Extension and Faster Implementation of the GRP Transform for Lossless Compression

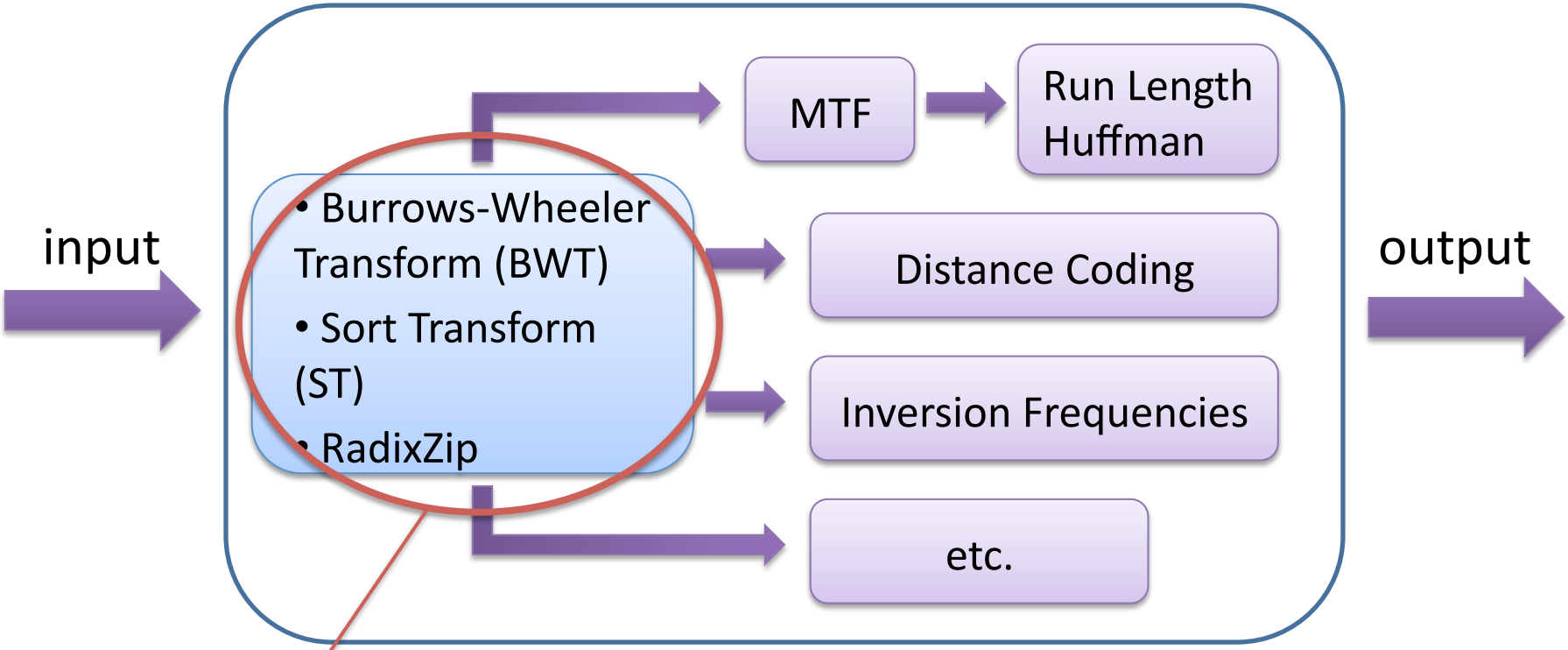
Hidetoshi Yokoo

Gunma University
Japan

Outline

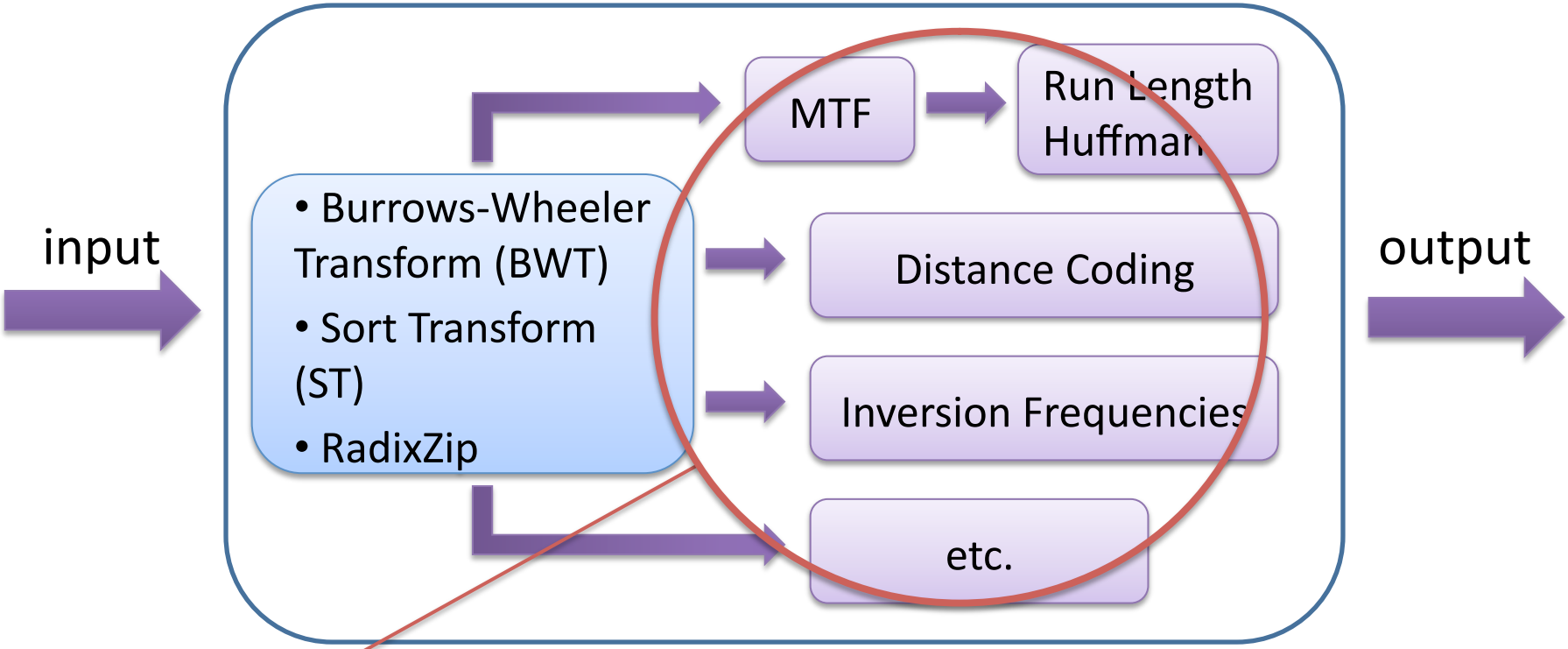
1. **What is the *GRP* Transform**
   - Sort-based Transforms for Lossless Data Compression
   - Existing Transforms -> Parametric Generalization
2. **Proposed Extension**
   - Forward Transformation
   - Faster Implementation of Inverse Trans.
3. **Compression Experiments**
4. **Conclusions and Future Work**

Lossless Compression with Sort-Based Transforms



Transform:
Conversion of one string to another

Lossless Compression with Sort-Based Transforms



[Second-Step Encoders for Transformed Strings]

Actual encoders for transformed text (LGT: Local to Global Transform, Adjeroh-Bell-Murkherjee '08)

Sort-based Transforms for Lossless Data Compression

Transforms	Context Length
BWT (Burrows & Wheeler 1994)	Unbounded
ST (Schindler 1997)	Constant
Permute in RadixZip (Vo & Manku 2008)	0, 1, …, Constant (Cyclically Changing)

- **Sorting = Context-Gathering**

Sort-based Transforms for Lossless Data Compression

Transforms	Context Length
BWT (Burrows & Wheeler 1994)	Unbounded
ST (Schindler 1997)	Constant
Permute in RadixZip (Vo & Manku 2008)	0, 1, …, Constant (Cyclically Changing)
Our Transform (*Generalized Radix Permutation*) (ITY, SPIRE2009)	$d, d+1, \ldots, d+\ell-1$

– **Parameterized Generalization**

Aims of Generalization

- Proper Generalization
- For Data Consisting of Subunits
  - Genetic Codes (Triplet Codons), Multi-byte Data, 2D Data
- Explicit Use of Context Information
  - Towards a Generalization of Second-Step Encoders

Proposed Extension (Generalized Radix Permutation)

Transforms	Context Lengths
BWT (Burrows & Wheeler 1994)	Unbounded
ST (Schindler 1997)	Constant
Permute in RadixZip (Vo & Manku 2008)	0, 1, …, Constant (Cyclically Changing)
Our Transform (*Generalized Radix Permutation*)	$d, d+1, \ldots, d+\ell-1$

- **Parameterized Generalization**

Proposed Extension (Generalized Radix Permutation)

Transform	Context Lengths
Our Transform (*Generalized Radix Permutation*)	$1 \le \ell \le n,\ 0 \le d \le n$ for text length n $d, d+1, \ldots d+\ell-1$

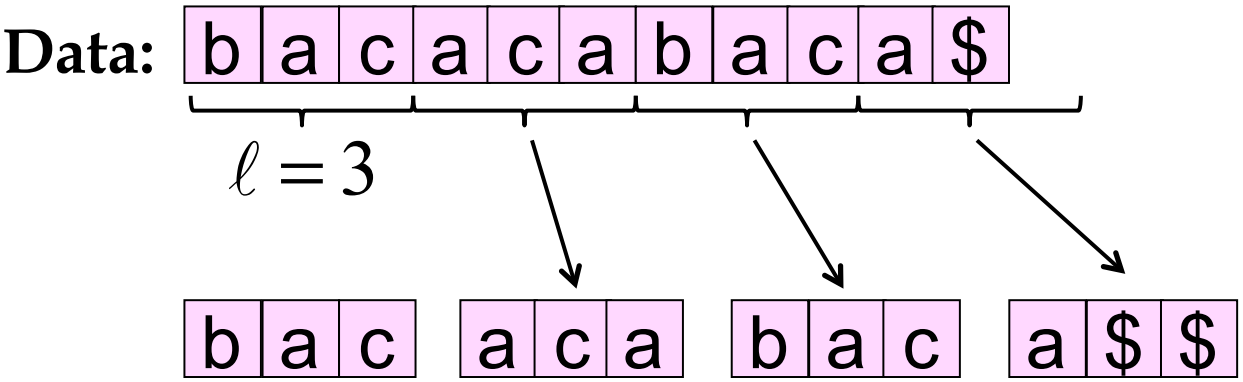
- Arbitrary Parameter Values
- Linear Time Algorithm for Inverse Transformation
- Parameterized Generalization
- Modification of MTF for the Transform

Outline

1. What is the *GRP* Transform
   - Sort-based Transforms for Lossless Data Compression
   - Existing Transforms -> Parametric Generalization
2. **Proposed Transform**
   - **Forward Transformation**
   - **Faster Implementation of Inverse Trans.**
3. Compression Experiments
4. Conclusions and Future Work

