Fine-grained complexity of Longest Common increasing Subsequence

Adam Polak

Jagiellonian University in Kraków

joint work with
Lech Duraj (Jagiellonian) and Marvin Künnemann (MPI)

Zürich, May 24th, 2018

Introduction

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

Example: LCS(edu**cat**ed, **c**o**a**u**t**hor) $= 3$

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]

- No LCS in $O(n^{2-\varepsilon})$ (unless SETH fails) [ABVW and BK '15]

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]

- No LCS in $O(n^{2-\varepsilon})$ (unless SETH fails) [ABVW and BK '15]

SETH (Strong Exponential Time Hypothesis)

No $O((2-\varepsilon)^N)$ time algorithm for CNF-SAT on N variables

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]

- No LCS in $O(n^{2-\varepsilon})$ (unless SETH fails) [ABVW and BK '15]

LCIS: Given two **integer** sequences of length n, determine the length of their **longest common increasing subsequence**

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]

- No LCS in $O(n^{2-\varepsilon})$ (unless SETH fails) [ABVW and BK '15]

LCIS: Given two **integer** sequences of length n, determine the length of their **longest common increasing subsequence**

Example: LCIS(4**1**23**37**, 4**13**63**7**) = 3

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]
- LCIS in $O(n^2)$ [YHC '05]
- No LCS in $O(n^{2-\varepsilon})$ (unless SETH fails) [ABVW and BK '15]

LCIS: Given two **integer** sequences of length n, determine the length of their **longest common increasing subsequence**

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]
- LCIS in $O(n^2)$ [YHC '05]
- No LCS in $O(n^{2-\varepsilon})$ (unless SETH fails) [ABVW and BK '15]

LCIS: Given two **integer** sequences of length n, determine the length of their **longest common increasing subsequence**

Question: Can we solve LCIS in $O(n^{2-\varepsilon})$?

Introduction

LCS: Given two sequences of length n, determine the length of their **longest common subsequence**

- LCS in $O(n^2)$ (textbook example of DP) [WF '74]
- LCS in $O(n^2/\log n)$ [MP '80]
- LCIS in $O(n^2)$ [YHC '05]
- No LCS in $O(n^{2-\varepsilon})$ (unless SETH fails) [ABVW and BK '15]

LCIS: Given two **integer** sequences of length n, determine the length of their **longest common increasing subsequence**

Question: Can we solve LCIS in $O(n^{2-\varepsilon})$?
Our answer: **NO** (unless SETH fails)

Proof overview

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Plan: reduce OV to LCIS

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Plan: reduce OV to LCIS, needs two ingredients:

(1) Vector gadgets:

(2) Glue:

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Plan: reduce OV to LCIS, needs two ingredients:

(1) Vector gadgets:

$u_i, v_i \longrightarrow U_i, V_i$

(vectors) (sequences)

(2) Glue:

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Plan: reduce OV to LCIS, needs two ingredients:

(1) Vector gadgets:

$u_i, v_i \longrightarrow U_i, V_i$

(vectors) (sequences)

large $\mathrm{LCIS}(U_i, V_j)$ iff $u_i \cdot v_j = 0$

(2) Glue:

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Plan: reduce OV to LCIS, needs two ingredients:

(1) Vector gadgets:

$$\underset{\text{(vectors)}}{u_i, v_i} \longrightarrow \underset{\text{(sequences)}}{U_i, V_i}$$

large $\mathrm{LCIS}(U_i, V_j)$ iff $u_i \cdot v_j = 0$

(2) Glue:

$$X = U_0 \circ U_1 \circ \cdots \circ U_{n-1}$$
$$Y = V_0 \circ V_1 \circ \cdots \circ V_{n-1}$$

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Plan: reduce OV to LCIS, needs two ingredients:

(1) Vector gadgets:

$$\underset{\text{(vectors)}}{u_i, v_i} \longrightarrow \underset{\text{(sequences)}}{U_i, V_i}$$

large $\mathrm{LCIS}(U_i, V_j)$ iff $u_i \cdot v_j = 0$

(2) Glue:

$$X = U_0 \circ U_1 \circ \cdots \circ U_{n-1}$$
$$Y = V_0 \circ V_1 \circ \cdots \circ V_{n-1}$$

large $\mathrm{LCIS}(X, Y)$ iff $\exists_{i,j}\ u_i \cdot v_j = 0$

Proof overview

Orthogonal Vectors problem (OV)

Input: Two sets of d-dimensional $(0,1)$-vectors $\{u_0, \ldots, u_{n-1}\}, \{v_0, \ldots, v_{n-1}\} \subset \{0,1\}^d$

Output: u_i, v_j such that $u_i \cdot v_j = 0$

Thm: OV in $O(n^{2-\varepsilon}\mathrm{poly}(d))$ time $\implies$ SETH fails [Williams '05]

Plan: reduce OV to LCIS, needs two ingredients:

(1) Vector gadgets:

$u_i, v_i \longrightarrow U_i, V_i$

(vectors) (sequences)

large $\mathrm{LCIS}(U_i, V_j)$ iff $u_i \cdot v_j = 0$

(You'd figure it out in 5 minutes)

(2) Glue:

$$X = U_0 \circ U_1 \circ \cdots \circ U_{n-1}$$
$$Y = V_0 \circ V_1 \circ \cdots \circ V_{n-1}$$

large $\mathrm{LCIS}(X, Y)$ iff $\exists_{i,j}\ u_i \cdot v_j = 0$

(Hard work happens here)

Vector gadgets

For u_i's:

$$0 \text{ at index } \ell \longrightarrow 2\ell, 2\ell+1$$
$$1 \text{ at index } \ell \longrightarrow 2\ell, 2\ell$$

For v_i's:

$$0 \text{ at index } \ell \longrightarrow 2\ell+1, 2\ell$$
$$1 \text{ at index } \ell \longrightarrow 2\ell+1, 2\ell+1$$

Vector gadgets

For u_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell+1 \\ 1 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell \end{aligned}$$

For v_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell \\ 1 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell+1 \end{aligned}$$

Example:

$$\begin{aligned} u_i = 0\ 1\ 0\ 1 &\longrightarrow U_i = 01\ 22\ 45\ 66 \\ v_j = 0\ 0\ 1\ 1 &\longrightarrow V_j = 10\ 32\ 55\ 77 \end{aligned}$$

Vector gadgets

For u_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell+1 \\ 1 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell \end{aligned}$$

For v_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell \\ 1 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell+1 \end{aligned}$$

Example:

$$\begin{aligned} u_i = 0\ 1\ 0\ 1 &\longrightarrow U_i = 01\ 22\ 45\ 66 \\ v_j = 0\ 0\ 1\ 1 &\longrightarrow V_j = 10\ 32\ 55\ 77 \end{aligned}$$

$$\begin{aligned} u_i \cdot v_j &= 0+0+0+1 \\ \mathrm{LCIS}(U_i, V_j) &= 1+1+1+0 \end{aligned}$$

Vector gadgets

For u_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell+1 \\ 1 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell \end{aligned}$$

For v_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell \\ 1 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell+1 \end{aligned}$$

