XÂU CON CHUNG DÀI NHẤT (Longest Common Subsequence – LCS)

Xâu ký tự S gọi là xâu con của xâu ký tự T nếu có thể xoá bớt một số ký tự trong xâu T để được xâu S. Cho hai xâu ký tự $X = x_1x_2 \dots x_m$ và $Y = y_1y_2 \dots y_n$. Tìm xâu Z có độ dài lớn nhất là xâu con của cả X và Y.

Dữ liệu: Vào từ file văn bản LCS.INP

- Dòng 1 chứa xâu X chỉ gồm các chữ cái hoa có độ dài không quá 10^3 ký tự
- Dòng 2 chứa xâu Y chỉ gồm các chữ cái hoa có độ dài không quá 10^6 ký tự

Kết quả: Ghi ra file văn bản LCS.OUT xâu Z tìm được

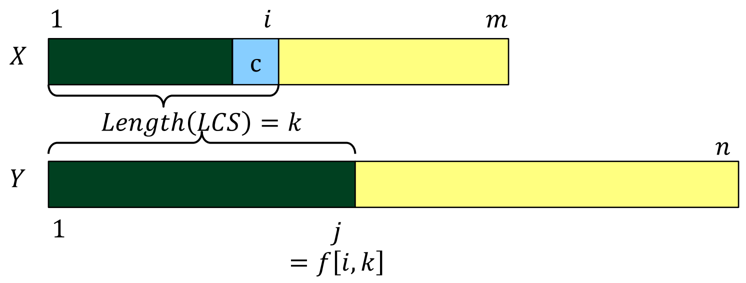
Ví dụ:

LCS.INP	LCS.OUT
ABCDEFGHIXYZ ABCXDEFYGHIZ	ABCDEFGHIZ

Để ý rằng trong bài toán này n khá lớn. Thuật toán truyền thống với độ phức tạp $\Theta(mn)$ không dùng được. Ta tìm thuật toán tận dụng đặc điểm m khá nhỏ ($m \leq 1000$)

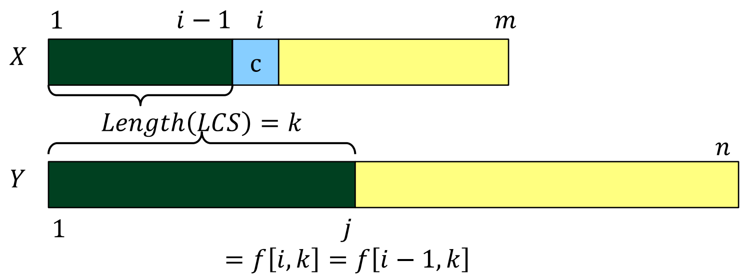
Trước hết ta quan tâm tới bài toán tìm LCS:

Đặt hàm mục tiêu: Gọi $f[i][k]$ là chỉ số j **nhỏ nhất** thỏa mãn: LCS của $x[1 \dots i]$ và $y[1 \dots j]$ có độ dài đúng bằng k. Mảng f có kích thước $[1 \dots maxM, 0 \dots maxM]$.

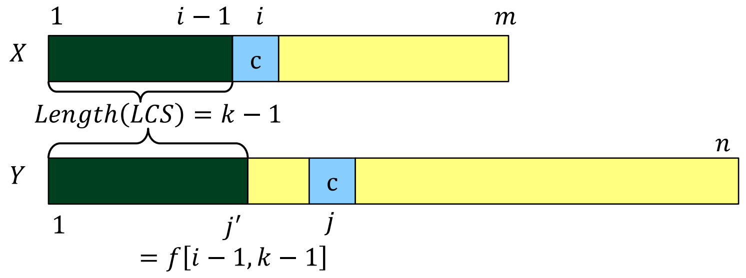


Để tính $j = f[i][k]$, ta phân tích xem LCS trong định nghĩa của $f[i][k]$ được xây dựng như thế nào? LCS này được xây dựng thuộc một trong 2 kiểu tùy thuộc vào quyết định: Có lấy ký tự $c = x[i]$ vào đuôi LCS hay không

Nếu không lấy $x[i]$ vào đuôi LCS, ta có $f[i][k] = f[i-1][k]$ bởi trong TH này ta có thể bỏ x_i ra trong việc tìm LCS:



Nếu có lấy $c = x[i]$ vào đuôi LCS. Hiển nhiên khi đó với $j = f[i][k]$ thì $y[j] = x[i] = c$ do ta cần chọn j nhỏ nhất. Hơn nữa nếu cắt bỏ ký tự c ở đuôi LCS, ta phải thu được một LCS độ dài $k-1$ của xâu $x[1 \dots i-1]$ và $y[1 \dots j-1]$, tức là $f[i-1][k-1]$ phải bằng j' nào đó đứng trước j. Vậy $f[i][k]$ được tính bằng cách tìm vị trí ký tự c đầu tiên trong dãy Y đứng sau vị trí $j' = f[i-1][k-1]$



Dĩ nhiên trong hai quyết định lấy hay không lấy $c = x[i]$ vào đuôi LCS ta chọn quyết định tốt hơn (cho $f[i][k]$ nhỏ hơn). Cơ sở của CT truy hồi là $f[i][0] = 0, \forall i$

Tiếp theo, ta xét bài toán hỗ trợ nhằm giảm độ phức tạp tính toán của lời giải: Đó là cho một vị trí j và ký tự c, tìm $g[j][c]$ là vị trí xuất hiện đầu tiên của ký tự c sau vị trí j trong dãy Y. Ta tiếp tục một bài toán giải công thức truy hồi đơn giản:

- Nếu $y[j+1] = c$, ta có $g[j][c] = j + 1$
- Nếu $y[j+1] \neq c$, ta có $g[j][c] = g[j+1][c]$

Mảng $g[0 \dots n]['A' \dots 'Z']$ được tính với $j = n-1, n-2, \dots 0$ và $c \in ['A' \dots 'Z']$. Cơ sở $g[n][\forall c] \coloneqq n + 1 (+\infty)$

Trong C++ ta có thể ánh xạ ký tự ‘A’ là 0, ‘B’ là 1, …, ‘Z’ là 25 cho phù hợp cách đánh chỉ số mảng.

Tóm lại:

Quy hoạch động tính mảng $g[0 \dots n]['A' \dots 'Z']$ ($O(26n)$)

Tiếp theo quy hoạch động $O(m^2)$ tính mảng $f[1 \dots m][0 \dots m]$ với cơ sở $f[i, 0] = 0$ và công thức truy hồi

$$f[i][k] = \min \begin{cases} f[i-1][k] \\ g\big[f[i-1][k-1], x[i]\big] \end{cases}$$

Truy vết: Tìm k lớn nhất sao cho $f[m][k] < +\infty (= n + 1)$

Nếu $f[m][k] = f[m-1][k]$, truy vết tiếp $f[m-1][k]$ bằng vòng lặp (đặt $m \coloneqq m - 1$)

Nếu không, thông báo chọn $x[m]$ vào LCS, truy vết tiếp $f[m-1][k-1]$ (đặt $m \coloneqq m-1; k \coloneqq k-1$)