SEQ-IC-LCS Computation of Labeled Graphs

Yuki Yonemoto, Yuto Nakashima
and Shunsuke Inenaga

Kyushu Univ.

Outline

- Labeled Graphs
- SEQ-IC-LCS (Constrained LCS)
- Computing SEQ-IC-LCS of Acyclic Labeled Graphs
- Computing SEQ-IC-LCS of Cyclic Labeled Graphs
- Conclusions and Future works

Outline

- Labeled Graphs
- SEQ-IC-LCS (Constrained LCS)
- Computing SEQ-IC-LCS of Acyclic Labeled Graphs
- Computing SEQ-IC-LCS of Cyclic Labeled Graphs
- Conclusions and Future works

Labeled Graphs

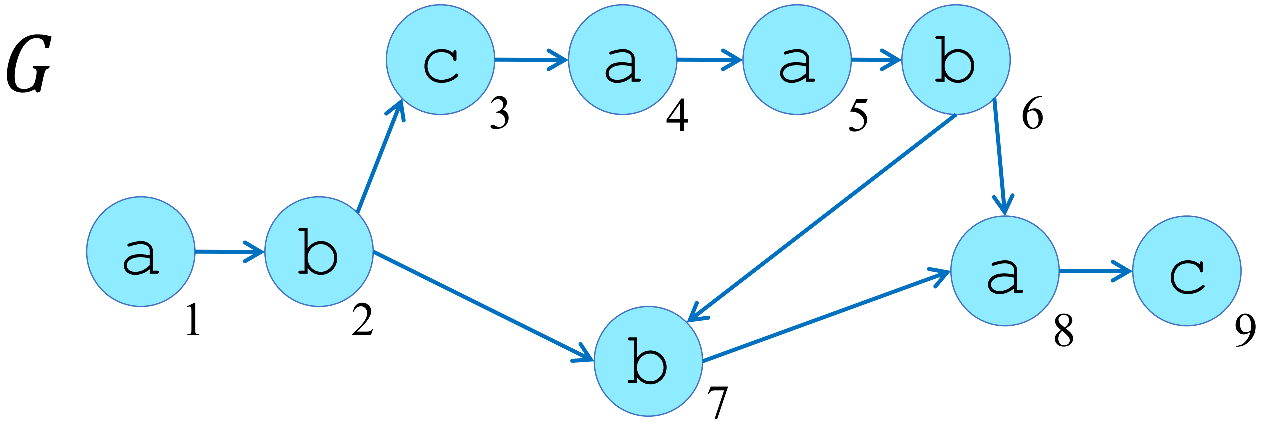
Labeled Graph $G = (V, E, L)$

A directed graph with vertices labeled by characters.

V : the set of vertices
E : the set of edges
$L : V \rightarrow \Sigma$: a labeling function

e.g. $V = \{ v_1, c_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9 \}$

$E = \{ (v_1, v_2), (v_2, v_3), (v_2, v_7), (v_3, v_4), (v_4, v_5),$
$(v_5, v_6), (v_6, v_7), (v_6, v_8), (v_7, v_8), (v_8, v_9) \}$



Labeled Graphs

$L(v)$: the character label of vertex v .

$P(v)$: the set of paths that end at vertex v.

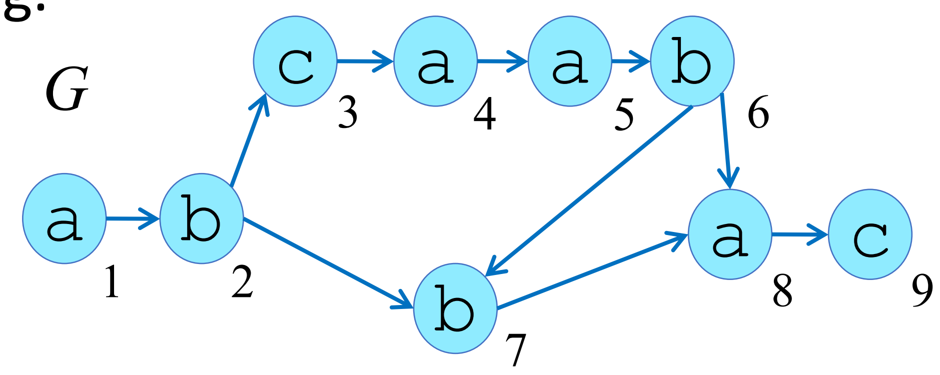
$L(P(v))$: the set of strings spelled by paths in $P(v)$.

$P(G)$: the set of paths in G. $\quad (P(G) = \{P(v) \mid v \in V \})$

$L(\pi)$: the set of strings spelled by paths in π (: the set of paths) .

$subseq(L(\pi))$: the set of subsequences of strings in $L(\pi)$.

e.g.



$L(v_7) = \mathtt{b}$

$P(v_7) = \{v_3v_4v_5v_6v_7 \text{ , } v_1v_2v_7 \text{ , } \ldots \}$

$L(P(v_7)) = \{\mathtt{caabb} \text{ , } \mathtt{abb}, \ldots \}$

$aca \in subseq(L(P(G)))$

Known Algorithms on Labeled Graphs

problem	text	pattern	time complexity
Pattern Matching	acyclic graph	string	$O(n+m\|E\|)$ [Park & Kim, 1995]
	tree	string	$O(n)$ [Akutsu, 1993]
	graph	string	$O(n+m\|E\|)$ [Amir et al, 1997]
Approximate Matching	graph with edit operations	string	NP-complete [Amir et al, 1997]
	graph	string with edit operations	$O(m(n+\|E\|))$ [Navarro, 2000]

n : sum of the length of strings in the text, m : length of the pattern.

problem	text 1	text 2	time complexity
Longest Common Substring	acyclic graph	acyclic graph	$O(\|E_1\|\|E_2\|)$ [Shimohira et al., 2011]
	graph	acyclic graph	
Longest Common Subsequence	acyclic graph	acyclic graph	$O(\|E_1\|\|E_2\|)$ [Shimohira et al., 2011]
	graph	graph	$O(\|E_1\|\|E_2\|+\|V_1\|\|V_2\|\log\|\Sigma\|)$ [Shimohira et al., 2011]

$|E_i|$: the number of edges in text i, $|V_i|$: the number of vertices in text i, $|\Sigma|$: the alphabet size.

Outline

- Labeled Graphs
- SEQ-IC-LCS (Constrained LCS)
- Computing SEQ-IC-LCS of Acyclic Labeled Graphs
- Computing SEQ-IC-LCS of Cyclic Labeled Graphs
- Conclusions and Future works

Constrained LCS [Chen et al., 2011]

In the field of molecular biology,
there are cases where the same sequence appears between different species, and there are demand to incorporate this into similarity measurements.

In recent years,
Constrained LCS problems for string inputs
derived from the LCS problem are considered.

Constrained LCS [Chen et al., 2011]

There exist four variants of the Constrained LCS problems.

Each of them is to compute a longest string Z such that Z includes/excludes the constraint pattern P as a substring/subsequence and Z is a common subsequence of the two target strings A and B.

Each problem is called,

STR-IC-LCS	（substring, include）
STR-EC-LCS	（substring, exclude）
SEQ-IC-LCS	（subsequence, include）
SEQ-EC-LCS	（subsequence, exclude）

Previous Work of Constrained LCS and Our Work

problem	text 1	text 2	text 3	time complexity
STR-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|)$ [Deorowicz, 2012]
STR-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Wang et al., 2013]
SEQ-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chin et al., 2004]
SEQ-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chen and Chao, 2011]

$|E_i|$: the number of edges in text i , $|V_i|$: the number of vertices in text i ,
$|\Sigma|$: the alphabet size .

Previous Work of Constrained LCS and Our Work

problem	text 1	text 2	text 3	time complexity
STR-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|)$ [Deorowicz, 2012]
STR-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Wang et al., 2013]
SEQ-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chin et al., 2004]
	acyclic graph	acyclic graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|)$ (this work)
	graph	graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|+\|V_1\|\|V_2\|\|V_3\|\log\|\Sigma\|)$ (this work)
SEQ-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chen and Chao, 2011]

$|E_i|$: the number of edges in text i , $|V_i|$: the number of vertices in text i ,
$|\Sigma|$: the alphabet size .

SEQ-IC-LCS

SEQ-IC-LCS of strings A, B and P is a longest string Z
such that Z includes P as a subsequence
and Z is a common subsequence of A and B.

SEQ-IC-LCS

SEQ-IC-LCS of strings A, B and P is a longest string Z such that Z includes P as a subsequence and Z is a common subsequence of A and B.

e.g. $A =$ b c d a b a b

$B =$ c b a c b a a b a

$P =$ c a a

c a b a b is an SEQ-IC-LCS of string A, B and P.

SEQ-IC-LCS

SEQ-IC-LCS of strings A, B and P is a longest string Z such that Z includes P as a subsequence and Z is a common subsequence of A and B.

e.g. $A =$ b c d a b a b

$B =$ c b a c b a a b a

$P =$ c a a

c a b a b is an SEQ-IC-LCS of string A, B and P.

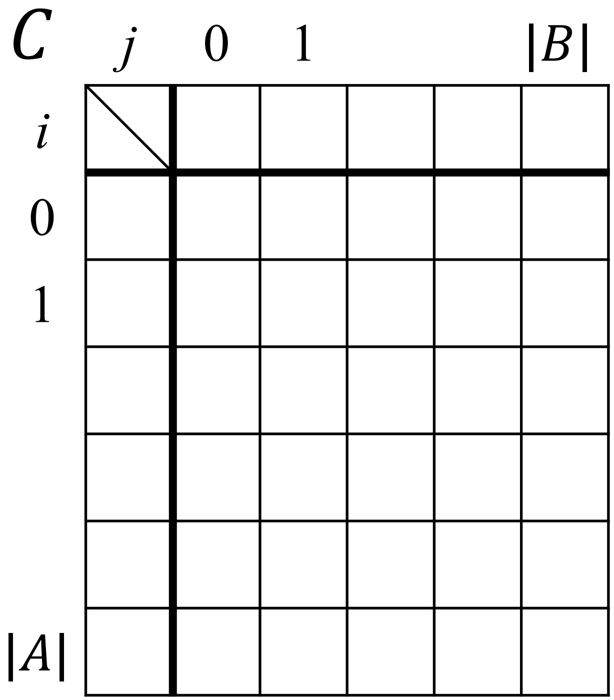
Algorithm for SEQ-IC-LCS of strings [Chin et al., 2004]

Previous work of the SEQ-IC-LCS problem for string inputs is based on dynamic programming.

Let C denote the three-dimensional table which stored the length of the SEQ-IC-LCS of $A[1..i], B[1..j]$ and $P[1..k]$ in $C(i,j,k)$ for any $0 \leq i \leq |A|, 0 \leq j \leq |B|, 0 \leq k \leq |P|$.

computing all $C(i,j,k)$ by using the recurrence.

$C(|A|,|B|,|P|)$ is the solution.



× $(|P| + 1)$ tables

Recurrence of SEQ-IC-LCS algorithm of strings [Chin et al., 2004]

$C(i,j,k)$: the length of SEQ-IC-LCS of $A[1..i], B[1..j]$ and $P[1..k]$.

$$C(i,j,k) = \begin{cases} 0 & \text{if } k = 0 \text{ and } (i = 0 \text{ or } j = 0); \\ -\infty & \text{if } k \neq 0 \text{ and } (i = 0 \text{ or } j = 0); \\ C(i-1, j-1, k-1) & \text{if } i,j,k > 0 \text{ and} A[i] = B[j] = P[k]; \\ C(i-1, j-1, k) & \text{if } i,j > 0 \text{ and} A[i] = B[j] \neq P[k]; \\ \max\big(C(i-1,j,k), C(i,j-1,k)\big) & \text{if } i,j > 0 \text{ and} A[i] \neq B[j]; \end{cases}$$

This algorithm computes the solution in $O(|A||B||P|)$ time.

