The Longest Common Wave Subsequence Problem [*]

Yu-Cheng Chang[a] and Chang-Biau Yang[a †]
[a]Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan

Abstract

In this paper, we first define the longest common wave subsequence (*LCWS*) *problem, which is the expansion of the* longest wave subsequence (*LWS*) *problem, first issued by Chen and Yang* [6] *in 2020. Given two sequences A, B of distinct values and a trend sequence T, the goal of LCWSt is to find the longest common wave subsequence of A and B, whose trend is identical to the prefix of T. The LCWSr problem aims to find the longest common wave subsequence of A and B within r segments. We propose an $O(mn)$-time and $O(n)$-space algorithm for solving LCWSt, where m and n are the lengths of A and B, respectively. The time and space complexities of our algorithm for solving LCWSr are $O(rmn)$ and $O(rn)$, respectively.*

Keywords: longest increasing subsequence, longest common wave subsequence, dynamic programming, trend sequence

1 Introduction

The *longest increasing subsequence* (LIS) problem has been studied extensively in the past decades [1, 4, 7, 8, 12, 13]. For a given sequence A of length n, the LIS problem aims to find an increasing subsequence with the maxima length. In 1961, the LIS problem was first defined by Schensted [12], who also gave an $\mathrm{O}(n \log n)$-time algorithm for solving it. In 1977, Hunt and Szmanski [8] proposed an $\mathrm{O}(n \log\log n)$-time algorithm with the van Emde Boas tree (vEB tree) [2] if sequence A consists of positive integers. And in 2000, Bespamyatnikh and Segal [4] also used the vEB tree and presented an idea for enumerating all of the LIS answers, which takes $\mathrm{O}(n \log\log n)$ time. Later, in 2010, Crochemore and Porat [7] showed that if the maximal length L of an LIS is given, then the LIS problem can be solved with $\mathrm{O}(n \log\log L)$ time in the RAM model. Soon after, in 2013, Alam and Rahman [1] gave a divide-and-conquer approach to solve the LIS problem in $\mathrm{O}(n \log n)$ time.

The *longest common increasing subsequence* (LCIS) problem also got the attention of researchers in recent years [5, 9, 11, 14, 15]. In 2005, the LCIS problem was first proposed by Yang *et al.* [14] and they presented a dynamic programming algorithm with $\mathrm{O}(mn)$ time and $\mathrm{O}(mn)$ space, where m and n denote the lengths of the two input sequences, respectively. In the same year, Chan *el al.* [5] gave an $\mathrm{O}(min(r \log L, nL + r) \log\log n + Sort(n))$-time algorithm with match pairs; and later, Brodal *et al.* [3, 9] presented an $\mathrm{O}((m + nL) \log\log |\Sigma| + Sort(n))$-time and linear-space algorithm with the bounded heap, where L denotes the LCIS length and r denotes the number of match pairs. Sakai [11] gave an $\mathrm{O}(mn)$-time and $\mathrm{O}(m + n)$-space algorithm for LCIS; and Zhu [15] gave a much simpler algorithm with $\mathrm{O}(mn)$-time and $\mathrm{O}(n)$-space than Sakai [11]. Lo *et al.* [10] presented a diagonal algorithm with $\mathrm{O}((n+L(m-L)) \log\log |\Sigma|)$-time, where $|\Sigma|$ is the size of alphabet. Practically, the method of Lo *et al.* has better time efficiency than the previously published papers in most parameter settings.

To generalize the LIS problem, in 2020, Chen and Yang [6] defined the new concept, called *longest wave subsequence* (LWS). Based on this new concept, two generalized problems are defined. One is the *longest wave subsequence with trend* (LWSt) problem, aiming to find the wave subsequence whose trend is identical to the prefix of a given trend sequence T. The other is the *longest wave subsequence within r segments* (LWSr) problem, whose goal is to find the wave subsequence within r segments. Chen and Yang [6] presented an $\mathrm{O}(n \log n)$-time algorithm for solving the LWSt problem and an $\mathrm{O}(rn \log n)$-time

[*]This research work was partially supported by the Ministry of Science and Technology of Taiwan under contract MOST 109-2221-E-110-040-MY2.

[†]Corresponding author. E-mail: cbyang@cse.nsysu.edu.tw (Chang-Biau Yang).

algorithm for the LWSr problem. In the conclusion of their paper, they mentioned the *longest common wave subsequence* (LCWS) problem of two sequences. This problem may be applied to obtaining the price trend relationship between two or more stocks.

The previously related studies are summaries in Table 1.

In this paper, we define two versions of the *longest common wave subsequence* (LCWS) problem, the generalization of the LCIS problem. The first one is the *longest common wave subsequence with trend* (LCWSt) problem, with two given numeric sequences A and B composed of distinct values, and a trend sequence T, where $|A| = |T| = m \leq n = |B|$. Its goal is to find the common wave subsequence with the maximal length between A and B, such that the common subsequence is identical to the prefix of T. Obviously, the LCWSt problem is degenerated into the LCIS problem if there is only one segment in T. For solving the LCWSt problem, we propose an $O(mn)$-time and $O(n)$-space algorithm based on Zhu *et al.*'s idea [15].

The second problem is the *longest common wave subsequence within r segments* (LCWSr) problem, defined as finding the longest common wave subsequence within r segments between two given numeric sequences A and B, such that every two consecutive segments are of reverse trend. Apparently, when $r = 1$, the LCWSr problem is degenerated into the LCIS problem or LCDS problem. We also propose an $O(rmn)$-time and $O(rn)$-space algorithm based on Zhu *et al.*'s idea [15].

