Longest Common Rollercoasters

Kosuke Fujita[1], Yuto Nakashima[1], Shunsuke Inenaga[1],

Hideo Bannai[2], and Masayuki Takeda[1]

1. Kyushu University
2. Tokyo Medical and Dental University

Background

- The Longest Common Subsequence (LCS) Problem is an important problem that appears in various fields.

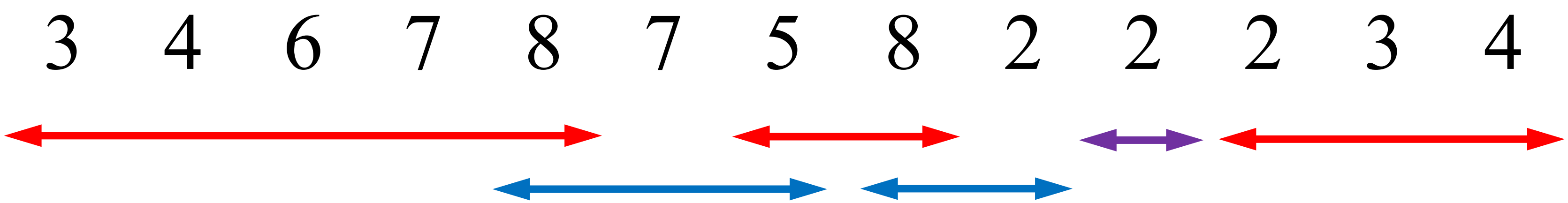
- Since 2018, the study on the sequence called rollercoaster has been conducted [Biedl et al., 2018].

- Longest Common Rollercoaster is natural extension of LCS.

Run

Definition [Biedl et al., 2018]

A substring is a ***run***, if it is a maximal strictly increasing ($+$-*run*) or a maximal strictly decreasing ($-$-*run*) substring.



 : $+$-run

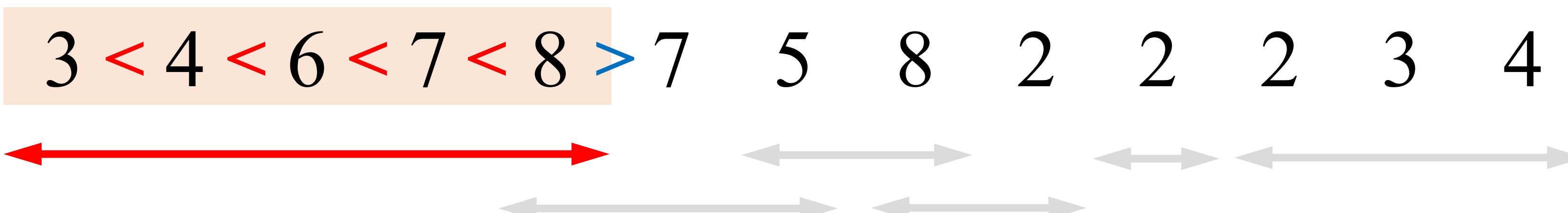
 : $-$-run

 : both $+$-run and $-$-run

Run

Definition [Biedl et al., 2018]

A substring is a ***run***, if it is a maximal strictly increasing (+-*run*) or a maximal strictly decreasing (−-*run*) substring.



: +-run

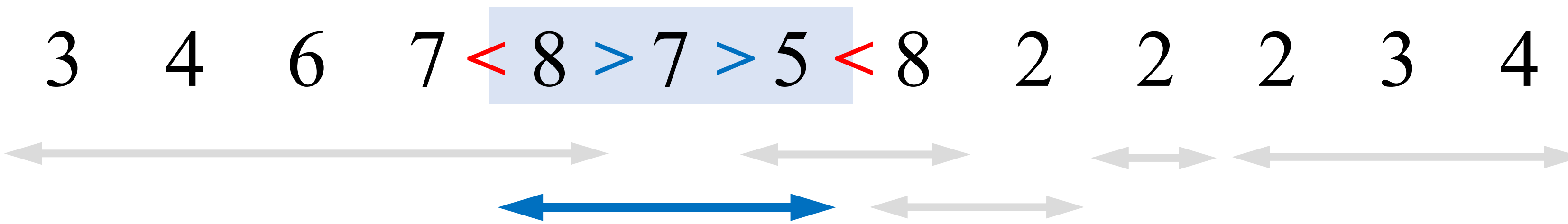
: −-run

: both +-run and −-run

Run

Definition [Biedl et al., 2018]

A substring is a ***run***, if it is a maximal strictly increasing ($+$-*run*) or a maximal strictly decreasing ($-$-*run*) substring.