Outline

- Labeled Graphs
- SEQ-IC-LCS (Constrained LCS)
- Computing SEQ-IC-LCS of Acyclic Labeled Graphs
- Computing SEQ-IC-LCS of Cyclic Labeled Graphs
- Conclusions and Future works

Definition about Labeled Graphs

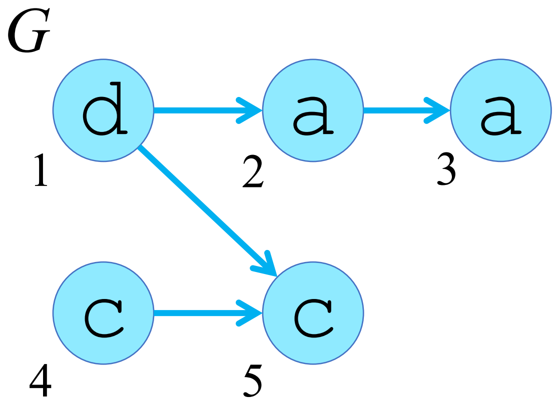
V_s : the set of vertices which has no in-coming edges.

V_e : the set of vertices which has no out-going edges.

$MP(v)$: the set of paths that start at v_s in V_s and end at vertex v .

$MP(G)$: the set of paths that start at v_s in V_s and end at vertex v_e in V_e in G (= maximal paths).

e.g.



$V_s = \{v_1 , v_4\}$ $V_e = \{v_3 , v_5\}$

$MP(v_5) = \{ v_1v_5 , v_4v_5 \}$

$L(MP(v_4)) = \{$ dc, cc $\}$

$MP(G) = \{ v_1v_2v_3 , v_1v_5 , v_4v_5 \}$

SEQ-IC-LCS Problem for Acyclic Labeled Graphs

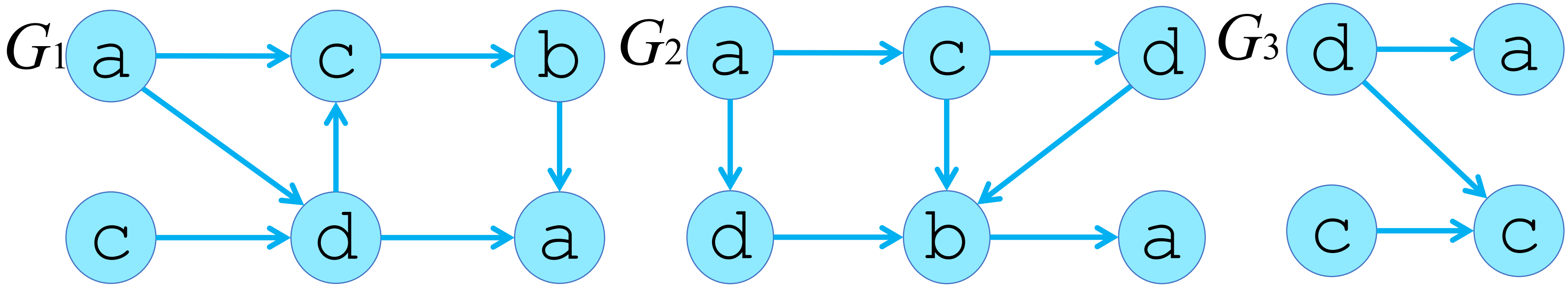
Problem 1

Input : Acyclic labeled graphs $G_1 = (V_1, E_1, L_1)$, $G_2 = (V_2, E_2, L_2)$
and $G_3 = (V_3, E_3, L_3)$

Output : Length of the longest string in the set $\{ z \mid \exists q \in L_3\big(MP(G_3)\big)$
such that $q \in Subseq(z)$ and $z \in Subseq(G_1) \cap Subseq(G_2) \}$

$MP(G)$: the set of maximal paths in G.
$subseq(G)$: the set of subsequences of strings in G .

e.g.



SEQ-IC-LCS Problem for Acyclic Labeled Graphs

Problem 1

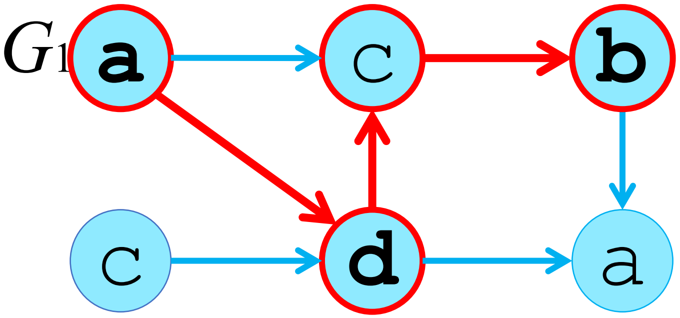
Input : Acyclic labeled graphs $G_1 = (V_1, E_1, L_1)$, $G_2 = (V_2, E_2, L_2)$
and $G_3 = (V_3, E_3, L_3)$

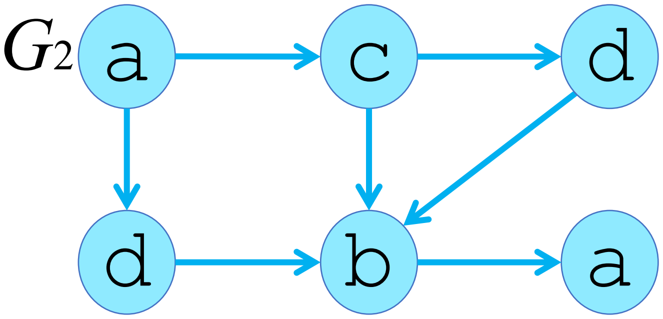
Output : Length of the longest string in the set $\{\, z \mid \exists q \in L_3(MP(G_3))$
such that $q \in Subseq(z)$ and $z \in Subseq(G_1) \cap Subseq(G_2) \,\}$

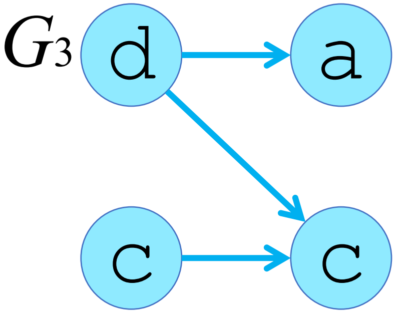
$MP(G)$: the set of maximal paths in G.

$subseq(G)$: the set of subsequences of strings in G .

e.g.







$Subseq(G_1)$ = {a, b, c, d, aa, ab, ac, ad, ba, ca, cb, cc, cd, da, db, dc, aba, aca, acb, ada, **adb**, adc, cba, cca, cda, dba, dca, acba, adba, ccba, cdba, cdca, adcba, cdcba}

$Subseq(G_2)$ = {a, b, c, d, aa, ab, ac, ad, ba, ca, cb, cd, da, db, aba, aca, acb, acd, ada, adb, cba, cda, cdb, dba, acba, adba, cdba, acdba}

$L_3(MP(G_3))$ = {cc, da, dc}

SEQ-IC-LCS Problem for Acyclic Labeled Graphs

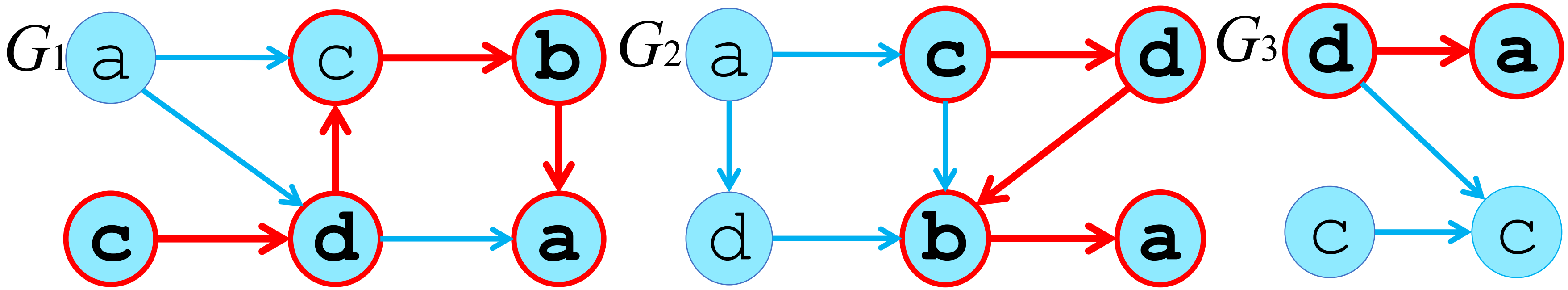
Problem 1

Input : Acyclic labeled graphs $G_1 = (V_1, E_1, L_1)$, $G_2 = (V_2, E_2, L_2)$
and $G_3 = (V_3, E_3, L_3)$

Output : Length of the longest string in the set $\{\, z \mid \exists q \in L_3(MP(G_3))$
such that $q \in \mathit{Subseq}(z)$ and $z \in \mathit{Subseq}(G_1) \cap \mathit{Subseq}(G_2) \,\}$

$MP(G)$: the set of maximal paths in G.
$\mathit{subseq}(G)$: the set of subsequences of strings in G .

e.g.



$\mathit{Subseq}(G_1) = \{$a, b, c, d, aa, ab, ac, ad, ba, ca, cb, cc, cd, da, db, dc, aba, aca, acb, ada, adb, adc, cba, cca, cda, dba, dca, acba, adba, ccba, **cdba**, cdca, adcba, cdcba$\}$

$\mathit{Subseq}(G_2) = \{$a, b, c, d, aa, ab, ac, ad, ba, ca, cb, cd, da, db, aba, aca, acb, acd, ada, adb, cba, cda, cdb, dba, acba, adba, **cdba**, acdba$\}$

$L_3(MP(G_3))$ = $\{$cc, **da**, dc$\}$

The solution is 4. (**cdba**)

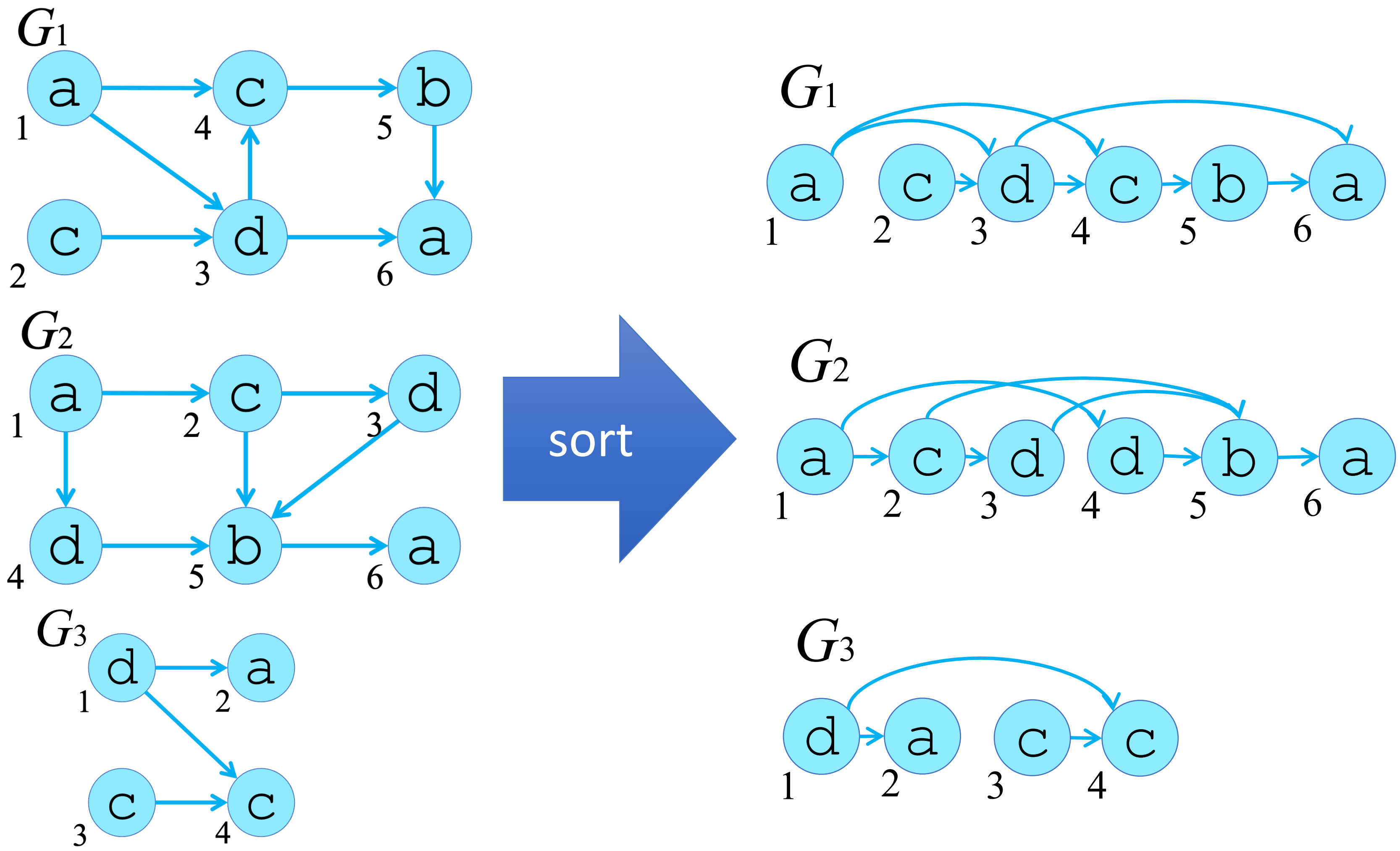
Main Idea of the Algorithm for SEQ-IC-LCS of Acyclic Labeled Graphs