The rest of this paper is organized as follows. In Section 2, we will review the basic definitions of the LWS problem. In Section 3, we give the definition of the LCWSt problem and propose an algorithm for solving it with $O(mn)$ time and $O(n)$ space. In Section 4, we define the LCWSr problem and propose an $O(rmn)$-time and $O(rn)$-space algorithm for solving it. Finally, Section 5 gives the conclusions.

2 Preliminaries

The *longest wave subsequence* (LWS) problem was first defined by Chen and Yang in 2020 [6].

Definition 1. [6](trend sequence) *Given a sequence* $W = \langle w_1, w_2, \ldots, w_n\rangle$, *the* trend sequence $T = \langle t_1, t_2, \ldots, t_n\rangle$ *of* W *is defined by* Equation 1.

$$t_i = \begin{cases} 0, & \text{if } w_{i-1} > w_i \text{ for } 2 \leq i \leq n; \\ 1, & \text{if } w_{i-1} < w_i \text{ for } 2 \leq i \leq n; \\ \text{complement of } t_2, & \text{if } i = 1. \end{cases} \tag{1}$$

Definition 2. [6](increasing segment and decreasing segment) *Given a sequence* $W = \langle w_1, w_2, \ldots, w_n\rangle$ *of distinct values, the substring* $W_{i..j}$ *is called an* increasing segment, *denoted as* $W^+_{i..j}$, *if* $w_i < w_{i+1} < w_{i+2} < \ldots < w_j$, *for* $1 \leq i < j \leq n$, *but* $w_{i-1} > w_i$ *and* $w_j > w_{j+1}$. *Reversely, a* decreasing segment *is a substring* $W^-_{i..j}$ *that* $w_i > w_{i+1} > w_{i+2} > \ldots > w_j$, *for* $1 \leq i < j \leq n$, *but* $w_{i-1} < w_i$ *and* $w_j < w_{j+1}$.

Suppose that $T = \langle t_1, t_2, \ldots, t_n\rangle$ is the trend sequence of $W = \langle w_1, w_2, \ldots, w_n\rangle$. By Definition 2, for an increasing segment $W^+_{i..j}$, all elements of its trend sequence $t_{i+1..j}$ are 1, but $t_i = t_{j+1} = 0$. For a decreasing segment $W^-_{i..j}$, all elements of its trend sequence $t_{i+1..j}$ are 0, but $t_i = t_{j+1} = 1$.

For example, $W = \langle 4, 6, 9, 8, 5, 1, 2, 3, 9\rangle$ has the trend sequence $T = \langle 0, 1, 1, 0, 0, 0, 1, 1, 1\rangle$. Therefore, W can be divided into three increasing or decreasing segments, which are $W^+_{1..3}$, $W^-_{3..6}$, and $W^+_{6..9}$. There is an overlapping element (turning point) between two consecutive segments, such as $w_3 = 9$ between $W^+_{1..3}$ and $W^-_{3..6}$, $w_6 = 1$ between $W^-_{3..6}$ and $W^+_{6..9}$.

Definition 3. [6](turning point) *Given a sequence* W, p *is a turning point if* $W_{i..p}$ *has different trend from* $W_{p..j}$, *where* $1 \leq i < p < j \leq n$. *That is,* w_p *is the overlapping element of two consecutive segments in* W.

Suppose that $T = \langle t_1, t_2, \ldots, t_n\rangle$ is the trend sequence of $W = \langle w_1, w_2, \ldots, w_n\rangle$. By Definition 3, p is a turning point if $t_p \neq t_{p+1}$.

Virtually, $p = 1$ is the first turning point of an arbitrary wave sequence, since t_1 is set as the complement of t_2 in Equation 1.

Definition 4. [6](LWSt problem) *Given a numeric sequence* $A = \langle a_1, a_2, \ldots, a_n\rangle$ *of distinct values and a trend sequence* $T = \langle t_1, t_2, \ldots, t_n\rangle$, *the* longest wave subsequence with trend (LWSt) *is a longest subsequence, obtained from* A, *whose trend sequence is identical to the prefix of* T.

For example, suppose that $A = \langle 6, 1, 8, 5, 7, 9, 2, 3\rangle$ and $T = \langle 1, 0, 0, 1, 0, 1, 1, 1\rangle$. The LWSt of A with T is $\langle 8, 7, 2, 3\rangle$, whose trend $\langle 1, 0, 0, 1\rangle$ is identical to the prefix of T.

Definition 5. [6](LWSr problem) *Given a numeric sequence* $A = \langle a_1, a_2, \ldots, a_n\rangle$ *of distinct*

Table 1: The time complexities of the LIS, LCIS, LWS and our algorithms. m and n: lengths of sequences A and B; L: length of the answer; r: number of match pairs; Σ: alphabet set.

