**Assignment 3. Matching with Signature Data**

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 executable file into ITA09\_Assignment\_3\Otherfiles\code

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

**About Code Explanation:**

The code primarily achieves the following key functionalities:

**1: Parsing SDT Files:** The parse\_sdt\_file function parses SDT files, extracting stroke data and storing it as a list of lists. The outer list represents characters, while the inner list represents strokes.

**2: Calculating Stroke Distance:** The calculate\_stroke\_distance function computes the distance between two strokes using the Euclidean distance.

**3: Calculating Character Distance:** The calculate\_character\_distance function calculates the distance between two characters by summing the distances between corresponding strokes.

**4: Dynamic Programming Matching:** The dp\_matching function implements dynamic programming to find the minimum distance between two sequences of strokes.

**5: Calculating DP Character Distance:** The calculate\_dp\_character\_distance function calculates the distance between two characters using dynamic programming matching.

**6: Plotting Matching Results:** The plot\_matching function plots three subplots. The first subplot displays the strokes of two characters, the second subplot shows the linear matching results, and the third subplot displays the dynamic programming matching results.

**7: Main Function:** The main function serves as the entry point of the program. It reads SDT files, calculates character distances, prints the results, and plots the matching results.

**some problems:**

In the two dataset files given, the implementation effect of one dataset is very messy. I implemented another dataset and the effect diagram is relatively ideal.

**Code**

import numpy as np  
import matplotlib.pyplot as plt  
  
  
def parse\_sdt\_file(file\_path):  
 data = []  
 current\_stroke = []  
 with open(file\_path, 'r') as file:  
 next(file)  
 for line in file:  
 parts = line.strip().split()  
 if parts[0] == '-1':  
 if current\_stroke:  
 data.append(current\_stroke)  
 current\_stroke = []  
 else:  
 x, y, pressure, direction, altitude, time = map(int, parts)  
 current\_stroke.append((x, y, pressure, direction, altitude, time))  
 if current\_stroke:  
 data.append(current\_stroke)  
 return data  
  
  
def calculate\_stroke\_distance(stroke1, stroke2):  
 total\_distance = 0  
 n = min(len(stroke1), len(stroke2))  
 for i in range(n):  
 x1, y1 = stroke1[i][:2]  
 x2, y2 = stroke2[i][:2]  
 distance = np.sqrt((x1 - x2) \*\* 2 + (y1 - y2) \*\* 2)  
 total\_distance += distance  
 average\_distance = total\_distance / n  
 return average\_distance  
  
  
def calculate\_character\_distance(signature1, signature2):  
 total\_distance = 0  
 N = min(len(signature1), len(signature2))  
 for i in range(N):  
 stroke\_distance = calculate\_stroke\_distance(signature1[i], signature2[i])  
 total\_distance += stroke\_distance  
 return total\_distance  
  
  
def dp\_matching(stroke1, stroke2):  
 n = len(stroke1)  
 m = len(stroke2)  
 dp = np.full((n + 1, m + 1), float('inf'))  
 dp[0, 0] = 0  
 for i in range(1, n + 1):  
 for j in range(1, m + 1):  
 cost = np.sqrt((stroke1[i - 1][0] - stroke2[j - 1][0]) \*\* 2 + (stroke1[i - 1][1] - stroke2[j - 1][1]) \*\* 2)  
 dp[i, j] = min(dp[i - 1, j - 1] + cost, dp[i - 1, j] + cost, dp[i, j - 1] + cost)  
 return dp[n, m]  
  
  
def calculate\_dp\_character\_distance(signature1, signature2):  
 total\_distance = 0  
 N = min(len(signature1), len(signature2))  
 for i in range(N):  
 stroke\_distance = dp\_matching(signature1[i], signature2[i])  
 total\_distance += stroke\_distance  
 return total\_distance  
  
  
def plot\_matching(signature1, signature2):  
 fig, axs = plt.subplots(1, 3, figsize=(18, 6))  
 scale = 1.0  
  
 # Plot signatures  
 for stroke1, stroke2 in zip(signature1, signature2):  
 x1, y1 = zip(\*[(x \* scale, y \* scale) for x, y, \_, \_, \_, \_ in stroke1])  
 x2, y2 = zip(\*[(x \* scale, y \* scale) for x, y, \_, \_, \_, \_ in stroke2])  
 axs[0].plot(x1, y1, 'r-')  
 axs[0].plot(x2, y2, 'b-')  
  
 axs[0].set\_title('Signatures')  
 axs[0].legend(['Signature 1', 'Signature 2'])  
  
 # Plot linear matching  
 for stroke1, stroke2 in zip(signature1, signature2):  
 x1, y1 = zip(\*[(x \* scale, y \* scale) for x, y, \_, \_, \_, \_ in stroke1])  
 x2, y2 = zip(\*[(x \* scale, y \* scale) for x, y, \_, \_, \_, \_ in stroke2])  
 for (x1i, y1i), (x2i, y2i) in zip(zip(x1, y1), zip(x2, y2)):  
 axs[1].plot([x1i, x2i], [y1i, y2i], 'grey', linestyle='--', linewidth=0.3)  
  
 axs[1].set\_title('Linear Matching')  
 axs[1].legend(['Matching'])  
  
 # Plot DP matching  
 for stroke1, stroke2 in zip(signature1, signature2):  
 x1, y1 = zip(\*[(x \* scale, y \* scale) for x, y, \_, \_, \_, \_ in stroke1])  
 x2, y2 = zip(\*[(x \* scale, y \* scale) for x, y, \_, \_, \_, \_ in stroke2])  
 for (x1i, y1i), (x2i, y2i) in zip(zip(x1, y1), zip(x2, y2)):  
 axs[2].plot([x1i, x2i], [y1i + 20, y2i + 20], 'green', linestyle='--', linewidth=0.3)  
  
 axs[2].set\_title('DP Matching')  
 axs[2].legend(['Matching'])  
  
 plt.show()  
  
  
def main():  
 signature\_data1 = parse\_sdt\_file('Otherfiles/code/database/ref.sdt')  
 signature\_data2 = parse\_sdt\_file('Otherfiles/code/database/ref.sdt')  
  
 character\_distance\_linear = calculate\_character\_distance(signature\_data1, signature\_data2)  
 print("Linear Matching Calculated Distance between characters:", character\_distance\_linear)  
  
 character\_distance\_dp = calculate\_dp\_character\_distance(signature\_data1, signature\_data2)  
 print("DP Matching Calculated Distance between characters:", character\_distance\_dp)  
  
 plot\_matching(signature\_data1, signature\_data2)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()