

Processing Compressed Texts: A Tractability Border

Yury Lifshits

Steklov Institute of Mathematics at St.Petersburg

<http://logic.pdmi.ras.ru/~yura>

Combinatorial Pattern Matching 2007

Consider two compressed texts



Consider two compressed texts



=?



Can we say whether they are equal
without unpacking them?

Outline

- 1 Popular topic in stringology: **algorithms for compressed texts**

Outline

- 1 Popular topic in stringology: **algorithms for compressed texts**
- 2 New algorithm for fully compressed pattern matching

Outline

- 1 Popular topic in stringology: **algorithms for compressed texts**
- 2 New algorithm for fully compressed pattern matching
- 3 Tractability border for processing compressed texts

Outline

- 1 Popular topic in stringology: **algorithms for compressed texts**
- 2 New algorithm for fully compressed pattern matching
- 3 Tractability border for processing compressed texts
- 4 Open problems

Part I

Formulating the problem

Straight-line Programs: Definition

Straight-line program

(SLP) is a context-free grammar generating **exactly one** string

Two types of productions:

$$X_i \rightarrow a \text{ and } X_i \rightarrow X_p X_q$$

Example

abaababaabaab

$$X_1 \rightarrow b$$

$$X_2 \rightarrow a$$

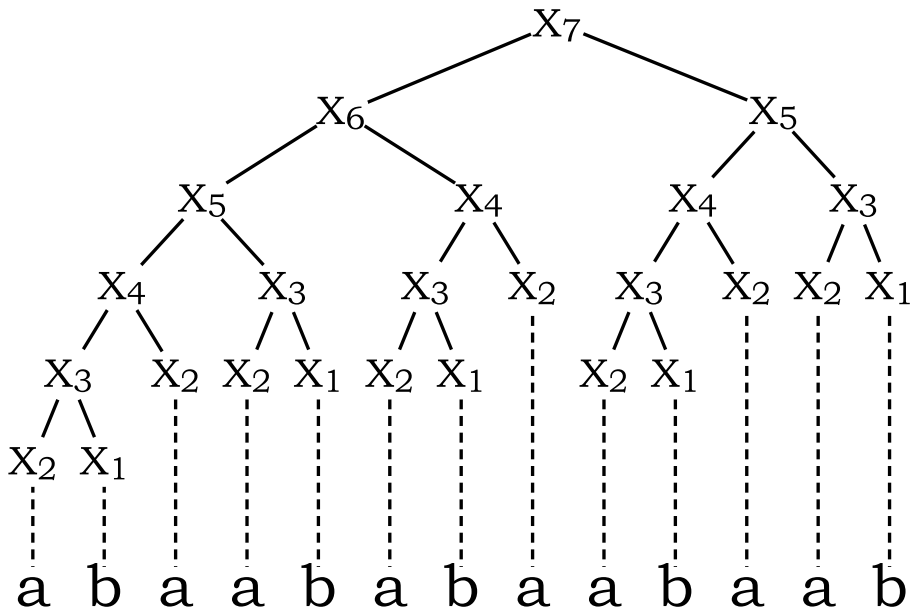
$$X_3 \rightarrow X_2 X_1$$

$$X_4 \rightarrow X_3 X_2$$

$$X_5 \rightarrow X_4 X_3$$

$$X_6 \rightarrow X_5 X_4$$

$$X_7 \rightarrow X_6 X_5$$



Rytter Theorem

Theorem (Rytter'03)

Resulting archive of most practical compression methods can be transformed into SLP generating the same original text. Two nice properties:

- 1 *SLP has almost the same size as initial archive*
- 2 *Transformation is fast*

Rytter Theorem

Theorem (Rytter'03)

Resulting archive of most practical compression methods can be transformed into SLP generating the same original text. Two nice properties:

- 1 *SLP has almost the same size as initial archive*
- 2 *Transformation is fast*

Notable exception: compression schemes based on Burrows-Wheeler transform

Why algorithms on compressed texts?

- **Problems with implicitly given objects:**
word equations [P04], program equivalence [LR06],
message sequence chart verification [GM02]

Why algorithms on compressed texts?

- **Problems with implicitly given objects:**
word equations [P04], program equivalence [LR06],
message sequence chart verification [GM02]
- **Acceleration by text compression:**
HMM training and decoding [MWZU07]

Why algorithms on compressed texts?

- **Problems with implicitly given objects:**
word equations [P04], program equivalence [LR06],
message sequence chart verification [GM02]
- **Acceleration by text compression:**
HMM training and decoding [MWZU07]
- **Connections to practical problems:**
software verification, database compression,
multimedia search

Why algorithms on compressed texts?