1. Sort vertices of G_1, G_2 and G_3 in topological order.

Topological Sort

1. Sort vertices of G_1, G_2 and G_3 in topological order.

e.g.



Main Idea of the Algorithm for SEQ-IC-LCS of Acyclic Labeled Graphs

1. Sort vertices of G_1, G_2 and G_3 in topological order.

Let C denote the three-dimensional table which stored the length of the SEQ-IC-LCS of $L_1\left(P\left(v_{1,i}\right)\right), L_2\left(P\left(v_{2,j}\right)\right)$ and $L_3\left(P\left(v_{3,k}\right)\right)$ in $C(i,j,k)$ for any $1 \le i \le |V_1|, 1 \le j \le |V_2|, 0 \le k \le |V_3|$.

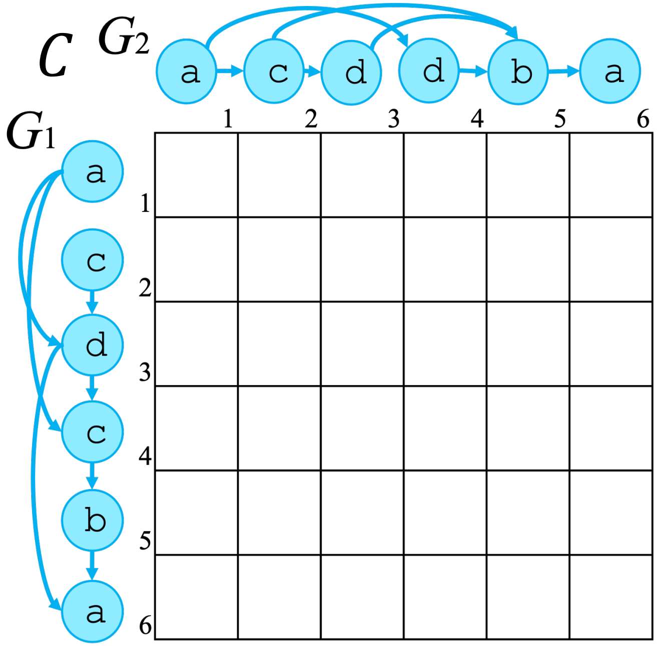
$\left(v_{1,i} \in V_1, v_{2,j} \in V_2, v_{3,k} \in V_3\right)$

2. Calculate $C(i,j,k)$ using the recurrence.

$$\max_{1 \le i \le |V_1|,\, 1 \le j \le |V_2|,\, v_{3,k}} C(i,j,k)$$

is the solution.

($v_{3,k}$ has no out-going edges in V_3.)



×$(|V_3| + 1)$ tables

Recurrence of SEQ-IC-LCS of Acyclic Labeled Graphs

$$
C_{i,j,k} =
$$

$$
\begin{cases}
\text{Recurrence of LCS of Acyclic Labeled Graph} & \text{if } k = 0; \\
\quad \text{[Shimohira et al., 2011]} & \\
1 + \max\left(\left\{ C_{x,y,z} : \left| \begin{array}{l} (v_{1,x}, v_{1,i}) \in E_1, \\ (v_{2,y}, v_{2,j}) \in E_2, \\ (v_{3,z}, v_{3,k}) \in E_3, \\ \text{ or } z = 0 \end{array} \right. \right\} \cup \{\gamma\} \right) & \begin{array}{l} \text{if } k > 0 \text{ and} \\ L_1(v_{1,i}) = L_2(v_{2,j}) \\ = L_3(v_{3,k}); \end{array} \\
\max\left(\left\{ 1 + C_{x,y,k} \left| \begin{array}{l} (v_{1,x}, v_{1,i}) \in E_1, \\ (v_{2,y}, v_{2,j}) \in E_2 \end{array} \right. \right\} \cup \{-\infty\} \right) & \begin{array}{l} \text{if } k > 0 \text{ and} \\ L_1(v_{1,i}) = L_2(v_{2,j}) \\ \neq L_3(v_{3,k}); \end{array} \\
\max\left(\begin{array}{l} \{ C_{x,j,k} \mid (v_{1,x}, v_{1,i}) \in E_1 \} \cup \\ \{ C_{i,y,k} \mid (v_{2,y}, v_{2,j}) \in E_2 \} \cup \{-\infty\} \end{array} \right) & \text{otherwise.}
\end{cases}
$$

where

$$
\gamma = \begin{cases} 0 & \begin{array}{l} \text{if } v_{1,i} \text{ does not have in-coming edges at all or } v_{2,j} \text{ does not have} \\ \text{in-coming edges at all, and } v_{3,k} \text{ does not have in-coming edges;} \end{array} \\ -\infty & \text{otherwise.} \end{cases}
$$

Main Idea of the Algorithm for SEQ-IC-LCS of Acyclic Labeled Graph

1. Sort vertices of G_1, G_2 and G_3 in topological order.

Let C denote the three-dimensional table which stored the length of the SEQ-IC-LCS of $L_1\left(P\left(v_{1,i}\right)\right), L_2\left(P\left(v_{2,j}\right)\right)$ and $L_3\left(P\left(v_{3,k}\right)\right)$ in $C(i,j,k)$ for any $1 \le i \le |V_1|, 1 \le j \le |V_2|, 0 \le k \le |V_3|$.

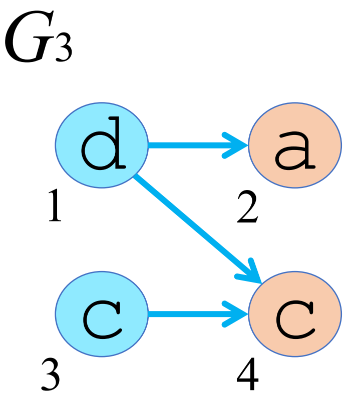
$\left(v_{1,i} \in V_1, v_{2,j} \in V_2, v_{3,k} \in V_3\right)$

2. Calculate $C(i, j, k)$ using the recurrence.

$$\max_{1\le i\le|V_1|,\, 1\le j\le|V_2|,\, v_{3,k}} C(i, j, k)$$

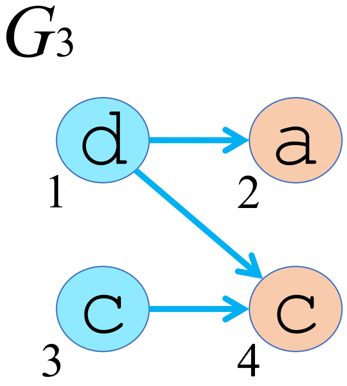
is the solution.

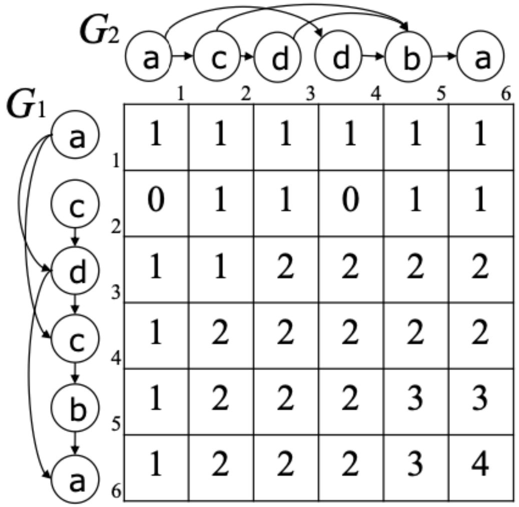
($v_{3,k}$ has no out-going edges in V_3.)



$L_3\left(MP\left(G_3\right)\right)$
$= \{$cc, da, dc$\}$

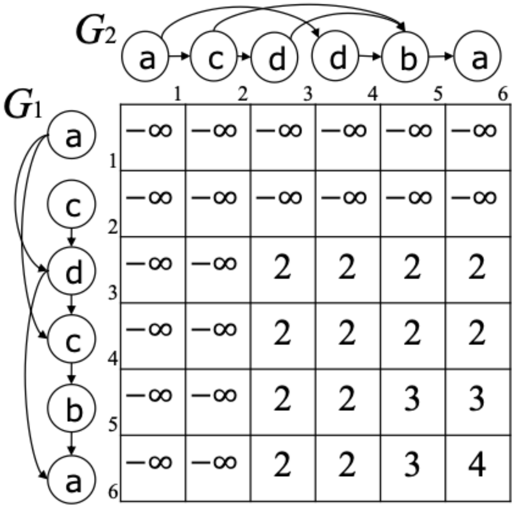
Tables computed by using the recurrence





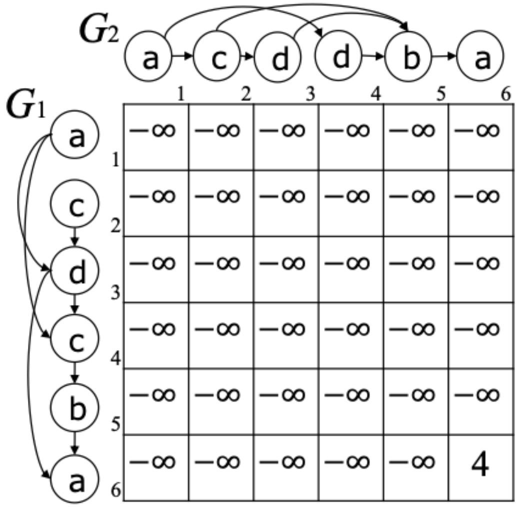
	1	2	3	4	5	6
1	1	1	1	1	1	1
2	0	1	1	0	1	1
3	1	1	2	2	2	2
4	1	2	2	2	2	2
5	1	2	2	2	3	3
6	1	2	2	2	3	4