Example:

$$\begin{aligned} u_i = 0\ 1\ 0\ 1 &\longrightarrow U_i = 01\ 22\ 45\ 66 \\ v_j = 0\ 0\ 1\ 1 &\longrightarrow V_j = 10\ 32\ 55\ 77 \end{aligned}$$

$$\begin{aligned} u_i \cdot v_j &= 0+0+0+1 \\ \mathrm{LCIS}(U_i, V_j) &= 1+1+1+0 \end{aligned}$$

- at most one of 2ℓ and $2\ell+1$ appears in LCIS

Vector gadgets

For u_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell+1 \\ 1 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell \end{aligned}$$

For v_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell \\ 1 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell+1 \end{aligned}$$

Example:

$$\begin{aligned} u_i = 0\ 1\ 0\ 1 &\longrightarrow U_i = 01\ 22\ 45\ 66 \\ v_j = 0\ 0\ 1\ 1 &\longrightarrow V_j = 10\ 32\ 55\ 77 \end{aligned}$$

$$\begin{aligned} u_i \cdot v_j &= 0+0+0+1 \\ \mathrm{LCIS}(U_i, V_j) &= 1+1+1+0 \end{aligned}$$

- at most one of 2ℓ and $2\ell+1$ appears in LCIS
- it appears iff $u_i[\ell]$ and $v_j[\ell]$ are not both 1

Vector gadgets

For u_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell+1 \\ 1 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell \end{aligned}$$

For v_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell \\ 1 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell+1 \end{aligned}$$

Example:

$$\begin{aligned} u_i = 0\ 1\ 0\ 1 &\longrightarrow U_i = 01\ 22\ 45\ 66 \\ v_j = 0\ 0\ 1\ 1 &\longrightarrow V_j = 10\ 32\ 55\ 77 \end{aligned}$$

$$\begin{aligned} u_i \cdot v_j &= 0+0+0+1 \\ \mathrm{LCIS}(U_i, V_j) &= 1+1+1+0 \end{aligned}$$

- at most one of 2ℓ and $2\ell+1$ appears in LCIS
- it appears iff $u_i[\ell]$ and $v_j[\ell]$ are not both 1
- $\mathrm{LCIS}(U_i, V_j) = d - (u_i \cdot v_j)$

Vector gadgets

For u_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell+1 \\ 1 \text{ at index } \ell &\longrightarrow 2\ell, 2\ell \end{aligned}$$

For v_i's:

$$\begin{aligned} 0 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell \\ 1 \text{ at index } \ell &\longrightarrow 2\ell+1, 2\ell+1 \end{aligned}$$

Example:

$$\begin{aligned} u_i = 0\ 1\ 0\ 1 &\longrightarrow U_i = 01\ 22\ 45\ 66 \\ v_j = 0\ 0\ 1\ 1 &\longrightarrow V_j = 10\ 32\ 55\ 77 \end{aligned}$$

$$\begin{aligned} u_i \cdot v_j &= 0+0+0+1 \\ \mathrm{LCIS}(U_i, V_j) &= 1+1+1+0 \end{aligned}$$

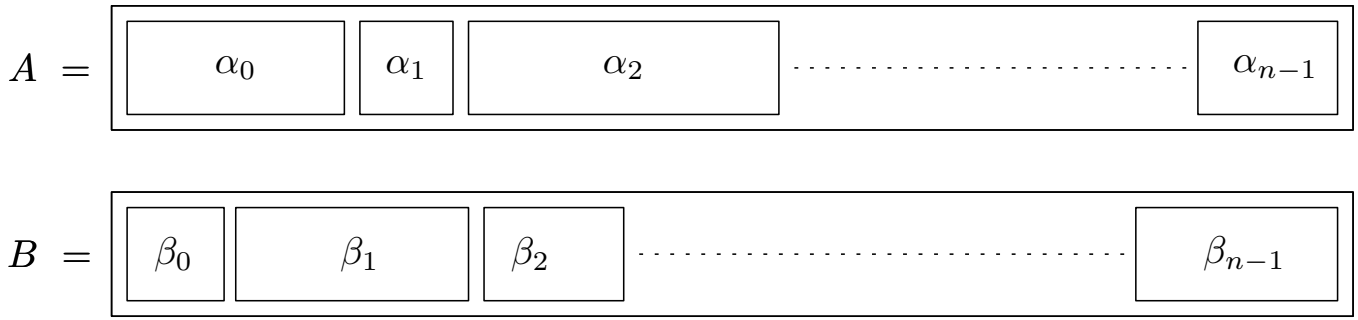
- at most one of 2ℓ and $2\ell+1$ appears in LCIS
- it appears iff $u_i[\ell]$ and $v_j[\ell]$ are not both 1
- $\mathrm{LCIS}(U_i, V_j) = d - (u_i \cdot v_j)$
- $\mathrm{LCIS}(U_i, V_j) \geqslant d$ iff $u_i \cdot v_j = 0$