Forward Transformation

Divide into ℓ - symbol blocks ($\ell = 3$)



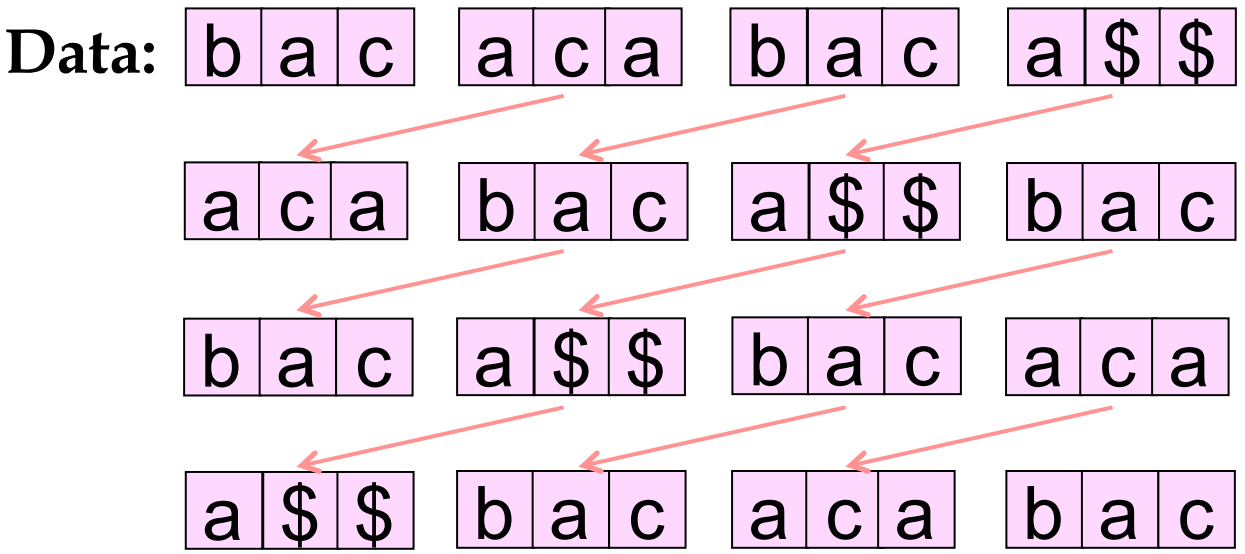
Forward Transformation

Data: | b | a | c | a | c | a | b | a | c | a | $ |

b	a	c		a	c	a		b	a	c		a	$	$

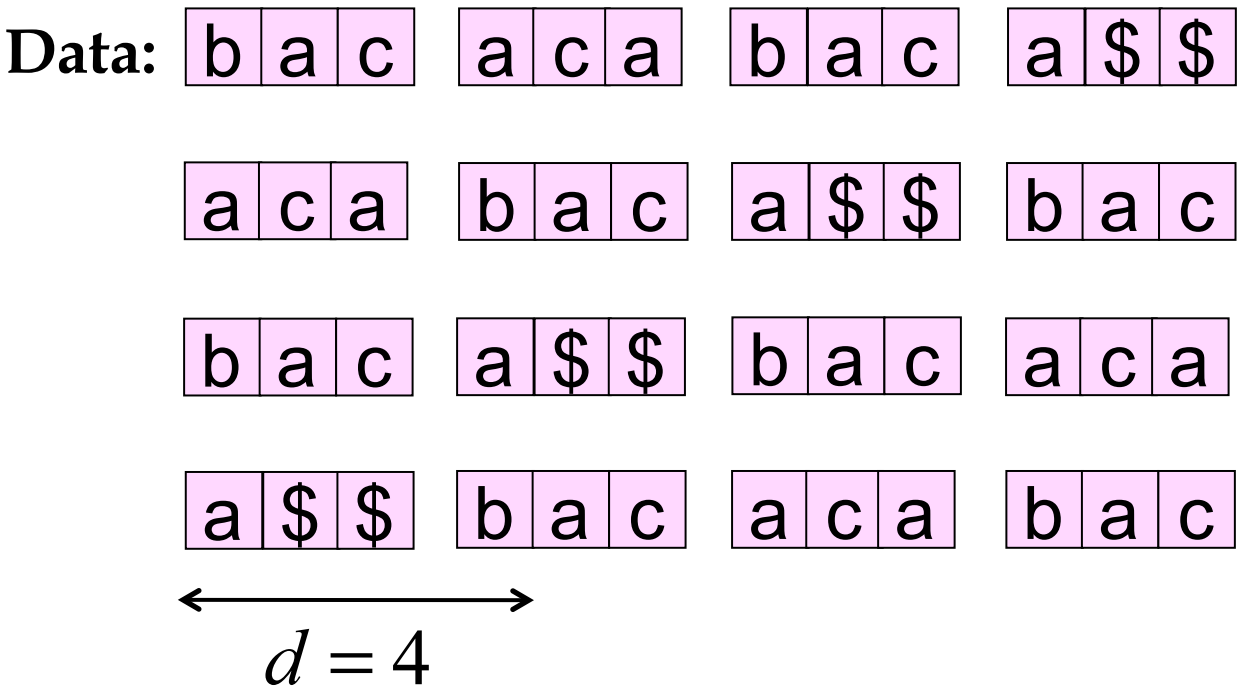
Forward Transformation

Generate all cyclic shifts of the blocks



Forward Transformation

Generate all cyclic shifts of the blocks

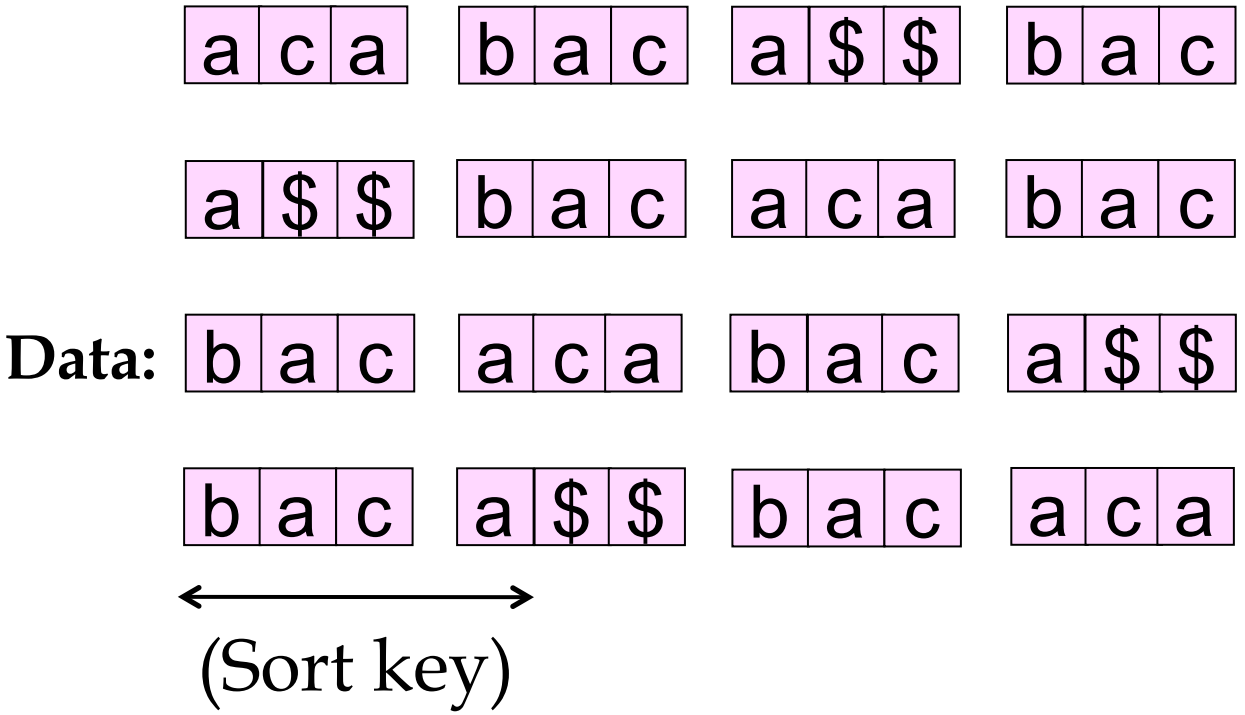


Sort the row vectors according to the left d symbols

($ is supposed to be the alphabetically largest symbol.)