: $+$-run

: $-$-run

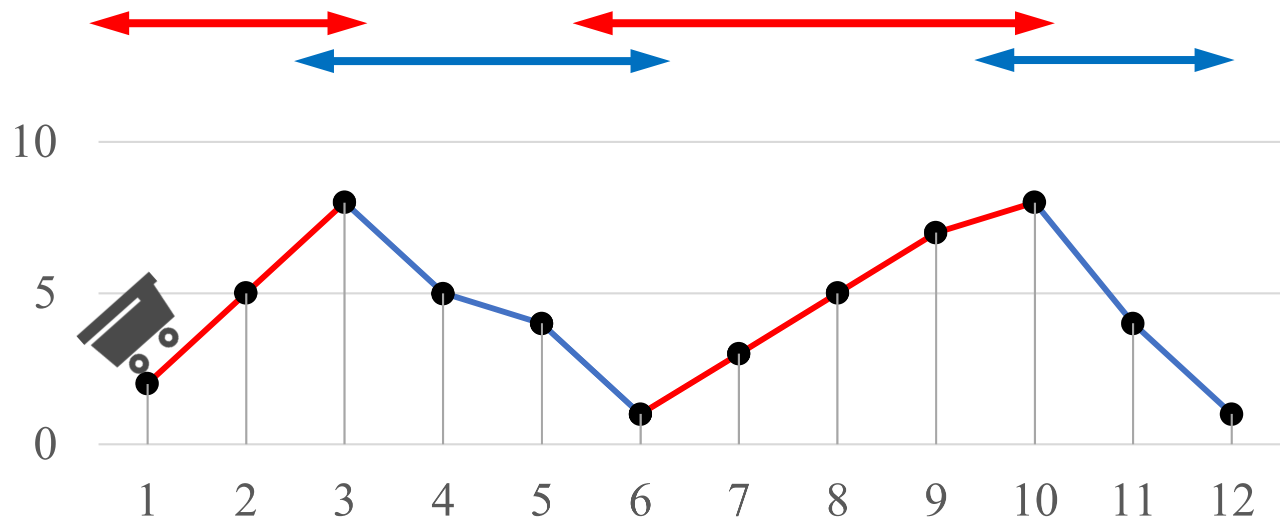
: both $+$-run and $-$-run

k-rollercoaster

Definition [Biedl et al., 2018]

A string X is a ***k-rollercoaster*** if any run in S is of length at least k.

$$X = 2 \quad 5 \quad 8 \quad 5 \quad 4 \quad 1 \quad 3 \quad 5 \quad 7 \quad 8 \quad 4 \quad 1$$



X is 3-rollercoaster

Previous work on k-rollercoasters

Longest k-Rollercoaster problem

Input :

String X of length n and

Positive integer k.

Output :

Longest k-rollercoaster that is a subsequence of X.

- $O(nk \log n)$ time [Biedl et al., 2018]
- $O(\min\{nk^2, n \log^2 n\})$ time [Gawrychowski et al., 2019]

Our Problem

Longest Common k-Rollercoaster problem

Input :

String S of length n,

String T of length m $(\leq n)$,

Positive integer k

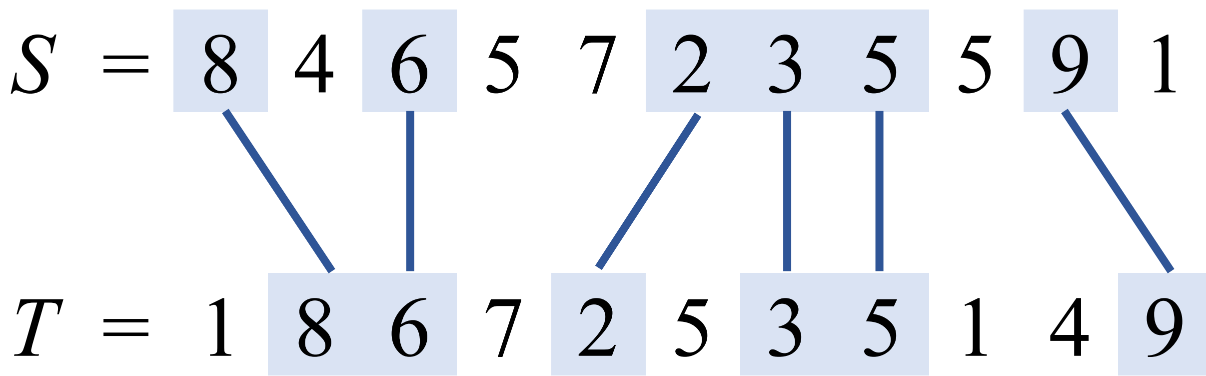
Output :

Longest k-rollercoaster that is a subsequence of both S and T.

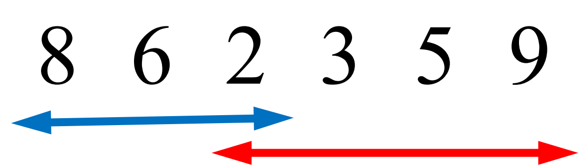
When $k = 1$, then this problem is equivalent to LCS.
So, this problem is a generalization of LCS.
In the following, I talk about the case where $k \geq 2$.

Longest common k-rollercoaster

When $k = 3$,



Longest common **3**-rollercoaster subsequence of S and T is



Our Contribution

Theorem 1.

A longest common k-rollercoaster of S and T can be computed in $O(nmk)$ time and space.

Theorem 2.

A longest common k-rollercoaster of S and T can be computed in $O(rk \log^3 m \log \log m)$ time and $O(rk)$ space.

r : the number of pairs (i, j) s.t. $S[i] = T[j]$

When S and T are random strings over $\{1, ..., \sigma\}$, then the expected value of r is nm / σ.

→ Theorem 2 is expected to be more space-efficient than Theorem 1.

Proof of Theorem 1

Theorem 1.

A longest common k-rollercoaster of S and T can be computed in $O(nmk)$ time and space.

Idea :

We use dynamic programming on S and T.

In our dynamic programming algorithm,
we use $(k, h)_w$-rollercoaster subsequences which are generalization of k-rollercoaster subsequences.

$(k, h)_w$-rollercoaster

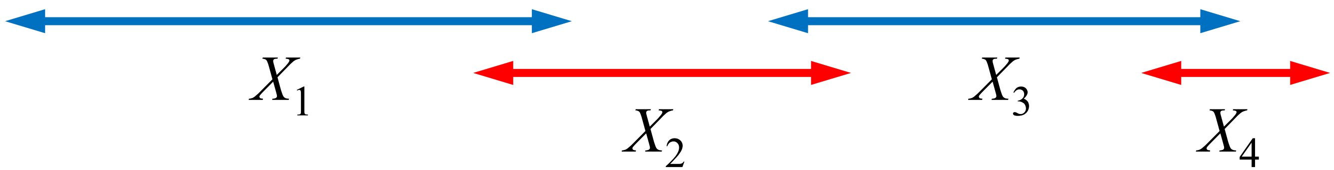
Definition [Biedl et al., 2018]

For an integer string X, let X_1, X_2, ..., X_x be the sequence of runs in X ordered by their occurrence in X. For $w \in \{+, -\}$ and integer $h \in [1, k]$, X is a ***$(k, h)_w$-rollercoaster*** if X_1, X_2, ..., X_x satisfies the following

1. The last run X_x is w-run.
2. $|X_i| \geq k$ for $i \in [1, x - 1]$.
3. If $h \in [1, k - 1]$, $|X_i| = h$, and $|X_i| \geq k$ otherwise.