Glue: separator sequences

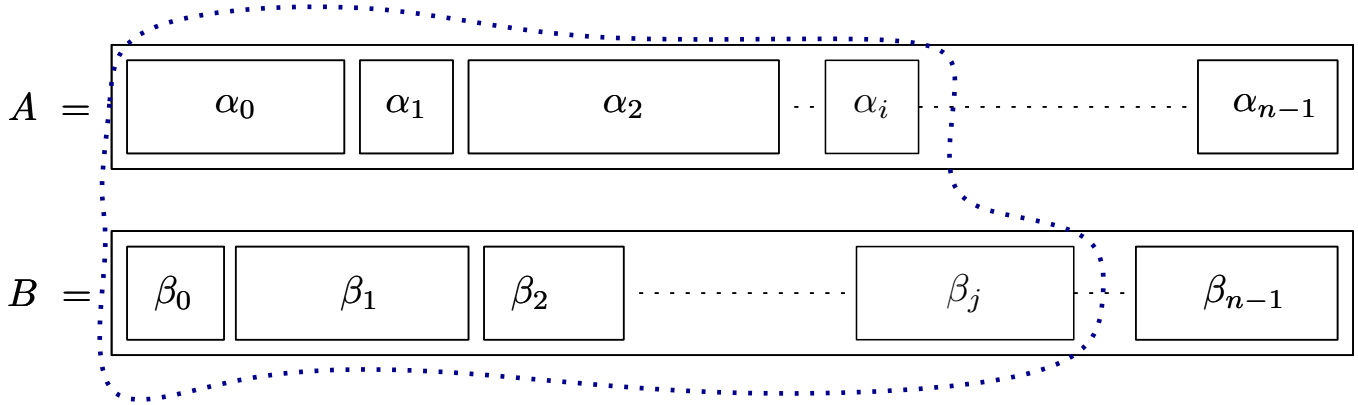
$A =$ 

$B =$ 

Glue: separator sequences

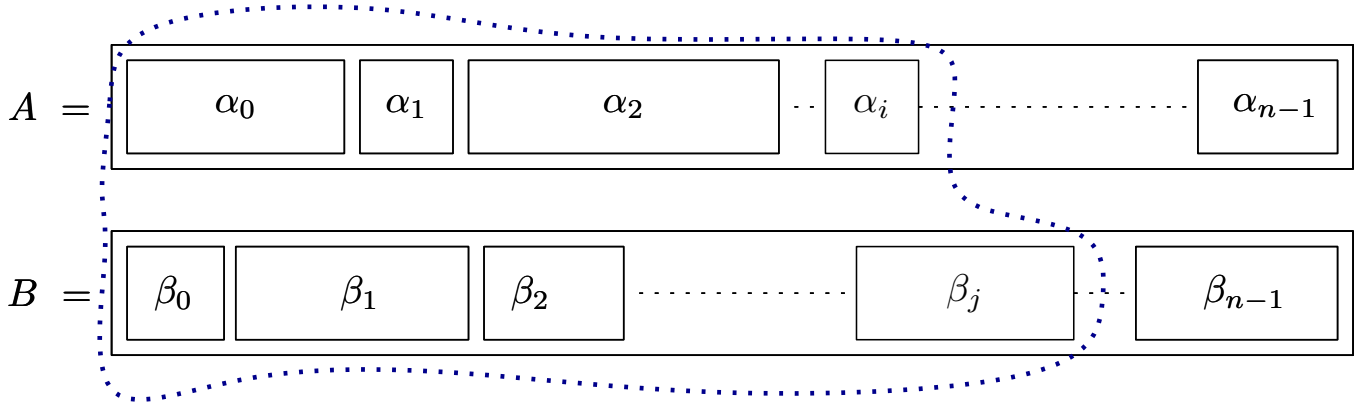


Glue: separator sequences



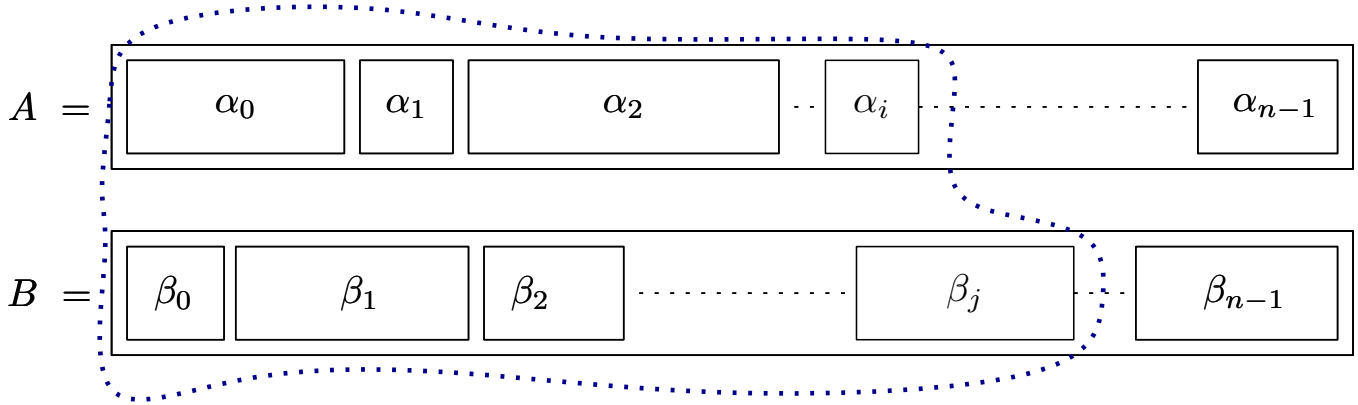
$$\mathrm{LCIS}(\alpha_0 \ldots \alpha_i, \beta_0 \ldots \beta_j) = i + j$$

Glue: separator sequences



$$\mathrm{LCIS}(\alpha_0 \ldots \alpha_i, \beta_0 \ldots \beta_j) = d \cdot (i + j) + const$$

Glue: separator sequences

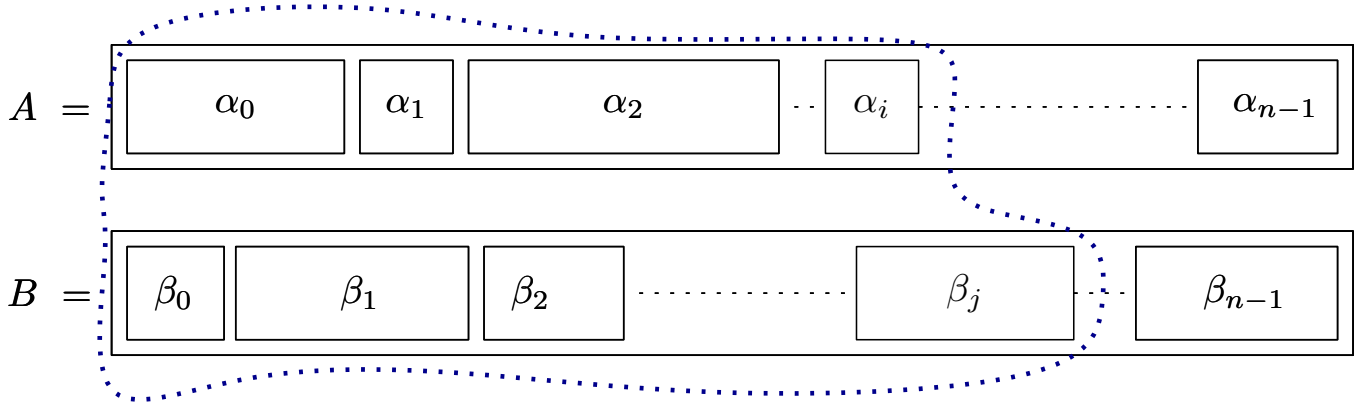


$$\mathrm{LCIS}(\alpha_0 \ldots \alpha_i, \beta_0 \ldots \beta_j) = d \cdot (i + j) + const$$

reverse and negate A, B to obtain $\hat{A}, \hat{B}$

$$\mathrm{LCIS}(\hat{\alpha}_i \ldots \hat{\alpha}_{n-1}, \hat{\beta}_j \ldots \hat{\beta}_{n-1}) = d \cdot (2n - i - j) + const$$

Glue: separator sequences



$$\mathrm{LCIS}(\alpha_0 \dots \alpha_i, \beta_0 \dots \beta_j) = d \cdot (i + j) + const$$

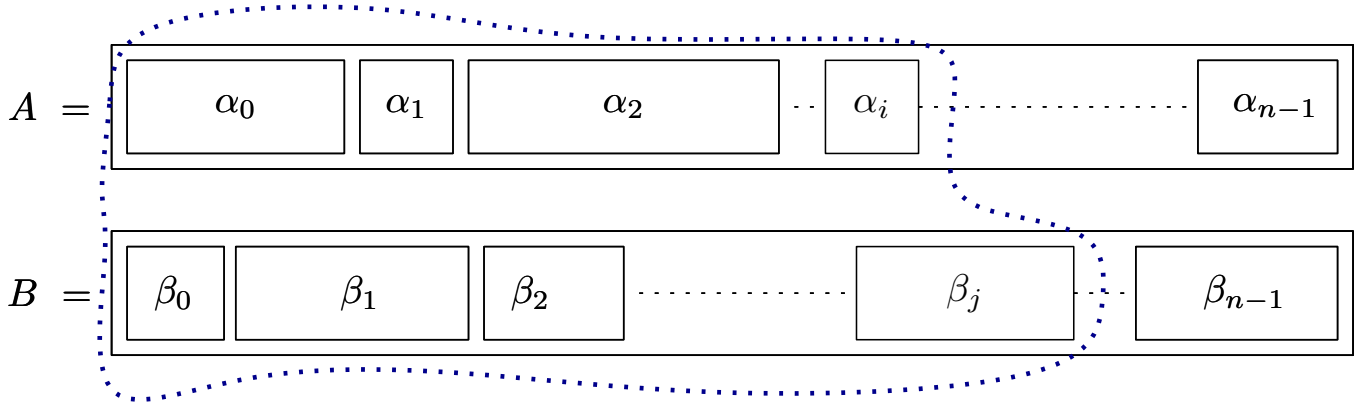
reverse and negate A, B to obtain $\hat{A}, \hat{B}$