The longest increasing subsequence (LIS) problem

Year	Author(s)	Time complexity	Note
1961	Schensted [12]	$O(n \log n)$	Young tableau, binary Search
1977	Hunt and Szymanski [8]	$O(n \log \log n)$	Match Pair, van Emde Boas tree
2000	Bespamyatnikh and Segal [4]	$O(n \log \log n)$	All answers
2010	Crochemore and Porat [7]	$O(n \log \log L)$	Split blocks
2013	Alam and Rahman [1]	$O(n \log n)$	Divide-and-conquer

The longest common increasing subsequence (LCIS) problem

Year	Author(s)	Time complexity	Note
2005	Yang *et al.* [14]	$O(mn)$	Dynamic programming
2005	Chan *et al.* [5]	$O(min(r \log L, nL + r) \log \log n + Sort(n))$	Match pair, binary Search
2006	Sakai [11]	$O(mn)$	Linear space, divide-and-conquer
2006	Brodal *et al.*[3, 9]	$O((m + nL) \log \log \lvert\Sigma\rvert + Sort(n))$	Bounded heap, divide-and-conquer
2018	Zhu *et al.* [15]	$O(mn)$	Linear space
2020	Lo *et al.*[10]	$O((n + L(m - L)) \log \log \lvert\Sigma\rvert)$	Diagonal

The longest wave subsequence (LWS) problem

Year	Author(s)	Time complexity	Note
2020	Chen and Yang [6]	$O(n \log n)$	The LWSt problem, greedy
2020	Chen and Yang [6]	$O(rn \log n)$	The LWSr problem, greedy
2021	This paper	$O(mn)$	The LCWSt problem
2021	This paper	$O(rmn)$	The LCWSr problem

values and a constant r of segments, the longest wave subsequence within r segments (LWSr) *problem aims to obtain a longest subsequence of A with at most r turning points.*

For example, suppose that $A = \langle 6, 1, 8, 5, 7, 9, 2, 3\rangle$ and $r = 3$. We can obtain an LWSr answer $\langle 1, 5, 7, 9, 2, 3\rangle$ or $\langle 6, 1, 5, 7, 9, 3\rangle$ with length 6.

3 The Longest Common Wave Subsequence Problem with Trend

3.1 The Problem Definition

The LCWSt problem is defined as follows.

Definition 6. (LCWSt problem) *Given two numeric sequences of distinct values, $A = \langle a_1, a_2, \ldots, a_m\rangle$ and $B = \langle b_1, b_2, \ldots, b_n\rangle$, where $m \le n$, and a trend sequence $T = \langle t_1, t_2, \ldots, t_m\rangle$, where $t_i \in \{0, 1\}$ for $1 \le i \le m$, and $t_1 \ne t_2$, the longest common wave subsequence (LCWSt) problem is to find a common wave subsequence W between A and B, whose trend is identical to the prefix of T.*

In Definition 6, suppose that $A' = \langle a'_1, a'_2, \ldots, a'_{m'}\rangle$ and $B' = \langle b'_1, b'_2, \ldots, b'_{m'}\rangle$ are subsequences of A and B, respectively. If $a'_i = b'_i$, for $1 \le i \le m' \le m$, and the trend of A' is identical to the prefix of T, then A' is a common wave subsequence of A and B with trend T.

For example, suppose that $A = \langle 7, 2, 8, 3, 1, 5, 6, 4, 9\rangle$, $B = \langle 2, 5, 6, 7, 8, 3, 1, 9, 4\rangle$, and $T = \langle 0, 1, 1, 0, 1, 0, 0, 0, 1\rangle$. The LCWSt is $\langle 2, 5, 6, 4\rangle$, whose trend $\langle 0, 1, 1, 0\rangle$ is identical to the prefix of T.

Note that the LCWSt problem is degenerated into the LCIS problem when there is only one turning point (one segment) in the trend sequence T.

LCWSt may or may not be a *longest common subsequence* (LCS) of A and B. For the above example, the LCS is $\langle 2(7), 8, 3, 1, 4(9)\rangle$, while the LCWSt is $\langle 2, 5, 6, 4\rangle$. Clearly, the trend of some subsequences of the LCSs may be identical to the prefix of T, but we still cannot obtain the LCWSt from the subsequence of an LCS.

3.2 The Dynamic Programming Algorithm

We illustrate our idea for solving the LCWSt problem in Table 2 with the above example. Each cell in Table 2 has three values as follows.

- $L(i, j)$: the LCWSt length of $A_{1..i}$ and $B_{1..j}$ with ending at b_j.
- $\ell^-(i, j)$: the maximal LCWSt length with ending at $b_{j'}$ that $b_{j'} < b_j$ and $j' < j$.
- $\ell^+(i, j)$: the maximal LCWSt length with ending at $b_{j'}$ that $b_{j'} > b_j$ and $j' < j$.

Table 2: An example of the LCWSt algorithm with $A = \langle 7, 2, 8, 3, 1, 5, 6, 4, 9\rangle$, $B = \langle 2, 5, 6, 7, 8, 3, 1, 9, 4\rangle$, and $T = \langle 0, 1, 1, 0, 1, 0, 0, 0, 1\rangle$, whose turning points $P = \langle 1, 3, 4, 5, 8\rangle$. The LCWSt answer is $\langle 2, 5, 6, 4\rangle$ with length 4.