Example of $(k, h)_+$-rollercoaster

$$X = 9\ 8\ 6\ 4\ 2\ 1\ 3\ 4\ 8\ 5\ 4\ 2\ 1\ 3$$



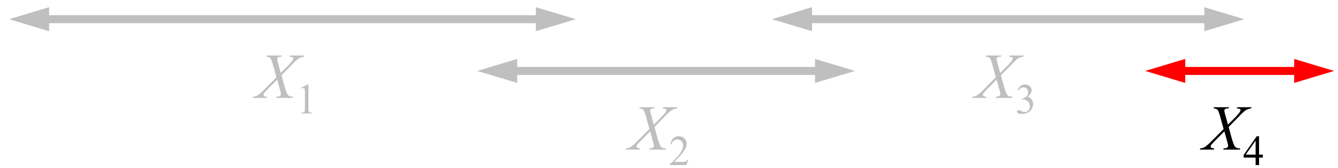
X is $(4, 2)_+$-rollercoaster

1. The last run X_4 is $+$-run
2. $|X_1|, |X_2|, |X_3| \geq 4$
3. $h = 2$ and $|X_4| = 2$

> 1. The last run X_x is $+$-run
> 2. $|X_i| \geq k$ for $i \in [1, x-1]$
> 3. If $h \in [1, k-1]$, $|X_i| = h$, and $|X_i| \geq k$ otherwise

Example of $(k, h)_+$-rollercoaster

$$X = 9\ 8\ 6\ 4\ 2\ 1\ 3\ 4\ 8\ 5\ 4\ 2\ 1\ 3$$



X is $(4, 2)_+$-rollercoaster

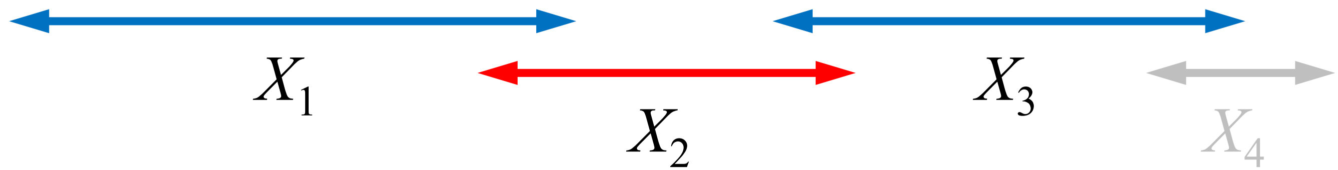
1. **The last run X_4 is $+$-run**
2. $|X_1|, |X_2|, |X_3| \geq 4$
3. $h = 2$ and $|X_4| = 2$

---

1. **The last run X_x is $+$-run**
2. $|X_i| \geq k$ for $i \in [1, x-1]$
3. If $h \in [1, k-1]$, $|X_i| = h$, and $|X_i| \geq k$ otherwise

Example of $(k, h)_+$-rollercoaster

$$X = 9\ 8\ 6\ 4\ 2\ 1\ 3\ 4\ 8\ 5\ 4\ 2\ 1\ 3$$



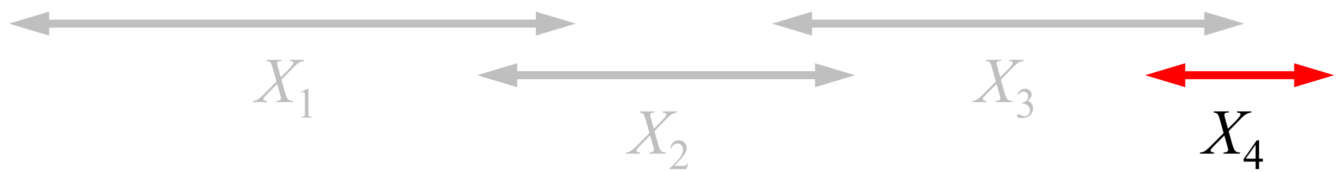
X is $(4, 2)_+$-rollercoaster

1. The last run X_4 is $+$-run.
2. $|X_1|, |X_2|, |X_3| \geq 4$
3. $h = 2$ and $|X_4| = 2$

1. The last run X_x is $+$-run
2. $|X_i| \geq k$ for $i \in [1, x-1]$
3. If $h \in [1, k-1]$, $|X_i| = h$, and $|X_i| \geq k$ otherwise

Example of $(k, h)_+$-rollercoaster

$$X = 9 \quad 8 \quad 6 \quad 4 \quad 2 \quad 1 \quad 3 \quad 4 \quad 8 \quad 5 \quad 4 \quad 2 \quad 1 \quad 3$$



X is $(4, 2)_+$-rollercoaster

1. The last run X_4 is $+$-run
2. $|X_1|, |X_2|, |X_3| \geq 4$
3. $\boldsymbol{h = 2}$ **and** $\boldsymbol{|X_4| = 2}$

1. The last run X_x is $+$-run
2. $|X_i| \geq k$ for $i \in [1, x-1]$
3. **If** $\boldsymbol{h \in [1, k-1], |X_i| = h}$, and $|X_i| \geq k$ otherwise