$k = 0$



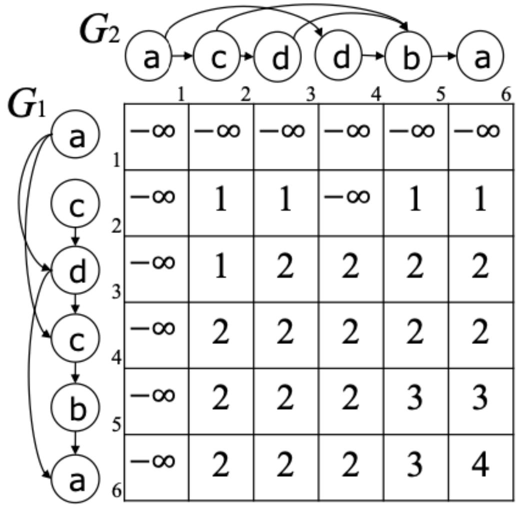
	1	2	3	4	5	6
1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
3	$-\infty$	$-\infty$	2	2	2	2
4	$-\infty$	$-\infty$	2	2	2	2
5	$-\infty$	$-\infty$	2	2	3	3
6	$-\infty$	$-\infty$	2	2	3	4

$k = 1$



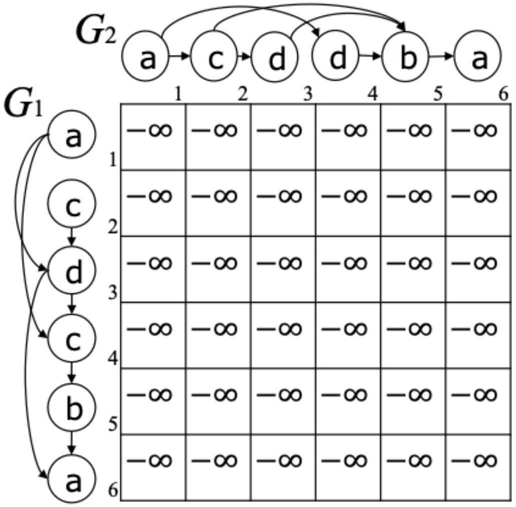
	1	2	3	4	5	6
1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	4

$k = 2$



	1	2	3	4	5	6
1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	$-\infty$	1	1	$-\infty$	1	1
3	$-\infty$	1	2	2	2	2
4	$-\infty$	2	2	2	2	2
5	$-\infty$	2	2	2	3	3
6	$-\infty$	2	2	2	3	4

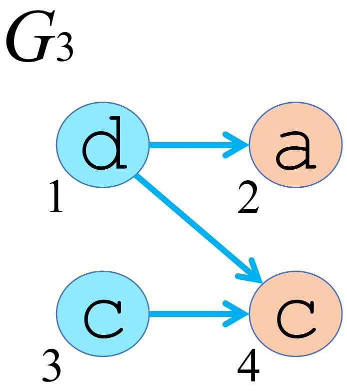
$k = 3$

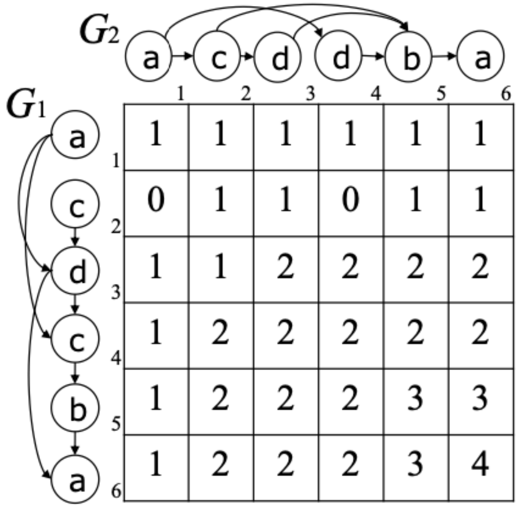


	1	2	3	4	5	6
1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

$k = 4$

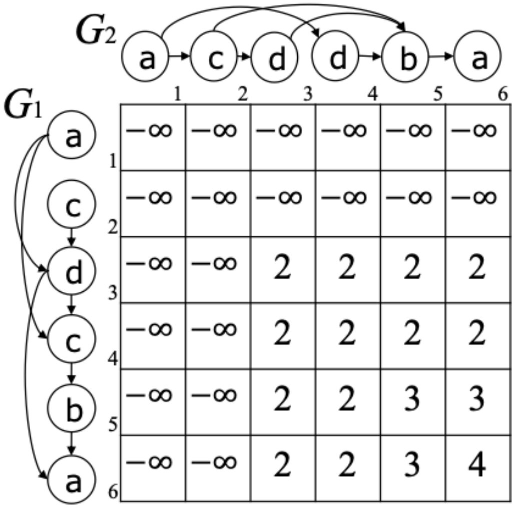
Tables computed by using the recurrence





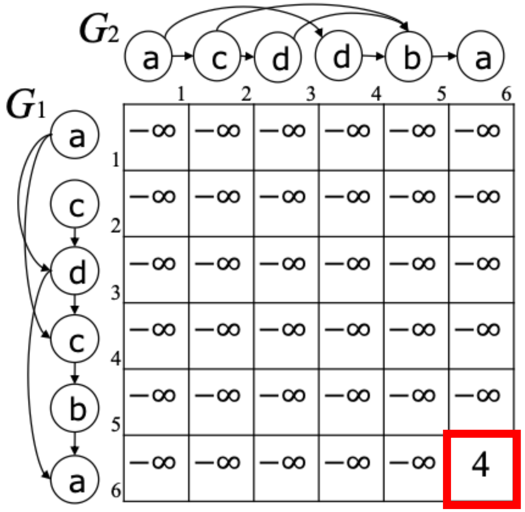
	a 1	c 2	d 3	d 4	b 5	a 6
a 1	1	1	1	1	1	1
c 2	0	1	1	0	1	1
d 3	1	1	2	2	2	2
c 4	1	2	2	2	2	2
b 5	1	2	2	2	3	3
a 6	1	2	2	2	3	4

$k = 0$



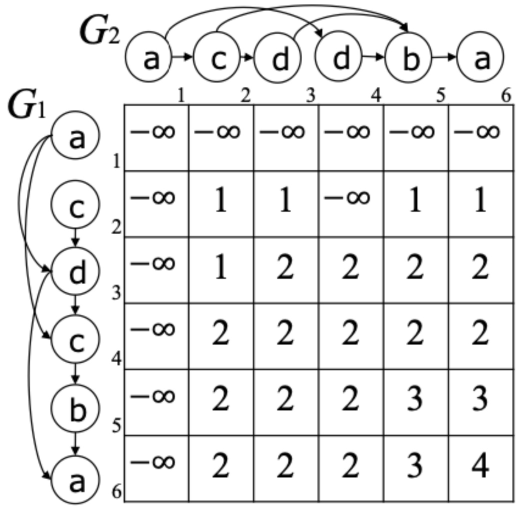
	a 1	c 2	d 3	d 4	b 5	a 6
a 1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
c 2	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
d 3	$-\infty$	$-\infty$	2	2	2	2
c 4	$-\infty$	$-\infty$	2	2	2	2
b 5	$-\infty$	$-\infty$	2	2	3	3
a 6	$-\infty$	$-\infty$	2	2	3	4

$k = 1$



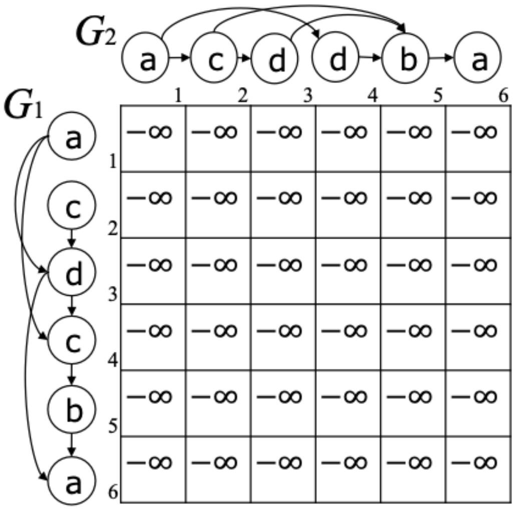
	a 1	c 2	d 3	d 4	b 5	a 6
a 1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
c 2	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
d 3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
c 4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
b 5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
a 6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$\boxed{4}$

$k = 2$



	a 1	c 2	d 3	d 4	b 5	a 6
a 1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
c 2	$-\infty$	1	1	$-\infty$	1	1
d 3	$-\infty$	1	2	2	2	2
c 4	$-\infty$	2	2	2	2	2
b 5	$-\infty$	2	2	2	3	3
a 6	$-\infty$	2	2	2	3	4

$k = 3$



	a 1	c 2	d 3	d 4	b 5	a 6
a 1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
c 2	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
d 3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
c 4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
b 5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
a 6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

$k = 4$

Time Complexity

1. Sort vertices of G_1, G_2 and G_3 in topological order.

Let C denote the three-dimensional ta which stored the length of the SEQ-IC-LCS of $L_1\left(P\left(v_{1,i}\right)\right), L_2$ $)\Big)$ in $C(i,j,k)$ for any $1 \le i \le |V_1|, 1 \le j$

$\left(v_{1,i} \in V_1, v_{2,j} \in V_2, v_{3,k} \in V_3\right)$

linear time

2. Calculate $C(i,j,k)$ using the recurrence.

$O(|E_1||E_2||E_3|)$ time

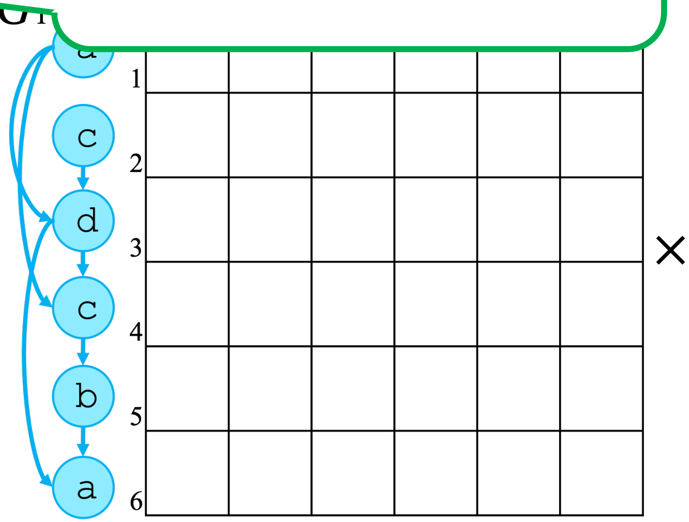
$$\max_{1 \le i \le |V_1|,\, 1 \le j \le |V_2|,\, v_{3,k}} C(i,j,k)$$

is the solution.

($v_{3,k}$ has no out-going edges in V_3.)

C

G_1



$\times(|V_3|+1)$ tables

Outline

- Labeled Graph
- SEQ-IC-LCS (Constrained LCS)
- Computing SEQ-IC-LCS of Acyclic Labeled Graphs
- Computing SEQ-IC-LCS of Cyclic Labeled Graphs
- Conclusions and Future works

SEQ-IC-LCS Problem for Cyclic Labeled Graphs

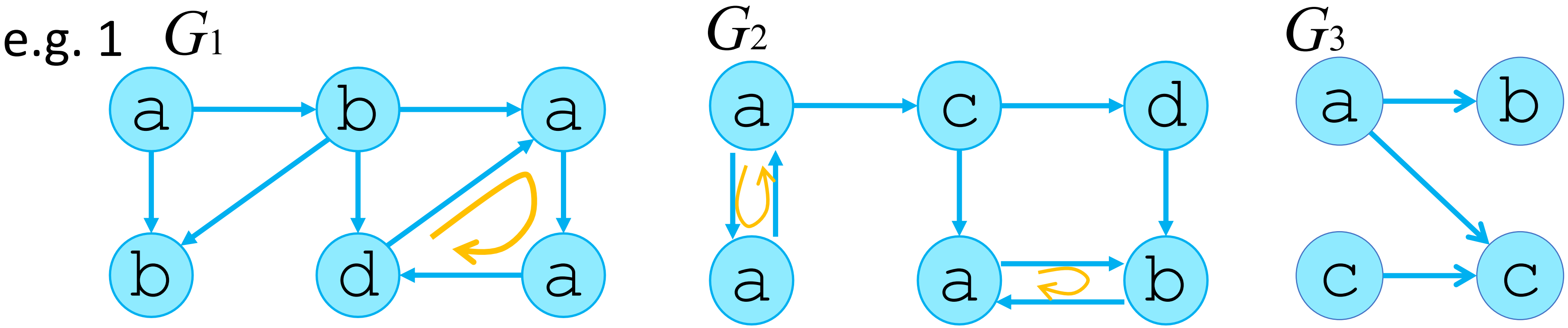
Problem 2

Input : Cyclic labeled graphs $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$, and Acyclic labeled graphs $G_3 = (V_3, E_3, L_3)$

Output : - ∞ (if some z are infinite.)