A \ B		2	5	6	7	8	3	1	9	4
	0	0	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-
	0	$\underline{\mathbf{0}}$	0	0	1	0	0	0	0	0
7	-	-	-	-	$\underline{\mathbf{0}}$	-	-	-	-	-
	-	-	-	-	0	-	-	-	-	-
	$\underline{\mathbf{0}}$	1	0	0	1	0	0	0	0	0
2	-	$\underline{\mathbf{0}}$	-	-	-	-	-	-	-	-
	-	0	-	-	-	-	-	-	-	-
	0	$\underline{\mathbf{1}}$	0	0	1	2	0	0	0	0
8	-	-	-	-	-	$\underline{\mathbf{1}}$	-	-	-	-
	-	-	-	-	-	0	-	-	-	-
	0	$\underline{\mathbf{1}}$	0	0	1	$\underline{\mathbf{2}}$	2	0	0	0
3	-	-	-	-	-	-	$\underline{\mathbf{1}}$	-	-	-
	-	-	-	-	-	-	$\underline{\mathbf{2}}$	-	-	-
	$\underline{\mathbf{0}}$	1	0	0	1	2	$\underline{\mathbf{2}}$	1	0	0
1	-	-	-	-	-	-	-	$\underline{\mathbf{0}}$	-	-
	-	-	-	-	-	-	-	$\underline{\mathbf{2}}$	-	-
	0	$\underline{\mathbf{1}}$	2	0	1	2	2	1	0	0
5	-	-	$\underline{\mathbf{1}}$	-	-	-	-	-	-	-
	-	-	0	-	-	-	-	-	-	-
	0	1	$\underline{\mathbf{2}}$	3	1	2	2	1	0	0
6	-	-	-	$\underline{\mathbf{2}}$	-	-	-	-	-	-
	-	-	-	0	-	-	-	-	-	-
	0	1	2	$\underline{\mathbf{3}}$	1	2	$\underline{\mathbf{2}}$	1	0	4
4	-	-	-	-	-	-	-	-	-	$\underline{\mathbf{2}}$
	-	-	-	-	-	-	-	-	-	$\underline{\mathbf{3}}$
	0	1	2	$\underline{\mathbf{3}}$	1	2	2	1	3	4
9	-	-	-	-	-	-	-	-	$\underline{\mathbf{3}}$	-
	-	-	-	-	-	-	-	-	0	-

Obviously, $L(i, j) = 0$ if b_j is not in $A_{1..i}$. For example, in Table 2, $L(3, 2) = 0$ since $b_2 = 5$ is not in $A_{1..3} = \langle 7, 2, 8\rangle$. $L(3, 5) = 2$ since $b_5 = 8$ is in $A_{1..3} = \langle 7, 2, 8\rangle$ and the LCWSt length of $A_{1..3}$ and $B_{1..5}$ is 2, which ends at b_5.

Definition 7. *Let $L(i, j)$ denote the LCWSt length of $A_{1..i}$ and $B_{1..j}$ with ending at b_j, for $1 \le i \le m$ and $1 \le j \le n$. Some notations are* *given as follows.*

$$\begin{aligned}\beta^-(i,j) &= \{u|b_u < a_i \text{ for } 1 \le u \le j-1\}\,, \\ \beta^+(i,j) &= \{u|b_u > a_i \text{ for } 1 \le u \le j-1\}\,, \\ \ell^-(i,j) &= \max\{L(i-1,u)|u \in \beta^-(i,j)\}, \\ \ell^+(i,j) &= \max\{L(i-1,u)|u \in \beta^+(i,j)\}, \\ \ell(i,j) &= \max\{\ell^-(i,j), \ell^+(i,j)\}.\end{aligned} \tag{2}$$

For example, in Table 2, $\beta^-(4,6) = \{1\}$ and $\beta^+(4,6) = \{2,\ 3,\ 4,\ 5\}$. Then, $\ell^-(4,6) = 1$, obtained from $A_{1..3} = \langle 7,\ 2,\ 8\rangle$ and $B_{1..5} = \langle 2,\ 5,\ 6,\ 7,\ 8\rangle$. Its LWCSt answer is $\langle 2\rangle$ with length 1, whose ending element 2 is smaller than $a_4 = 3$. Similarly, $\ell^+(4,6) = 2$ with LCWSt answer $\langle 7,\ 8\rangle$, whose ending element 8 is larger than $a_4 = 3$. Furthermore, $\ell(4,6) = \max\{1,\ 2\} = 2$. In the table, for more clarity, we only show the values of $\ell^-(i,j)$ and $\ell^+(i,j)$ when $a_i = b_j$ since the LCWSt length may be increased only at those cells.

In fact, $\ell(i,j)$ is the LCWSt length of $A_{1..i-1}$ and $B_{1..j-1}$, not counting a_i and b_j. With $\ell(i,j)$, we can know the state (increasing, decreasing or turning point) of the wave subsequence when a_i and b_j are to be computed.

For simplicity, we use ℓ, ℓ^- and ℓ^+ to denote $\ell(i,j)$, $\ell^-(i,j)$ and $\ell^+(i,j)$, respectively, when there is no ambiguity. $L(i,j)$ can be computed by the following dynamic programming (DP) formula, for $0 \le i \le m$ and $0 \le j \le n$.

$$L(i,j) = \begin{cases} 0 & \text{,if } i=0 \text{ or } j=0; \\ L(i-1,j) & \text{,if } i,j>0 \text{ and } a_i \ne b_j; \\ 1+\ell^- & \text{,if } t_\ell = t_{\ell+1} = 1 \text{ and } a_i = b_j; \\ 1+\ell^+ & \text{,if } t_\ell = t_{\ell+1} = 0 \text{ and } a_i = b_j; \\ \max\{1+\ell^-, \ell^+\} & \text{,if } t_\ell \ne t_{\ell+1} = 1 \text{ and } a_i = b_j; \\ \max\{\ell^-, 1+\ell^+\} & \text{,if } t_\ell \ne t_{\ell+1} = 0 \text{ and } a_i = b_j; \end{cases} \tag{3}$$