Example of $(k, h)_-$-rollercoaster

$$Y = 9\ 8\ 6\ 4\ 2\ 1\ 3\ 4\ 8\ 5\ 4\ 2\ 1$$

Y_1 Y_2 Y_3

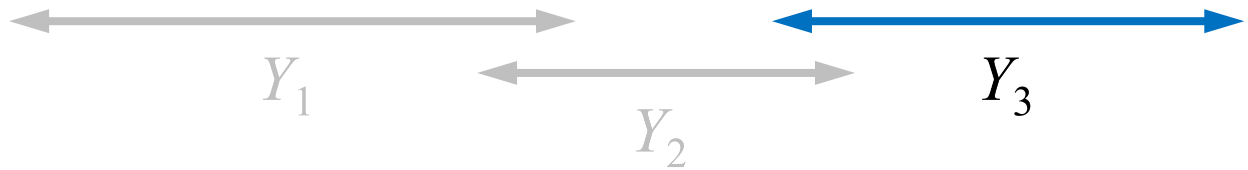
Y is $(4, 4)_-$-rollercoaster

1. The last run Y_3 is $-$-run
2. $|Y_1|, |Y_2| \geq 4$
3. $h = 4$ and $|Y_3| \geq 4$

1. The last run X_x is $-$-run
2. $|X_i| \geq k$ for $i \in [1, x-1]$
3. If $h \in [1, k-1]$, $|X_i| = h$, and $|X_i| \geq k$ otherwise

Example of $(k, h)_-$-rollercoaster

$$Y = 9\ \ 8\ \ 6\ \ 4\ \ 2\ \ 1\ \ 3\ \ 4\ \ 8\ \ 5\ \ 4\ \ 2\ \ 1$$



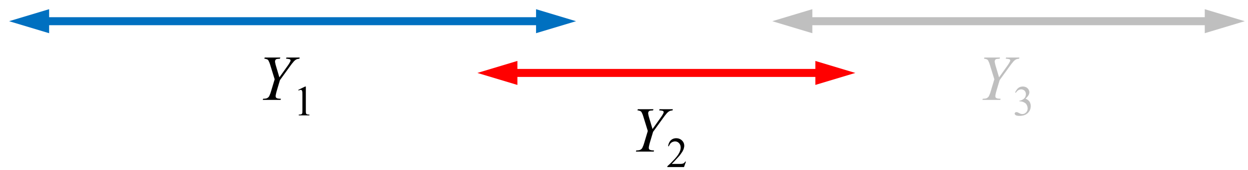
Y is $(4, 4)_-$-rollercoaster

1. **The last run Y_3 is $-$-run**
2. $|Y_1|, |Y_2| \geq 4$
3. $h = 4$ and $|Y_3| \geq 4$

1. **The last run X_x is $-$-run**
2. $|X_i| \geq k$ for $i \in [1, x-1]$
3. If $h \in [1, k-1]$, $|X_i| = h$, and $|X_i| \geq k$ otherwise

Example of (k, h)_-rollercoaster

$$Y = 9\ 8\ 6\ 4\ 2\ 1\ 3\ 4\ 8\ 5\ 4\ 2\ 1$$



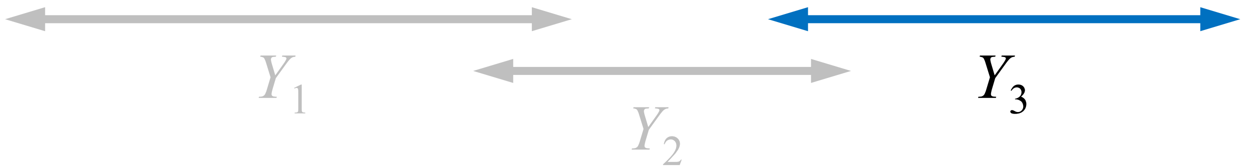
Y is (4, 4)_-rollercoaster

1. The last run Y_3 is $-$-run
2. **$|Y_1|, |Y_2| \geq 4$**
3. $h = 4$ and $|Y_3| \geq 4$

1. The last run X_x is $-$-run
2. **$|X_i| \geq k$ for $i \in [1, x - 1]$**
3. If $h \in [1, k - 1]$, $|X_i| = h$, and $|X_i| \geq k$ otherwise

Example of (k, h)_-rollercoaster

$Y = 9\ 8\ 6\ 4\ 2\ 1\ 3\ 4\ 8\ 5\ 4\ 2\ 1$



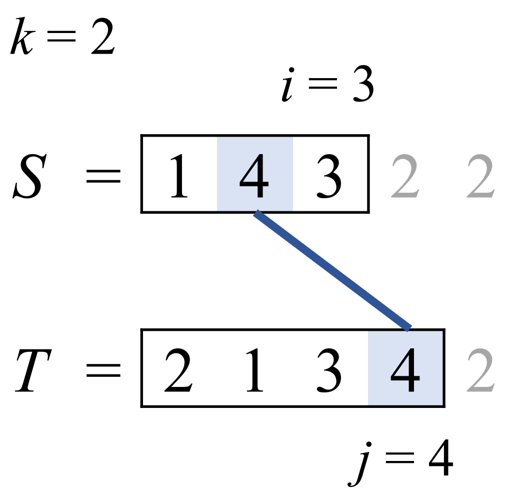
Y is $(4, 4)$_-rollercoaster

1. The last run Y_3 is $-$-run
2. $|Y_1|, |Y_2| \geq 4$
3. **$h = 4$ and $|Y_3| \geq 4$**

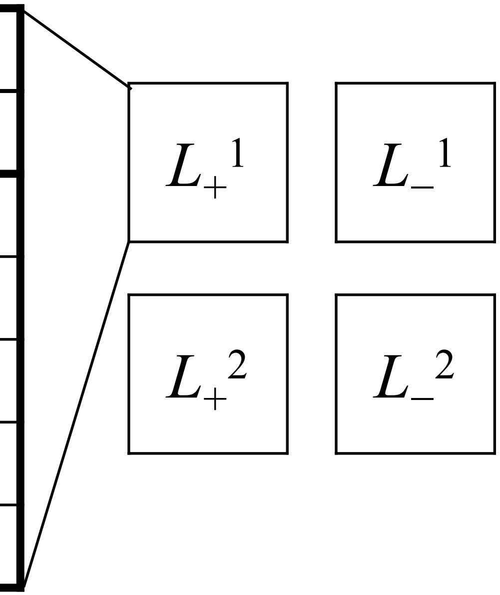
1. The last run X_x is $-$-run
2. $|X_i| \geq k$ for $i \in [1, x-1]$
3. If $h \in [1, k-1]$, $|X_i| = h$, and **$|X_i| \geq k$ otherwise**

Dynamic Programming

For $w \in \{+, -\}$, $h \in [1, k]$, $i \in [1, n]$, $j \in [1, m]$,
$L_w{}^h[i, j]$ = the length of longest common $(k, h)_w$-rollercoaster subsequence of $S[1..i]$ and $T[1..j]$ that ends with $T[j]$.

