KHAI THÁC DỮ LIỆU

Nguyễn Xuân Việt Đức - 22280012

Bài tập thực hành - Lần 30

Bài 3: SỰ TƯƠNG ĐỒNG VÀ CÁC KHOẢNG CÁCH (TT)

III. Nội dung thực hành:

1. Khoảng cách thay đổi (Edit distance):

Cài đặt và thực thi mục 1 trên máy tính.

Edit distance là số lượng thao tác tối thiểu (thêm, xóa, thay thế) cần thiết để biến đổi một chuỗi ký tự $X = x_1x_2 \dots x_m$ thành chuỗi $Y = y_1y_2 \dots y_n$.

- Chèn (Insertion): Chi phí = 1
- Xóa (Deletion): Chi phí = 1
- Thay thế (Substitution): Chi phí = 2
- Không thay đổi (Match): Chi phí = 0

Định nghĩa hàm chi phí Gọi I_{ij} là hàm chỉ báo nhị phân:

$$I_{ij} = \begin{cases} 0 & \text{nếu } x_i = y_j \\ 1 & \text{nếu } x_i \neq y_j \end{cases}$$

Công thức quy hoạch động Gọi $\text{Edit}(i, j)$ là chi phí tối thiểu để biến đổi X_1^i thành Y_1^j, ta có công thức đệ quy:

$$\text{Edit}(i,j) = \min \begin{cases} \text{Edit}(i-1,j) + \text{Deletion cost} \\ \text{Edit}(i,j-1) + \text{Insertion cost} \\ \text{Edit}(i-1,j-1) + I_{ij} \times \text{Replacement cost} \end{cases}$$

Khởi tạo

$$\text{Edit}(i,0) = i \quad (\text{xóa toàn bộ ký tự trong } X_1^i)$$
$$\text{Edit}(0,j) = j \quad (\text{chèn toàn bộ ký tự vào để thành } Y_1^j)$$

Kết quả Giá trị Edit(m, n) là khoảng cách thay đổi giữa chuỗi X và Y.

Truy vết (Backtracking) Khi đi ngược lại từ ô Edit(m, n), ta xác định được chuỗi thao tác (insert, delete, replace) để biến đổi chuỗi ban đầu thành chuỗi đích.

Output:

```
Enter the source string :
 INTENTION
Enter the target string :
 EXECUTION
Minimum edit distance : 8
Number of insertions : 1
Number of deletions : 1
Number of substitutions : 3
Total number of operations : 5
Actual Operations :
[1] DELETE : I
[2] SUBSTITUTE : N by E
[3] SUBSTITUTE : T by X
[4] INSERT : C
[5] SUBSTITUTE : N by U
```

Hình 1: Edit distance

2. Khoảng cách dãy con chung dài nhất (Longest Common Subsequence-LCSS):

Viết chương trình tính khoảng cách dãy con chung dài nhất

Định nghĩa Cho hai chuỗi:

$$X = x_1 x_2 \dots x_m \quad \text{và} \quad Y = y_1 y_2 \dots y_n$$

Gọi $LCS(i, j)$ là độ dài của dãy con chung dài nhất giữa X_1^i và Y_1^j.
Công thức quy hoạch động

$$LCS(i,j) = \begin{cases} 0 & \text{nếu } i = 0 \text{ hoặc } j = 0 \\ LCS(i-1, j-1) + 1 & \text{nếu } x_i = y_j \\ \max(LCS(i-1, j), LCS(i, j-1)) & \text{nếu } x_i \neq y_j \end{cases}$$

Khởi tạo

$$LCS(i, 0) = 0 \quad \text{với mọi } iLCS(0, j) = 0 \quad \text{với mọi } j$$

Kết quả: Giá trị tại ô $LCS(m, n)$ trong bảng động là độ dài của dãy con chung dài nhất.

Dựa trên kỹ thuật quy hoạch động (dynamic programming), ta xây dựng một bảng 2 chiều $L[i][j]$, trong đó:

$$L[i][j] \text{ là độ dài LCS giữa chuỗi } X[1..i] \text{ và } Y[1..j]$$

Truy vết (Backtracking) Bắt đầu từ ô $LCS(m, n)$, ta lần lượt thực hiện:

- Nếu $x_i = y_j$, thêm ký tự đó vào kết quả và di chuyển về ô $(i - 1, j - 1)$
- Nếu không, di chuyển theo hướng có giá trị lớn hơn giữa $(i - 1, j)$ và $(i, j - 1)$

Kết quả cuối cùng là chuỗi con chung dài nhất.

