New Perspectives on the BWT

Hideo Bannai

hdbn.dsc@tmd.ac.jp

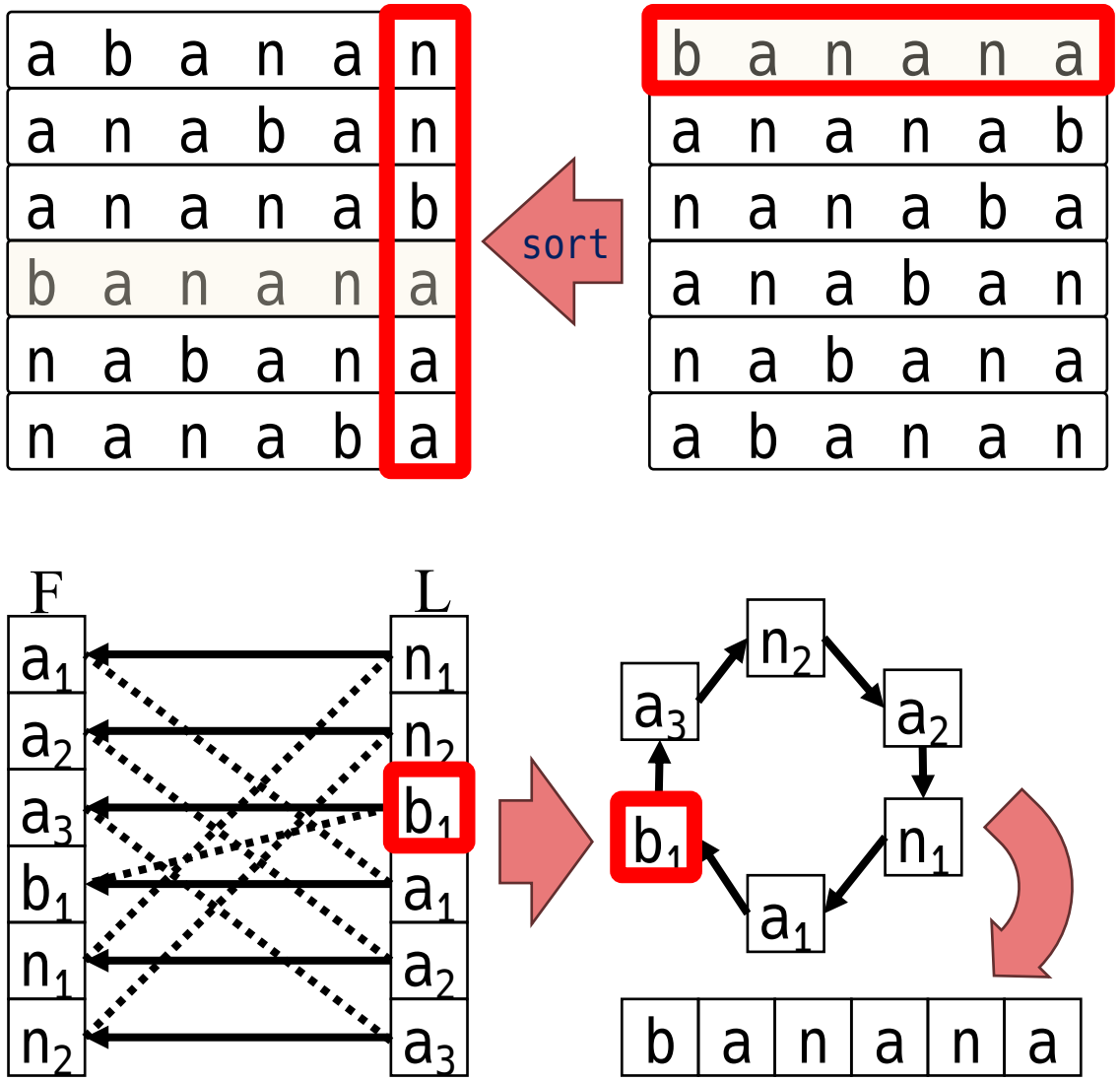
Institute of SCIENCE TOKYO

September 8-11, 2025 SPIRE 2025 @ London, UK

Burrows-Wheeler Transform (BWT) [Burrows&Wheeler 1994]