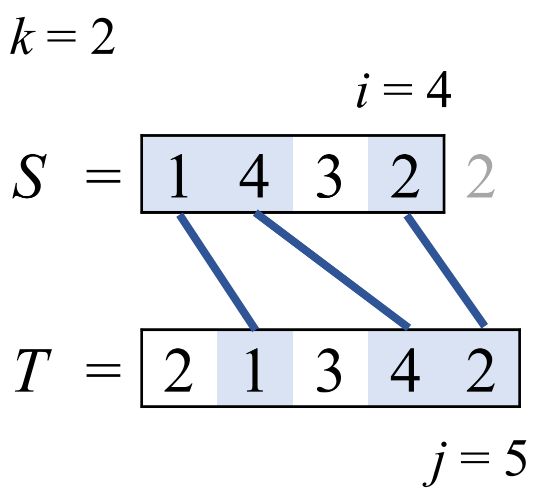


$L_+{}^1$	j=1	j=2	j=3	j=4	j=5
i=1	0	1	0	0	0
i=2	0	1	0	1	0
i=3	0	1	1	1	0
i=4	1	1	1	1	1
i=5	1	1	1	1	1

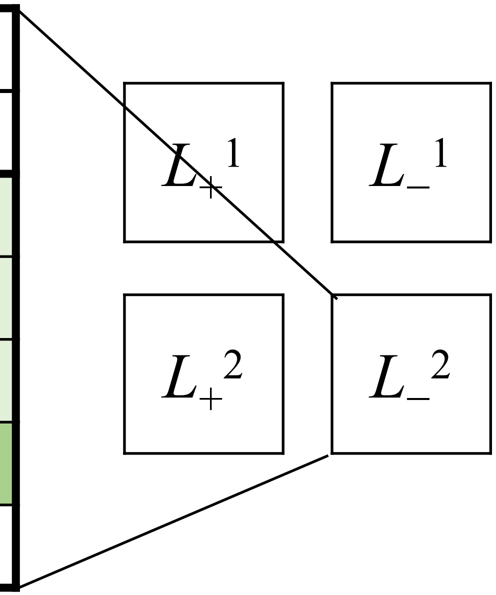


Dynamic Programming

For $w \in \{+, -\}$, $h \in [1, k]$, $i \in [1, n]$, $j \in [1, m]$,
$L_w{}^h[i, j]$ = the length of longest common $(k, h)_w$-rollercoaster subsequence of $S[1..i]$ and $T[1..j]$ that ends with $T[j]$.



L_-^2		*j* = 1	2	3	4	**5**
i	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	**3**
	5	0	0	0	0	3

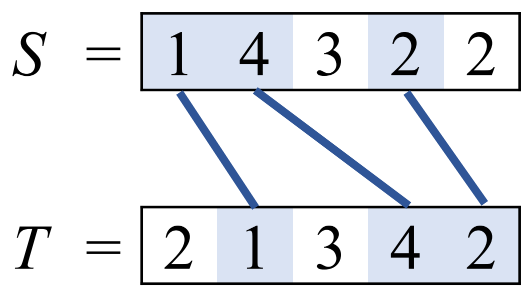


Dynamic Programming

For $w \in \{+, -\}$, $h \in [1, k]$, $i \in [1, n]$, $j \in [1, m]$,
$L_w^h[i, j]$ = the length of longest common $(k, h)_w$-rollercoaster of $S[1..i]$ and $T[1..j]$ that ends with $T[j]$.

The length of longest common k-rollercoaster of S and T is

$$\max\{L_w^k[n, j] \mid w \in \{+, -\}, j \in [1, m]\}$$



L_+^2		j=1	2	3	4	5
i	1	0	0	0	0	0
	2	0	0	0	2	0
	3	0	0	2	2	0
	4	0	0	2	2	2
	5	0	0	2	2	2

L_-^2		j=1	2	3	4	5
i	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	3
	5	0	0	0	0	3

Recurrence for $L_w^h[i, j]$

Consider the case for $w = +$.

The case for $w = -$ can be shown in a symmetric fashion.

Consider the following cases.

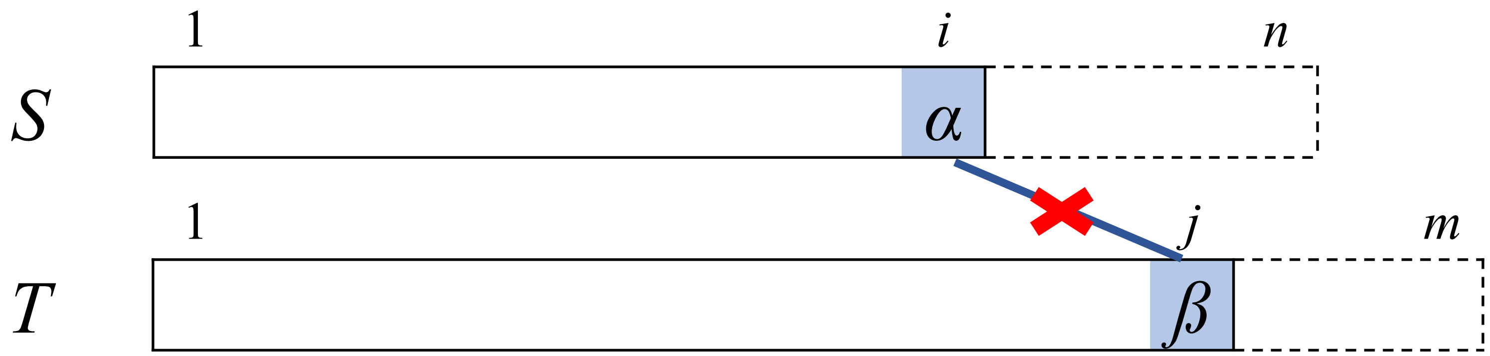
1. $S[i] \neq T[j]$
2. $S[i] = T[j]$ and $h = 1$
3. $S[i] = T[j]$ and $h \in [2, k-1]$
4. $S[i] = T[j]$ and $h = k$