Ví dụ:

- X = "`ABCBDAB`"
- Y = "`BDCAB`"
- LCS(X, Y) có thể là: "`BDAB`" hoặc "`BCAB`" với độ dài là 4.

Output:

```
Longest Common Subsequence (LCS) Program
----------------------------------------
Enter the first string:  ACADB
Enter the second string:  CBDA

Results:
Length of Longest Common Subsequence: 2
Longest Common Subsequence: 'CB'

Illustration of finding LCS:
String X: ACADB
String Y: CBDA

Dynamic Programming Table (values of L[i][j]):
       C  B  D  A
   0  0  0  0  0
A  0  0  0  0  1
C  0  1  1  1  1
A  0  1  1  1  2
D  0  1  1  2  2
B  0  1  2  2  2
```

Hình 2: Longest Common Subsequence-LCSS

3. Khoảng cách biến đổi thời gian động (Dynamic Time Warping-DTW):

Viết chương trình tính khoảng cách biến đổi thời gian động

Dynamic Time Warping (DTW) là một thuật toán cho phép đo mức độ tương đồng giữa hai chuỗi thời gian, ngay cả khi chúng có độ dài khác nhau hoặc bị lệch theo thời gian. Kỹ thuật này kéo dãn chuỗi thời gian một cách linh hoạt để tìm sự khớp tốt nhất.

Giả sử ta có hai chuỗi thời gian:

$$X = (x_1, x_2, \ldots, x_m), \quad Y = (y_1, y_2, \ldots, y_n)$$

Khoảng cách giữa hai phần tử x_i và y_j được tính bằng:

$$\text{dist}(x_i, y_j) = |x_i - y_j|$$

Gọi $DTW(i, j)$ là khoảng cách DTW tối ưu giữa $x_1, \ldots, x_i$ và $y_1, \ldots, y_j$. Khi đó công thức quy hoạch động là:

$$DTW(i,j) = \begin{cases} 0, & \text{nếu } i = j = 0 \\ \infty, & \text{nếu } i = 0 \text{ hoặc } j = 0 \\ |x_i - y_j| + \min \begin{cases} DTW(i-1, j) & \text{(lặp lại } x_i\text{)} \\ DTW(i, j-1) & \text{(lặp lại } y_j\text{)} \\ DTW(i-1, j-1) & \text{(không lặp)} \end{cases}, & \text{ngược lại} \end{cases}$$

Ý nghĩa các bước:

- Lặp lại x_i: khớp x_i với nhiều phần tử trong Y
- Lặp lại y_j: khớp y_j với nhiều phần tử trong X
- Không lặp: khớp từng cặp phần tử một-một

Truy vết đường đi tối ưu Sau khi điền đầy đủ ma trận chi phí DTW, ta truy vết ngược từ ô (m, n) về $(0, 0)$ bằng cách tại mỗi bước chọn ô có giá trị nhỏ nhất trong ba ô lân cận:

$$(i-1, j), \quad (i, j-1), \quad (i-1, j-1)$$

Đường đi này thể hiện cách "co dãn" chuỗi thời gian để đạt được sự tương đồng tốt nhất.

Output:

```
DYNAMIC TIME WARPING (DTW) DISTANCE CALCULATOR
==============================================
Enter source sequence (A):  1748296520
Enter target sequence (B):  1285519465

DTW Distance: 17.0

DTW Cost Matrix:
 0  36.0 29.0 32.0 22.0 22.0 16.0 24.0 16.0 18.0 17.0
 2  35.0 27.0 24.0 17.0 17.0 15.0 21.0 12.0 14.0 12.0
 5  34.0 27.0 18.0 14.0 14.0 18.0 14.0 10.0 10.0 9.0
 6  30.0 24.0 15.0 14.0 14.0 18.0 10.0 9.0 9.0 10.0
 9  25.0 20.0 13.0 13.0 13.0 15.0 7.0 12.0 13.0 17.0
 2  17.0 13.0 12.0 9.0 9.0 7.0 14.0 10.0 14.0 17.0
 8  16.0 13.0 6.0 6.0 6.0 11.0 8.0 12.0 14.0 17.0
 4  9.0 7.0 6.0 3.0 4.0 7.0 12.0 12.0 14.0 15.0
 7  6.0 5.0 2.0 4.0 6.0 12.0 14.0 17.0 18.0 20.0
 1  0.0 1.0 8.0 12.0 16.0 16.0 24.0 27.0 32.0 36.0
    1   2   8   5   5   1   9   4   6   5

Warping Path:
[0, 0, 1, 2, 3, 6, 7, 7, 9, 9, 9, 12, 17]
```

Hình 3: Dynamic Time Warping- DTW)