修士学位論文要約（平成29年３月）

最長順序保存部分列の計算の効率化に関する研究

上木 庸平
指導教員：篠原 歩

Longest Common Subsequence in at Least k Length Order-Isomorphic Substrings

Yohei UEKI
Supervisor: Ayumi SHINOHARA

In this paper, we consider the longest common subsequence (LCS) problem with the restriction that the common subsequence is required to consist of substrings of length at least k. First, we show an $O(mn)$ time algorithm for the problem which gives a better worst-case running time than existing algorithms, where m and n are lengths of the input strings. Moreover, we also consider the LCS in at least k length order-isomorphic substrings problem. We show that the problem can also be solved in $O(mn)$ worst-case time by an easy-to-implement algorithm.

1. はじめに

生命情報学や時系列データ解析などにおいて，文字列の類似度を定量的に測る手段として**最長共通部分列** (LCS) がよく用いられている．LCS を類似度の尺度として用いる際の大きな欠点の一つとして，LCS は文字の連続的一致を類似度に反映できない点がある．

Benson ら [1] は，**長さ k の部分文字列から構成される最長共通部分列**（LCS_k）問題を提案した．LCS_k 問題では，共通部分列は長さ k の共通部分文字列の連接で構成される必要がある．例えば，文字列 $X = \mathtt{ATTTGG}$, $Y = \mathtt{ATTAAGG}$ に対して，$X[1:2] = Y[1:2] = \mathtt{AT}$, $X[5:6] = Y[6:7] = \mathtt{GG}$ が成り立ち，かつこのように選ぶときが最長となる場合の一つなので，X と Y の LCS_2 の 1 つは $\mathtt{ATGG}$ である．LCS_k 問題に対して，n, m を入力文字列の長さとすると，Benson らは $O(mn)$ 時間アルゴリズムを提案した．また，Deorowicz と Grabowski [2] は k の大きさによらない $O(mn)$ 時間アルゴリズムや平均的に速く動作するアルゴリズムなどを提案した．

さらに，Pavetić ら [6] は**長さ k 以上の部分文字列から構成される共通部分列**（LCS_{k^+}）問題を提案した．LCS_{k^+} 問題では共通部分列は長さ k 以上の共通部分文字列の連接で構成される必要がある．例えば，文字列 $X = \mathtt{ATTTGG}$, $Y = \mathtt{ATTAAGG}$ に対して，$X[1:3] = Y[1:3] = \mathtt{ATT}$, $X[5:6] = Y[6:7] = \mathtt{GG}$ が成り立つので，X, Y の LCS_{2^+} は $\mathtt{ATTGG}$ である．LCS_{k^+} 問題は生命情報学に応用されている [7]．m, n を入力文字列の長さ，r を入力文字列間において一致している長さ k の部分文字列の数とすると，Pavetić らは LCS_{k^+} 問題が $n \geq m$ を仮定して $O(n + r \log r + r \log n)$ 時間で解けることを示した．また，Benson ら [1] は最悪 $O(kmn)$ 時間のアルゴリズムを提案した．

本稿ではまず LCS_{k^+} 問題に対して，動的計画法を用いた最悪 $O(mn)$ 時間アルゴリズムを提案する．次に，文字列の一致関係を**順序同型** [5] に置き換えた**長さ k 以上の順序同型部分文字列から構成される最長共通部分列**（$op\text{-}LCS_{k^+}$）問題について考える．順序アルファベット上の同じ長さ l の文字列 S, T に対して，任意の $1 \leq i, j \leq l$ で $S[i] \leq S[j] \iff T[i] \leq T[j]$ が成り立つとき，S と T は**順序同型**であるといい，$S \approx T$ と表記する．

2. LCS_{k^+} 問題

文字列 X に対して，$X\langle i, +l\rangle = X[i : i + l - 1]$, $X\langle j, -l\rangle = X[j-l+1 : j]$ と表記する．$C[i, j]$ を $X[: i]$ と $Y[: j]$ の LCS_{k^+} の長さとする．$Match(i, j, l)$ を $X\langle i, -l\rangle = Y\langle j, -l\rangle$ のとき，$Match(i, j, l) = 1$，それ以外のとき，$Match(i, j, l) = 0$ と定義する．

$L[i, j] = \max\{l : X\langle i, -l\rangle = Y\langle j, -l\rangle\}$，$A_{i,j} = \{C[i-l, j-l]+l \cdot Match(i, j, l) : k \leq l \leq L[i, j]\}\}$ とすると，$C[i, j] = \max(\{C[i, j-1], C[i-1, j]\} \cup A_{i,j})$ が成り立つ．それぞれの i, j に対して，$\max A_{i,j}$ を計算する素朴なアルゴリズムは $n \geq m$ を仮定して $O(m)$ 時間かかるため，この漸化式を基にした素朴な動的計画法のアルゴリズムは $O(m^2 n)$ 時間かかる．本稿では，問題を $O(mn)$ 時間で解くためにそれぞ

れの i, j に対して $\max A_{i,j}$ を定数時間で計算する方法を考える．

$i,j>0$ かつ $X[i]=Y[j]$ のとき，$L[i,j]=L[i-1,j-1]+1$，それ以外のとき $L[i,j]=0$ が成り立つので，表 L は動的計画法により $O(mn)$ 時間で計算することができる．