1. $S[i] \neq T[j]$

For any $h \in [1, k]$

$$L_+{}^h[i, j] = L_+{}^h[i-1, j]$$

$L_+{}^h[i, j]$ = the length of longest common $(k, h)_+$-rollercoaster of $S[1..i]$ and $T[1..j]$ that ends with $T[j]$.

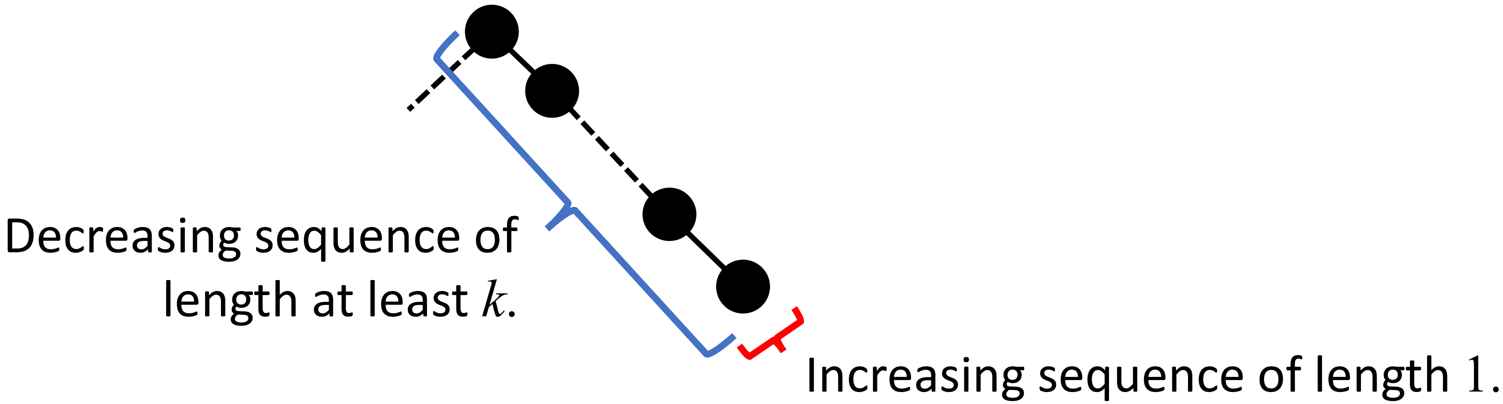


2. $S[i] = T[j]$ and $h = 1$

$$L_+^1[i,j] = \begin{cases} L_-^k[i,j] & \text{if } L_-^k[i,j] \neq 0, \\ 1 & \text{otherwise.} \end{cases}$$

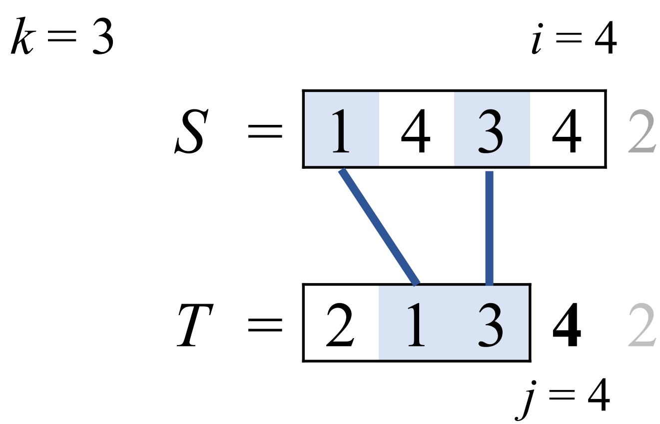
Any $(k,\ 1)_+$-rollercoaster is either

- a $(k,\ k)_-$-rollercoaster subsequence, or
- a sequence of length 1.



3. $S[i] = T[j]$ and $h \in [2, k-1]$

$M_+^h[i, j]$ = the length of longest common $(k, h)_+$-rollercoaster of $S[1..i]$ and $T[1..j-1]$ that ends with an element which is less than $T[j]$



M_+^2		j				
		1	2	3	**4**	5
i	1	0	0	1	1	1
	2	0	0	1	1	1
	3	0	0	1	2	1
	4	0	0	1	**2**	1
	5	0	0	1	2	1

$$L_+^h[i, j] = \begin{cases} M_+^{h-1}[i, j] + 1 & \text{if } M_+^{h-1}[i, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm for Case 3

$M_+^{h\text{-}1}[i, j]$ = the length of longest common $(k, h - 1)_+$-rollercoaster of $S[1..i]$ and $T[1..j - 1]$ that ends with an element that is less than $T[j]$

$$L_+^h[i, j] = \begin{cases} M_+^{h\text{-}1}[i - 1, j] + 1 & \text{if } M_+^{h\text{-}1}[i - 1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

L_+^{h-1}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
$i - 1$	0				5		3			7			...	
⋮														

$S[i] > T[j]$

L_+^h

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i	0			?					?			?	...	
⋮														

$S[i] = T[j]$

Algorithm for Case 3

$$M_+^{h-1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^{h}[i, j] = \begin{cases} M_+^{h-1}[i-1, j] + 1 & \text{if } M_+^{h-1}[i-1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

L_+^{h-1}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i - 1	0				5		3			7			...	
⋮														

$S[i] > T[j]$

L_+^{h}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i	0			?					?			?	...	
⋮														

