 将未匹配到的点用前一个匹配点填充

for i in range(len(new\_data\_b)):

    if new\_data\_b[i] is None:

        new\_data\_b[i] = new\_data\_b[i - 1]

# 绘制更改后的图像

axs[1].plot(data\_a, color='red', label='A')

axs[1].plot(new\_data\_b, color='blue', label='B')

# 标记对应点

axs[1].legend()

axs[1].set\_title('Modified')

plt.show()