Compression for Data Structures

An Invitation to Start a New Research Area

Yury Lifshits

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

Steklov Institute of Mathematics at St.Petersburg

October 2006

Want to store date in the smallest possible **space**?
Use **data compression**!

LZ-FAMILY, ARITHMETIC ENCODING, BWT, ...

Want to store date in the smallest possible **space**?
Use **data compression**!

LZ-FAMILY, ARITHMETIC ENCODING, BWT, ...

Want to have the fastest **query processing** time?
Use **data structures**!

BALANCED TREES, SUFFIX ARRAYS, HEAPS, HASH TABLES, ...

Want to store date in the smallest possible **space**?
Use **data compression**!

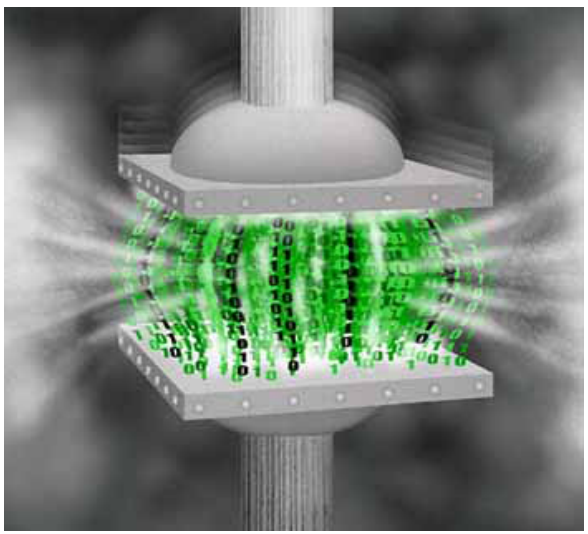
LZ-FAMILY, ARITHMETIC ENCODING, BWT, ...

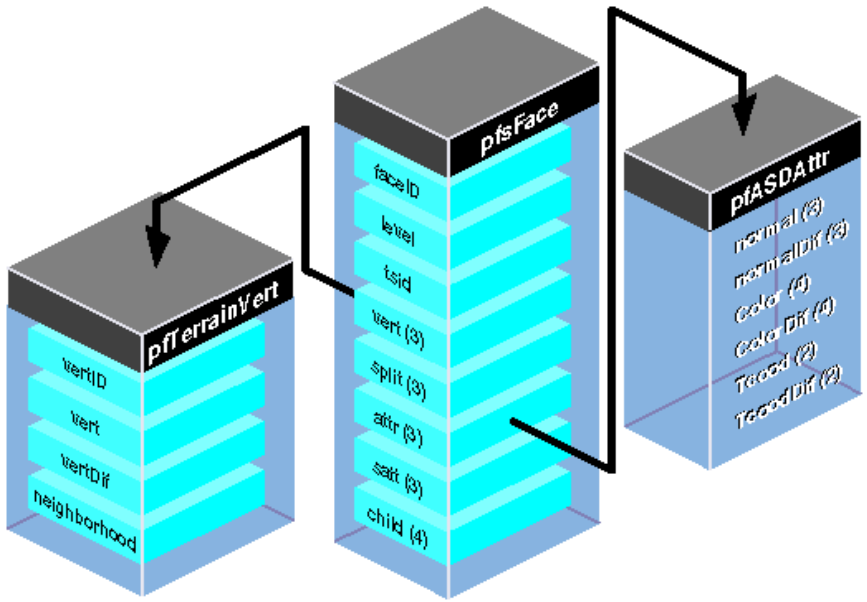
Want to have the fastest **query processing** time?
Use **data structures**!

BALANCED TREES, SUFFIX ARRAYS, HEAPS, HASH TABLES, ...

Can we get **BOTH** these advantages?