$S[i] = T[j]$

Algorithm for Case 3

$$M_+^{h\text{-}1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^{h}[i, j] = \begin{cases} M_+^{h\text{-}1}[i - 1, j] + 1 & \text{if } M_+^{h\text{-}1}[i - 1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

L_+^{h-1}

	0	1	2	3	4	5	6	7	8	9	10	11	…	m
⋮														
i - 1	0				5		3			7			…	
⋮														

$S[i] > T[j]$

L_+^{h}

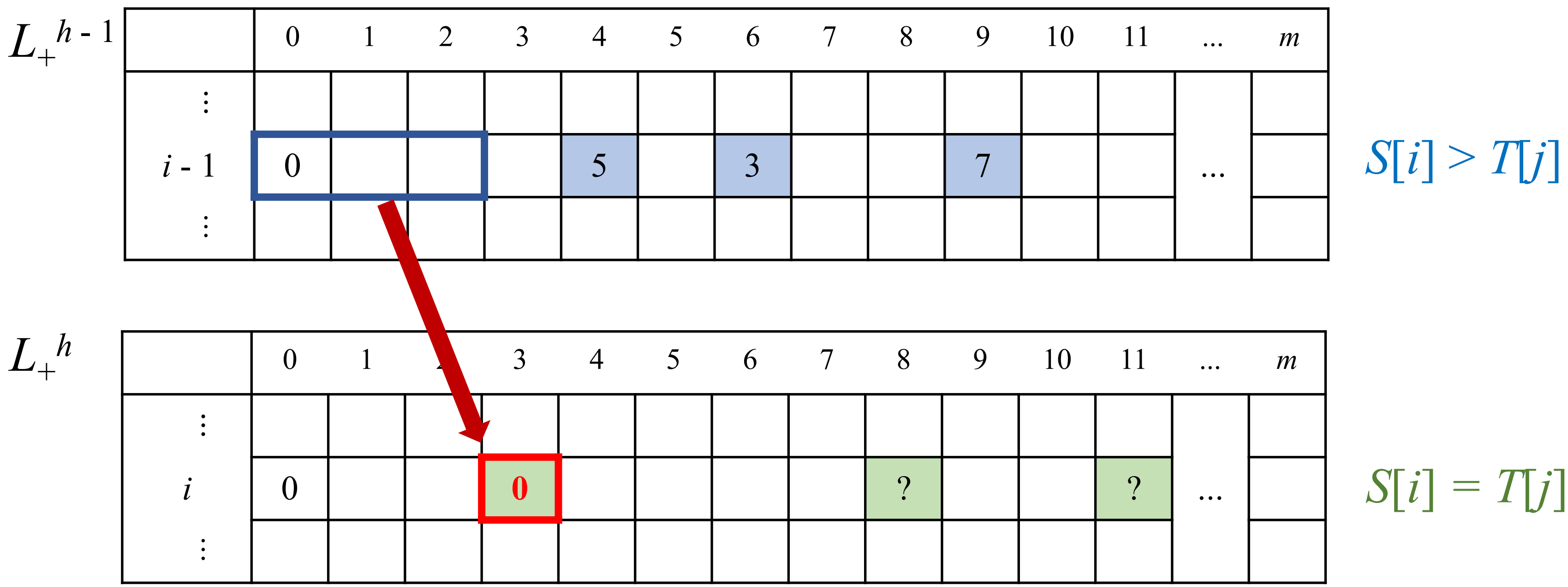
	0	1	2	3	4	5	6	7	8	9	10	11	…	m
⋮														
i	0			?					?			?	…	
⋮														

$S[i] = T[j]$

Algorithm for Case 3

$$M_+^{h-1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^{h}[i, j] = \begin{cases} M_+^{h-1}[i-1, j] + 1 & \text{if } M_+^{h-1}[i-1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$



$$M_+^{h-1}[i, 3] = 0$$

Algorithm for Case 3

$$M_+^{h\text{-}1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \dots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \dots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^{h}[i, j] = \begin{cases} M_+^{h\text{-}1}[i - 1, j] + 1 & \text{if } M_+^{h\text{-}1}[i - 1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

$L_+^{h - 1}$

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i - 1	0				5		3			7			...	
⋮														

$S[i] > T[j]$

L_+^{h}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i	0			0					?			?	...	
⋮														

$S[i] = T[j]$

Algorithm for Case 3

$$M_+^{h\text{-}1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^{h}[i, j] = \begin{cases} M_+^{h\text{-}1}[i - 1, j] + 1 & \text{if } M_+^{h\text{-}1}[i - 1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

L_+^{h-1}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i - 1	0				5		3			7			...	
⋮														

$S[i] > T[j]$

L_+^{h}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i	0			0					?			?	...	
⋮														

$S[i] = T[j]$

Algorithm for Case 3

$$M_+^{h-1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^{h}[i, j] = \begin{cases} M_+^{h-1}[i-1, j] + 1 & \text{if } M_+^{h-1}[i-1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

L_+^{h-1}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i - 1	0				5		3			7			...	
⋮														

$S[i] > T[j]$

L_+^{h}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i	0			0					6			?	...	
⋮														

$S[i] = T[j]$

$$M_+^{h-1}[i, 8] = 5$$

Algorithm for Case 3

$$M_+^{h-1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^h[i, j] = \begin{cases} M_+^{h-1}[i-1, j] + 1 & \text{if } M_+^{h-1}[i-1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

L_+^{h-1}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i - 1	0				5		3			7			...	
⋮														

$S[i] > T[j]$

L_+^h

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i	0			0					6			?	...	
⋮														

$S[i] = T[j]$

Algorithm for Case 3

$$M_+^{h\text{-}1}[i, j] = \begin{cases} \text{maximum of the blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \\ 0 \text{ if there are no blue cells in row } i-1 \text{ and columns } 1, 2, \ldots, j-1 \text{ in } L_+^{h-1} \end{cases}$$

$$L_+^{h}[i, j] = \begin{cases} M_+^{h\text{-}1}[i-1, j] + 1 & \text{if } M_+^{h\text{-}1}[i-1, j] \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

