Small-Space 2D Compressed Dictionary Matching

Shoshana Neuburger
and
Dina Sokol

City University of New York

Problem Definition

Compressed 2D Dictionary Matching

Input:

- Compressed text of uncompressed size $n \times n$.
- Dictionary containing k compressed patterns .
- Each pattern is of uncompressed size $m \times m$.

Output:

- All positions in text at which a dictionary pattern occurs.

Problem Definition

An algorithm is strongly inplace if the amount of extra space it uses is proportional to the optimal compression of the data.

Our algorithm is both linear time and strongly inplace .

It uses only $O(km)$ space for a dictionary that occupies $O(km^2)$ space when uncompressed.

2D Dictionary Matching

Bird (1977) and Baker (1978)

- Convert 2D data to 1D representation.
- Name patterns rows.
- Name text positions.
- Use Aho-Corasick to find pattern occurrences.

2D Dictionary Matching

Bird (1977) and Baker (1978)

- Convert 2D data to 1D representation.
- Name patterns rows.
- Name text positions.
- Use Aho-Corasick to find pattern occurrences.

Text is processed once!

2D Dictionary Matching

Bird (1977) and Baker (1978)

- Convert 2D data to 1D representation.
- Name patterns rows.
- Name text positions.
- Use Aho-Corasick to find pattern occurrences.

Text is processed once!

New algorithm is like Bird/Baker in small space.

2D Dictionary Matching

Other 2D dictionary matching algorithms:

- Amir, Landau, Farach (1992)
- Idury, Schaffer (1993)

2D Dictionary Matching

Other 2D dictionary matching algorithms:

- Amir, Landau, Farach (1992)
- Idury, Schaffer (1993)

Problem: not sequential.

Not easily adapted to a strongly-inplace algorithm.

Related Work

Many 2D compressed matching algorithms

- linear time-complexity, yet conserve space.

Use 2D compressed matching algorithms for dictionary matching

- requires scanning text several times.

Compressed dictionary matching not addressed even in 1D.

Compressed Matching

Key property of LZ78:
can decompress using constant space in time linear in the uncompressed string.

Compressed Matching

Key property of LZ78:
can decompress using constant space in time linear in the uncompressed string.

- Amir, Landau, Sokol (2003) algorithm for strongly-inplace 2D pattern matching in LZ78-compressed data.
- We also consider the row-by-row linearization of 2D data.
- Mark text blocks to conserve space.

Our Method

New 2D Dictionary Matching Algorithm

- Linearize the 2D patterns.
- Process the text in blocks.
- Linearize the 2D text.
- Use 1D dictionary matching.
- Search for all patterns simultaneously.
- Cannot access complete dictionary when processing text.

Our Method

Bird and Baker:

- Aho-Corasick automaton of pattern rows.
- $O(km^2)$ preprocessing space.

New algorithm:

- Groups pattern rows into equivalence classes.
- $O(km)$ preprocessing space.

1D Periodicity

Definition

A string p is *periodic* in u if $p = u'u^k$ where u' is a suffix of u, u is primitive, and $k \geq 2$.

Our algorithm is for patterns in which the pattern rows are periodic with period size $\leq m/4$.

Lyndon Words

Definition

Two words x, y are conjugate if $x = uv$, $y = vu$ for some u, v.

Definition

A Lyndon word is a primitive string which is lexicographically smaller than any of its conjugates.

Canonization computes the least conjugate of a word.

Pattern Preprocessing

Examine one pattern row in the dictionary at a time

1. decompress
2. compute period and canonize
3. name row by the Lyndon word of its period
4. store period size, name, first Lyndon word occurrence ($LYpos$).

Naming

New technique for naming rows:

same name if periods are conjugate .

a	a	b	b	a	a	b	b
a	a	b	c	a	a	b	c
a	b	c	a	a	b	c	a
a	b	c	a	b	c	a	b
a	b	a	b	a	b	a	b
a	b	b	a	a	b	b	a
c	a	a	b	c	a	a	b
a	b	a	b	a	b	a	b

1
2
2
3
4
1
2
4

Naming

New technique for naming rows:

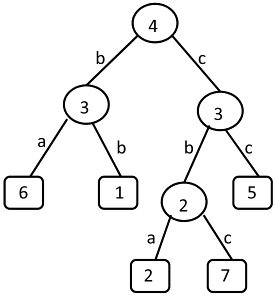
same name if periods are conjugate .

How is this done in linear time, yet small space? witness tree

- Witness tree stores a distinction between two names.
- To name a new row, it is compared to only one other row.

Witness Tree

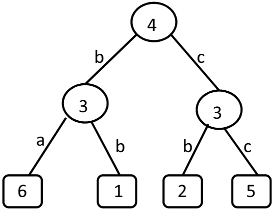
Witness tree for Lyndon words of length 4.