- An "invertible" transform
  - The last (or previous) symbol of all rotations, in lex. order of the rotations
    - BWT(banana) = nnbaaa
- The inverse transform: via LF-mapping induced by a stable sorting of the symbols
  - We need more information, e.g.
    - index of starting position or
    - end-of-string marker $

  to get original string.
  BWT(banana) = (nnbaaa, 3)
  BWT^{-1}((nnbaaa, 3)) = banana

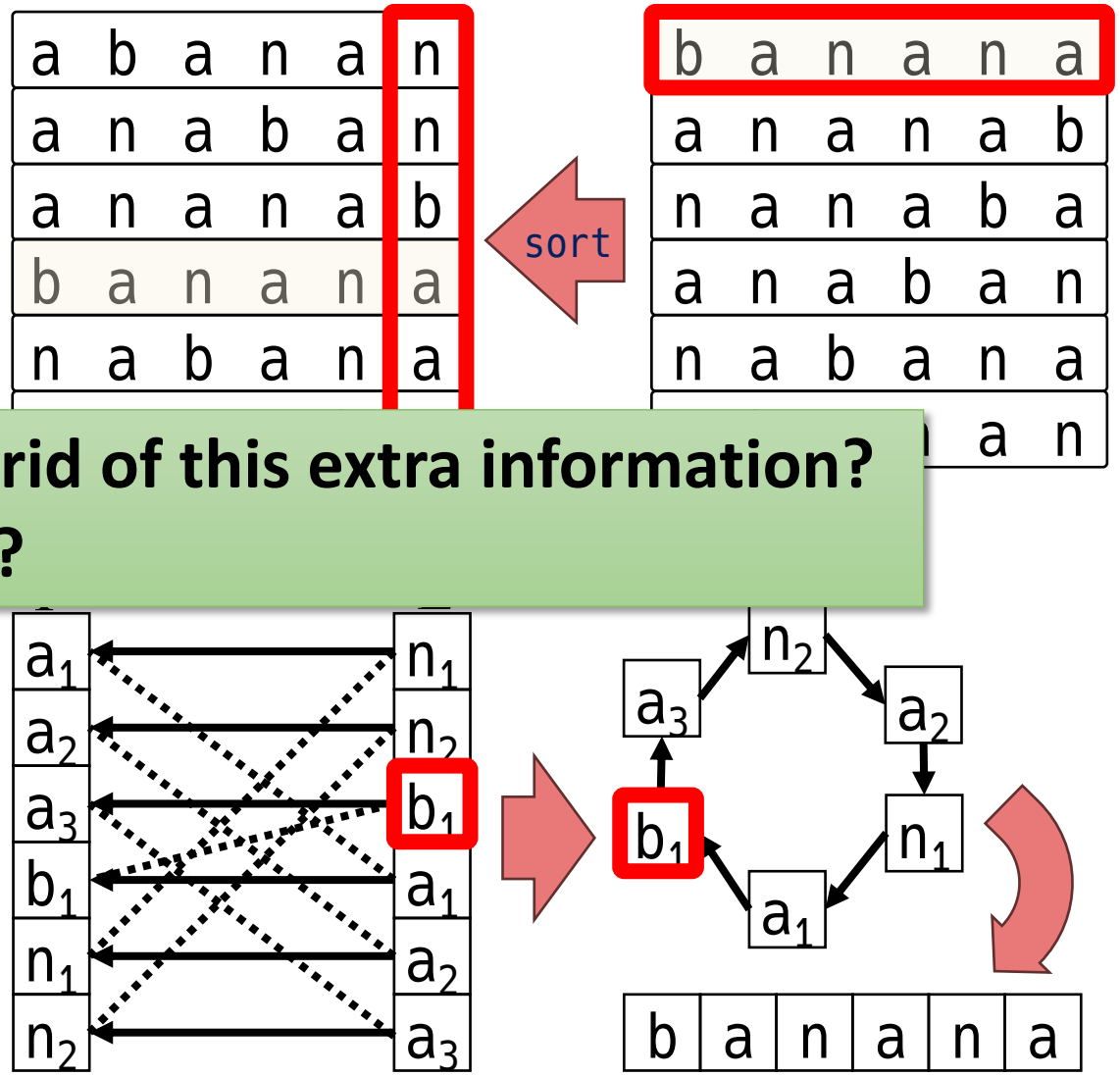


Burrows-Wheeler Transform (BWT) [Burrows&Wheeler 1994]