The calculation of Table 2 is done with the row-major manner. The first segment $T_{1..3}$ of the trend sequence T is increasing. First, in the row of $a_1 = 7$, $L(1,4)$ is updated to 1 because $a_1 = b_4$. Then $L(2,1) = 1$ due to $a_2 = 2 = b_1$. In the row of $a_3 = 8$, we have $\ell^-(3,5) = 1$. Accordingly, $L(3,5)$ is set as 2 because the LCWSt length can be extended with $a_3 = 8 = b_5$ from either $b_1 = 2$ or $b_4 = 7$. In the row of $a_4 = 3$, we set $L(4,6) = 2$ since $a_2 = 2 = b_1$ can be extended with $a_4 = 3 = b_6$. Similarly, we obtain $L(5,7) = 1$, $L(6,2) = 2$ and $L(7,3) = 3$.

For computing $L(8,9)$ in the row of $a_8 = 4$, we have $t_3 \ne t_4 = 0$ in the trend sequence T, indicating a turning point. We have $\ell^-(8,9) = 2$, obtained from $\beta^-(8,9) = \{1,\ 6,\ 7\}$, representing the longest subsequence $\langle 2,\ 3\rangle$; and $\ell^+(8,9) = 3$, obtained from $\beta^+(8,9) = \{2,\ 3,\ 4,\ 5,\ 8\}$, representing the longest subsequence $\langle 2,\ 5,\ 6\rangle$. Therefore, $L(8,9) = \max\{\ell^-(8,9), 1+\ell^+(8,9)\} = \max\{2, 1+3\} = 4$. It means that $a_8 = 4 = b_9$ may either substitute the second element ($\ell^-(8,9) = 2$) of $\langle 2,\ 3\rangle$ to form $\langle 2,\ 4\rangle$ in the increasing segment, or pass over the turning point and be appended to $\langle 2,\ 3,\ 6\rangle$ to form a longer answer. Here we get $\langle 2,\ 3,\ 6,\ 4\rangle$, with $L(8,9) = 4$. The reason why we do not append 4 to $\langle 2,\ 3\rangle$ to form $\langle 2,\ 3,\ 4\rangle$ is that we know the state of the current wave subsequence is at turning point by $\ell = 3$. Therefore, it is a decreasing segment currently and we can only substitute the turning point or append to it. If we want to obtain $\langle 2,\ 3,\ 4\rangle$, ℓ would indicate that it is an increasing segment rather than a turning point.

For computing $L(9,8)$ in the row of $a_9 = 9$, we also encounter the turning point $t_3 \ne t_4 = 0$. We have $\ell^-(9,8) = 3$, obtained from $\beta^-(9,8) = \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\}$, representing the longest subsequence $\langle 2,\ 3,\ 6\rangle$; and $\ell^+(9,8) = 0$. Therefore, $L(9,8) = \max\{\ell^-(9,8), 1+\ell^+(9,8)\} = \max\{3, 1+0\} = 3$. It means that we cannot pass over the turning point and can only substitute $a_7 = 6 = b_3$ with $a_9 = 9 = b_8$. The answer for $L(9,8) = 3$ is $\langle 2,\ 3,\ 9\rangle$.

After all, we can obtain the LCWSt length of A and B, which is $\max_{1 \le j \le n}\{L(m,j)\}$. It is clear that both time and space complexities are $O(mn)$. By recording the predecessor of each element during the process in an extra $O(n)$ space, we can construct the LCWSt content easily. The LCWSt algorithm is presented in Algorithm 1.

Note that the calculation of each $\ell^-(i,j)$ with $\beta^-(i,j)$ can be done incrementally. Thus we need only $O(1)$ time for calculating each $\ell^-(i,j)$. Since the calculation is performed with the row-major manner and incrementally, the total space complexity can be easily reduced to $O(n)$.

Theorem 1. *$L(i,j)$ can be calculated correctly with Algorithm 1, where $L(i,j)$ stores the LCWSt length of $A_{1..i}$ with ending at b_j, for $1 \le i \le m$ and $1 \le j \le n$.*

4 The Longest Common Wave Subsequence Problem within r Segments

The definition of the LCWSr problem is given as follows.

Definition 8. (LCWSr problem) *Given two numeric sequences of distinct values, $A = \langle a_1,\ a_2,\ \ldots,\ a_m\rangle$ and $B = \langle b_1,\ b_2,\ \ldots,\ b_n\rangle$, where $m \le n$, and a constant r of segments, the* longest common