- Length of longest string in the set $\{ z \mid \exists q \in L_3(MP(G_3))$ such that $q \in Subseq(z)$ and $z \in Subseq(G_1) \cap Subseq(G_2) \}$ (otherwise)

e.g. 1



SEQ-IC-LCS Problem for Cyclic Labeled Graphs

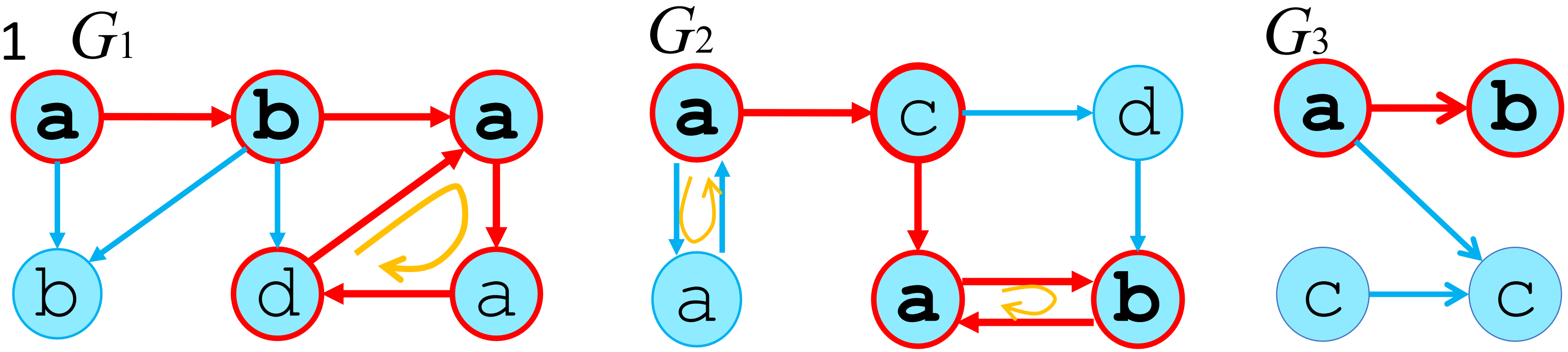
Problem 2

Input : Cyclic labeled graphs $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$, and Acyclic labeled graphs $G_3 = (V_3, E_3, L_3)$

Output : - ∞ (if some z are infinite.)

- Length of longest string in the set $\{ z \mid \exists q \in L_3(MP(G_3))$ such that $q \in Subseq(z)$ and $z \in Subseq(G_1) \cap Subseq(G_2) \}$ (otherwise)

e.g. 1



$Subseq(G_1) \cap Subseq(G_2)$ = {a, b, d, … , **aba**$^\infty$, …}

$L_3(MP(G_3))$ = {**ab**, ac, cc}

The solution is ∞. (**aba**$^\infty$)

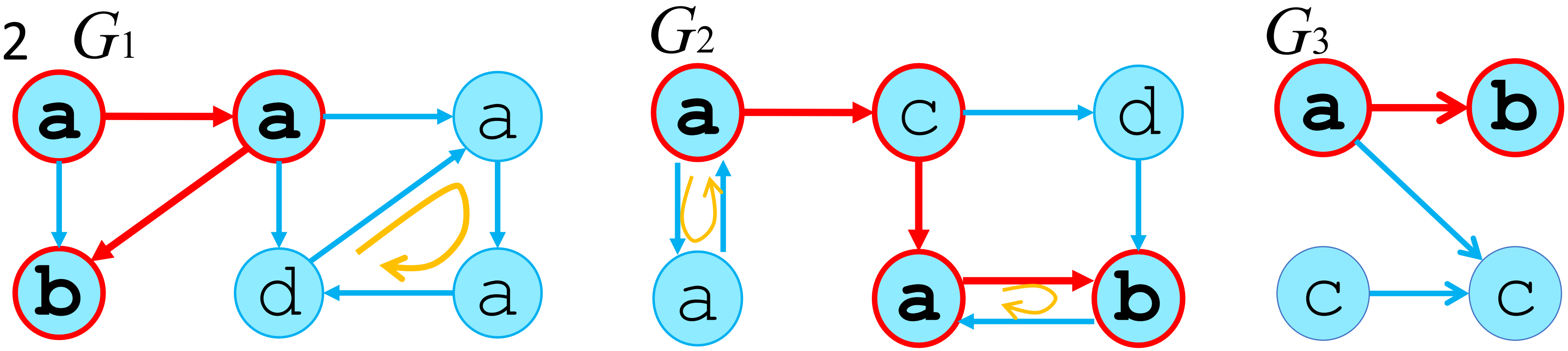
SEQ-IC-LCS Problem for Cyclic Labeled Graphs

Problem 2

Input : Cyclic labeled graphs $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$,
and Acyclic labeled graphs $G_3 = (V_3, E_3, L_3)$

Output : - ∞ (if some z are infinite.)
- Length of longest string in the set $\{ z \mid \exists q \in L_3(MP(G_3))$ such that $q \in Subseq(z)$ and $z \in Subseq(G_1) \cap Subseq(G_2) \}$ (otherwise)

e.g. 2



$Subseq(G_1) \cap Subseq(G_2)$ = {a, b, d, … , **aab**, …}

$L_3(MP(G_3))$
= {**ab**, ac, cc}

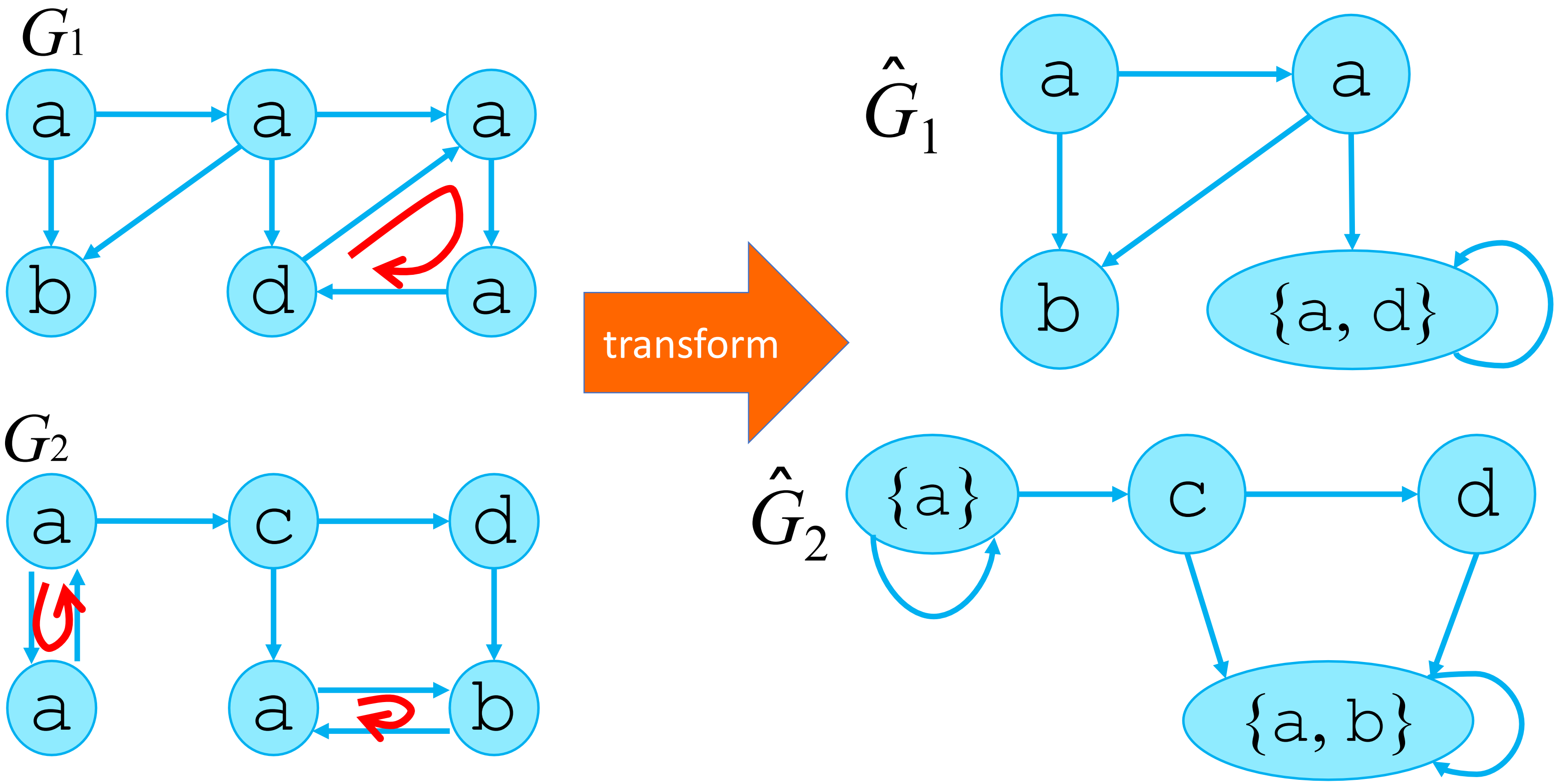
The solution is 3. (**aab**)

Main Idea of the Algorithm for SEQ-IC-LCS of Cyclic Labeled Graphs

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

Strongly Connected Components

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.



Cyclic Labeled Graphs

Acyclic Labeled Graphs

Main Idea of the Algorithm for SEQ-IC-LCS of Cyclic Labeled Graphs

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

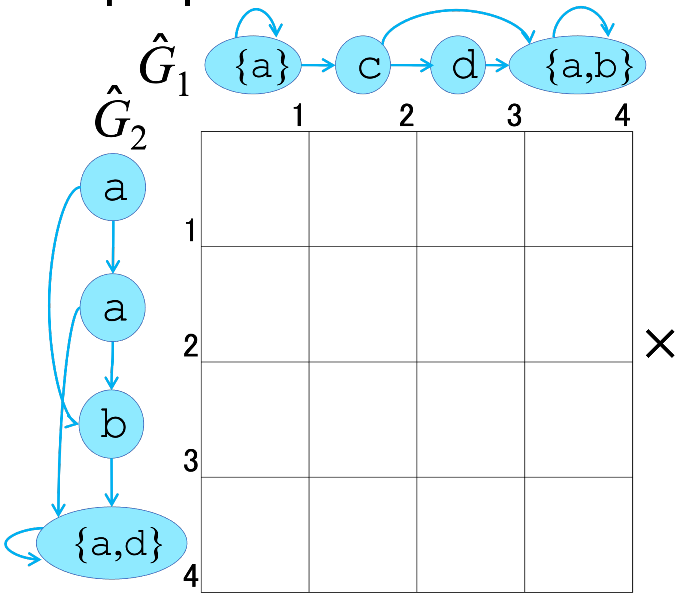
2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

Main Idea of the Algorithm for SEQ-IC-LCS of Cyclic Labeled Graphs

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

Let C denote the three-dimensional table which stored the length of the SEQ-IC-LCS of $\hat{L}_1\left(P\left(\hat{v}_{1,i}\right)\right), \hat{L}_2\left(P\left(\hat{v}_{2,j}\right)\right)$ and $L_3\left(P\left(v_{3,k}\right)\right)$ in $C(i,j,k)$ for any $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq |V_3|$.
$\left(\hat{v}_{1,i} \ \in \hat{V}_1, \hat{v}_{2,j} \ \in \hat{V}_2, v_{3,k} \ \in V_3\right)$



$\times\ (|V_3| + 1)$ tables

Main Idea of the Algorithm for SEQ-IC-LCS of Cyclic Labeled Graphs

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

Let C denote the three-dimensional table which stored the length of the SEQ-IC-LCS of $\hat{L}_1\left(P\left(\hat{v}_{1,i}\right)\right), \hat{L}_2\left(P\left(\hat{v}_{2,j}\right)\right)$ and $L_3\left(P\left(v_{3,k}\right)\right)$ in $C(i,j,k)$ for any $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq |V_3|$. $\left(\hat{v}_{1,i} \in \hat{V}_1, \hat{v}_{2,j} \in \hat{V}_2, v_{3,k} \in V_3\right)$

3. Precompute the result of conditional expression of recurrence for all $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq \left|\hat{V}_3\right|$.