We Want to Join Them





Mathematical Challenge

For a given “query problem” to develop a **data structure** such that:

- Query time comparing to the classical data structures is linear
- For some kind of “regular data” the size of our data structure **is smaller than the original data size**

Mathematical Challenge

For a given “query problem” to develop a **data structure** such that:

- Query time comparing to the classical data structures is linear
- For some kind of “regular data” the size of our data structure **is smaller than the original data size**

We might be interested in additional properties:

- Construction time for our DS should be reasonable
- Update time should be small
- Avoid “one bit catastrophe”: small update should not lead to a large expansion of DS

Motivation

Three reasons for “Compression for Data Structures”:

- Potential **applications** in all kind of databases. Real web and biological data sets are now extremely huge
- **Interplay** of two famous fields: compression and data structures
- Looks like a **fresh** topic! Basic problems are still open

Name for the Topic

Working title:

Compression for data structures

Also used:

Data optimization

Queriable compression

Name for the Topic

Working title:

Compression for data structures

Also used:

Data optimization

Queriable compression

What name for this topic do you suggest?

Outline

1. Compression for Specific Problems
   - Membership Test
   - Pattern Matching
   - Graph Navigation
   - Kolmogorov Lower Bound

Outline

1. Compression for Specific Problems
   - Membership Test
   - Pattern Matching
   - Graph Navigation
   - Kolmogorov Lower Bound
2. Workflow for Further Research

Part I

Let us start with two query problems

How to store **sets** if we want to run **membership test** in logarithmic time?

How to store **texts** if we want to run **pattern matching** in time proportional to the **pattern length**?

Set Compression with Fast Membership Test

Problem formalization:

- Given a set A of k integers from the interval $[1..n]$
- Computational model: one space unit for numbers from $[1..n]$, arithmetical operations also in one step
- Query "whether $x \in A$" should be answered in $O(\log k)$
- Data structure should use $o(k)$ space for some **reasonable** class of **"regular"** sets

Set Compression with Fast Membership Test

Problem formalization:

- Given a set A of k integers from the interval $[1..n]$
- Computational model: one space unit for numbers from $[1..n]$, arithmetical operations also in one step
- Query "whether $x \in A$" should be answered in $O(\log k)$
- Data structure should use $o(k)$ space for some **reasonable** class of **"regular"** sets

Do not go to the next slide
What compression method do you suggest for membership test?

I Love DAGs!

DAG = directed acyclic graph:

- Natural idea: if some object is represented by tree and some branches are similar, then we should **merge them**
- **[Rytter, 2003]** For a given text T of length n we can compute $\log n$-approximation of the minimal DAG-representation for T using time $O(n \log n)$. Moreover, this DAG is always **balanced**

Solution for Membership Test

Construction and query:

- Rewrite the set A as a run-length encoding of n-long $0, 1$-string
- Apply Rytter's transformation for getting DAG representation
- Compute "shift values" on the DAG's edges and "key values" for vertices
- Query is simple: use DAG as a search tree!

Compression for Suffix Trees

Problem formalization:

- Given a text T in a constant alphabet
- Computational model: one space unit for characters, all comparison/search operations on characters require one step
- Query "whether P is a substring of T" should be answered in $O(|P|)$ time
- Data structure should use $o(|T|)$ space for some **reasonable** class of **"regular"** texts

Motivation: compact index for desktop search

Compression for Suffix Trees

Problem formalization:

- Given a text T in a constant alphabet
- Computational model: one space unit for characters, all comparison/search operations on characters require one step
- Query "whether P is a substring of T" should be answered in $O(|P|)$ time
- Data structure should use $o(|T|)$ space for some **reasonable** class of **"regular"** texts

Motivation: compact index for desktop search

What compression method do you suggest for pattern matching?

Looks Like a Challenge

I do not know how to solve “Compressed Suffix Tree” problem

Looks Like a Challenge

I do not know how to solve “Compressed Suffix Tree” problem

As a first step I suggest to solve it for two classes of **low informative** texts:

- Texts of length n where only $\log n$ letters are different from a
- Texts of the type $T = S^{\sqrt{n}}$, where $|S| = \sqrt{n}$

Compression for Maps

Problem formalization:

- Graph G
- Query "what is the shortest path from i to j in G?" should be answered in $O(|D_{ij}|)$ time (proportional to the output)
- Data structure should use $o(|V| + |E|)$ space for some **reasonable** class of **"regular"** graphs

Compression for Maps

Problem formalization:

- Graph G
- Query "what is the shortest path from i to j in G?" should be answered in $O(|D_{ij}|)$ time (proportional to the output)
- Data structure should use $o(|V| + |E|)$ space for some **reasonable** class of **"regular"** graphs

Do not go to the next slide
What compression method do you suggest for map compression?