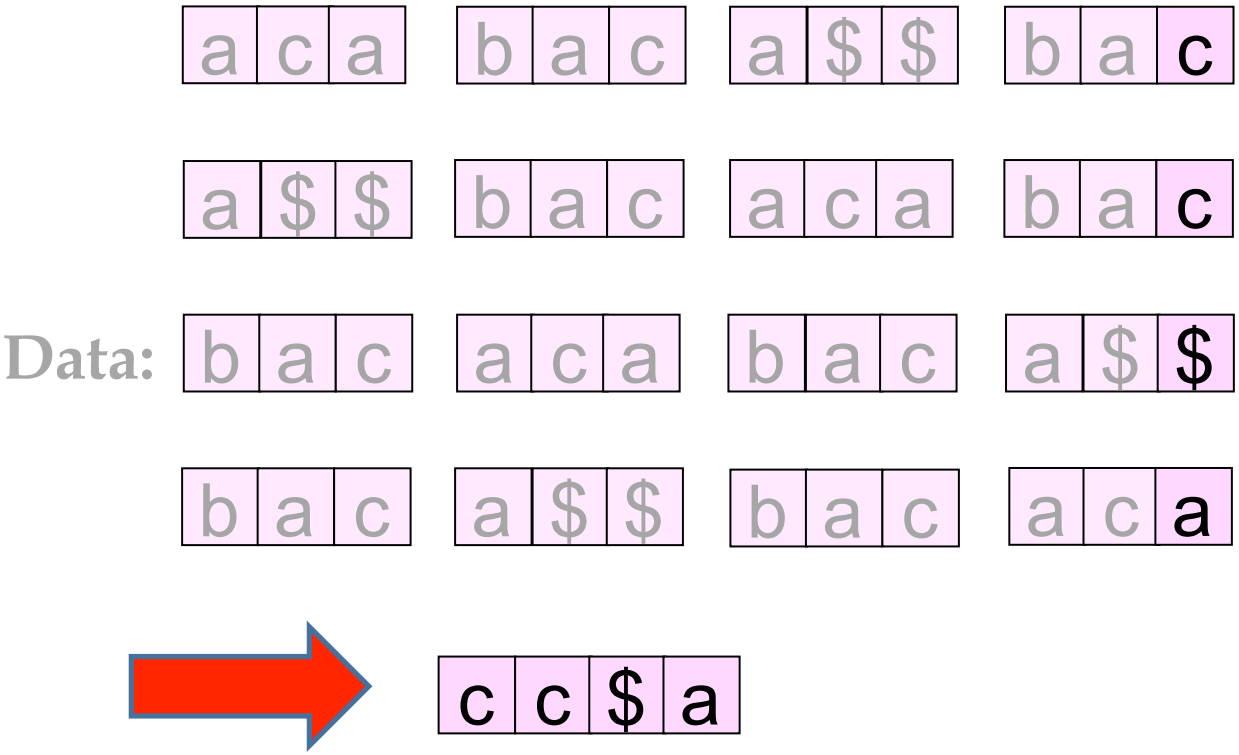
Forward Transformation

Sorted row vectors according to the left *d* symbols



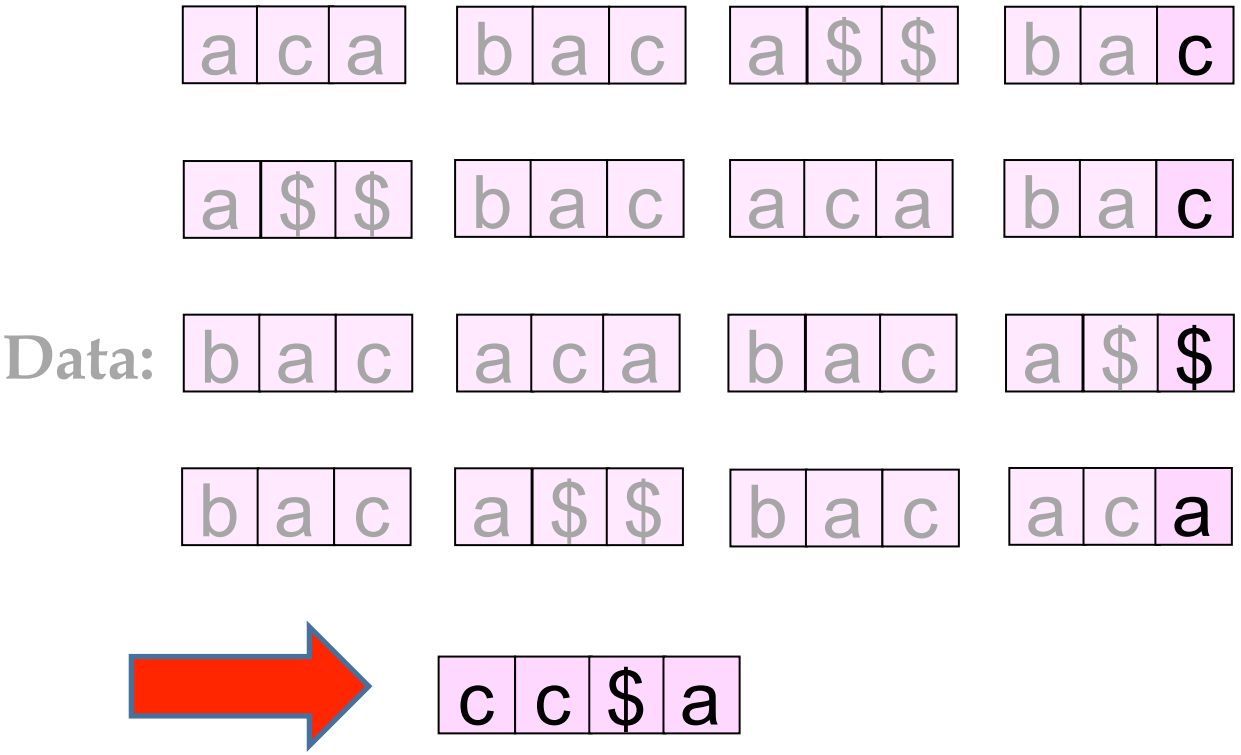
Forward Transformation

Output the symbols in the rightmost column



Forward Transformation

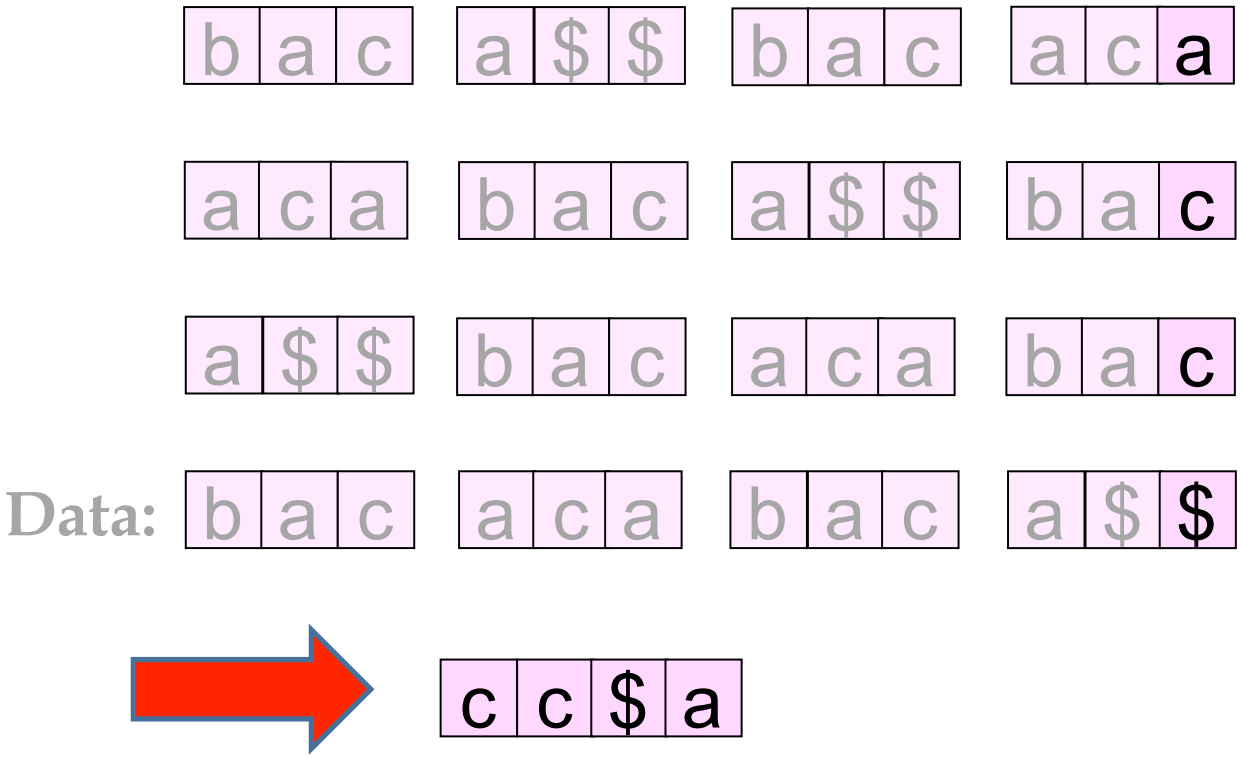
Output the symbols in the rightmost column



Sort again the row vectors according to the rightmost symbols. (Perform a **stable** sort.)

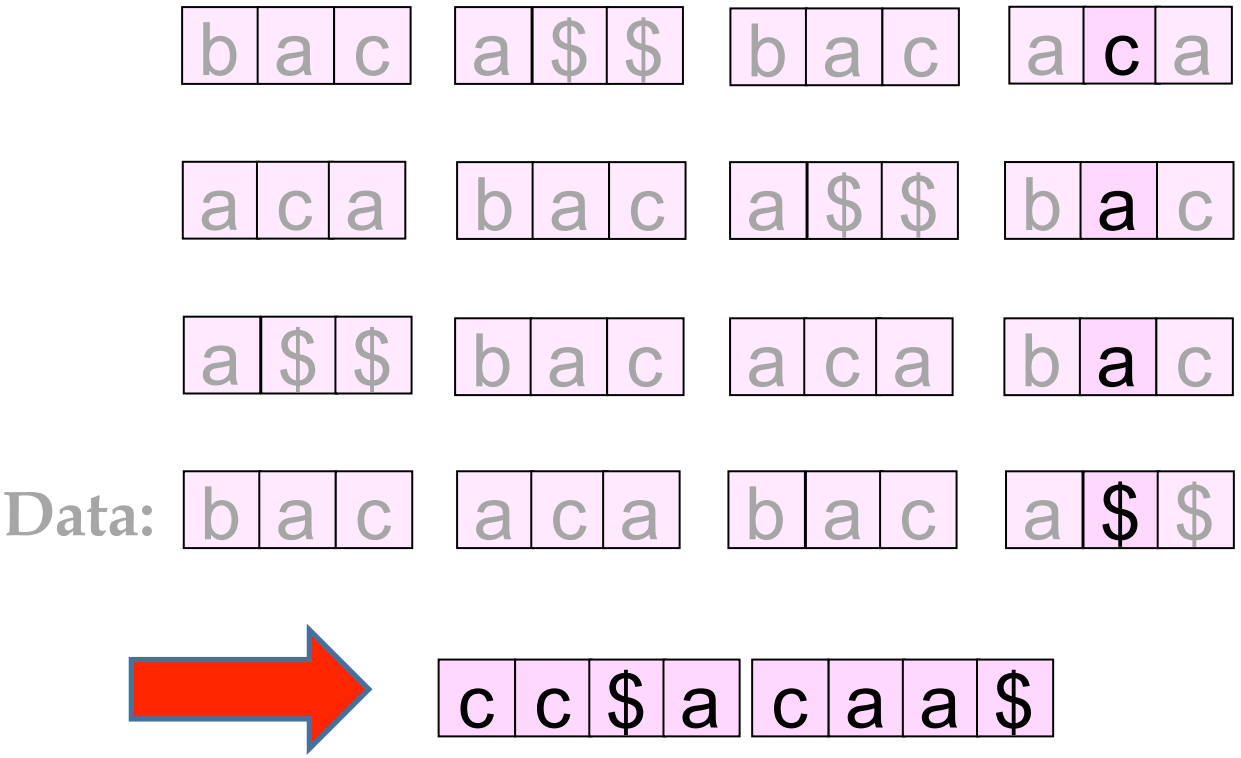
Forward Transformation

Perform a **stable** sort according to the rightmost symbols.