Algorithm 1 Computing the LCWSt length

```
Input: Two numeric sequences of distinct values $A = \langle a_1, a_2, \ldots, a_m \rangle$ and $B = \langle b_1, b_2, \ldots, b_n \rangle$, where $m \leq n$, and a trend sequence $T = \langle t_1, t_2, \ldots, t_m \rangle$, where $t_i \in \{0, 1\}$ and $t_1 \neq t_2$
Output: The LCWSt length
 1: $L(i, 0) \leftarrow 0$, for $1 \leq i \leq m$
 2: $L(0, j) \leftarrow 0$, for $1 \leq j \leq n$
 3: for $i = 1$ to $m$ do
 4:     for $j = 1$ to $n$ do
 5:         if $a_i \neq b_j$ then
 6:             $L(i, j) \leftarrow L(i-1, j)$
 7:         else
 8:             Compute $\beta^-(i, j)$ and $\beta^+(i, j)$.
 9:             $\ell^- \leftarrow \max\{L(i-1, u) | u \in \beta^-(i, j)\}$
10:             $\ell^+ \leftarrow \max\{L(i-1, u) | u \in \beta^+(i, j)\}$
11:             $\ell \leftarrow \max\{\ell^-, \ell^+\}$
12:             if $t_\ell = t_{\ell+1} = 1$ then
13:                 $L(i, j) \leftarrow 1 + \ell^-$
14:             else if $t_\ell = t_{\ell+1} = 0$ then
15:                 $L(i, j) \leftarrow 1 + \ell^+$
16:             else if $t_\ell \neq t_{\ell+1} = 1$ then
17:                 $L(i, j) \leftarrow \max\{1 + \ell^-, \ell^+\}$
18:             else if $t_\ell \neq t_{\ell+1} = 0$ then
19:                 $L(i, j) \leftarrow \max\{\ell^-, 1 + \ell^+\}$
20: return $\max\{L(m, j) | 1 \leq j \leq n\}$
```

wave subsequence within r segments (*LCWSr*) *aims to find the common wave subsequence between A and B with the maximal length and at most r turning points.*

For example, suppose that two numeric sequences $A = \langle 6, 2, 9, 4, 3, 7, 8, 1, 5 \rangle$, $B = \langle 7, 5, 1, 6, 4, 2, 9, 3, 8 \rangle$, and a constant $r = 3$ of segments. We can get the length of LCWSr between A and B is 4, which is $\langle 6, 4, 3, 8 \rangle$ with 2 segments or $\langle 6, 9, 3, 8 \rangle$ with 3 segments.

4.1 The Dynamic Programming Algorithm

Basically, the main idea for solving the LCWSr problem is same as the LCWSt algorithm. Here, we have to calculate $\beta^-(i, j)$, $\beta^+(i, j)$, $L_1(i, j)$, $L_2(i, j), \cdots, L_r(i, j)$, for $1 \leq i \leq m$ and $1 \leq j \leq n$.

Definition 9. *For $1 \leq k \leq r$, let $\ell_k^-(i, j) = \max\{L_k(i-1, u) | u \in \beta^-(i, j)\}$, $\ell_k^+(i, j) = \max\{L_k(i-1, u) | u \in \beta^+(i, j)\}$. Let $L_k(i, j)$ denote the LCWSr length of $A_{1..i}$ and $B_{1..j}$ ending at b_j within k segments.*

Without loss of generality, it is assumed that the first segment in the answer is increasing. The LCWSr problem can be solved by calculating $L_k(i, j)$ with the DP formula of Equation 4, as shown in Figure 1.

It is clear that the answer within k segments must not be shorter than that within $k-1$ segments. Therefore, when $a_i = b_j$ for computing $L_k(i, j)$, we have to examine $L_1, L_2, \cdots, L_{k-1}$. In fact, it is sufficient to check L_{k-1}, since the computation is done recursively.

In the following, we illustrate our algorithm with an example in Table 3. Note that, in each cell, we group the three values as a 3-tuple $(L_k(i, j),\ \ell_k^-(i, j),\ \ell_k^+(i, j))$. The number colored by blue represents that it is the length inheriting from its previous segment. In other words, the number of segments is not increased.

For the first row of $a_1 = 6$, because $a_1 = 6 = b_4$ is the first common number, the first 3-tuple is $(1, 0, 0)$. The second segment inherits the length from the first segment, so the second 3-tuple is $(1, 0, 0)$, whose 1 is colored by blue. And the third segment also inherits from the second segment, obtaining 3-tuple $(1, 0, 0)$.

In the next row, for computing $a_2 = 2 = b_6$, the first tuple is $(1, 0, 1)$. In the second segment, which is decreasing, we can append $b_6 = 2$ to $b_4 = 6$ to form an increasing segment (with length 1) and then a decreasing segment. Thus, the second tuple is $(2, 0, 1)$. As for the third segment, it inherits the length of the second segment, obtaining $(2, 0, 1)$, whose 2 is marked by blue. For more details, $L_3(2, 6) = 1 + \max\{\ell_2^+(2, 6), \ell_3^-(2, 6)\} = 1 + \max\{1, 0\} = 2$.

In the third row of $a_3 = 9$, it sets $L_1(3, 7) = 2$ by appending b_7 to $b_4 = 6$. For $L_2(3, 7)$, it inherits from $L_1(3, 7)$. As for $L_3(3, 7)$, it appends b_7 to $b_6 = 2$. We get an LCWSr answer $\langle 6, 2, 9 \rangle$ until now. Note that there are 3 segments in $\langle 6, 2, 9 \rangle$, which consists of an increasing segment with length 1, a decreasing segment with length 2 and an increasing segment with length 2. The overlapped element (at the turning point) is counted twice, for two neighboring segments.

The following rows can be computed by Equation 4 and we can obtain $L_3(9, j)$ finally. Due to the inheritance, the maximum in $L_r(m, j)$ should be the length of the LCWSr. The number of segments is within r, but we cannot guarantee the exact number of segments involved in th answer. In addition, the trace back technique may be a little complex because we have to record not only the predecessors but the inheritances.