Precompute the Result of Conditional Expressions of Recurrence

$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \neq \emptyset$

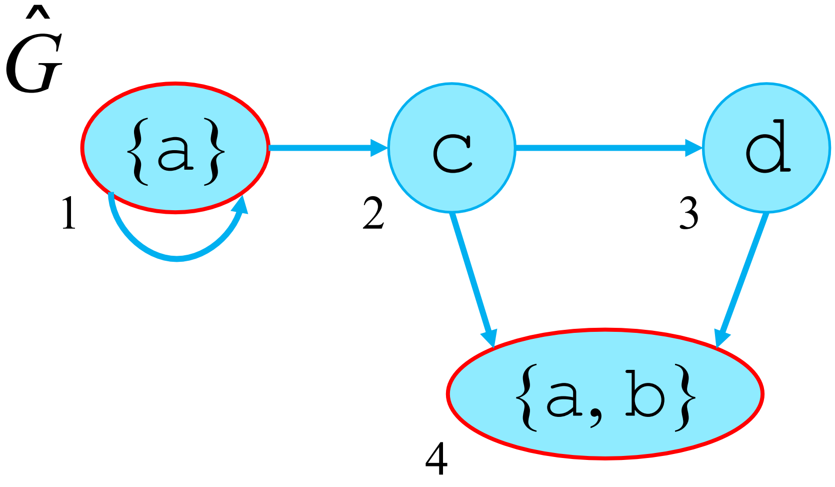
$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$

$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset$

$L(v)$: the character label of vertex v .

$\hat{v}$: the vertex transformed G_1 based on the strongly connected components.

$\hat{L}(\hat{v})$: the set of characters labeled to vertex $\hat{v}$.



$\hat{L}(\hat{v}_1) = \{ \text{a} \}$

$\hat{L}(\hat{v}_4) = \{ \text{a, b} \}$

$\hat{L}(\hat{v}_1) \cap \hat{L}(\hat{v}_4) = \{ \text{a} \}$

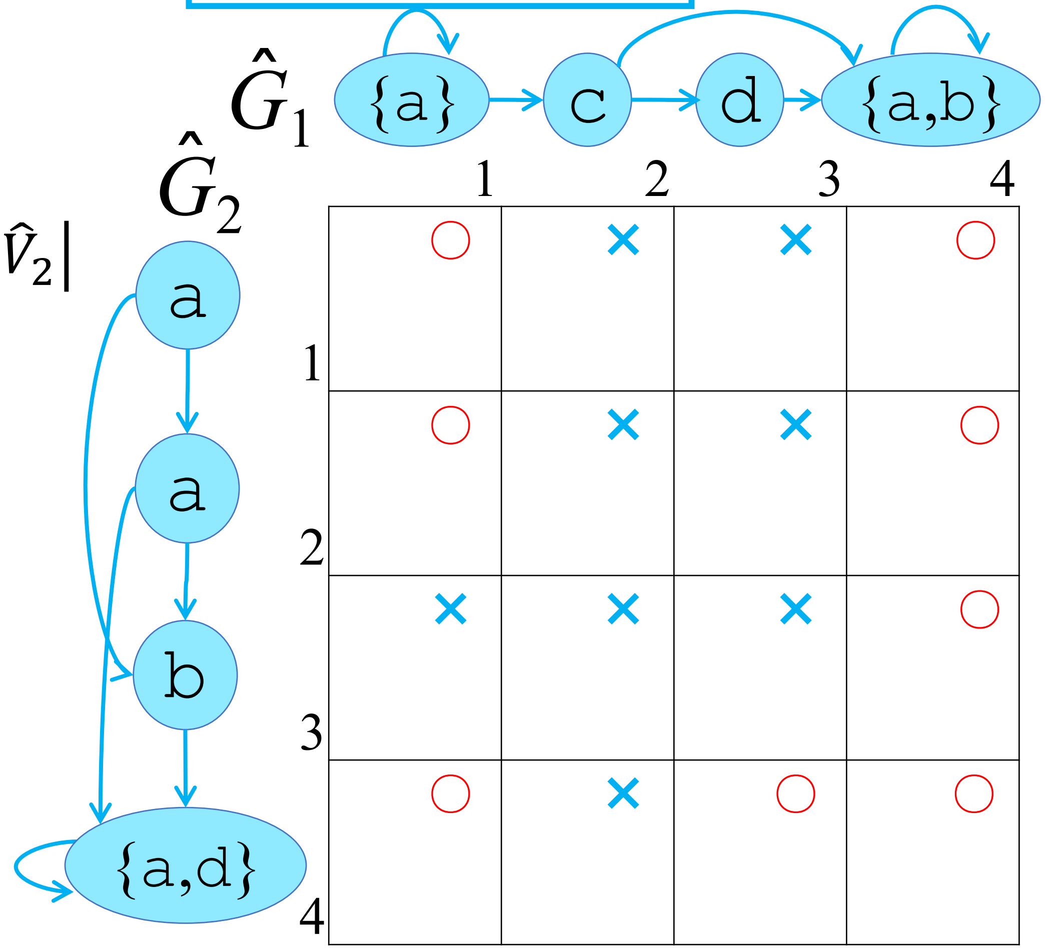
Precompute the Result of Conditional Expressions of Recurrence

$$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \neq \emptyset$$

$$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset \text{ and } \boxed{\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset}$$

$$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset \text{ and } \boxed{\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset}$$

Compute $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j})$
for all $1 \leq i \leq |\hat{V}_1|$ and $1 \leq i \leq |\hat{V}_2|$
using balanced tree.



	1	2	3	4
1	○	×	×	○
2	○	×	×	○
3	×	×	×	○
4	○	×	○	○

Precompute the Result of Conditional Expressions of Recurrence

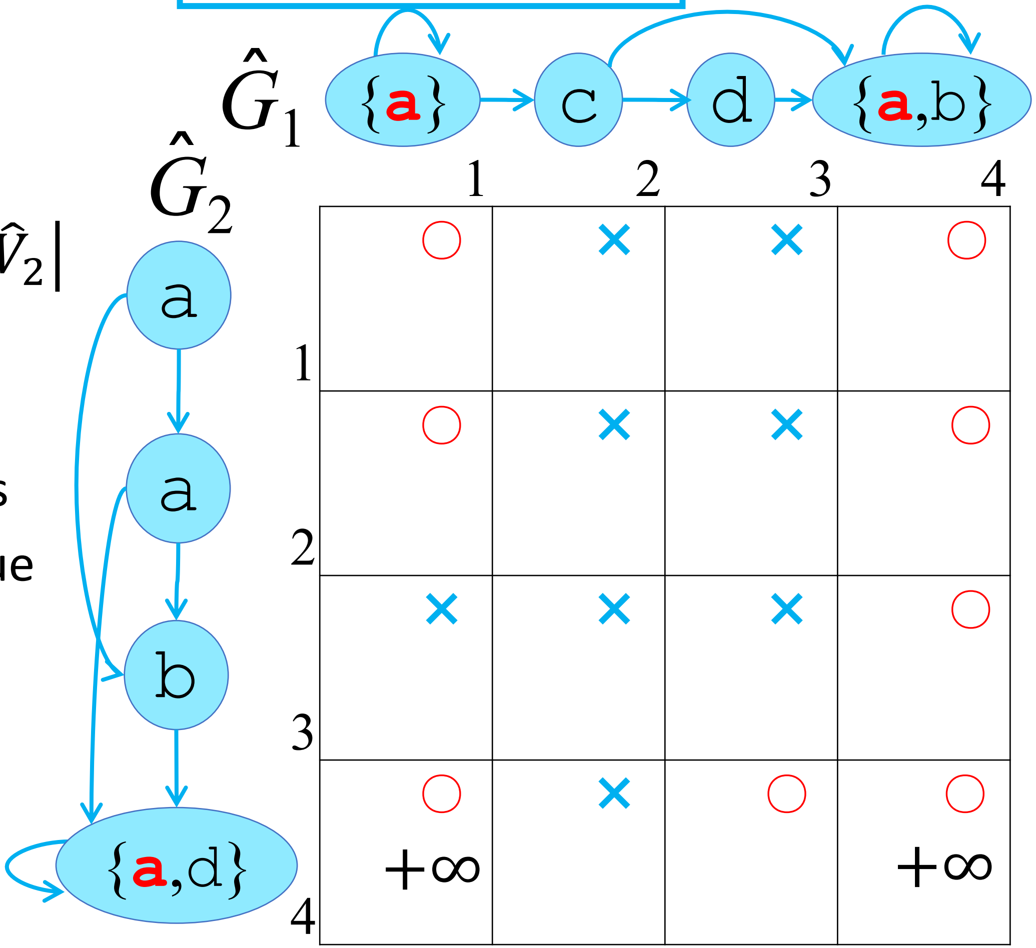
$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \neq \emptyset$

$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\boxed{\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset}$

$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\boxed{\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset}$

Compute $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j})$
for all $1 \leq i \leq |\hat{V}_1|$ and $1 \leq i \leq |\hat{V}_2|$
using balanced tree.

(If both $\hat{v}_{1,i}$ and $\hat{v}_{2,j}$ are cyclic vertices
and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$, the value
$C_{i,j,k}$ is incremented by ∞.)