Kolmogorov Lower Bound

For the classical compression there is a natural lower bound for any class of texts:

SIZE OF SOME COMPRESSED OBJECT IS GREATER THAN OR EQUAL TO THE KOLMOGOROV COMPLEXITY OF THAT OBJECT

Kolmogorov Lower Bound

For the classical compression there is a natural lower bound for any class of texts:

SIZE OF SOME COMPRESSED OBJECT IS GREATER THAN OR EQUAL TO THE KOLMOGOROV COMPLEXITY OF THAT OBJECT

We can formulate a similar proposition for data structures compression:

SIZE OF SOME COMPRESSED DATA STRUCTURE IS GREATER THAN OR EQUAL TO THE KOLMOGOROV COMPLEXITY OF THE **LIST OF ALL QUERY ANSWERS**

Compressibility of Query Problems

Let us define **compressibility for query problems** as the **average ratio** between the kolmogorov complexity of the list of all query answers and the kolmogorov complexity of the data **through all data instances**

Part II

What should be done now?

Mathematical open problems?

Open Problems

- Solve suffix tree compression in general

Open Problems

- Solve suffix tree compression in general
- Solve suffix tree compression for two specific cases

Open Problems

- Solve suffix tree compression in general
- Solve suffix tree compression for two specific cases
- Theoretical evaluation (pick a regularity definition, prove some upper bounds on the compressed size) for DAG-method applied to membership problem

Open Problems

- Solve suffix tree compression in general
- Solve suffix tree compression for two specific cases
- Theoretical evaluation (pick a regularity definition, prove some upper bounds on the compressed size) for DAG-method applied to membership problem
- Find **compressibility** of some famous query problems

Learn More!

- Find relevant papers (idea is natural — something has been already done)
- Make a list of “query problems”
- We know what is a “regular text” (entropy, short automata-description, low kolmogorov complexity...). But what do we mean by “regular” for sets, binary relations, ... ?
- Get a feedback from industry people

References (1/2)

Further results will appear at

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

References:

Giorgio Busatto, Markus Lohrey, Sebastian Maneth
Efficient Memory Representation of XML Documents
http://inf.informatik.uni-stuttgart.de/fmi/ti/personen/Lohrey/05-XML.pdf

Yury Lifshits
Solving Classical String Problems on Compressed Texts
http://xxx.lanl.gov/pdf/cs.DS/0604058

JK Min, MJ Park, CW Chung
XPRESS: a queriable compression for XML data
http://islab.kaist.ac.kr/ jkmin/papers/SIGMOD03-min.pdf

Wojtech Rytter
Application of Lempel-Ziv factorization to the approximation of grammar-based compression
http://citeseer.ist.psu.edu/rytter02application.html

References (2/2)

More references:

David Salomon
Data Compression: The Complete Reference (2004)
http://books.google.com/books?id=FlWjiShUstOC

Images sources:
http://www.cs.rochester.edu and http://plus.maths.org/

Main points

Today we learn:

- New field and new challenge: small size together with the fast query time

Main points

Today we learn:

- New field and new challenge: small size together with the fast query time
- The only technique so far: replace trees by DAGs

Main points

Today we learn:

- New field and new challenge: small size together with the fast query time
- The only technique so far: replace trees by DAGs
- Plenty of work to be done. Join this research!

Main points

Today we learn:

- New field and new challenge: small size together with the fast query time
- The only technique so far: replace trees by DAGs
- Plenty of work to be done. Join this research!

Main points

Today we learn:

- New field and new challenge: small size together with the fast query time
- The only technique so far: replace trees by DAGs
- Plenty of work to be done. Join this research!

Questions?