$$\mathrm{LCIS}(\hat{\alpha}_i \dots \hat{\alpha}_{n-1}, \hat{\beta}_j \dots \hat{\beta}_{n-1}) = d \cdot (2n - i - j) + const$$

shift alphabets so that $\Sigma_{A,B} < \Sigma_{U_i,V_j} < \Sigma_{\hat{A},\hat{B}}$

$$X = \alpha_0 U_0 \hat{\alpha}_0 \ \alpha_1 U_1 \hat{\alpha}_1 \ \dots\dots\dots\dots\dots \ \alpha_{n-1} U_{n-1} \hat{\alpha}_{n-1}$$
$$Y = \beta_0 V_0 \hat{\beta}_0 \ \beta_1 V_1 \hat{\beta}_1 \ \dots\dots\dots\dots\dots \ \beta_{n-1} V_{n-1} \hat{\beta}_{n-1}$$

Glue: separator sequences



$$\mathrm{LCIS}(\alpha_0 \dots \alpha_i, \beta_0 \dots \beta_j) = d \cdot (i + j) + const$$

reverse and negate A, B to obtain $\hat{A}, \hat{B}$

$$\mathrm{LCIS}(\hat{\alpha}_i \dots \hat{\alpha}_{n-1}, \hat{\beta}_j \dots \hat{\beta}_{n-1}) = d \cdot (2n - i - j) + const$$

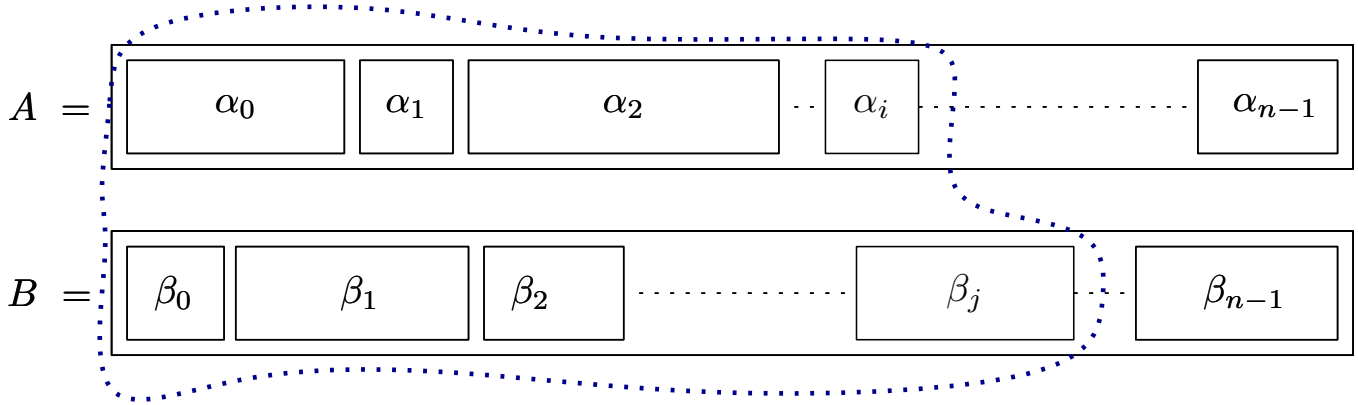
shift alphabets so that $\Sigma_{A,B} < \Sigma_{U_i,V_j} < \Sigma_{\hat{A},\hat{B}}$

$$X = \alpha_0 U_0 \hat{\alpha}_0 \ \alpha_1 U_1 \hat{\alpha}_1 \ \dots\dots\dots\dots\dots \ \alpha_{n-1} U_{n-1} \hat{\alpha}_{n-1}$$

$$Y = \beta_0 V_0 \hat{\beta}_0 \ \beta_1 V_1 \hat{\beta}_1 \ \dots\dots\dots\dots\dots \ \beta_{n-1} V_{n-1} \hat{\beta}_{n-1}$$

$$\mathrm{LCIS}(X, Y) = const' + \max_{i,j} \mathrm{LCIS}(U_i, V_j) \quad (\text{large iff } \exists_{i,j}\ u_i \cdot v_j = 0)$$

Glue: separator sequences



$$\mathrm{LCIS}(\alpha_0 \dots \alpha_i, \beta_0 \dots \beta_j) = d \cdot (i + j) + const$$

reverse and negate A, B to obtain $\hat{A}, \hat{B}$

$$\mathrm{LCIS}(\hat{\alpha}_i \dots \hat{\alpha}_{n-1}, \hat{\beta}_j \dots \hat{\beta}_{n-1}) = d \cdot (2n - i - j) + const$$

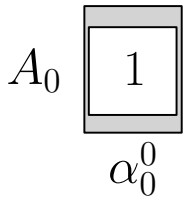
shift alphabets so that $\Sigma_{A,B} < \Sigma_{U_i,V_j} < \Sigma_{\hat{A},\hat{B}}$

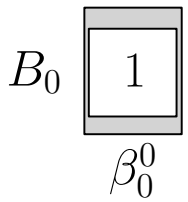
$$X = \alpha_0 U_0 \hat{\alpha}_0 \ \alpha_1 U_1 \hat{\alpha}_1 \ \dots\dots \alpha_i U_i \hat{\alpha}_i \dots \ \alpha_{n-1} U_{n-1} \hat{\alpha}_{n-1}$$

$$Y = \beta_0 V_0 \hat{\beta}_0 \ \beta_1 V_1 \hat{\beta}_1 \ \dots \beta_j V_j \hat{\beta}_j \dots\dots \ \beta_{n-1} V_{n-1} \hat{\beta}_{n-1}$$