The algorithm for calculating the LCWSr length is given in Algorithm 2. Here, without loss of generality, it is assumed that the first segment

$$L_k(i,j) = \begin{cases} 0 & \text{,if } i = 0 \text{ or } j = 0; \\ L_k(i-1,j) & \text{,if } i,j > 0 \text{ and } a_i \neq b_j; \\ 1 + \max(\ell^{+}_{k-1}(i,j), \ell^{-}_{k}(i,j)) & \text{,if } i,j > 0,\ a_i = b_j \text{ and segment } k \text{ is increasing.} \\ 1 + \max(\ell^{-}_{k-1}(i,j), \ell^{+}_{k}(i,j)) & \text{,if } i,j > 0,\ a_i = b_j \text{ and segment } k \text{ is decreasing,} \end{cases} \tag{4}$$

where $\ell^{-}_{0}(i,j) = 0$ and $\ell^{+}_{0}(i,j) = 0$.

Figure 1: The DP formula for solving the LCWSr problem.

Table 3: An example of the LCWSr algorithm with $A = \langle 6, 2, 9, 4, 3, 7, 8, 1, 5\rangle$, $B = \langle 7, 5, 1, 6, 4, 2, 9, 3, 8\rangle$, and $r = 3$. The LCWSr answer is $\langle 6, 4, 3, 8\rangle$ with length 4. Here, $*$ represents $(0, 0, 0)$.

A \ B		7	5	1	6	4	2	9	3	8
	* * *	* * *	* * *	* * *	* * *	* * *	* * *	* * *	* * *	* * *
6	* * *	* * *	* * *	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	* * *	* * *	* * *	* * *	* * *
2	* * *	* * *	* * *	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	* * *	(1, 0, 1) (2, 0, 1) (2, 0, 1)	* * *	* * *	* * *
9	* * *	* * *	* * *	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	* * *	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(2, 1, 0) (2, 1, 0) (3, 2, 0)	* * *	* * *
4	* * *	* * *	* * *	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(2, 1, 0) (2, 1, 0) (3, 2, 0)	* * *	* * *
3	* * *	* * *	* * *	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(2, 1, 0) (2, 1, 0) (3, 2, 0)	(2, 1, 2) (3, 1, 2) (3, 2, 3)	* * *
7	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	* * *	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(2, 1, 0) (2, 1, 0) (3, 2, 0)	(2, 1, 2) (3, 1, 2) (3, 2, 3)	* * *
8	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	* * *	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(2, 1, 0) (2, 1, 0) (3, 2, 0)	(2, 1, 2) (3, 1, 2) (3, 2, 3)	(3, 2, 2) (3, 3, 2) (4, 3, 3)
1	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	* * *	(1, 0, 0) (2, 0, 1) (2, 0, 1)	(1, 0, 0) (1, 0, 0) (1, 0, 0)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(2, 1, 0) (2, 1, 0) (3, 2, 0)	(2, 1, 2) (3, 1, 2) (3, 2, 3)	(3, 2, 2) (3, 3, 2) (4, 3, 3)
5	* * *	(1, 0, 0) (1, 0, 0) (1, 0, 0)	(1, 0, 0) (2, 0, 1) (2, 0, 1)	(1, 0, 0) (2, 0, 1) (2, 0, 1)	(1, 0, 0) (1, 0, 0) (1, 0, 0)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(1, 0, 1) (2, 0, 1) (2, 0, 1)	(2, 1, 0) (2, 1, 0) (3, 2, 0)	(2, 1, 2) (3, 1, 2) (3, 2, 3)	(3, 2, 2) (3, 3, 2) (4, 3, 3)

is increasing. It is easily to modify the algorithm for the case that the first segment is decreasing.

It is clear that the time complexity and space complexity of our algorithm are both O(rmn) and O(rmn). It is not difficult to reduce the space to O(mn).

Theorem 2. *$L_{1..k}(i,j)$ can be computed correctly with Algorithm 2, where $L_{1..k}(i,j)$ stores the LCWSr lengths of $A_{1..i}$ and $B_{1..j}$ within k segments with ending at b_j, for $1 \leq i \leq m$, $1 \leq j \leq n$ and $1 \leq k \leq r$.*

5 Conclusion

In this paper, we solve two versions of the *longest common wave subsequence* (LCWS) problem. One is the *longest common wave subsequence with trend* (LCWSt) and the other is the *longest common wave subsequence within r segments* (LCWSr) problem. As mentioned by Chen and Yang [6], LCWS may be applied to a stock market when the price trend relationship between two or more stocks are considered. Furthermore, we may try to solve the LCWS problem with three or more sequences. We may try other ways to