$M[i,j]$ を，$L[i,j]\geq k$ のとき，$M[i,j]=\max A_{i,j}$，それ以外のとき $M[i,j]=-1$ とする．$L[i,j]>k$ のとき，$M[i,j]=\max\{M[i-1,j-1]+1, C[i-k,j-k]+k\}$，$L[i,j]=k$ のとき，$M[i,j]=C[i-k,j-k]+k]$，$L[i,j]<k$ のとき，$M[i,j]=-1$ が成り立つので，表 M は動的計画法により $O(mn)$ 時間で計算することができる．

提案アルゴリズムは大きさ $(m+1)\times(n+1)$ の表 C, L, M を計算するため $O(mn)$ 時間・領域必要である．なお，LCS_{k^+} の長さのみを計算したい場合，領域計算量は $O(km)$ に容易に削減できる．以上より次の定理を得る．

定理 1. LCS_{k^+} 問題は $O(mn)$ 時間と $O(km)$ 領域で解ける．

3. op-LCS_{k^+} 問題

$C[i,j]$ を $X[:i]$ と $Y[:j]$ の op-LCS_{k^+} の長さ，$L[i,j]=\max\{l : X\langle i,-l\rangle \approx Y\langle j,-l\rangle\}$ と再定義する．前章の表 C に関する漸化式は順序同型に関しても成り立つが，表 L の漸化式は成り立たない．

代わりに，表 L を計算するために **Z 表**を効率よく計算する **Z アルゴリズム** [4] を用いる．文字列 S の Z 表 Z_S を，それぞれの $1\leq i\leq|S|$ に対して $Z_S[i]=\max\{l : S\langle 1,+l\rangle \approx S\langle i,+l\rangle\}$ と定義する．定義より，$L[m-i+1,n-j+1]=\min\{Z_{(X^{\mathrm{R}}Y^{\mathrm{R}})[i:]}[m-i+j+1],\ m-i+1\}$ が成り立つ．文字列 S の Z 表 Z_S は $O(|S|\log|S|)$ 時間かかるため，Z アルゴリズムと上式を素朴に用いると $O(mn\log(m+n))$ 時間必要である．文字列 S に対する Z アルゴリズムでは，前処理として表 $Prev_S$, $Next_S$ を計算するが，それらの計算に $O(|S|\log|S|)$ 時間かかり，それ以外のすべての操作は $O(|S|)$ 時間しかかからない．したがって，$O(|S|\log|S|)$ 時間の前処理を 1 回行うことでそれぞれの $1\leq i\leq|S|$ に対して表 $Prev_{S[i:]}$, $Next_{S[i:]}$ を $O(|S|)$ 時間で求めることができれば，表 L は $O(mn)$ 時間で計算できる．表 $Prev_{S[i:]}$, $Next_{S[i:]}$ を計算するために，平衡二分探索木を用いる Hasan らのアルゴリズムの代わりに，Kubica ら [5] が提案した，ソートを基にしたアルゴリズムを修正したものを用いる．まず前処理として，それぞれ昇順と降順に S の位置を S の要素に基づいて安定ソートした表 S', S'' を $O(|S|\log|S|)$ 時間で計算する．それぞれの $1\leq i\leq|S|$ に対して表 $Prev_{S[i:]}$, $Next_{S[i:]}$ は，表 S', S'' の i より小さい要素を無視しながら Kubica らのアルゴリズムを適用することにより，$O(|S|)$ 時間で計算できる．

それぞれの i, j に対して $\max A_{i,j}$ を定数時間で計算する方法を考える．表 T を保持し，次の 2 つの操作に対応する**準動的区間最大値問い合わせ（準動的 RMQ）** [3] のためのデータ構造を用いる．

- `prepend`(x): 表 T の先頭に要素 x を償却定数時間で追加する．
- `rmq`(i_1,i_2): $T[i_1:i_2]$ の最大値を定数時間で返す．

準動的 RMQ のデータ構造を用いることで，すべての i, j に対して $\max A_{i,j}$ を合計 $O(mn)$ 時間で計算できる．

上で議論したように，提案アルゴリズムは $O(mn)$ 時間で動作する．それぞれの準動的 RMQ のデータ構造は線形領域必要で，合計 $O(mn)$ 個の要素が準動的 RMQ のデータ構造で保持される．したがって，合計の領域計算量は $O(mn)$ である．ゆえに次の定理を得る．

定理 2. op-LCS_{k^+} 問題は $O(mn)$ 時間・領域で解ける．

文献

1) G. Benson et al. LCSk: A refined similarity measure. *Theor. Comput. Sci.*, 638:11–26, 2016.
2) S. Deorowicz and S. Grabowski. Efficient algorithms for the longest common subsequence in k-length substrings. *Inf. Process. Lett.*, 114(11):634–638, 2014.
3) J. Fischer and V. Heun. Space-efficient preprocessing schemes for range minimum queries on static arrays. *SIAM J. Comput.*, 40(2):465–492, 2011.
4) M. M. Hasan et al. Order preserving pattern matching revisited. *Pattern Recogn. Lett.*, 55:15–21, 2015.
5) M. Kubica et al. A linear time algorithm for consecutive permutation pattern matching. *Inf. Process. Lett.*, 113(12):430–433, 2013.
6) F. Pavetić et al. *LCSk++*: Practical similarity metric for long strings. *CoRR*, abs/1407.2407, 2014.
7) I. Sović et al. Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nat. Commun.*, 7(11307), 2016.