- **Problems with implicitly given objects:**
word equations [P04], program equivalence [LR06],
message sequence chart verification [GM02]
- **Acceleration by text compression:**
HMM training and decoding [MWZU07]
- **Connections to practical problems:**
software verification, database compression,
multimedia search
- **Complexity surprises:**
many problems have unexpected computational
complexity when input is given in compressed form

Comparison Problems



Consider two straight-line programs \mathcal{P}, \mathcal{Q} generating texts P and Q . Tasks:

- 1 Determine whether $P = Q$
- 2 **Fully compressed pattern matching:**
check whether P is a substring of Q
- 3 Compute Hamming distance between P and Q

Comparison Problems



Consider two straight-line programs \mathcal{P}, \mathcal{Q} generating texts P and Q . Tasks:

- 1 Determine whether $P = Q$
- 2 **Fully compressed pattern matching:**
check whether P is a substring of Q
- 3 Compute Hamming distance between P and Q

Let m, n be the sizes of \mathcal{P}, \mathcal{Q} . Best previously known result is $\mathcal{O}(n^2m^2)$ algorithm for (1) and (2) **MST'97**

Example

Do they generate the same text?

$$X_1 \rightarrow b$$

$$X_2 \rightarrow a$$

$$X_3 \rightarrow X_2 X_1$$

$$X_4 \rightarrow X_3 X_2$$

$$X_5 \rightarrow X_4 X_3$$

$$X_6 \rightarrow X_5 X_4$$

$$Y_1 \rightarrow b$$

$$Y_2 \rightarrow a$$

$$Y_3 \rightarrow Y_1 Y_2$$

$$Y_4 \rightarrow Y_2 Y_3$$

$$Y_5 \rightarrow Y_3 Y_4$$

$$Y_6 \rightarrow Y_4 Y_5$$

Example

Do they generate the same text?

$$X_1 \rightarrow b$$

$$X_2 \rightarrow a$$

$$X_3 \rightarrow X_2 X_1$$

$$X_4 \rightarrow X_3 X_2$$

$$X_5 \rightarrow X_4 X_3$$

$$X_6 \rightarrow X_5 X_4$$

$$X_7 \rightarrow X_6 X_5$$

$$Y_1 \rightarrow b$$

$$Y_2 \rightarrow a$$

$$Y_3 \rightarrow Y_1 Y_2$$

$$Y_4 \rightarrow Y_2 Y_3$$

$$Y_5 \rightarrow Y_3 Y_4$$

$$Y_6 \rightarrow Y_4 Y_5$$

$$Y_7 \rightarrow Y_5 Y_6$$

Part II

Fully compressed pattern matching

New FCPM Algorithm

MAIN RESULT 1

Fully compressed pattern matching can be solved in $\mathcal{O}(n^2m)$ time

Dynamic Programming (1/2)

$$X_1 \rightarrow b$$

$$X_2 \rightarrow a$$

$$X_3 \rightarrow X_2 X_1$$

$$X_4 \rightarrow X_3 X_2$$

$$X_5 \rightarrow X_4 X_3$$

$$X_6 \rightarrow X_5 X_4$$

$$Y_1 \rightarrow b$$

$$Y_2 \rightarrow a$$

$$Y_3 \rightarrow Y_1 Y_2$$

$$Y_4 \rightarrow Y_2 Y_3$$

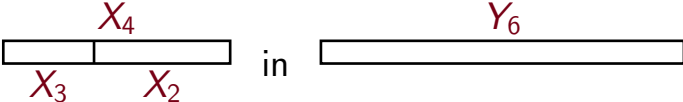

$$Y_5 \rightarrow Y_3 Y_4$$

$$Y_6 \rightarrow Y_4 Y_5$$

$$Y_7 \rightarrow Y_5 Y_6$$

For every i, j we compute occurrences of X_i in Y_j in lexicographic order of pairs: from $(1, 1)$ to $(6, 7)$

Dynamic Programming (2/2)

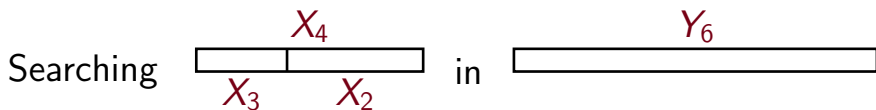
Searching  in 

From previous steps of dynamic programming:

X_3 occurrences in Y_6

X_2 occurrences in Y_6 .

Dynamic Programming (2/2)



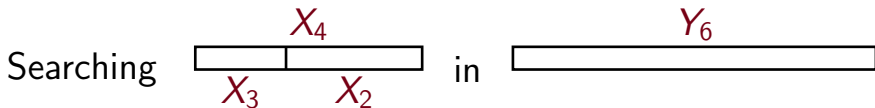
From previous steps of dynamic programming:

X_3 occurrences in Y_6

X_2 occurrences in Y_6 .