Algorithm 2 Computing the LCWSt length

```
Input: Two numeric sequence $A = \langle a_1, a_2, \ldots, a_m \rangle$ and $B = \langle b_1, b_2, \ldots, b_n \rangle$, where $m \le n$, and a constant $r$ of segments.
Output: The LCWSr length
 1: $L_k(i, 0) = 0$, for $1 \le k \le r$ and $1 \le i \le m$
 2: $L_k(0, j) = 0$, for $1 \le k \le r$ and $1 \le j \le n$
 3: for $i = 1$ to $m$ do
 4:     for $j = 1$ to $n$ do
 5:         $\beta^-(i, j) \leftarrow \{u | b_u < a_i$ for $1 \le u \le j-1\}$
 6:         $\beta^+(i, j) \leftarrow \{u | b_u > a_i$ for $1 \le u \le j-1\}$
 7: for $i = 1$ to $m$ do
 8:     for $j = 1$ to $n$ do
 9:         Construct $\beta^-(i, j)$ and $\beta^+(i, j)$
10:         for $k = 1$ to $r$ do
11:             if $a_i \ne b_j$ then
12:                 $L_k(i, j) \leftarrow L_k(i-1, j)$
13:             else
14:                 $\ell^-_{k-1} \leftarrow \max\{L_{k-1}(i-1, u) | u \in \beta^-(i, j)\}$
15:                 $\ell^+_{k-1} \leftarrow \max\{L_{k-1}(i-1, u) | u \in \beta^+(i, j)\}$
16:                 $\ell^-_k \leftarrow \max\{L_k(i-1, u) | u \in \beta^-(i, j)\}$
17:                 $\ell^+_k \leftarrow \max\{L_k(i-1, u) | u \in \beta^+(i, j)\}$
18:                 if segment $k$ is increasing then
19:                     $\ell_k = \max\{\ell^+_{k-1}, \ell^-_k\}$
20:                 else
21:                     $\ell_k = \max\{\ell^-_{k-1}, \ell^+_k\}$
22:                 $L_k(i, j) = 1 + \ell_k$
23: return $\max\{L_r(m, j) | 1 \le j \le n\}$
```

solve the two LCWS problems with a lower time complexity and try to find out whether the lower bound of the LCWS problem is same as the LCIS problem or not.

It is an open question whether we can obtain the LCWSr by erasing some elements from one of the LCSs. We ensure that when $r = 1$, the LCWSr is degenerated into the LCIS problem, and it is clear that the LCIS answer cannot be obtained from the LCS. However, when $r \ge r'$, where r' is the number of the turning points in the LCS, the LCWSr must be same as the LCS. If the LCWSr answer can be obtained from one of the LCSs, then we can first get the LCS answer and then apply the LWSr algorithm proposed by Chen and Yang [6]. If the LCWSr can be got with this way, it is still not easy to perform the algorithm, since the possible LCS answers may be numerous.

References

[1] M. R. Alam and M. S. Rahman, "A divide and conquer approach and a work-optimal parallel algorithm for the lis problem," *Information Processing Letters*, Vol. 113, No. 13, pp. 470–476, 2013.

[2] P. v. E. Boas, R. Kaas, and E. Zijlstra, "Design and implementation of an efficient priority queue," *Mathematical Systems Theory*, Vol. 10, No. 1, pp. 99–127, 1976.

[3] G. S. Brodal, K. Kaligosi, I. Katriel, and M. Kutz, "Faster algorithms for computing longest common increasing subsequences," *Proceedings of the 17th Annual Symposium on Combinatorial Pattern Matching (CPM)*, Barcelona, Spain, pp. 330–341, 2006.

[4] S. Camyatnikh and M. Segal, "Enumerating longest increasing subsequences and patience sorting," *Information Processing Letters*, Vol. 76, No. 1-2, pp. 7–11, 2000.

[5] W.-T. Chan, Y. Zhang, S. P. Y. Fung, D. Ye, and H. Zhu, "Efficient algorithms for finding a longest common increasing subsequence," *Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC 2005)*, Sanya, Hainan, China, 2005. Also in *Lecture Notes in Computer Science*, Vol. 3827, pp. 665–674, 2005.

[6] G.-Z. Chen and C.-B. Yang, "The longest wave subsequence problem: Generalizations of the longest increasing subsequence problem," *The 37th Workshop on Combinatorial Mathematics and Computation Theory*, Kaohsiung, Taiwan, pp. 28–33, 2020.

[7] M. Crochemore and E. Porat, "Fast computation of a longest increasing subsequence and application," *Information and Computation*, Vol. 208, No. 9, pp. 1054–1059, 2010.

[8] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences," *Communications of the ACM*, Vol. 20, pp. 350–353, 1977.

[9] M. Kutz, G. S. Brodal, K. Kaligosi, and I. Katriel, "Faster algorithms for computing longest common increasing subsequences," *Journal of Discrete Algorithms*, Vol. 9, No. 4, pp. 314–325, 2011.

[10] S.-F. Lo, K.-T. Tseng, C.-B. Yang, and K.-S. Huang, "A diagonal-based algorithm for the longest common increasing subsequence problem," *Theoretical Computer Science*, Vol. 815, pp. 69–78, 2020.

[11] Y. Sakai, "A linear space algorithm for computing a longest common increasing subsequence," *Information Processing Letters*, Vol. 99, No. 5, pp. 203–207, 2006.

[12] C. Schensted, "Longest increasing and decreasing subsequences," *Canadian Journal of Mathematics*, Vol. 13, pp. 179–191, 1961.

[13] C.-T. Tseng, C.-B. Yang, and H.-Y. Ann, "Minimum height and sequence constrained longest increasing subsequence," *Journal of Internet Technology*, Vol. 10, No. 2, pp. 173–178, 2009.

[14] I.-H. Yang, C.-P. Huang, and K.-M. Chao, "A fast algorithm for computing a longest common increasing subsequence," *Information Processing Letters*, Vol. 93, No. 5, pp. 249–253, 2005.

[15] D. Zhu, L. Wang, T. Wang, and X. Wang, "A simple linear space algorithm for computing a longest common increasing subsequence," *IAENG International Journal of Computer Science*, Vol. 45, No. 3, pp. 472–477, 2018.