- An "invertible" transform
  - The last (or previous) symbol of all rotations, in lex. order of the rotations
    - BWT(banana) = nnbaaa
- The i… via L… stable sorting of the symbols
  - We need more information, e.g.
    - index of starting position or
    - end-of-string marker $

  to get original string.
  BWT(banana) = (nnbaaa, 3)
  BWT^{-1}((nnbaaa, 3)) = banana

1. Can we modify BWT to get rid of this extra information?
2. If yes, how meaningful is it?



This Talk

Revisit BWT as function on strings from the perspectives of both bijectivity and compression, introducing some of our recent results on BWT and Bijective BWT.
(ignoring things like algorithms, text indexing, practicality)

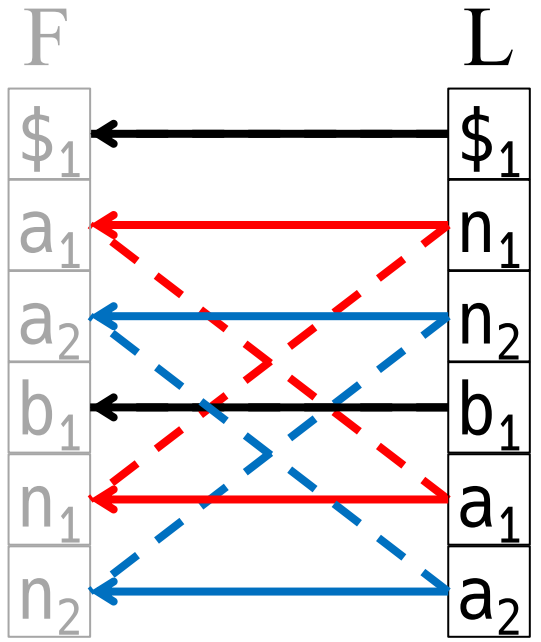
For references, see:

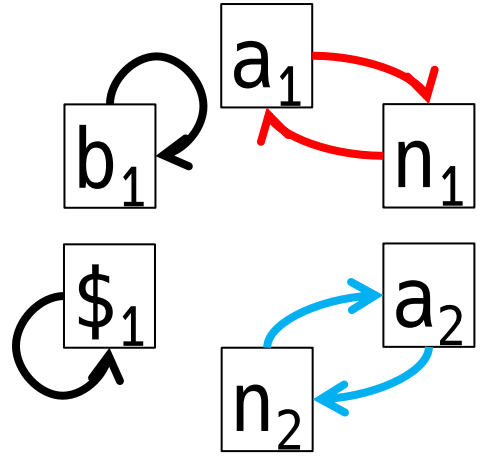
B., Köppl, Lipták: **A Survey of the Bijective Burrows-Wheeler Transform.** The Expanding World of Compressed Data: A Festschrift for Giovanni Manzini's 60th Birthday: 2:1-2:26, 2025.

Common misunderstanding on Bijectivity of BWT

Isn't BWT bijective if we add the index or End-of-String $?

- Invertibility only implies **injectivity**, not surjectivity
- LF-mapping can lead to multiple cycles if applied to strings not in BWT image (even with starting pos or $)





We would like a bijective transform
BBWT: $\Sigma^n \rightarrow \Sigma^n$, not so different from BWT.

Bijective BWT (BBWT) a.k.a. BWT *Scottified* (BWTS)

History of Bijective BWT (1) [Scott 2007]

Can be found at:
https://groups.google.com/g/comp.compression

2007/12/13	**A truely BIJECTIVE BWT is here!**

I just created what I think is the first truely bijective BWT through my Scottification process. My code is a mod of Marks BWT and should function in most cases as a drop in replacement. I call it BWTS. my bijective BWT at:

http://bijective.dogma.net/bwts.zip (accessible via https://web.archive.org/)

it does what BWT should have done the transform with out an index. N bytes to N bytes with only the order changed. I don't think anyone else has a practical working bijective BWT transform it took a lot of trail and error to make it so that on the average it will lead to bijective BWT compression with out the stupid INDEX and it will lead to better compression on the average.