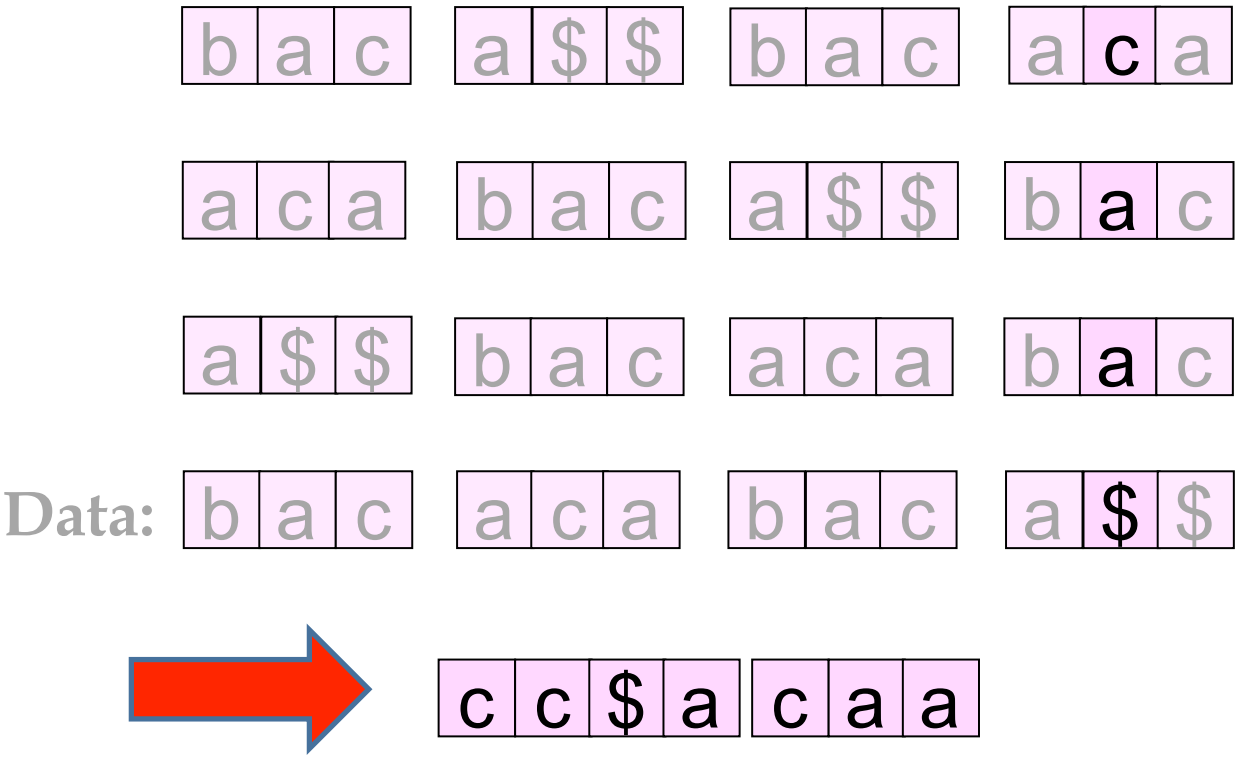
Forward Transformation

Output the symbols in the column, second from right.



Forward Transformation

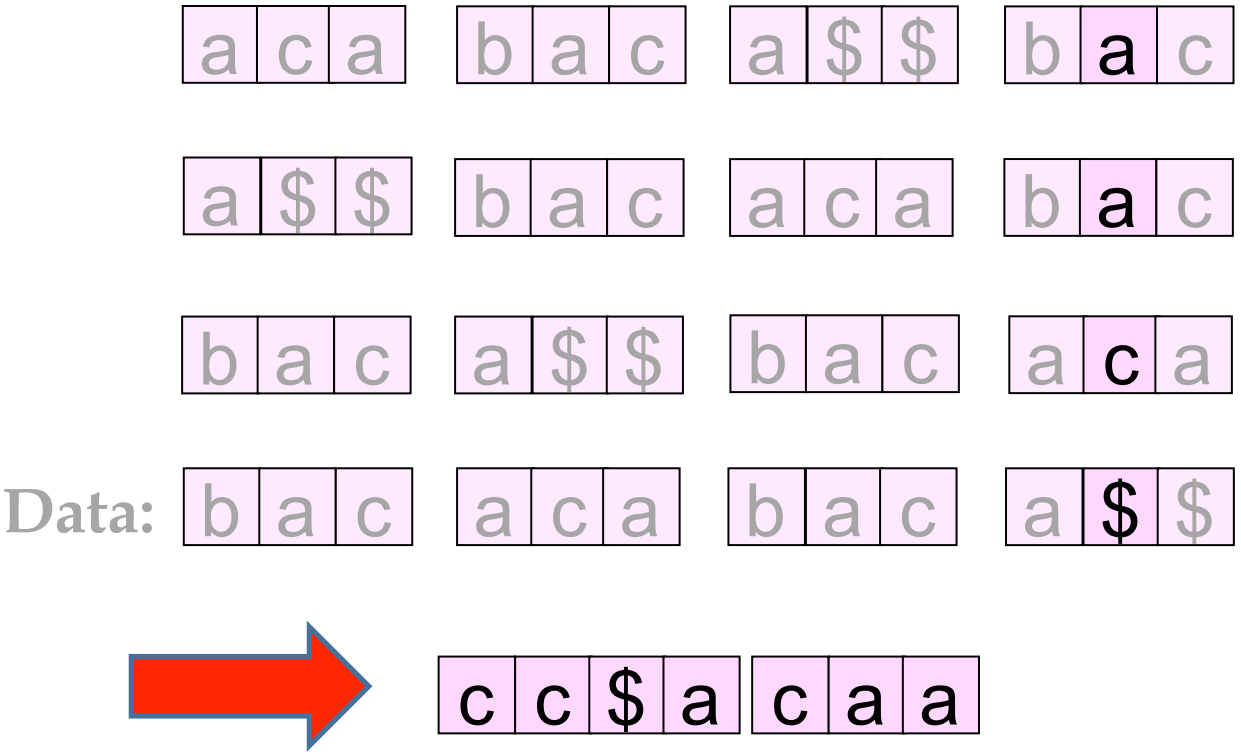
Output the symbols in the column, second from right.



Perform a stable sort again on the row vectors according to the **outputted** symbols.

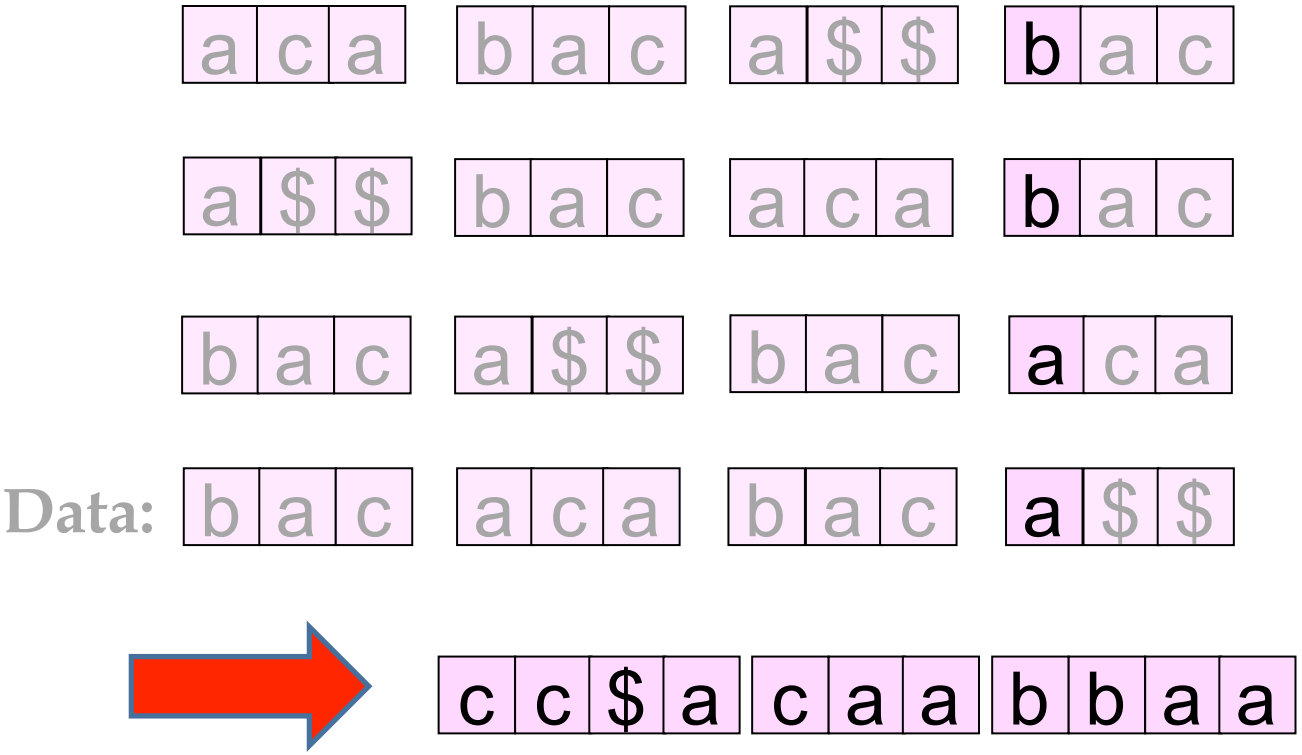
Forward Transformation

Perform a stable sort according to the outputted symbols.



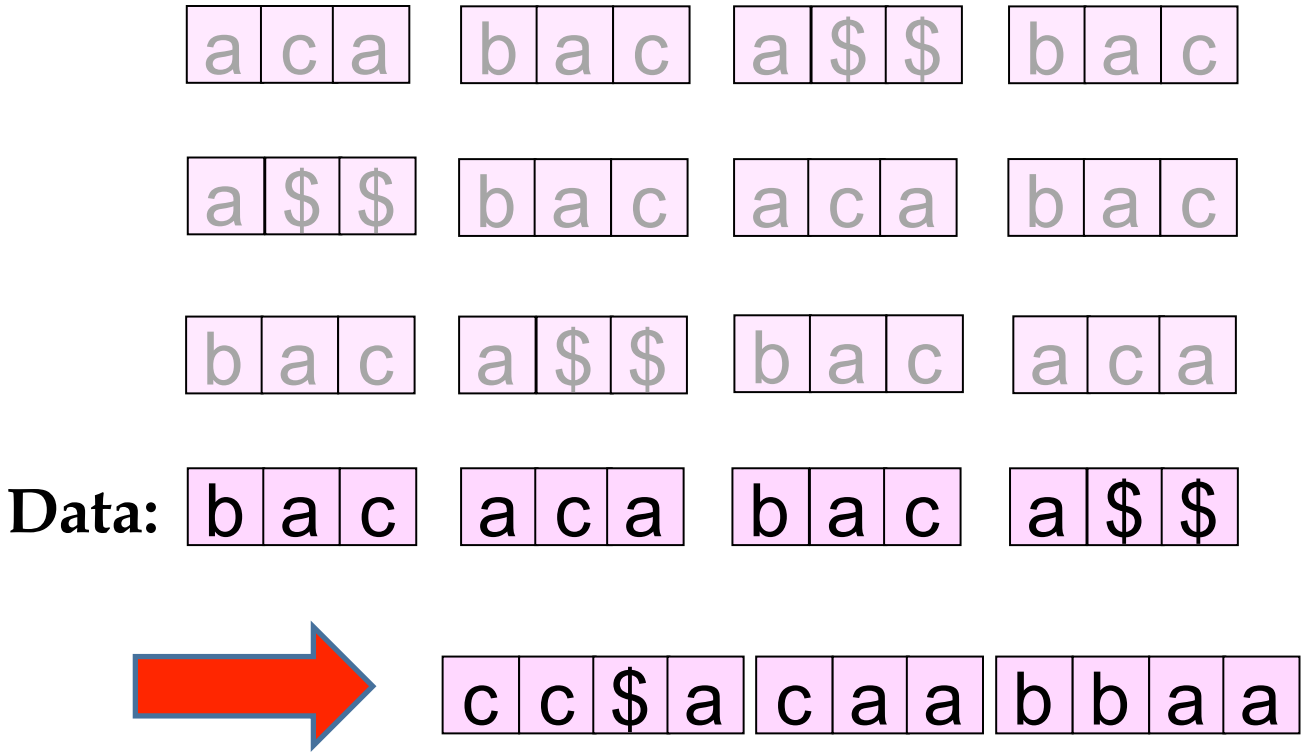
Forward Transformation

Output the symbols in the column, third from right.