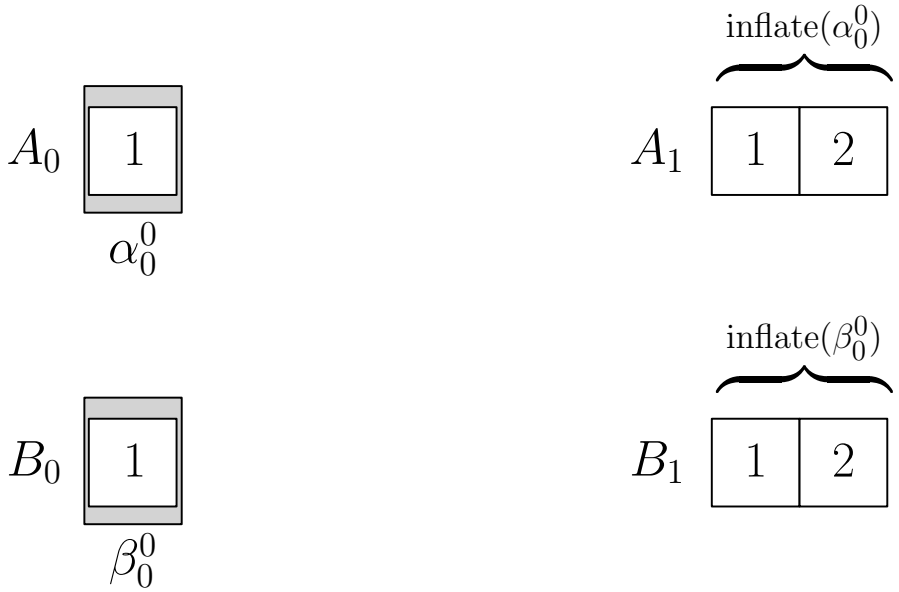
$$\mathrm{LCIS}(X, Y) = const' + \max_{i,j} \mathrm{LCIS}(U_i, V_j) \quad \text{(large iff } \exists_{i,j}\ u_i \cdot v_j = 0\text{)}$$