Thank You
Merry Christmas
David A. Scott

History of Bijective BWT (1) [Scott 2007]

Can be found at:
https://groups.google.com/g/comp.compression

2007/12/13	**A truely BIJECTIVE BWT is here!**

I just created what I think is the first truely bijective BWT through my Scottification process. My code is a mod of Marks BWT and should function in most cases as a drop in replacement

Bio of Scott for an article on his website (dated 1999) :

https://web.archive.org/web/20011122075435/http://bijective.dogma.net/dspaper.htm

David A. Scott is not a writter and this is his first attempt at writting an article of this type. I graduated from Arizona State University with a BSEE in Electrical Engineering with a major in Fields and Waves. This was the closest thing to a Math Degree without all the English requirements. My main carreer was writting and debugging algorithms for the NAVY at CHINA LAKE CA. form 1970 to 1996. While at CHINA LAKE I was sent to The University of Southern California in 1972 to recieve my MSEE in Electrical Engineering the major field of study was control theroy. I am currently retired living in El Paso, Texas; in the near future I may pursue more education in Jaurez to become a medical doctor.

History of Bijective BWT (1) [Scott 2007]

Can be found at:
https://groups.google.com/g/comp.compression

20(

I j
M
I c

ht

it
or
tra
bi
on the average.

Thank You
Merry Christmas
David A. Scott

I suspect this is impossible. In order to achieve an improvement in compression, BWT needs to map less-compressible vectors onto more-compressible vectors.
However, there are fewer of the latter. It gets away with this by mapping multiple source vectors onto the same output vector, and then adding an auxiliary index such that the process is reversible. BWT as such cannot work without that index.
Now it's entirely possible that you've created a new transform that encourages "better compression on the average" from the back end, but the fact that it is better is almost certainly unrelated to a lack of an index, and is probably harmed by that lack.
Care to outline the algorithm here?

History of Bijective BWT (1) [Scott 2007]

Can be found at:
https://groups.google.com/g/comp.compression

20(

I j
M
I c

ht

it
or
tra
bi
on the average.

Thank You
Merry Christmas
David A. Scott

I suspect this is impossible. In order to achieve an improvement in compression, BWT needs to map less-compressible vectors onto more-compressible vectors.
However, there are fewer of the latter. It gets away with this by mapping multiple source vectors onto the same output vector, and then adding an auxiliary index such that the process is reversible. BWT as such cannot work without that index.
Now it's entirely possible that you've created a new transform that encourages "better compression on the average" from the back end, but the fact that it is better is almost

The numbers in your READ.ME file are certainly pulled out of your arse, I'm disinclined to read any more after that.

History of Bijective BWT (1) [Scott 2007]

Can be found at:
https://groups.google.com/g/comp.compression

20

I j
M
I c

ht

it
or
tra
bi
or

Th
M
Da

…
The reason that I am suspicious of David's claim is just that I don't see how the transform can be reversible without additional information.
Of course, I'm reluctant to flag it as impossible. If you were naively reading the original paper without having ever heard of it before, I would be very surprised if you were to get through the description of the transform and then say "Aha! This would be totally reversible if I just had a pointer to the new position of the last character in the input sequence!" Well, maybe you would, but I sure didn't, and it still seems pretty remarkable to me that it works at all.
I haven't even looked at David's code yet but I was hoping he could give a words-only description of what he did to achieve his claim. But I'll give David a break, he says his attempts at exposition are rarely successful, and that's not likely to change overnight. What worries me is my own personal conviction that there is a bijection between clear writing in spoken language and clear writing in code.

History of Bijective BWT (2) [Scott 2009a]

Can be found at:
https://groups.google.com/g/comp.compression

2009/07/08	**The paper "A Bijective String Sorting Transform"**

Look it's obvious I did not write this. Yossi did what are the odds it gets accepted. I don't know so far I have batted zero on any paper actually being published with my name on it. I actually think the transform has more application in crypto than compression. But there are niche places where it will do very well for compression time will tell.