Forward Transformation

Output the symbols in the column, third from right.



Transformation completed!

Inverse Transformation

1. **Main contribution of the present work**
2. **Linear time/space implementation**
   - Application of the technique of Nong-Zhang-Chan for the ST [CPM'08]
   - *Context switch vector* and *cycles*
3. **We need not reconstruct the matrix representation, instead**
4. **We use only auxiliary quantities all of size** $O(n/\ell)$

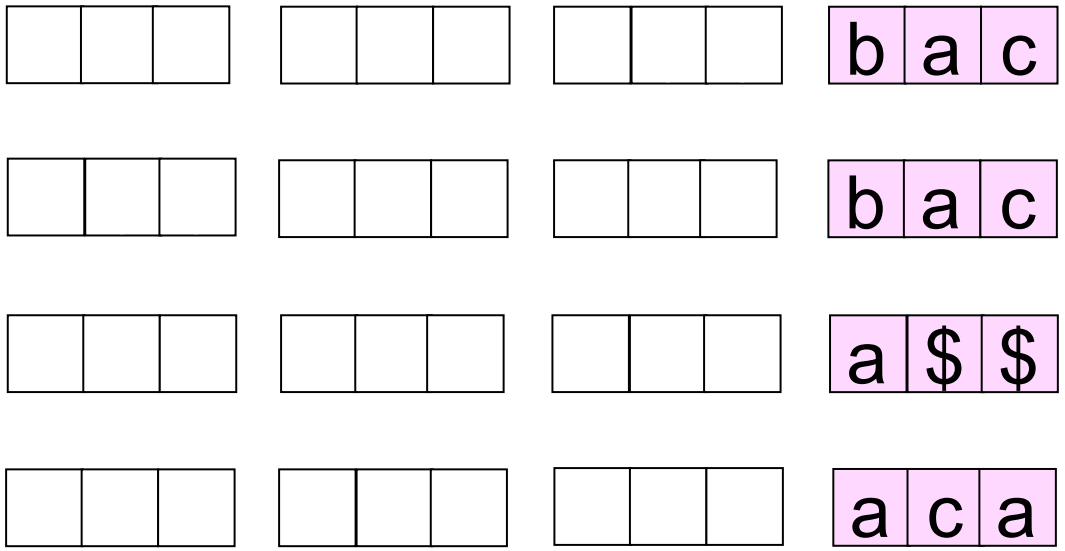
Inverse Transformation

Actual Procedure (*Conceptually quite simple*)

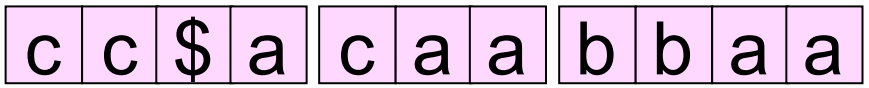
1. **Reconstruction of the output part**
2. **Computation of four auxiliary vectors**
   - Permutation vector $Q[1..n/\ell]$
   - Context switch vector $D[1..n/\ell]$
   - Index vector $T_d[1..n/\ell]$
   - Counter vector $C_d[1..n/\ell]$
3. **Restoring an original string**

Inverse Transformation

Reconstruction of the **output part**

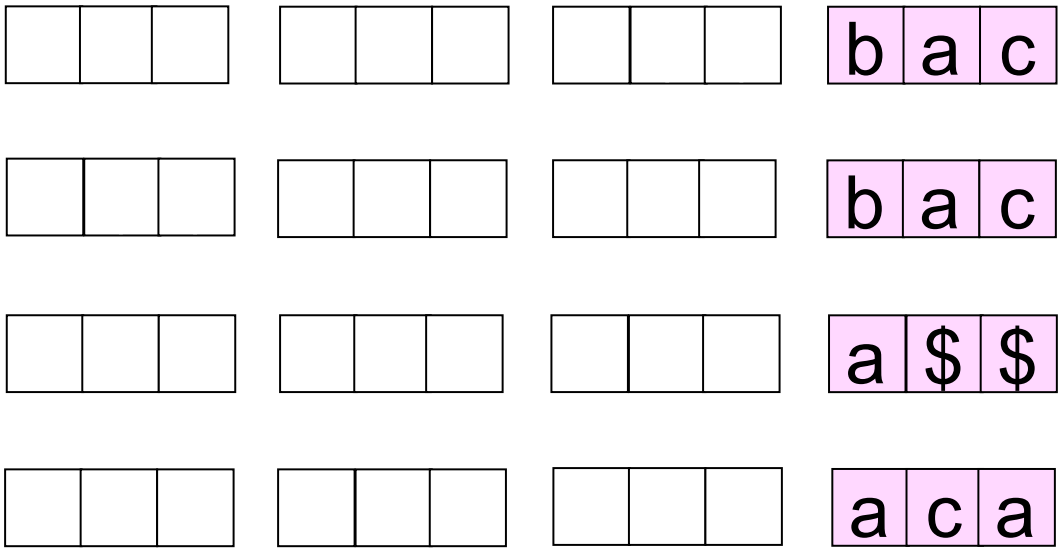


Transformed string:



Inverse Transformation

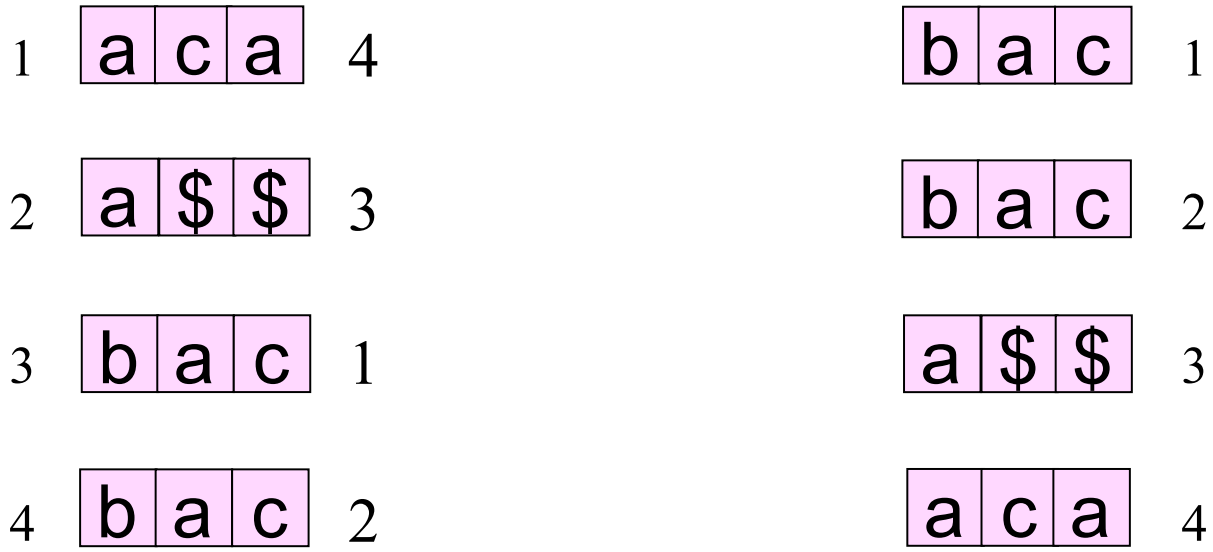
Reconstruction of the **output part**



Sort the obtained output part lexicographically

Inverse Transformation

Computation of auxiliary vectors



Inverse Transformation

i	$Q[i]$		$T_d[i]$		
1	a c a	4	3	b a c	1
2	a $ $	3	3	b a c	2
3	b a c	1	2	a $ $	3
4	b a c	2	1	a c a	4

Inverse Transformation

i		Q	D	C_d	T_d		
1	a c a	4	1	1	3	b a c	1
2	a $ $	3	1	1	3	b a c	2
3	b a c	1	1	2	2	a $ $	3
4	b a c	2	0	0	1	a c a	4

$C_d[i]$: Counter vector

$D[i]$: Context switch vector

Inverse Transformation

i	Q	D	C_d	T_d	
1	4	1	1	3	b a c
2	3	1	1	3	b a c
3	1	1	2	2	a $ $
4	2	0	0	1	a c a

$i :=$ index of the row that includes \$;

Repeat :

Stack the ith row;

$i := T_d[i]$;

$C_d[i] := C_d[i] - 1$;

$i := i + C_d[i]$;

Inverse Transformation

i	Q	D	C_d	T_d	
1	4	1	1	3	b a c
2	3	1	1	3	b a c
3	1	1	2	2	a $ $
4	2	0	0	1	a c a

$i :=$ index of the row that includes \$;

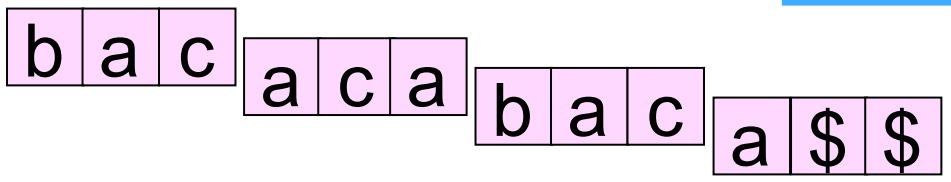
Repeat :

Stack the ith row;

$i := T_d[i]$;

$C_d[i] := C_d[i] - 1$;

$i := i + C_d[i]$;