Separator sequences

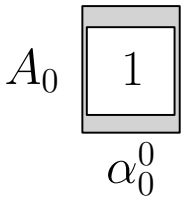


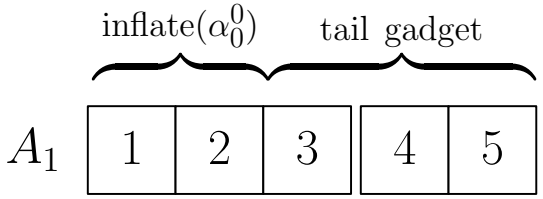


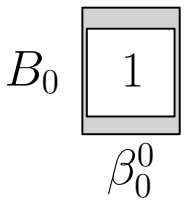
Separator sequences

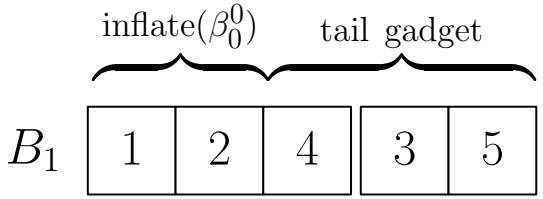


Separator sequences

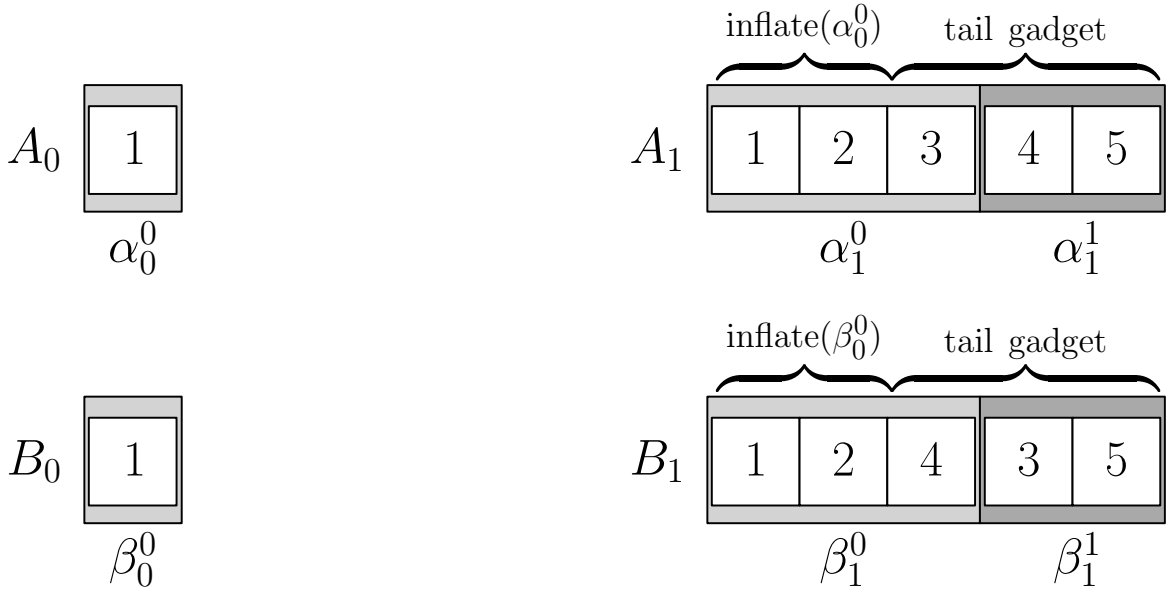




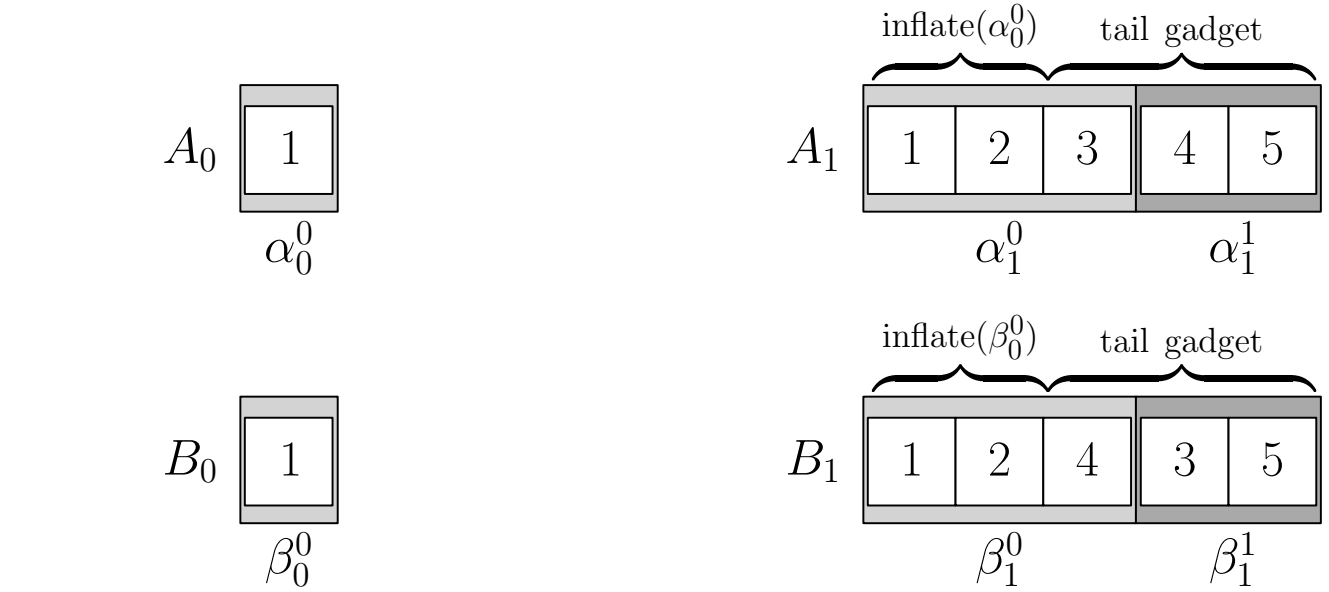


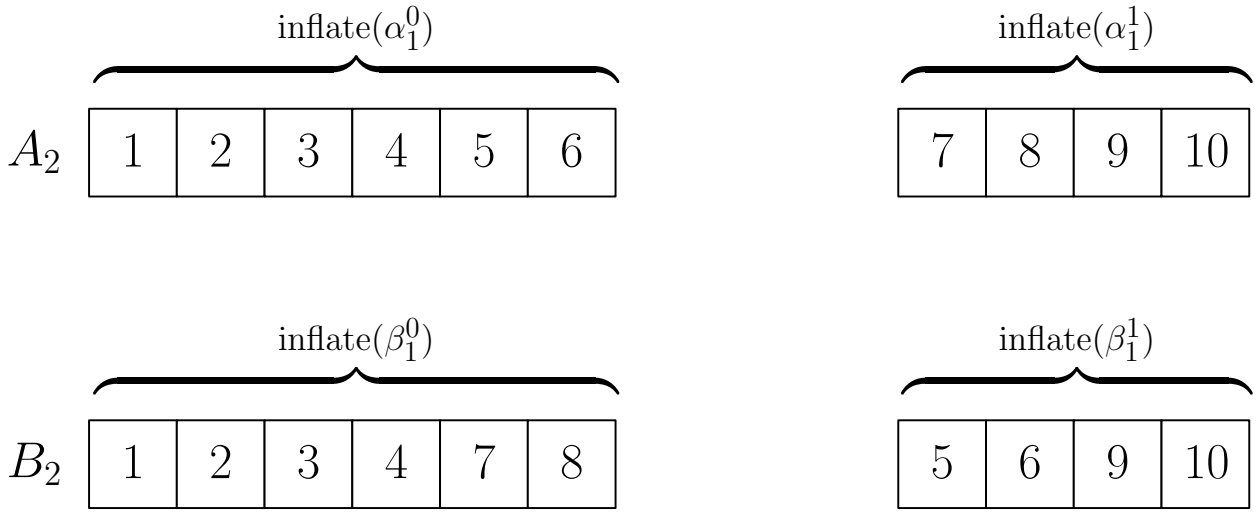


Separator sequences

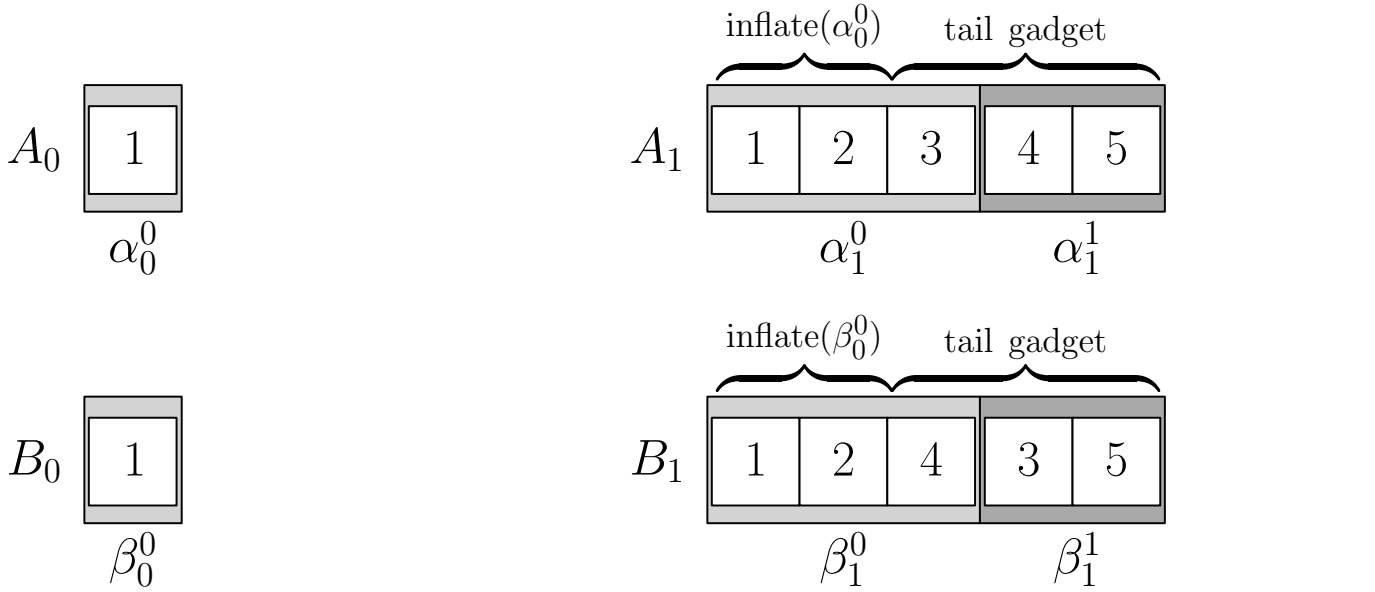


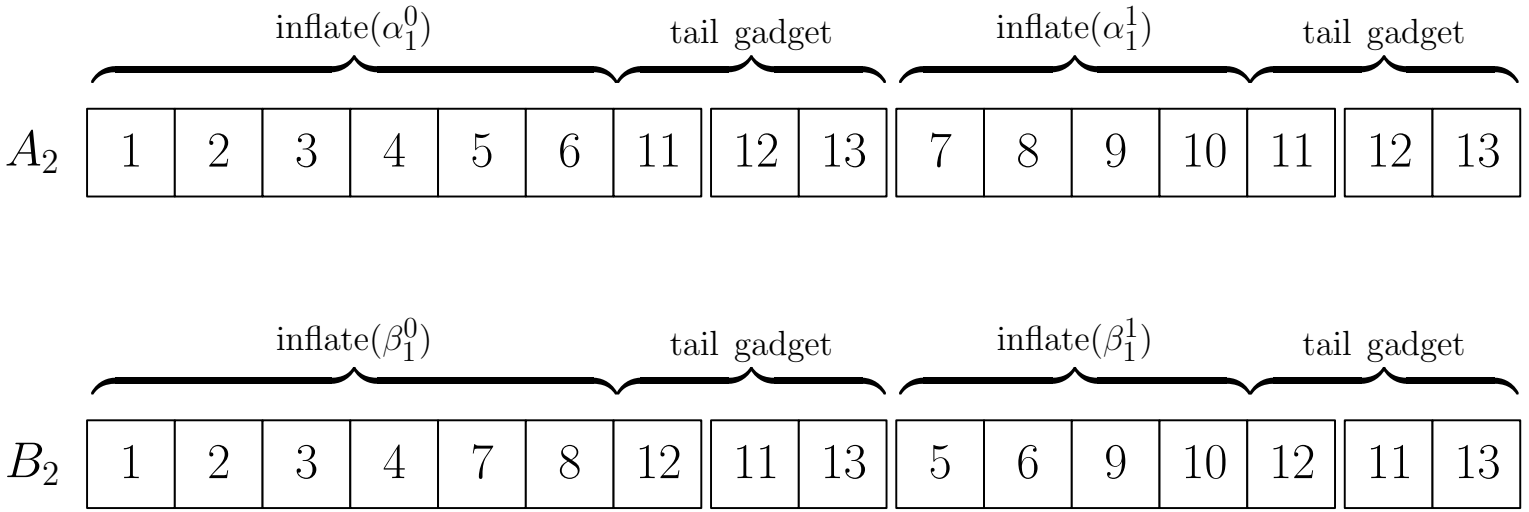
Separator sequences



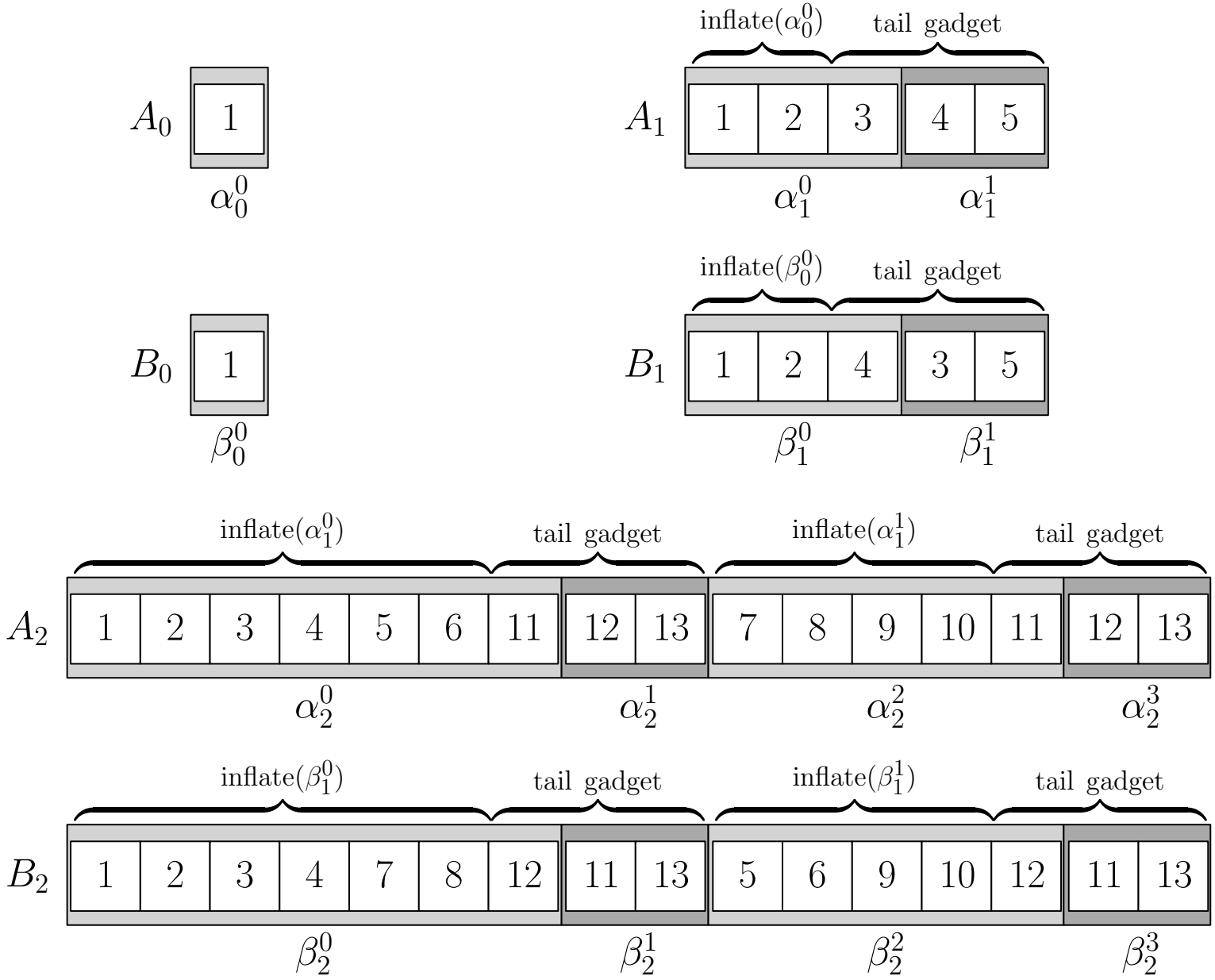


Separator sequences

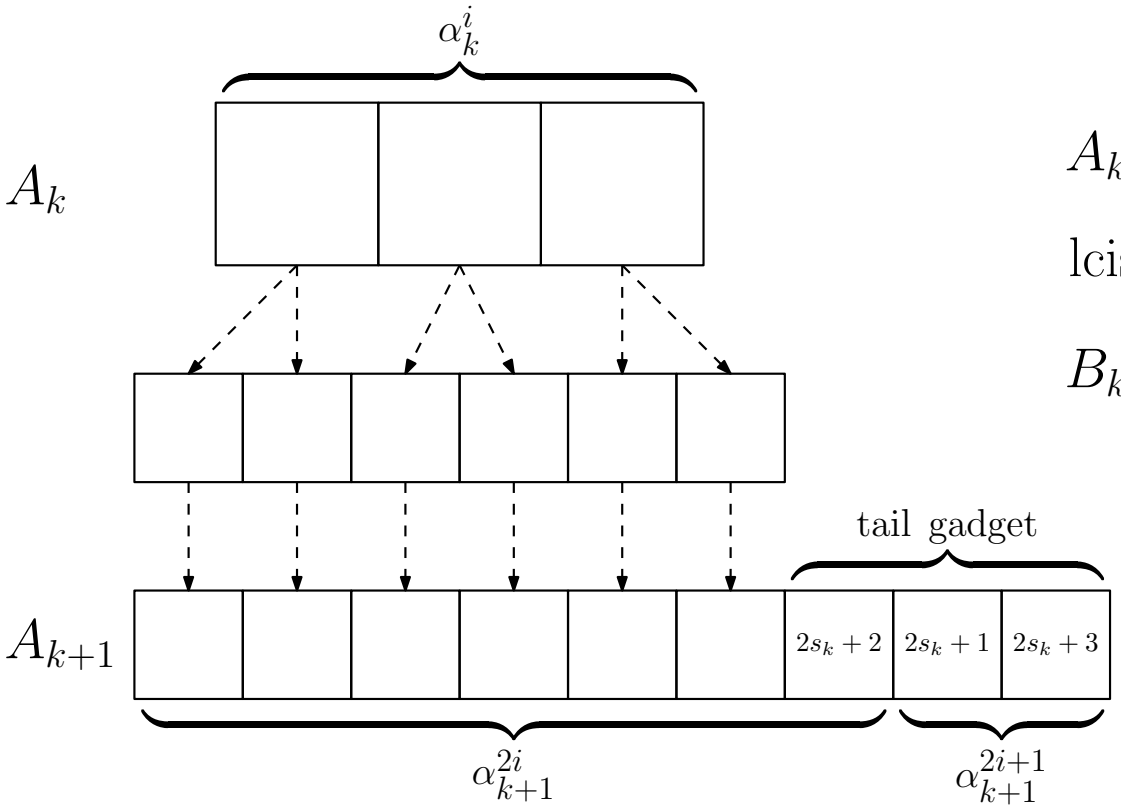


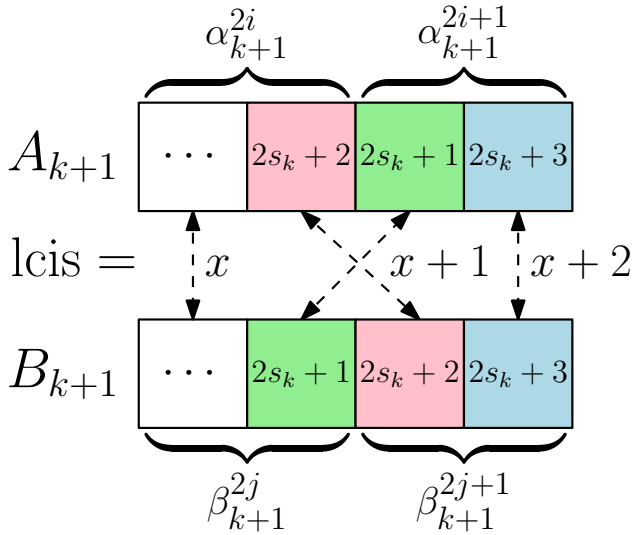


Separator sequences



Separator sequences





$$x = 2i + 2j + 2^{k+1}$$

Our results

Our results

Unless SETH fails:

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

SETH: Brute-force optimal for SAT on **CNF formulas**

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

SETH: Brute-force optimal for SAT on **CNF formulas**

No LCS in $O(n^{2-\varepsilon})$ time (unless SETH fails) [ABVW and BK '15]

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

SETH: Brute-force optimal for SAT on **CNF formulas**

No LCS in $O(n^{2-\varepsilon})$ time (unless SETH fails) [ABVW and BK '15]

No LCS in $O(n^{2-\varepsilon})$ time (unless **BP**-SETH fails) [AHVWW '16]

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

SETH: Brute-force optimal for SAT on **CNF formulas**

No LCS in $O(n^{2-\varepsilon})$ time (unless SETH fails) [ABVW and BK '15]

No LCS in $O(n^{2-\varepsilon})$ time (unless **BP**-SETH fails) [AHVWW '16]

BP-SETH: Brute-force optimal for SAT on **branching programs**

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

SETH: Brute-force optimal for SAT on **CNF formulas**

No LCS in $O(n^{2-\varepsilon})$ time (unless SETH fails) [ABVW and BK '15]

No LCS in $O(n^{2-\varepsilon})$ time (unless **BP**-SETH fails) [AHVWW '16]