No matter how far his goes I would like to thank Yossi for trying to follow my rantings on the phone while he was thousands of miles away. I would lilke to thank Mark Nelson who planted the dream of this many years ago. There are many others most of whom I never meet and most likely never will. I don't like being in public but if the paper is accepted I guess I have to buy a tie and a clown suit so I can mingle with people.

http://bijective.dogma.net/00yyy.pdf (accessible via https://web.archive.org/)

Joseph Yossi Gil, David Allen Scott: A Bijective String Sorting Transform, July 7, 2009.
Uploaded to Scott's website, later uploaded to arXiv on Jan 15, 2012. https://arxiv.org/abs/1201.3077

History of Bijective BWT (3) [Scott 2009b]

Can be found at:
https://groups.google.com/g/comp.compression

2009/09/11	**BIJECTIVE BWTS PAPER**

Many of you know that a paper was submitted to the SODA conference on the BWTS. I did not write it but worked with a guy good at writing.

It was rejected. I am not surprised! here is sentence from one of the reviews.

"Of course one can easily achieve invertability in Burrows-Wheeler transform, by appending a special character to the input string. "

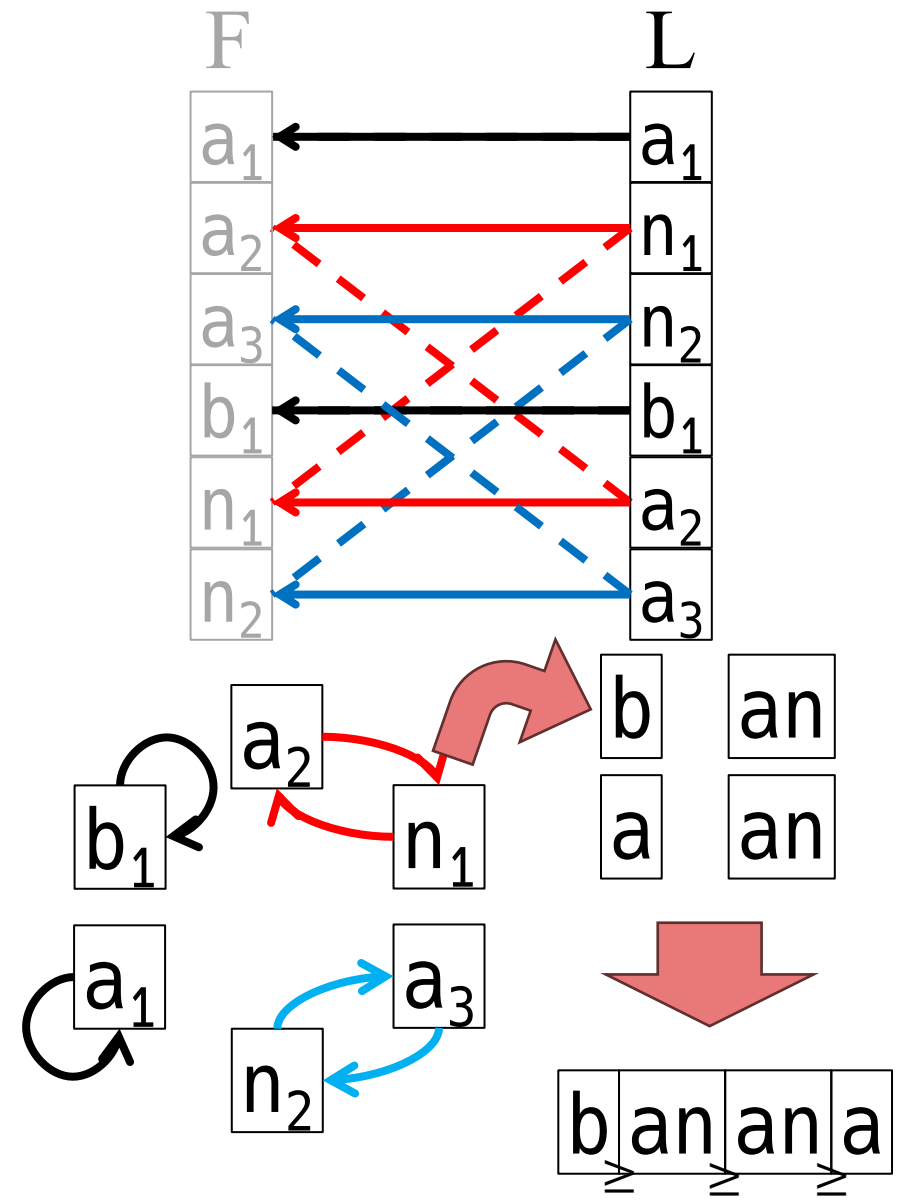
Golly Gee why didn't I think of that of course that's the solution to every thing.