L_+^{h-1}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i - 1	0				5		3			7			...	
⋮														

$S[i] > T[j]$

L_+^{h}

	0	1	2	3	4	5	6	7	8	9	10	11	...	m
⋮														
i	0			0					6			8	...	
⋮														

$S[i] = T[j]$

$$M_+^{h\text{-}1}[i, 11] = 7$$

4. $S[i] = T[j]$ and $h = k$

In a similar way to Case 3, we obtain the following:

$$L_+^k[i, j] = \begin{cases} \max\{M_+^{k-1}[i, j], M_+^k[i, j]\} + 1 & \\ & \text{if } \max\{M_+^{k-1}[i, j], M_+^k[i, j]\} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Recurrence for $L_+^h[i, j]$

When $h = 1$,

$$L_+^1[i,j] = \begin{cases} L_-^k[i,j] & \text{if } S[i] = T[j] \text{ and } L_-^k[i,j] \neq 0, \\ 1 & \text{if } S[i] = T[j] \text{ and } L_-^k[i,j] = 0, \\ L_+^1[i-1,j] & \text{otherwise.} \end{cases}$$

When $2 \le h \le k - 1$,

$$L_+^h[i,j] = \begin{cases} M_+^{h-1}[i,j] + 1 & \text{if } S[i] = T[j] \text{ and } M_+^{h-1}[i,j] \neq 0, \\ 0 & \text{if } S[i] = T[j] \text{ and } M_+^{h-1}[i,j] = 0, \\ L_+^h[i-1,j] & \text{otherwise.} \end{cases}$$

When $h = k$,

$$L_+^k[i,j] = \begin{cases} \max\{M_+^{k-1}[i,j], M_+^k[i,j]\} + 1 & \\ & \text{if } S[i] = T[j] \text{ and } \max\{M_+^{k-1}[i,j], M_+^k[i,j]\} \neq 0, \\ 0 & \text{if } S[i] = T[j] \text{ and } \max\{M_+^{k-1}[i,j], M_+^k[i,j]\} = 0, \\ L_+^k[i-1,j] & \text{otherwise.} \end{cases}$$

Retrieve

$k = 3$

$S = $ | 8 | 4 | 6 | 2 | 7 |

$T = $ | 1 | 8 | 6 | 7 | 2 |

Calculate $L_+^1[5, 5]$

$S[5] \neq T[5]$, so we adapt case 1

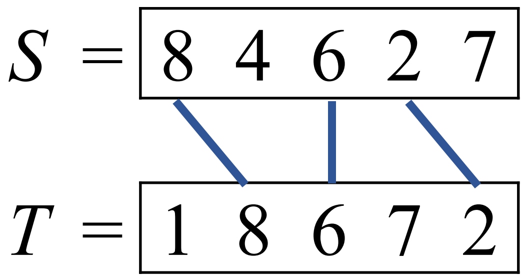
$L_+^1[5, 5] = L_+^1[4, 5] = 3$

L_+^1		j				
		1	2	3	4	5
i	1	0	1	0	0	0
	2	0	1	0	0	0
	3	0	1	1	0	0
	4	0	1	1	0	3
	5	0	1	1	1	**3**

L_-^3		j				
		1	2	3	4	5
i	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	3
	5	0	0	0	0	3

Retrieve

$k = 3$



Calculate $L_+^1[5, 5]$

$S[5] \neq T[5]$, so we adapt case 1

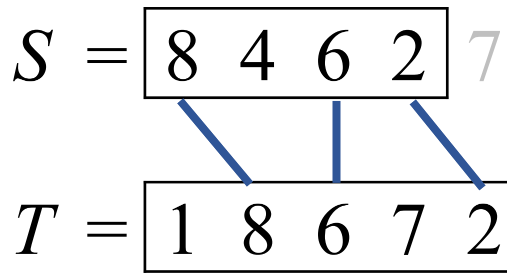
$L_+^1[5, 5] = L_+^1[4, 5] = 3$

L_+^1		j = 1	2	3	4	5
i	1	0	1	0	0	0
	2	0	1	0	0	0
	3	0	1	1	0	0
	4	0	1	1	0	3
	5	0	1	1	1	**3**

L_-^3		j = 1	2	3	4	5
i	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	3
	5	0	0	0	0	3

Retrieve

$k = 3$



Calculate $L_+^1[4, 5]$

$S[5] = T[5]$, so we adapt case 2

$L_+^1[4, 5] = L_-^3[4, 5] = 3$

L_+^1		j				
		1	2	3	4	5
	1	0	1	0	0	0
	2	0	1	0	0	0
i	3	0	1	1	0	0
	4	0	1	1	0	**3**
	5	0	1	1	1	3

L_-^3		j				
		1	2	3	4	5
	1	0	0	0	0	0
	2	0	0	0	0	0
i	3	0	0	0	0	0
	4	0	0	0	0	3
	5	0	0	0	0	3

Complexity of our algorithm

1. Initialize $L_+^1, \ldots, L_+^k, L_-^1, \ldots, L_-^k$ to 0

   $O(nmk)$ time and space

2. For $i = 1, \ldots, n$,
   a. For $h = 2, \ldots, k$, compute L_+^h, L_-^h
   b. Compute L_+^k, L_-^k
   c. Compute L_+^1, L_-^1

   $O(nmk)$ time

3. Compute $\max\{L_w^k[n, j] \mid w \in \{+, -\}, j \in [1, m]\}$

   $O(m)$ time

4. Retrieve longest common k-rollercoaster

   $O(m)$ time and space

Conclusions

Theorem 1.

A longest common k-rollercoaster of S and T can be computed in $O(nmk)$ time and space.

Theorem 2.

A longest common k-rollercoaster of S and T can be computed in $O(rk \log^3 m \log \log m)$ time and O(rk) space.

r : the number of pairs (i, j) s.t. $S[i] = T[j]$