BP-SETH: Brute-force optimal for SAT on **branching programs**, in particular: NC-circuits, $o(\sqrt{n})$-memory non-deterministic TM

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

Same lower bounds also hold under **BP**-SETH

Our results

Unless SETH fails:

- **No** $O(n^{2-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If output size L is small:
  - $O(nL \log\log n + n \log n)$ time algorithm [KBKK '11]
  - **No** $O((nL)^{1-\varepsilon})$ time algorithm for LCIS, for any $\varepsilon > 0$
- If number of input sequences k is greater than two:
  - $O(n^k)$ time algorithm for k-LCIS (folklore)
  - **No** $O(n^{k-\varepsilon})$ time algorithm for k-LCIS, for any $\varepsilon > 0$

Same lower bounds also hold under **BP**-SETH

Same lower bounds also hold for **LCWIS**
(Longest Common **Weakly** Increasing Subsequence)

Open problems

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?
  - For $|\Sigma| = 3$ we can, in linear time! [Duraj '13]
  - What about $|\Sigma| > 3$?

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?
- Can we approximate LCIS faster than $O(n^{1.5})$?

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?
- Can we approximate LCIS faster than $O(n^{1.5})$?

$(1+\varepsilon)$-approximation of LCIS in $O(n^{1.5}\mathrm{polylog}(n))$ time

- Case 1: $L \leqslant \sqrt{n/\varepsilon}$, use $O(nL \log\log n)$ algorithm
- Case 2: $L > \sqrt{n/\varepsilon}$
  - delete elements occuring more than $\frac{1}{2}\sqrt{n/\varepsilon}$ times
  - at most $\sqrt{n\varepsilon}$ such elements, output decreased by at most εL
  - reduced to $\frac{n^{1.5}}{\sqrt{\varepsilon}}$ matching pairs, solve it in $O(\frac{n^{1.5}}{\sqrt{\varepsilon}}\mathrm{polylog}(n))$

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?
- Can we approximate LCIS faster than $O(n^{1.5})$?