Unfortunately, it seems the BBWT paper by Scott was not subsequently published in peer reviewed proceedings/journal.

(but maybe this is a good sign, considering the parallel with the original BWT paper ☺)

(inverse) Bijective BWT (BBWT) [Scott 2007]

- BWT is **not** bijective on set of all strings
- LF-mapping can lead to multiple cycles (even with starting pos or $)
  - actually, it is a bijection between a **string** and a **multiset of cyclic primitive strings**
    [Gessel&Reutenauer 1993][Crochemore et al. 2004]
- We can make it a bijection on strings by
  - Taking lex. smallest rotation of each cycle (⇔ Lyndon words [Lyndon 1954])
  - Concatenating in non-increasing lex order (⇔ Lyndon factorization [Chen-Fox-Lyndon 1958])



primitive: not an integer power of a shorter string

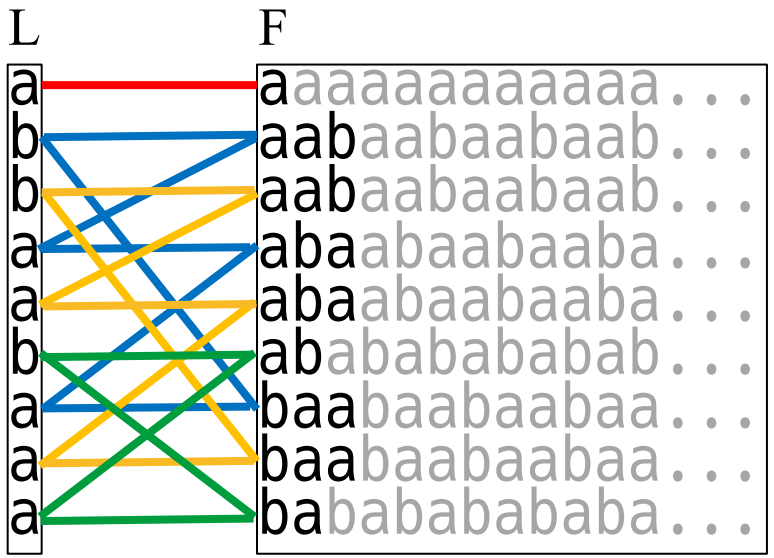
Defining BBWT (Forward Transform)

- A string is a *Lyndon word* if it is primitive and lex. smallest rotation.
- $T = L_1^{e_1} \cdot \ldots \cdot L_\ell^{e_\ell}$ is the *Lyndon factorization* of T iff:
  - *Lyndon factor* L_i $(1 \le i \le \ell)$ are Lyndon words satisfying $L_i > L_{i+1}$ $(1 \le i < \ell)$

Following the LF-mapping is essentially radix sort on the cyclic string. Thus, BBWT(T) can be defined as: the last (or previous) symbols of all rotations of the multiset of all Lyndon factors of T, in the *cyclic lex. order* (a.k.a. omega order) of the rotations.