Name	Period size	Lyndon word
1	4	aabb
2	4	aabc
3	3	abc
4	2	ab
5	4	aacc
6	4	aaab
7	4	acbc

Witness Tree

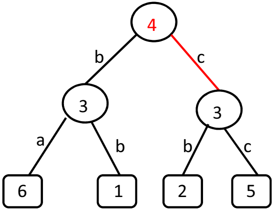
To give acbc name 7 and append to witness tree:



Name	Period size	Lyndon word
1	4	aabb
2	4	aabc
3	3	abc
4	2	ab
5	4	aacc
6	4	aaab
7	4	acbc

Witness Tree

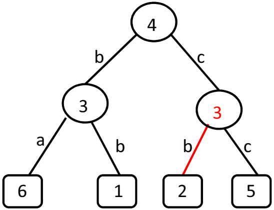
To give acbc name 7 and append to witness tree:



Name	Period size	Lyndon word
1	4	aabb
2	4	aabc
3	3	abc
4	2	ab
5	4	aacc
6	4	aaab
7	4	acbc

Witness Tree

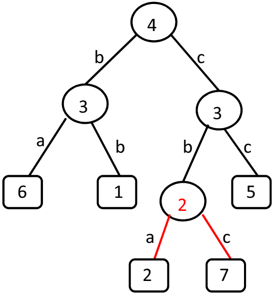
To give acbc name 7 and append to witness tree:



Name	Period size	Lyndon word
1	4	aabb
2	4	aabc
3	3	abc
4	2	ab
5	4	aacc
6	4	aaab
7	4	acbc

Witness Tree

To give acbc name 7 and append to witness tree:



Name	Period size	Lyndon word
1	4	aabb
2	4	aabc
3	3	abc
4	2	ab
5	4	aacc
6	4	aaab
7	4	acbc

Preprocess 1D Patterns

1. Linearize 2D patterns in dictionary.
2. Construct AC automaton of 1D patterns.
3. Find LCM of each 1D pattern.
4. For multiple patterns of same 1D name, build offset tree.

Preprocess 1D Patterns

1. Linearize 2D patterns in dictionary.
2. Construct AC automaton of 1D patterns.
3. Find LCM of each 1D pattern.
4. For multiple patterns of same 1D name, build offset tree.

LCM

Why store the Least Common Multiple (LCM) of 1D patterns?

- Text can have more pattern occurrences than space we allow.
- However, they occur at regular intervals.
- Summarized by occurrence's left and right endpoints + LCM of pattern.

Pattern Preprocessing

Summary of pattern preprocessing:

1. For each pattern row,
   1. decompress
   2. compute period and canonize
   3. name row
   4. store period size, name, first Lyndon word occurrence (*LYpos*).
2. Construct AC automaton of 1D patterns.
3. Find LCM of each 1D pattern.
4. For multiple patterns of same 1D name, build offset tree.

Complexity: $O(km^2)$ time and $O(km)$ space.

Text Scanning

Text scanning stage:

1. Name rows of text block.
2. Identify candidates.
3. Verify candidates.

Naming Text

Lemma

At most one maximal periodic substring of length $\geq m$ with period $\leq m/4$ can occur in a text block row of size $3m/2$.

Text block rows are named the same way as pattern rows.

Text Scanning

Text scanning stage:

1. For each text block row,
   1. decompress
   2. compute period and canonize
   3. name row
   4. store period size, name, first Lyndon word occurrence ($LYpos$).
2. Identify candidates with 1D dictionary matching algorithm.

Use Aho-Corasick automaton of 1D patterns to find rows of text block that can contain a pattern.

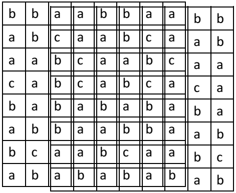
Verification

Consistent Patterns

- have the same 1D representation

  and

- can occur at overlapping positions on a text row



Consistent patterns are horizontal cyclic shifts of one another.

Verification

Consistent Patterns

- have the same 1D representation

  and

- can occur at overlapping positions on a text row

Consistent patterns are horizontal cyclic shifts of one another.

Lemma

Two patterns are consistent iff the LYpos of all their rows are shifted by C mod period size of the row, where C is a constant.

Verification

Single pass verification:

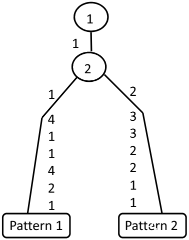
- Group consistent patterns in one class.
- Shift *LYpos* array of pattern to obtain class representative.
- Store shifted arrays in offset tree .
- Shift *LYpos* array of text and compare to edge labels.

		X	X	X	X		
			X	X	X	X	
X	X	X	X				
	X	X	X				
	X	X					
			X	X	X	X	
		X	X	X	X		
X	X						

X	X	X	X				
	X	X	X	X			
		X	X	X	X		
		X	X	X			
	X	X					
	X	X	X	X			
X	X	X	X				
X	X						

Offset Tree

- Pattern1 and Pattern2 have the same 1D name.
- *LYpos* of Pattern 1 not shifted since first entry is 1.
- *LYpos* of Pattern2 shifted by *2 mod period size of row*.



Name	Period size	Pattern1 LYpos	Pattern2 LYpos
1	4	1	3
2	4	1	4
2	4	4	1
3	3	1	2
4	2	1	2
1	4	4	4
2	4	2	3
4	2	1	1

Text Scanning

Text scanning stage:

1. For each text block row,
   1. decompress
   2. compute period and canonize
   3. name row
   4. store period size, name, first Lyndon word occurrence (*LYpos*).
2. Identify candidates with 1D dictionary matching algorithm.
3. Verify candidates of each text row using offset tree.

Complexity: $O(n^2 \log \sigma)$ time and $O(m)$ space.

Summary

- New approach to 2D dictionary matching in small-space.
- Time complexity is optimal.

Summary

- New approach to 2D dictionary matching in small-space.
- Time complexity is optimal.
- Preprocess dictionary in time proportional to uncompressed dictionary size.
- Scan text in time proportional to uncompressed text, independent of dictionary size.

Future Work

Completing the algorithm to cover all patterns.

Patterns with a row of periodic row $> m/4$, possibly aperiodic:

- Many 1D names can overlap in a text block row.
- Identification of candidates is not a problem.
- Difficulty of verifying many candidates per row without access to original dictionary.

Thank you!