	1	2	3	4
1	○	×	×	○
2	○	×	×	○
3	×	×	×	○
4	○ +∞	×	○	○ +∞

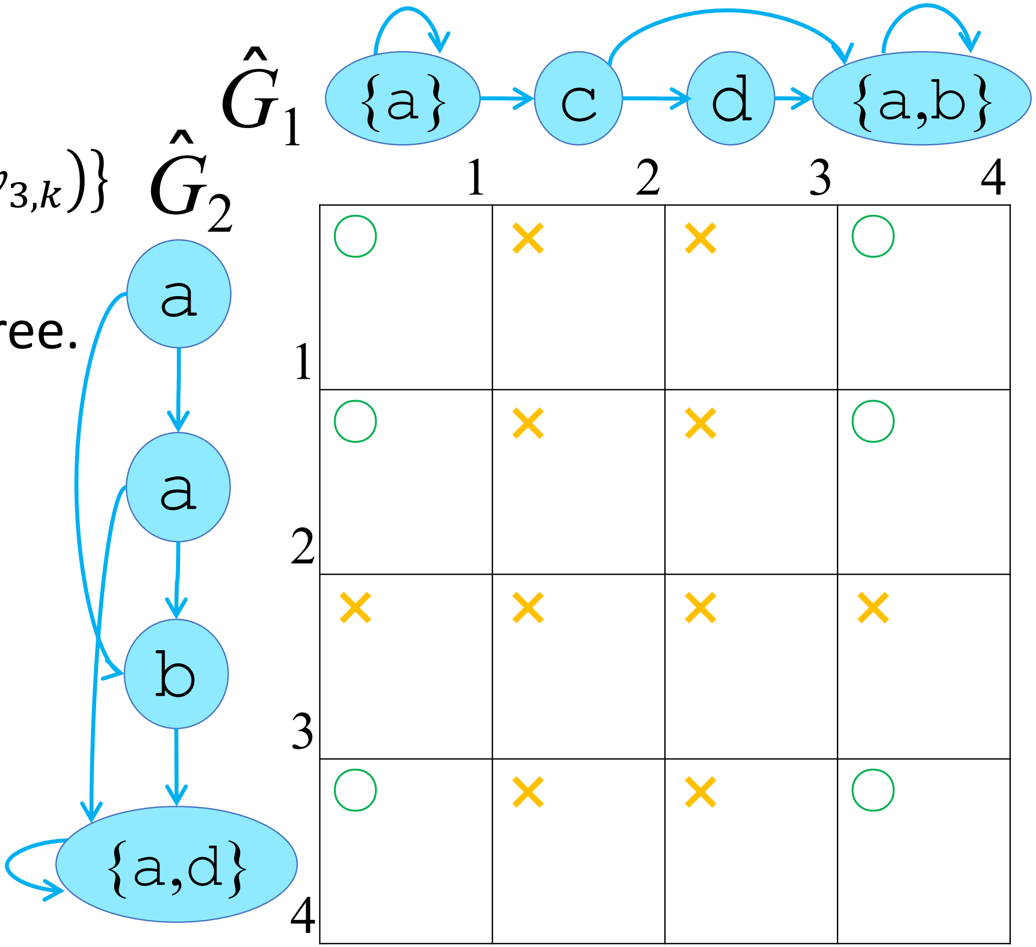
Precompute the Result of Conditional Expressions of Recurrence

$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \neq \emptyset$

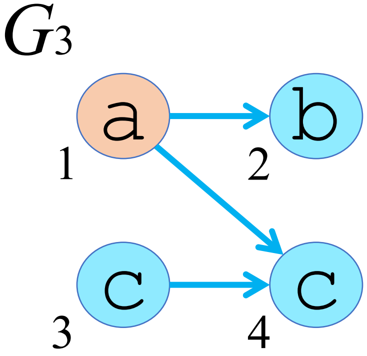
$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset$

$\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} = \emptyset$ and $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset$

Compute $\hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\}$ for all $1 \leq i \leq |\hat{V}_1|, 1 \leq j \leq |\hat{V}_2|$ and $1 \leq k \leq |V_3|$ using balanced tree.



	1	2	3	4
1	○	×	×	○
2	○	×	×	○
3	×	×	×	×
4	○	×	×	○



$k = 1$

Main Idea of the Algorithm for SEQ-IC-LCS of Cyclic Labeled Graphs

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

Let C denote the three-dimensional table which stored the length of the SEQ-IC-LCS of $\hat{L}_1\left(P\left(\hat{v}_{1,i}\right)\right), \hat{L}_2\left(P\left(\hat{v}_{2,j}\right)\right)$ and $L_3\left(P\left(v_{3,k}\right)\right)$ in $C(i,j,k)$ for any $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq |V_3|$.
$\left(\hat{v}_{1,i} \in \hat{V}_1, \hat{v}_{2,j} \in \hat{V}_2, v_{3,k} \in V_3\right)$

3. Precompute the result of conditional expression of recurrence for all $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq \left|\hat{V}_3\right|$.

4. Calculate $C(i, j, k)$ using the recurrence.

$\max_{1 \leq i \leq |V_1|,\, 1 \leq j \leq |V_2|,\, v_{3,k}} C(i, j, k)$ is the solution.

($v_{3,k}$ has no out-going edges in V_3.)

Recurrence of SEQ-IC-LCS of Cyclic Labeled Graphs

$$C_{i,j,k} =$$

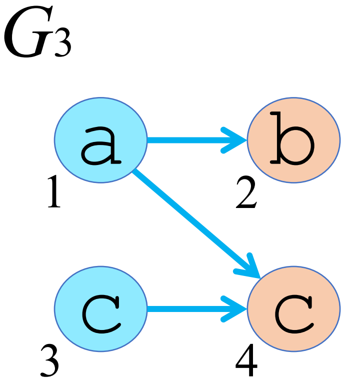
$$\begin{cases}
\delta + \max\left(\left\{ C_{i,j,k} \;\middle|\; \begin{array}{l} (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1, \\ (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2 \end{array} \right\} \cup \{0\}\right) & \begin{array}{l}\text{if } k = 0 \text{ and} \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset;\end{array} \\
\max\left(\begin{array}{l} \{ C_{x,j,k} \mid (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1 \} \cup \\ \{ C_{i,y,k} \mid (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2 \} \cup \{0\} \end{array}\right) & \begin{array}{l}\text{if } k = 0 \text{ and} \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) = \emptyset;\end{array} \\
\delta + \max\left(\left\{ C_{x,y,z} \;\middle|\; \begin{array}{l} (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1, \\ (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2, \\ (v_{3,z}, v_{3,k}) \in E_3 \\ \text{or } z = 0 \end{array} \right\} \cup \{\gamma\}\right) & \begin{array}{l}\text{if } k > 0 \text{ and} \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3([v_{3,k}])\} \\ \neq \emptyset;\end{array} \\
\max\left(\left\{ \delta + C_{x,y,k} \;\middle|\; \begin{array}{l} (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1, \\ (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2 \end{array} \right\} \cup \{-\infty\}\right) & \begin{array}{l}\text{if } k > 0, \\ \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \cap \{L_3(v_{3,k})\} \\ = \emptyset, \text{ and } \hat{L}_1(\hat{v}_{1,i}) \cap \hat{L}_2(\hat{v}_{2,j}) \neq \emptyset;\end{array} \\
\max\left(\begin{array}{l} \{ C_{x,j,k} \mid (\hat{v}_{1,x}, \hat{v}_{1,i}) \in \hat{E}_1 \} \cup \\ \{ C_{i,y,k} \mid (\hat{v}_{2,y}, \hat{v}_{2,j}) \in \hat{E}_2 \} \cup \{-\infty\} \end{array}\right) & \text{otherwise,}
\end{cases}$$

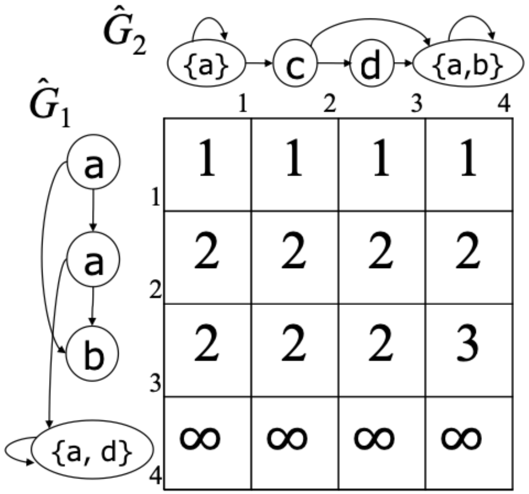
where

$$\delta = \begin{cases} \boxed{\infty \quad \text{if both } \hat{L}_1(\hat{v}_{1,i}) \text{ and } \hat{L}_2(\hat{v}_{2,j}) \text{ are cyclic vertices;}} \\ 1 \quad \text{otherwise,} \end{cases}$$

$$\gamma = \begin{cases} 0 & \begin{array}{l}\text{if } \hat{v}_{1,i} \text{ does not have in-coming edges at all or } \hat{v}_{2,j} \text{ does not have} \\ \text{in-coming edges at all, and } v_{3,k} \text{ does not have in-coming edges;}\end{array} \\ -\infty & \text{otherwise.} \end{cases}$$

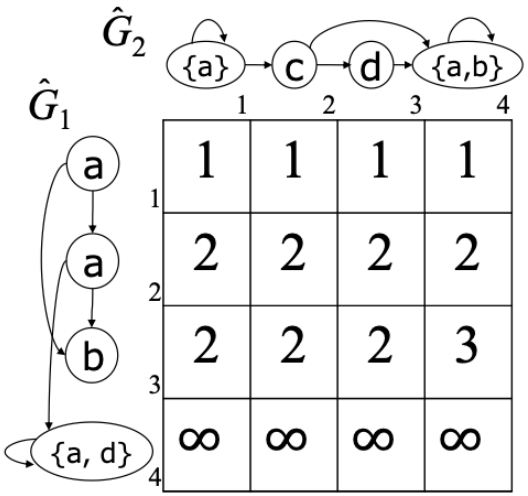
Tables computed by using the recurrence





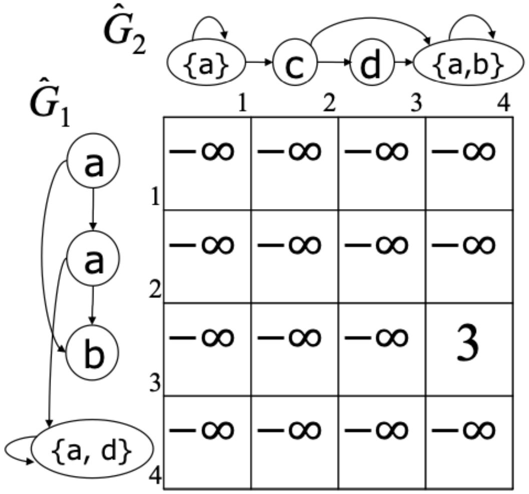
	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	2	2	2	3
4	∞	∞	∞	∞

$k = 0$



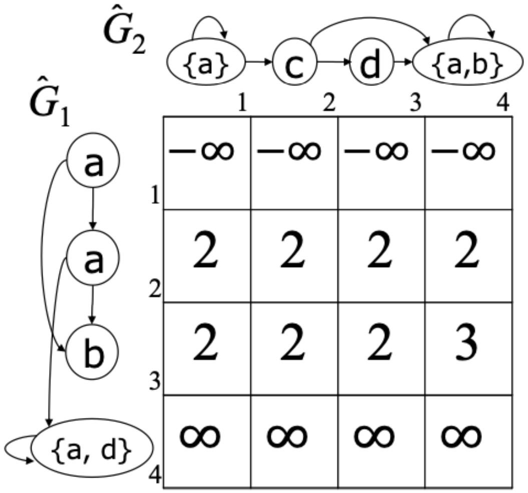
	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	2	2	2	3
4	∞	∞	∞	∞

$k = 1$



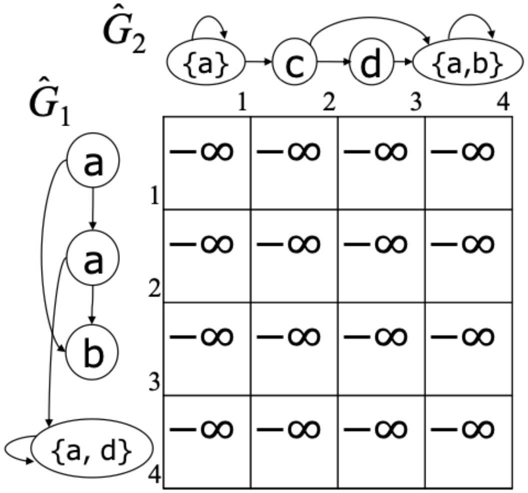
	1	2	3	4
1	−∞	−∞	−∞	−∞
2	−∞	−∞	−∞	−∞
3	−∞	−∞	−∞	3
4	−∞	−∞	−∞	−∞