Take X_3 occurrences, X_2 occurrences, shift and intersect!

Dynamic Programming (2/2)



From previous steps of dynamic programming:

X_3 occurrences in Y_6

X_2 occurrences in Y_6 .

Take X_3 occurrences, X_2 occurrences, shift and intersect!

Complexity of one step of dynamic programming: $\mathcal{O}(n)$

Number of steps in dynamic programming: $\mathcal{O}(nm)$

Part III

Tractability Border

Check Your Intuition

When two SLPs of sizes n, m are given we can check equivalence between texts generated by them in time $\mathcal{O}(\min(n^2m, m^2n))$

Check Your Intuition

When two SLPs of sizes n, m are given we can check equivalence between texts generated by them in time $\mathcal{O}(\min(n^2m, m^2n))$

What is the complexity of computing Hamming distance between two SLP-generated texts?

Check Your Intuition

When two SLPs of sizes n, m are given we can check equivalence between texts generated by them in time $\mathcal{O}(\min(n^2m, m^2n))$

What is the complexity of computing Hamming distance between two SLP-generated texts?

MAIN RESULT 2

The problem of computing Hamming distance between SLP-generated texts is $\#P$ -complete

Tractability Border

When input texts are compressed:

∃ **poly algorithm:**

GKPR'96 Equivalence
GKPR'96 Regular Language
Membership
KPR'95 Fully Compressed
Pattern Matching
CGLM'06 Window Subsequence
Matching
GKPR'96 Shortest Period
L'07 Shortest Cover

At least NP-hard:

L'07 Hamming distance
Lohrey'04 Context-Free
Language Membership
LL'06 Fully Compressed
Subsequence Matching
BKLPR'02 Two-dimensional
Compressed Pattern Matching

Part IV

Open Problems

OP1: Longest Common Substring

Input: Two SLPs generating texts P and Q

Task: Compute the length of the longest common substring of P and Q

OP1: Longest Common Substring

Input: Two SLPs generating texts P and Q

Task: Compute the length of the longest common substring of P and Q

Can we do it in polynomial time?

OP2: SLP for OR

Input: Two SLPs of sizes n, m generating binary strings P and Q of the same length

Task: Compute a close-to-minimal SLP generating “bitwise OR between P and Q ”

OP2: SLP for OR

Input: Two SLPs of sizes n, m generating binary strings P and Q of the same length

Task: Compute a close-to-minimal SLP generating “bitwise OR between P and Q ”

Can we do it in time $\text{poly}(n + m + \text{output})$?

OP3: Edit Distance

Input: Text P of length l and SLP of size m generating text Q

Input: Compute edit distance between P and Q

OP3: Edit Distance

Input: Text P of length l and SLP of size m generating text Q

Input: Compute edit distance between P and Q

Can we do it in $\mathcal{O}(lm)$ time?

Summary

- Straight-line programs is a wide-accepted model of compressed text

Summary

- Straight-line programs is a wide-accepted model of compressed text
- Fully compressed pattern matching can be solved in $\mathcal{O}(n^2m)$ time

Summary

- Straight-line programs is a wide-accepted model of compressed text
- Fully compressed pattern matching can be solved in $\mathcal{O}(n^2m)$ time
- Computing Hamming distance between compressed texts is $\#P$ -complete

Summary

- Straight-line programs is a wide-accepted model of compressed text
- Fully compressed pattern matching can be solved in $\mathcal{O}(n^2m)$ time
- Computing Hamming distance between compressed texts is $\#P$ -complete
- Open problems: longest common substring, bitwise OR and edit distance for compressed texts

Last Slide

Search “**Lifshits**” or visit <http://logic.pdmi.ras.ru/~yura/>



Yury Lifshits

Processing Compressed Texts: A Tractability Border

CPM'07 // [on-line version](#)



Yury Lifshits and Markus Lohrey

Querying and Embedding Compressed Texts

MFCS'06 // [on-line version](#)



Patrick Cégielski, Irène Guessarian, Yury Lifshits and Yuri Matiyasevich

Window Subsequence Problems for Compressed Texts

CSR'06 // [on-line version](#)

Last Slide

Search “**Lifshits**” or visit <http://logic.pdmi.ras.ru/~yura/>



Yury Lifshits

Processing Compressed Texts: A Tractability Border

CPM'07 // [on-line version](#)



Yury Lifshits and Markus Lohrey

Querying and Embedding Compressed Texts

MFCS'06 // [on-line version](#)



Patrick Cégielski, Irène Guessarian, Yury Lifshits and Yuri Matiyasevich

Window Subsequence Problems for Compressed Texts

CSR'06 // [on-line version](#)

Thank you for your attention! Questions?