Querying and Embedding Compressed Texts

Yury Lifshits[1], Markus Lohrey[2]

[1]Steklov Institute of Mathematics at St.Petersburg,
yura@logic.pdmi.ras.ru

[2]Stuttgart University,
lohrey@informatik.uni-stuttgart.de

Stara Lesna
August 2006

Subsequence Matching (Embedding)

INPUT: pattern **TEAM** and text

I N T E R N A T I O N A L S Y M P O S I U M M F C S

TASK: to check whether the text contains the pattern as a **subsequence** (i.e. gaps are allowed)

OUTPUT: Yes

I N **T** **E** R N **A** T I O N A L S Y **M** P O S I U M M F C S

Problem for this talk:

Given a COMPRESSED text and a COMPRESSED pattern can we solve embedding faster than just "unpack-and-search"?

Outline of the Talk

1. New topic in computer science: **algorithms for compressed texts**
2. Our problems and our results
3. Some proof ideas

Part I

What are **compressed** texts?

Can we do something interesting **without unpacking**?

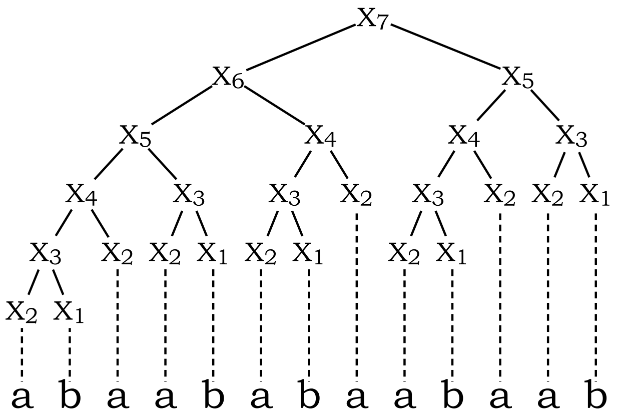
Straight-line Programs: Definition

Straight-line program (SLP) is a
- Context-free grammar generating **exactly one** string
- Two types of productions: $X_i \rightarrow a$ and $X_i \rightarrow X_p X_q$

Example

abaababaabaab

$X_1 \rightarrow b$
$X_2 \rightarrow a$
$X_3 \rightarrow X_2 X_1$
$X_4 \rightarrow X_3 X_2$
$X_5 \rightarrow X_4 X_3$
$X_6 \rightarrow X_5 X_4$
$X_7 \rightarrow X_6 X_5$



SLP = Compressed Text

Rytter, 2003: Consider the archive of size z obtained by LZ78,LZW or some dictionary-based compression method. Then we can in time $O(z)$ convert it to SLP of size $O(z)$ generating the same text.

Rytter, 2003: Consider the LZ77-compressed or RLE-compressed text T of original length n and the archive of size z. Then we can in time $O(z \log n)$ convert it to SLP of the size $O(z \log n)$ generating the same text.

In the following by compressed text we mean an **SLP generating it**

Why algorithms on compressed texts?

Answer for algorithms people:
- Might be faster than "unpack-and-search"
- Saving storing space and transmitting costs
- Many fields with highly compressible data: statistics (internet log files), automatically generated texts, message sequence charts for parallel programs

Answer for complexity people:
- Some problems are hard in worst case. But they might be easy for **compressible** inputs
- New complexity relations. Similar problems have different complexities on compressed inputs

Problems on SLP-generated texts

∃ poly algorithms:

GKPR'96 Equivalence
GKPR'96 Regular Language Membership
GKPR'96 Shortest Period
L'06 Shortest Cover
L'06 Fingerprint Table
GKPR'96 Fully Compressed Pattern Matching
CGLM'06 Window Subsequence Matching

At least NP-hard:

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

Part II

What are embedding and querying problems on compressed texts?

How computationally hard are they?

Querying and Embedding Compressed Texts

Compressed Embedding Problem:
INPUT: Two SLPs generating strings T and P
OUTPUT: YES if T contains P as a subsequence, otherwise NO

Compressed Querying Problem:
INPUT: A SLP generating string T, position i, character a
OUTPUT: YES if $T_i = a$, otherwise NO

Compressed Embedding is Hard

GKPR'96 proved that **string matching** when both the text and the pattern are compressed has a polynomial algorithm.

Natural question: then what about **subsequence matching**?

MAIN RESULT 1:

Compressed Embedding problem is NP-hard
Compressed Embedding problem is co-NP-hard.
Compressed Embedding problem is Θ_2-hard

Compressed Querying is Hard

The most used operation on compressed texts is decompressing.

Natural question: can it be done efficiently by a parallel algorithm?

MAIN RESULT 2:

Compressed Querying problem is P-complete.

Part III

How to prove NP-hardness of Embedding?

How to prove co-NP-hardness of Embedding?

Proving NP-hardness

Classical reduction:

1. Take an NP-complete problem (**Subset Sum**)
2. For every instance of Subset Sum construct two straight line programs such that

Embedding holds ⇔ Subset Sum has answer "Yes"

Proving co-NP-hardness

Lemma (Yes-No symmetry):
For every SLPs X and Y we can in polynomial time construct SLPs X' and Y' such that:

Embedding holds for X and Y
⇔
Embedding does not hold for X' and Y'

Corollary: NP-hardness implies co-NP-hardness

Summary

Main points:

- Compressed text = text generated by SLP
- For compressed texts querying is P-complete, embedding is Θ_2-hard
- Method: reduction from subset sum problem, “yes-no” symmetry

Open Problems:

- What is exact complexity of Compressed Embedding problem (we know that it is somewhere between Θ_2 and PSPACE)?
- To construct $O(nm)$ algorithms for edit distance, where n is the length of T_1 and m is the **compressed size** of T_2

Last Slide

Yury Lifshits http://logic.pdmi.ras.ru/~yura/

Our relevant papers:

Yury Lifshits and Markus Lohrey
Querying and Embedding Compressed Texts
MFCS'06.

Yury Lifshits
Solving Classical String Problems on Compressed Texts
preprint at Arxiv:cs.DS/0604058, 2006.

P. Cégielski, I. Guessarian, Yu. Lifshits and Yu. Matiyasevich
Window Subsequence Problems for Compressed Texts
CSR'06.

Markus Lohrey
Word Problems and Membership Problems on Compressed Words
ICALP'04.

Thanks for attention!