$k = 2$



	1	2	3	4
1	−∞	−∞	−∞	−∞
2	2	2	2	2
3	2	2	2	3
4	∞	∞	∞	∞

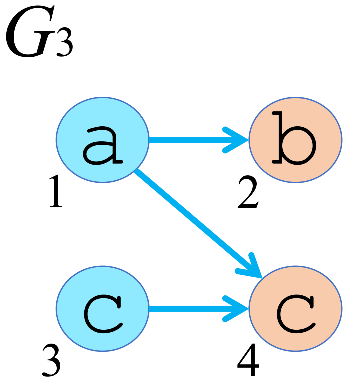
$k = 3$

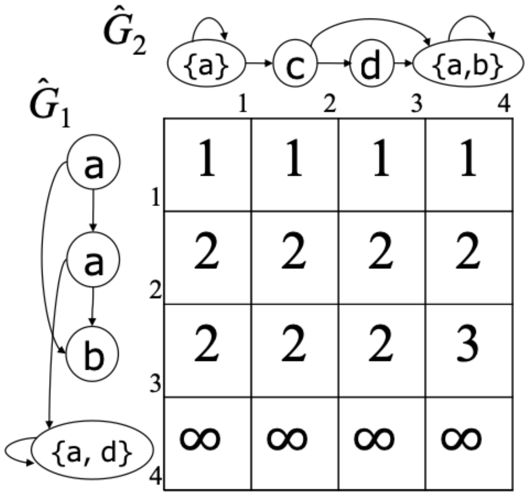


	1	2	3	4
1	−∞	−∞	−∞	−∞
2	−∞	−∞	−∞	−∞
3	−∞	−∞	−∞	−∞
4	−∞	−∞	−∞	−∞

$k = 4$

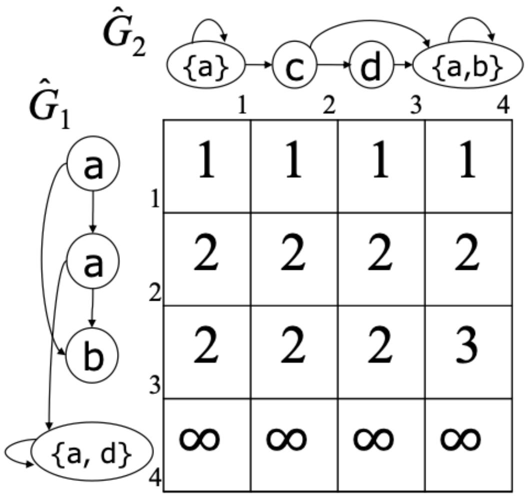
Tables computed by using the recurrence





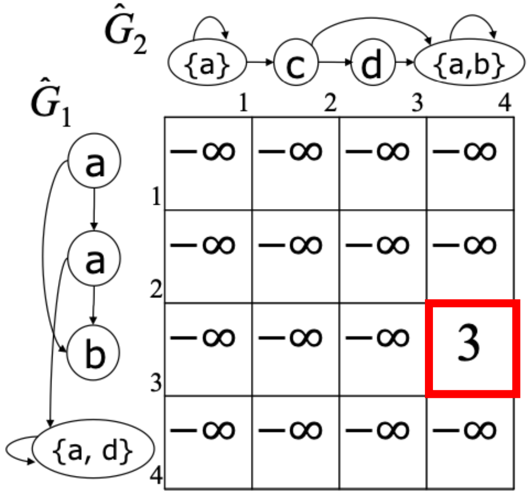
	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	2	2	2	3
4	∞	∞	∞	∞

$k = 0$



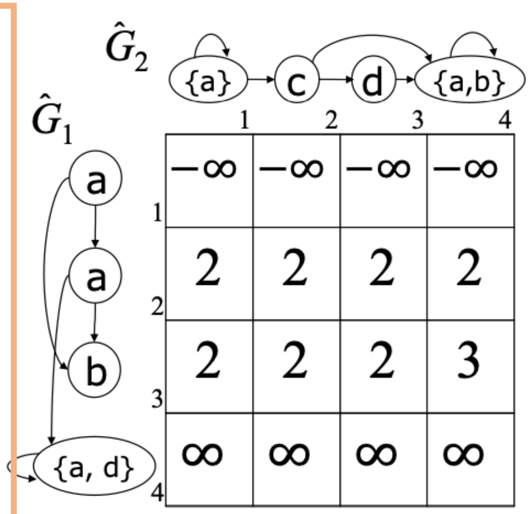
	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	2	2	2	3
4	∞	∞	∞	∞

$k = 1$



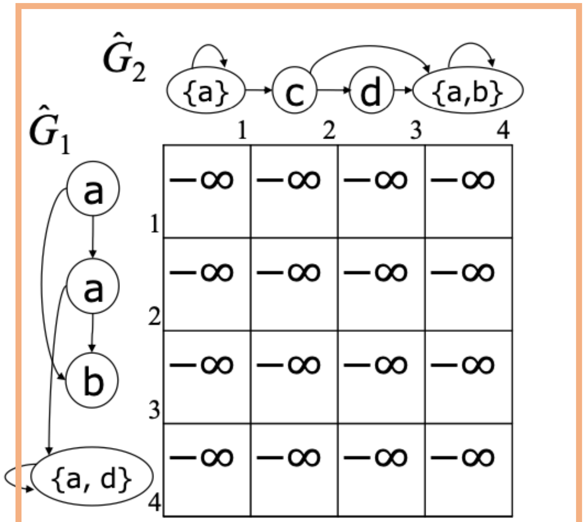
	1	2	3	4
1	−∞	−∞	−∞	−∞
2	−∞	−∞	−∞	−∞
3	−∞	−∞	−∞	3
4	−∞	−∞	−∞	−∞

$k = 2$



	1	2	3	4
1	−∞	−∞	−∞	−∞
2	2	2	2	2
3	2	2	2	3
4	∞	∞	∞	∞

$k = 3$



	1	2	3	4
1	−∞	−∞	−∞	−∞
2	−∞	−∞	−∞	−∞
3	−∞	−∞	−∞	−∞
4	−∞	−∞	−∞	−∞

$k = 4$

Time Complexity

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

linear time

Let C denote the three-dimensional table w

the SEQ-IC-LCS of $\hat{L}_1\left(P\left(\hat{v}_{1,i}\right)\right), \hat{L}_2\left(P\left(\hat{v}_{2,j}\right)\right.$

$C(i,j,k)$ for any $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq |V_3|$.
$\left(\hat{v}_{1,i} \in \hat{V}_1, \hat{v}_{2,j} \in \hat{V}_2, v_{3,k} \in V_3\right)$

3. Precompute the result of conditional expression of recurrence for all $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq \left|\hat{V}_3\right|$.

4. Calculate $C(i, j, k)$ using the recurrence.

$\max\limits_{1 \leq i \leq |V_1|,\, 1 \leq j \leq |V_2|,\, v_{3,k}} C(i, j, k)$ is the solution.

($v_{3,k}$ has no out-going edges in V_3.)

Time Complexity

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

linear time

Let C denote the three dimensional table w

$_2\left(P(\hat{v}_{2,j})\right.$

$j \le |\hat{V}_2|, 0 \le k \le |V_3|$.

$(\hat{v}_{1,i} \in V_1, \hat{v}_{2,j} \in V_2,$ $\in V_3)$

$O\left(|\hat{V}_1||\hat{V}_2||V_3|\log|\Sigma|\right)$ time

(Balanced tree can search a character in $O(\log|\Sigma|)$ time.)

3. Precompute the result of conditional expression of recurrence for all $1 \le i \le |\hat{V}_1|, 1 \le j \le |\hat{V}_2|, 0 \le k \le |\hat{V}_3|$.

4. Calculate $C(i,j,k)$ using the recurrence.

$\max_{1 \le i \le |V_1|,\, 1 \le j \le |V_2|,\, v_{3,k}} C(i,j,k)$ is the solution.

($v_{3,k}$ has no out-going edges in V_3.)

Time Complexity

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

linear time

2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

Let C denote the three dimensional table w

$_2\left(P\left(\hat{v}_{2,j}\right)\right.$

$j \leq \left|\hat{V}_2\right|, 0 \leq k \leq |V_3|$.

$\left(\hat{v}_{1,i} \in V_1, \hat{v}_{2,j} \in V_2, \quad \in V_3\right)$

$O\left(\left|\hat{V}_1\right|\left|\hat{V}_2\right||V_3| \log|\Sigma|\right)$ time

(Balanced tree can search a character in $O(\log|\Sigma|)$ time.)

3. Precompute the result of conditional expression of recurrence for all $1 \leq i \leq \left|\hat{V}_1\right|, 1 \leq j \leq \left|\hat{V}_2\right|, 0 \leq k \leq \left|\hat{V}_3\right|$.

4. Calculate $C(i,j,k)$ using the recurrence.

$O\left(\left|\hat{E}_1\right|\left|\hat{E}_2\right||E_3|\right)$ time

$$\max_{1 \leq i \leq |V_1|,\, 1 \leq j \leq |V_2|,\, v_{3,k}} C(i,j,k)$$ is the solution.

($v_{3,k}$ has

Time Complexity

1. Transform G_1 and G_2 into $\hat{G}_1$ and $\hat{G}_2$ based on the strongly connected components.

2. Sort vertices of $\hat{G}_1, \hat{G}_2$ and G_3 in topological order.

linear time

Let C denote the three dimensional table w

$O\left(|\hat{V}_1||\hat{V}_2||V_3| \log|\Sigma|\right)$ time

(Balanced tree can search a character in $O(\log|\Sigma|)$ time.)

$_2\left(P(\hat{v}_{2,j})\right)$

$j \leq |\hat{V}_2|, 0 \leq k \leq |V_3|$.

$(\hat{v}_{1,i}$

3. Pr

recu

4. Ca

The total time complexity is $O(|E_1||E_2||E_3|+|V_1||V_2||V_3|\log|\Sigma|)$ time.

$\max_{1\leq i\leq|V_1|,\ 1\leq j\leq|V_2|,\ v_{3,k}} C(i,j,k)$ is the solution.

$O\left(|\hat{E}_1||\hat{E}_2||E_3|\right)$ time

$(v_{3,k}$ has

Outline

- Labeled Graphs
- SEQ-IC-LCS (Constrained LCS)
- Computing SEQ-IC-LCS of Acyclic Labeled Graphs
- Computing SEQ-IC-LCS of Cyclic Labeled Graphs
- Conclusions and Future works

Conclusions

problem	text 1	text 2	text 3	Time complexity
STR-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|)$ [Deorowicz, 2012]
STR-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Wang et al., 2013]
SEQ-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chin et al., 2004]
	acyclic graph	acyclic graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|)$ (this work)
	graph	graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|+\|V_1\|\|V_2\|\|V_3\|\log\|\Sigma\|)$ (this work)
SEQ-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chen and Chao, 2011]

$|E_i|$: the number of edges in text i , $|V_i|$: the number of vertices in text i ,
$|\Sigma|$: the alphabet size .

Conclusions

problem	text 1	text 2	text 3	Time complexity
STR-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|)$ [Deorowicz, 2012]
STR-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Wang et al., 2013]
SEQ-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chin et al., 2004]
	acyclic graph	acyclic graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|)$ (this work)
	graph	graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|+\|V_1\|\|V_2\|\|V_3\|\log\|\Sigma\|)$ (this work)
SEQ-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chen and Chao, 2011]

$|E_i|$: the number of edges in text i , $|V_i|$: the number of vertices in text i ,
$|\Sigma|$: the alphabet size .

It is likely that these SEQ-IC-LCS algorithms are optimal as we proved $O(n^{3-\varepsilon})$ ($\varepsilon > 0$) time conditional lower bound based on SETH (Strongly Exponential Time Hypothesis).

Future work

problem	text 1	text 2	text 3	Time complexity
STR-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|)$ [Deorowicz, 2012]
STR-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Wang et al., 2013]
SEQ-IC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chin et al., 2004]
	acyclic graph	acyclic graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|)$ (this work)
	graph	graph	acyclic graph	$O(\|E_1\|\|E_2\|\|E_3\|+\|V_1\|\|V_2\|\|V_3\|\log\|\Sigma\|)$ (this work)
SEQ-EC-LCS	string	string	string	$O(\|E_1\|\|E_2\|\|E_3\|)$ [Chen and Chao, 2011]

- SEQ-EC-LCS problem for labeled graphs could be solved by similar methods.
- STR-IC/EC-LCS problems for labeled graphs are open.