$(1+\varepsilon)$-approximation of LCIS in $O(n^{1.5}\mathrm{polylog}(n))$ time

- Case 1: $L \leqslant \sqrt{n/\varepsilon}$, use $O(nL \log\log n)$ algorithm
- Case 2: $L > \sqrt{n/\varepsilon}$
  - delete elements occuring more than $\frac{1}{2}\sqrt{n/\varepsilon}$ times
  - at most $\sqrt{n\varepsilon}$ such elements, output decreased by at most εL
  - reduced to $\frac{n^{1.5}}{\sqrt{\varepsilon}}$ matching pairs, solve it in $O(\frac{n^{1.5}}{\sqrt{\varepsilon}}\mathrm{polylog}(n))$
- Can we improve it?

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?
- Can we approximate LCIS faster than $O(n^{1.5})$?

$(1+\varepsilon)$-approximation of LCIS in $O(n^{1.5}\mathrm{polylog}(n))$ time

- Case 1: $L \leqslant \sqrt{n/\varepsilon}$, use $O(nL \log\log n)$ algorithm
- Case 2: $L > \sqrt{n/\varepsilon}$
  - delete elements occuring more than $\frac{1}{2}\sqrt{n/\varepsilon}$ times
  - at most $\sqrt{n\varepsilon}$ such elements, output decreased by at most εL
  - reduced to $\frac{n^{1.5}}{\sqrt{\varepsilon}}$ matching pairs, solve it in $O(\frac{n^{1.5}}{\sqrt{\varepsilon}}\mathrm{polylog}(n))$
- Can we improve it? Would imply subcubic approx for 3-LCS

Open problems

- Can we solve LCIS in $O(n^2/\log n)$ time?
- Can we solve LCWIS in $O(n^{2-\varepsilon})$ for constant size alphabet?
- Can we approximate LCIS faster than $O(n^{1.5})$?

$(1+\varepsilon)$-approximation of LCIS in $O(n^{1.5}\mathrm{polylog}(n))$ time

- Case 1: $L \leqslant \sqrt{n/\varepsilon}$, use $O(nL \log\log n)$ algorithm
- Case 2: $L > \sqrt{n/\varepsilon}$
  - delete elements occuring more than $\frac{1}{2}\sqrt{n/\varepsilon}$ times
  - at most $\sqrt{n\varepsilon}$ such elements, output decreased by at most εL
  - reduced to $\frac{n^{1.5}}{\sqrt{\varepsilon}}$ matching pairs, solve it in $O(\frac{n^{1.5}}{\sqrt{\varepsilon}}\mathrm{polylog}(n))$
- Can we improve it? Would imply subcubic approx for 3-LCS

Thank you!