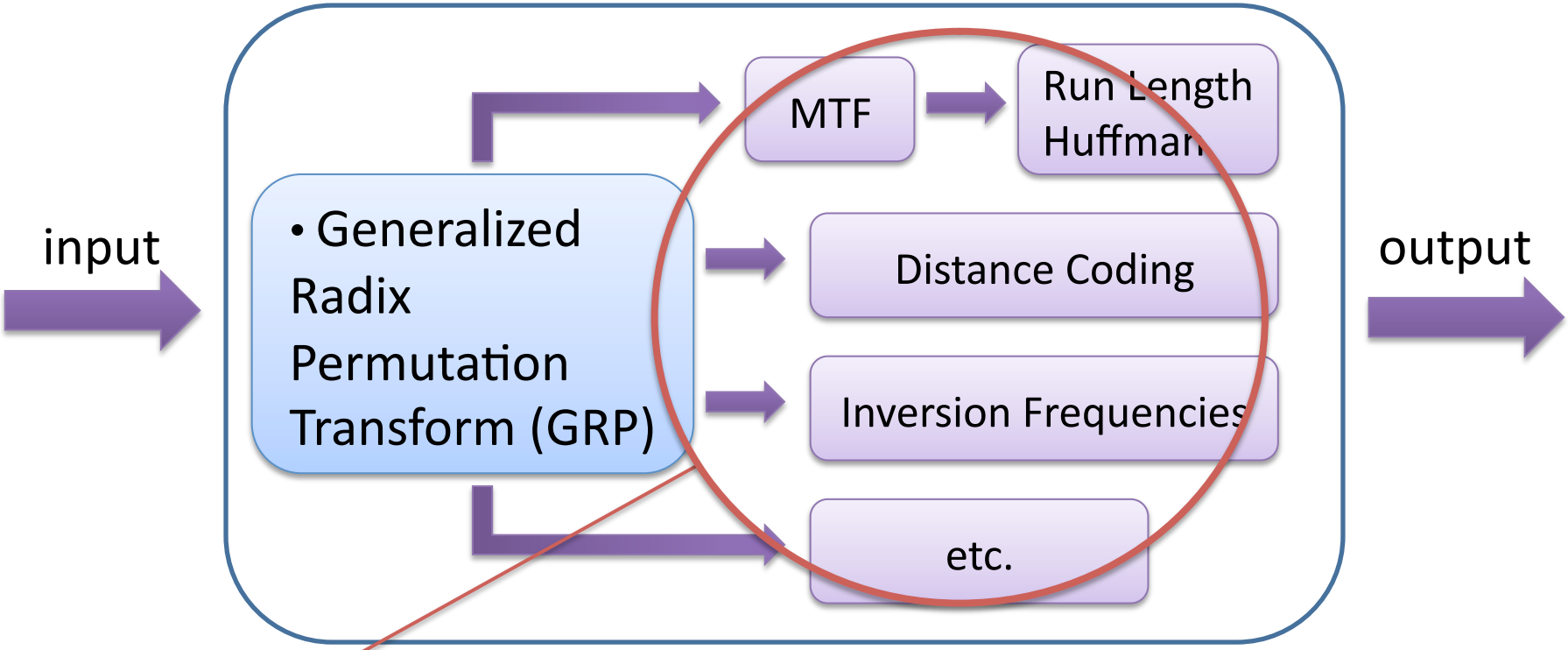


Original string recovered!

Outline

1. What is the *GRP* Transform
   - Sort-based Transforms for Lossless Data Compression
   - Existing Transforms -> Parametric Generalization
2. Proposed Transform
   - Forward Transformation
   - Faster Implementation of Inverse Trans.
3. **Compression Experiments**
4. Conclusions and Future Work

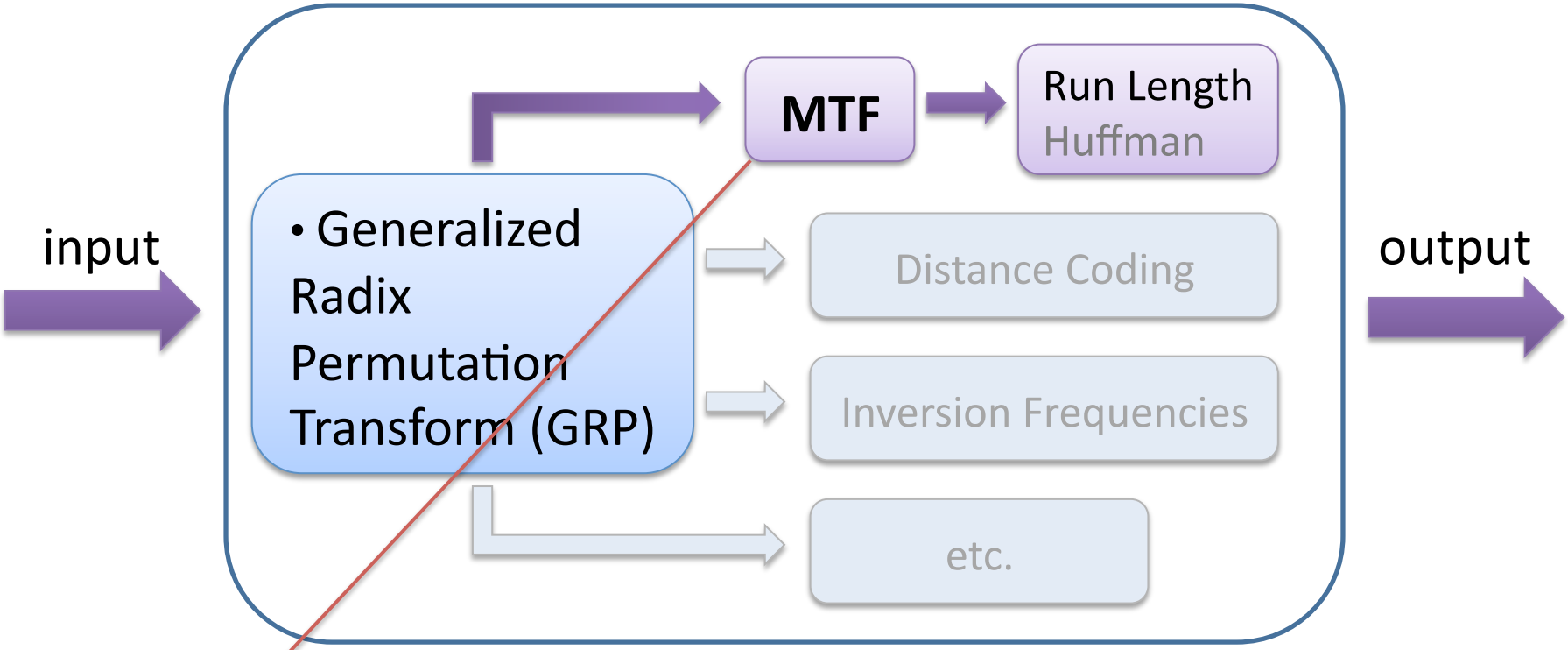
Compression Experiments



[Second-Step Encoders for Transformed String]

Actual encoders for transformed text (LGT: Local to Global Transform, Adjeroh-Bell-Murkherjee '08)

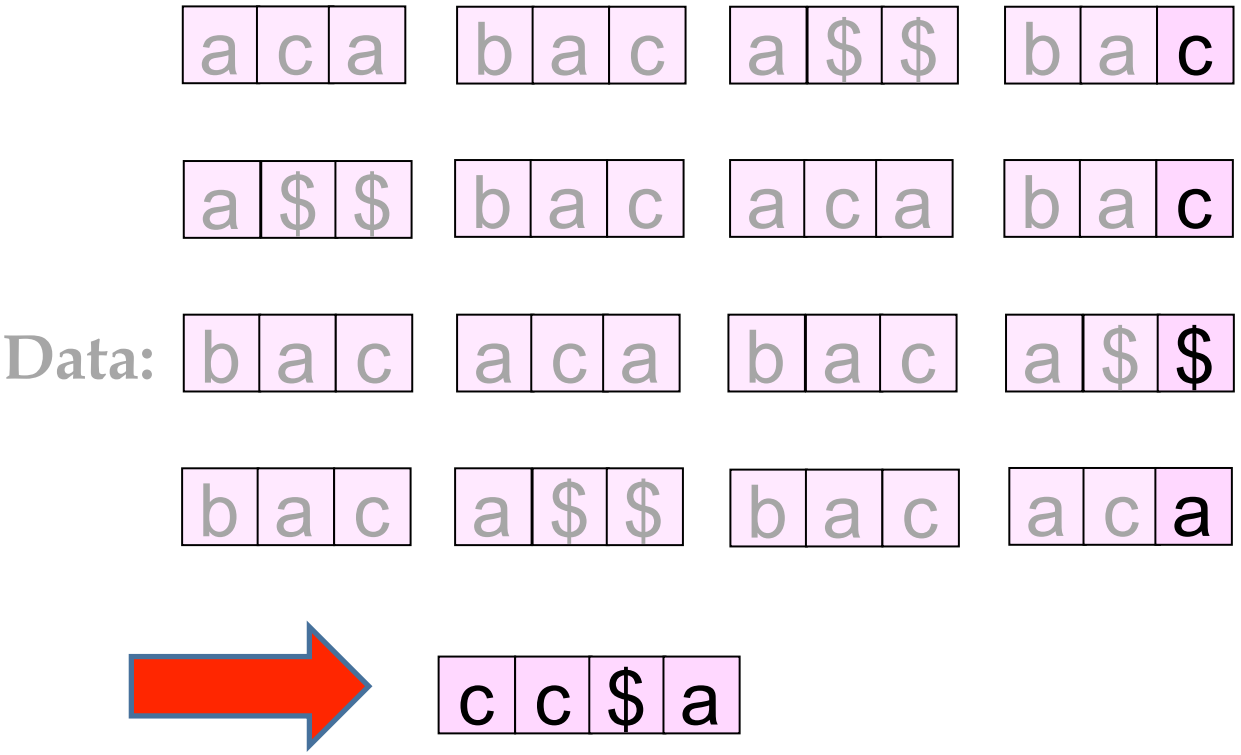
Compression Experiments



We have modified the MTF (Move-to-Front) heuristics so that it takes advantage of the output characteristics of the GRP transform. (**AMTF: Adaptive MTF**)

Output Characteristics

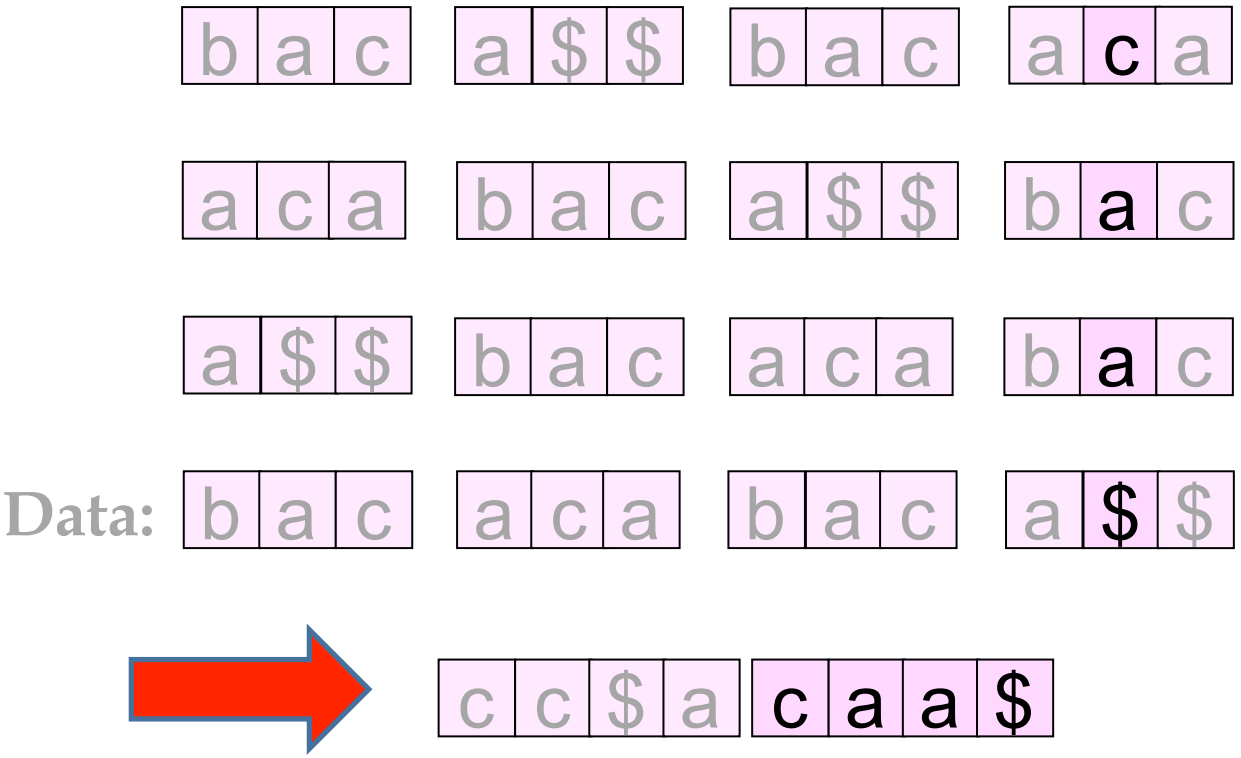
The symbols in the rightmost column:



The situation is the same as in the BWT in the sense that their contexts are not observable.

Output Characteristics

The symbols in the second column, second from right:



However, the situation is changed. **We can observe their immediate right symbols.**

Output Characteristics

When we encode the symbols in the *k*th column, *k*th from right,

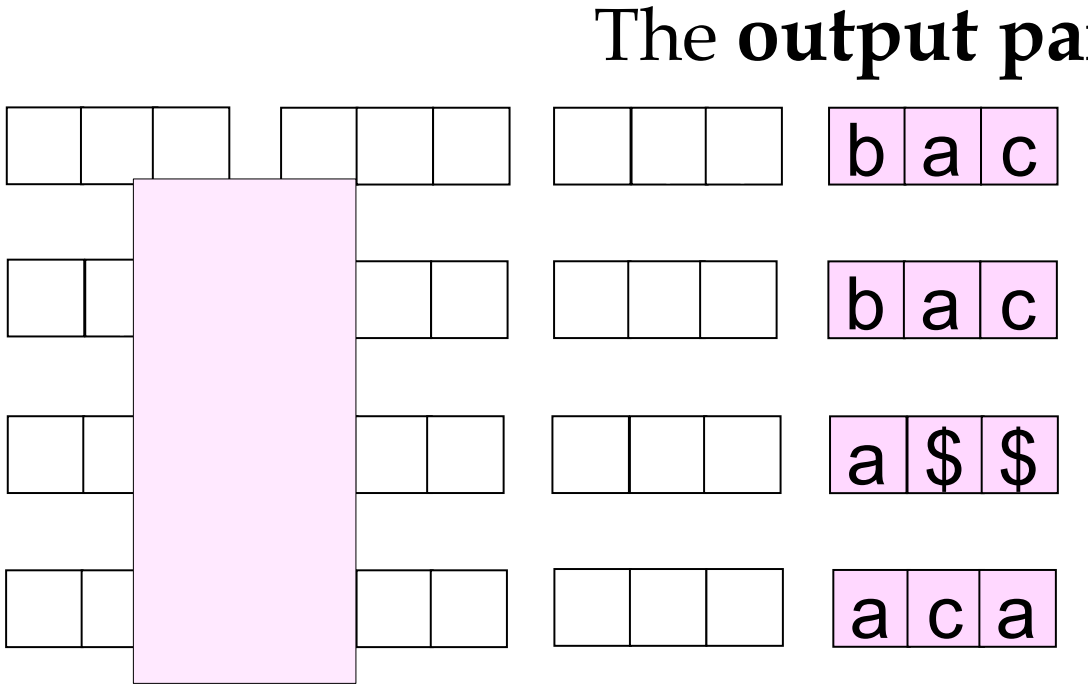
we can use their $k-1$ right symbols as their immediate contexts.

AMTF = MTF + Contextual information

adaptively changes the MTF list when the context changes according to:

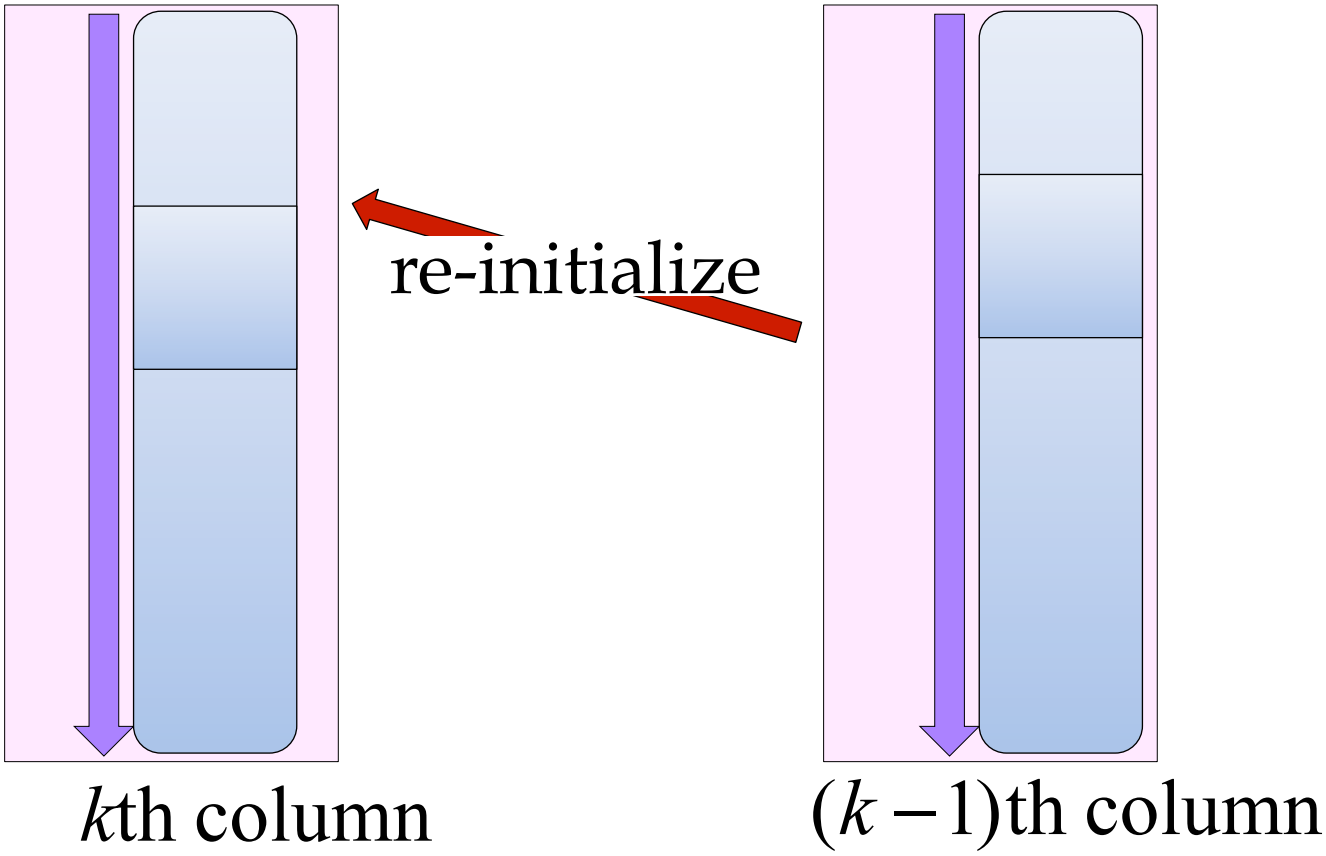
When the context changes completely, the MTF list of symbols is initialized so that the order of symbols corresponds to the frequency of symbols occurred in shorter similar contexts.

AMTF: Adaptive MTF



AMTF: Adaptive MTF

Encoding the *k*th column

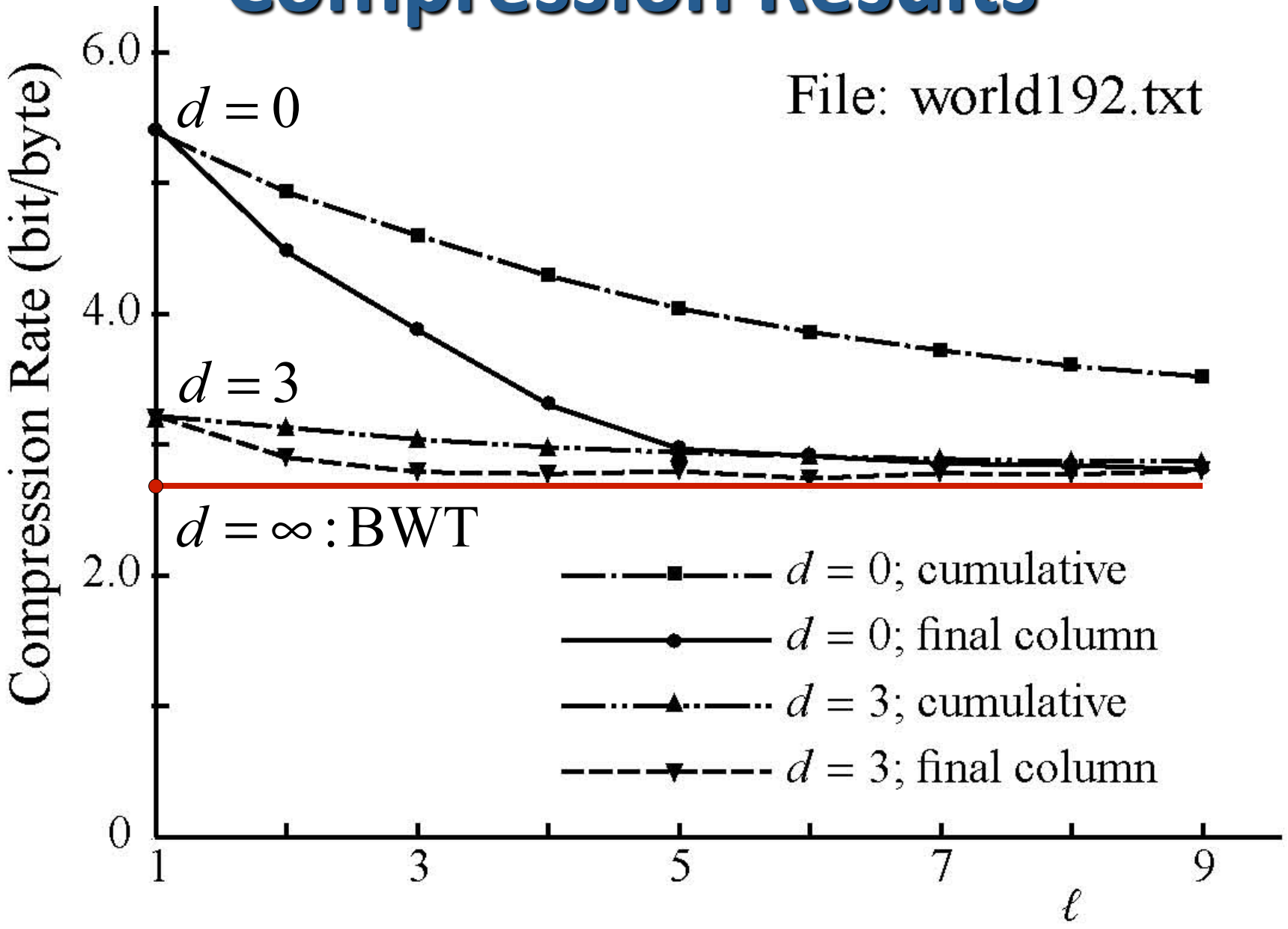


The MTF list of symbols is initialized using the symbol frequencies in shorter similar contexts.

Compression Results

		File	cp.html	alice29	lcet10	plrabn12	ptt5	kennedy	world192	bible	E.coli
d	ℓ	Size	24603	152089	426754	481861	513216	1029744	2473400	4047392	4638690
6	1	MTF	2.68	2.80	2.76	3.16	1.00	0.88	2.73	2.46	2.28
∞	1	MTF	2.68	2.80	2.75	3.15	0.90	1.22	2.69	2.45	2.27
3	3	MTF	3.22	2.96	2.98	3.24	1.04	0.70	3.00	2.68	2.20
3	3	AMTF	3.18	2.95	2.97	3.24	1.04	0.70	2.99	2.68	2.20
6	3	MTF	3.20	2.89	2.90	3.20	1.01	0.90	2.82	2.58	2.18
6	3	AMTF	3.17	2.89	2.89	3.20	1.00	0.90	2.82	2.57	2.18
3	4	MTF	3.33	3.03	2.99	3.23	1.03	0.88	2.98	2.69	2.16
3	4	AMTF	3.26	3.01	2.98	3.22	1.03	0.87	2.97	2.68	2.16
6	4	MTF	3.32	2.98	2.94	3.19	1.00	0.99	2.85	2.60	2.17
6	4	AMTF	3.25	2.97	2.93	3.18	1.00	0.98	2.83	2.60	2.17
0	3	AMTF	4.69	4.31	4.30	4.33	1.11	2.38	4.58	4.10	2.14
1	3	AMTF	3.74	3.53	3.58	3.67	1.04	1.33	3.75	3.34	2.18
10	3	AMTF	3.17	2.88	2.89	3.20	0.97	1.16	2.79	2.57	2.18

Compression Results

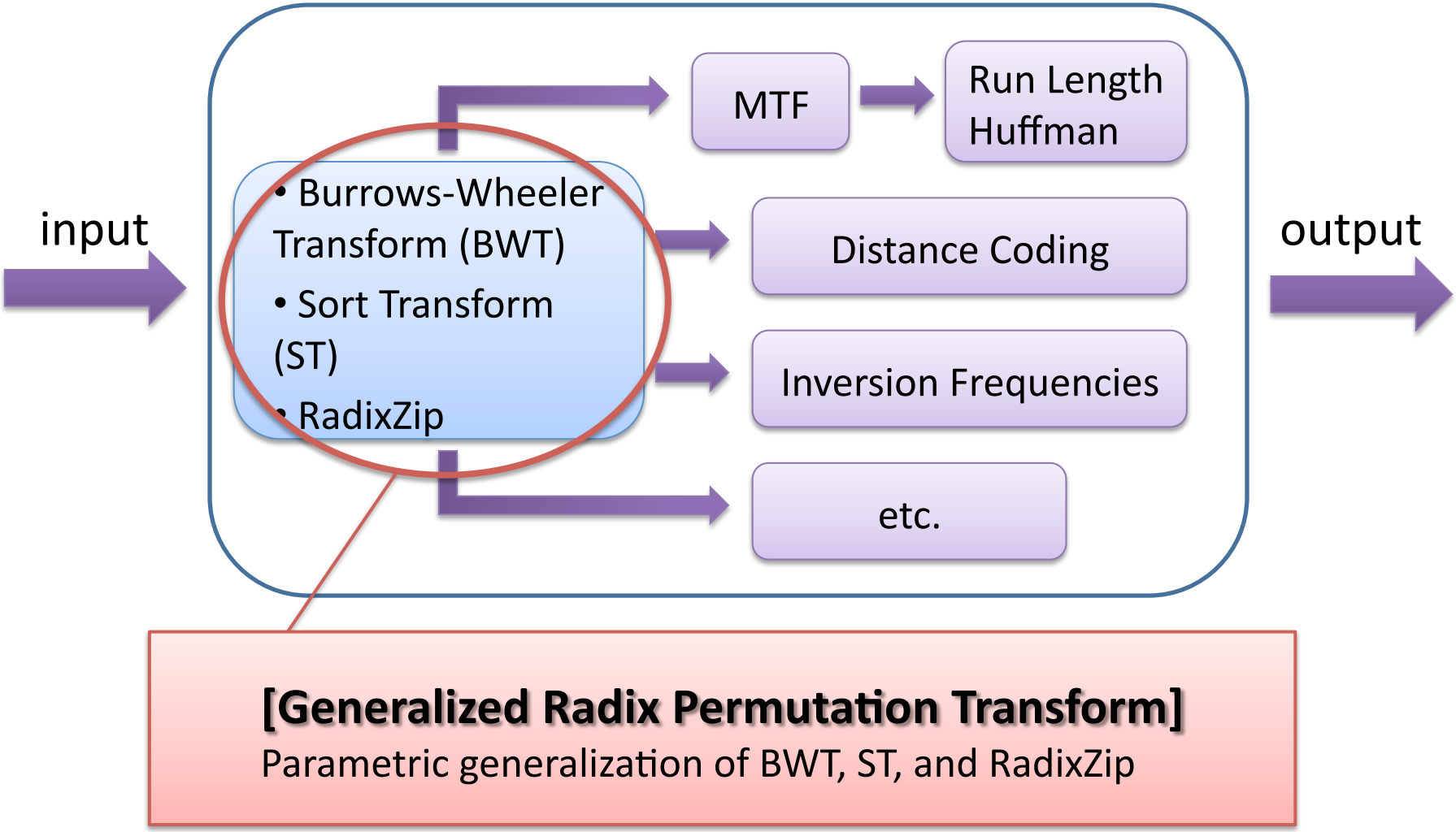


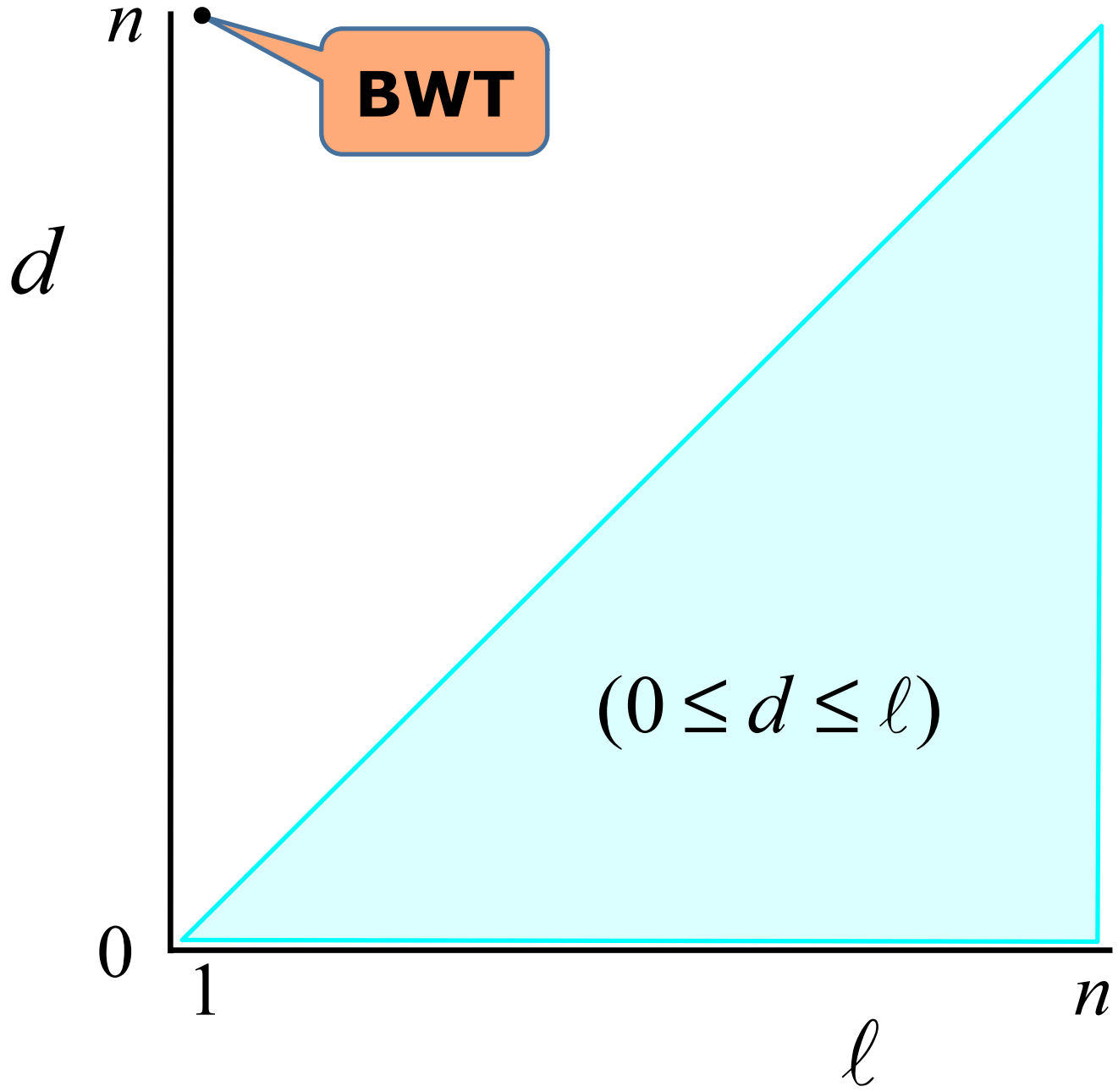
Conclusions and Future Work

Transform	Context Lengths
We developed the ***Generalized Radix Permutation*** transform [2009].	$d, d+1, \ldots, d+\ell-1$

- We have extended it to a more general version (applicable to arbitrary parameter values).
- Linear time implementation of the inverse transformation
- Development of a new second-step encoder (AMTF)
- Its compression performance is slightly better than the conventional MTF when combined with the GRP transform.
- Development of other second-step encoders

Lossless Compression with Sort-Based Transforms




n
BWT
d
$(0 \le d \le \ell)$
0
1
n
ℓ

Compression Experiments