Example:

$T = \texttt{abaabaaba} = (\texttt{ab})^1 \cdot (\texttt{aab})^2 \cdot \texttt{a}^1$
$BBWT(T) = \texttt{abbaabaaa}$



Defining BBWT (Forward Transform)

- A string is a *Lyndon word* if it is primitive and lex. smallest rotation.
- $T = L_1^{e_1} \cdot \ldots \cdot L_\ell^{e_\ell}$ is the *Lyndon factorization* of T iff:
  - *Lyndon factor* L_i $(1 \leq i \leq \ell)$ are Lyndon words satisfying $L_i > L_{i+1}$ $(1 \leq i < \ell)$

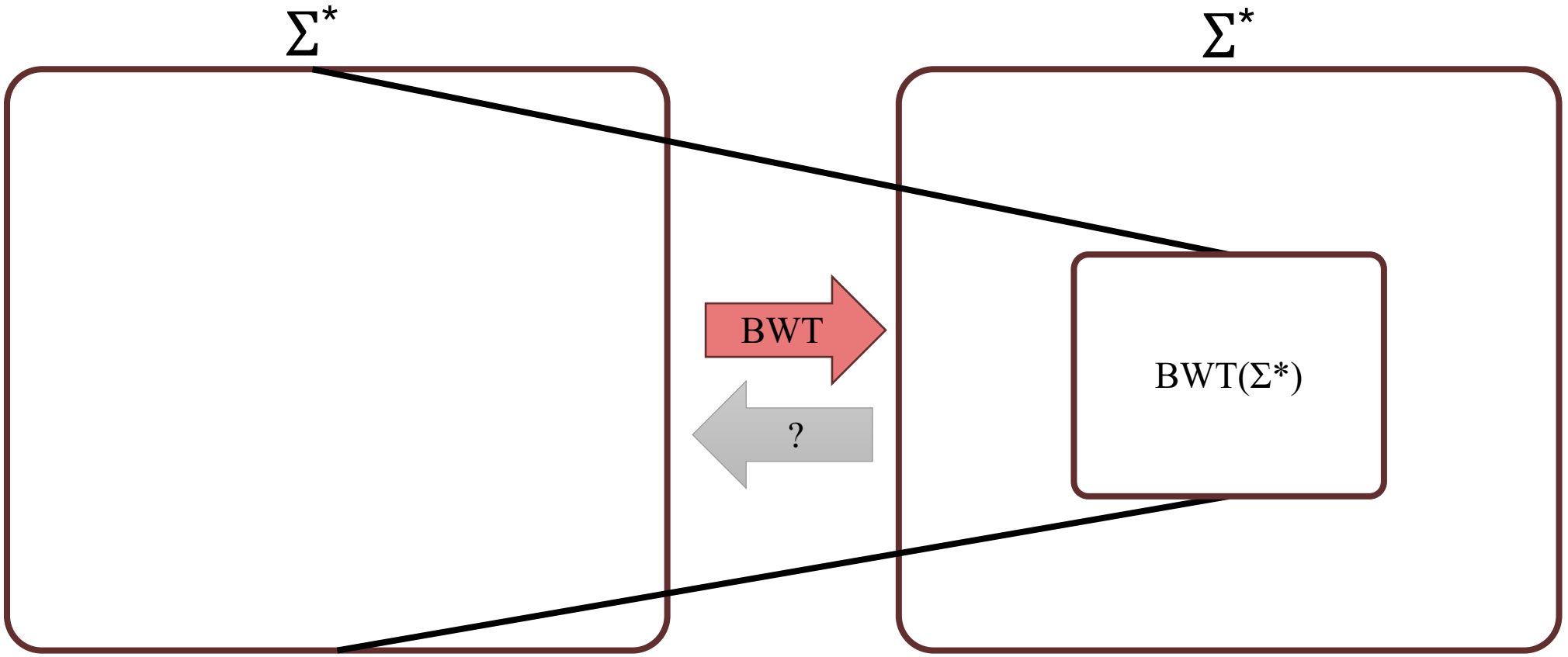
Following the LF-mapping is essentially radix sort on the cyclic string. Thus, BBWT(T) can be defined as: the last (or previous) symbols of all rotations of the multiset of all Lyndon factors of T, in the *cyclic lex. order* (a.k.a. omega order) of the rotations.

BBWT can be:

- Restated as the eBWT [Mantaci et al. 2007] of Lyndon factors of T
- Computed in linear time [B., Kärkkäinen, Köppl, Piątkowski 2021,2024, Boucher *et al.* 2021]
- Used as an index (like BWT) [B., Kärkkäinen, Köppl, Piątkowski 2019, 2024]

BBWT is a generalization of BWT

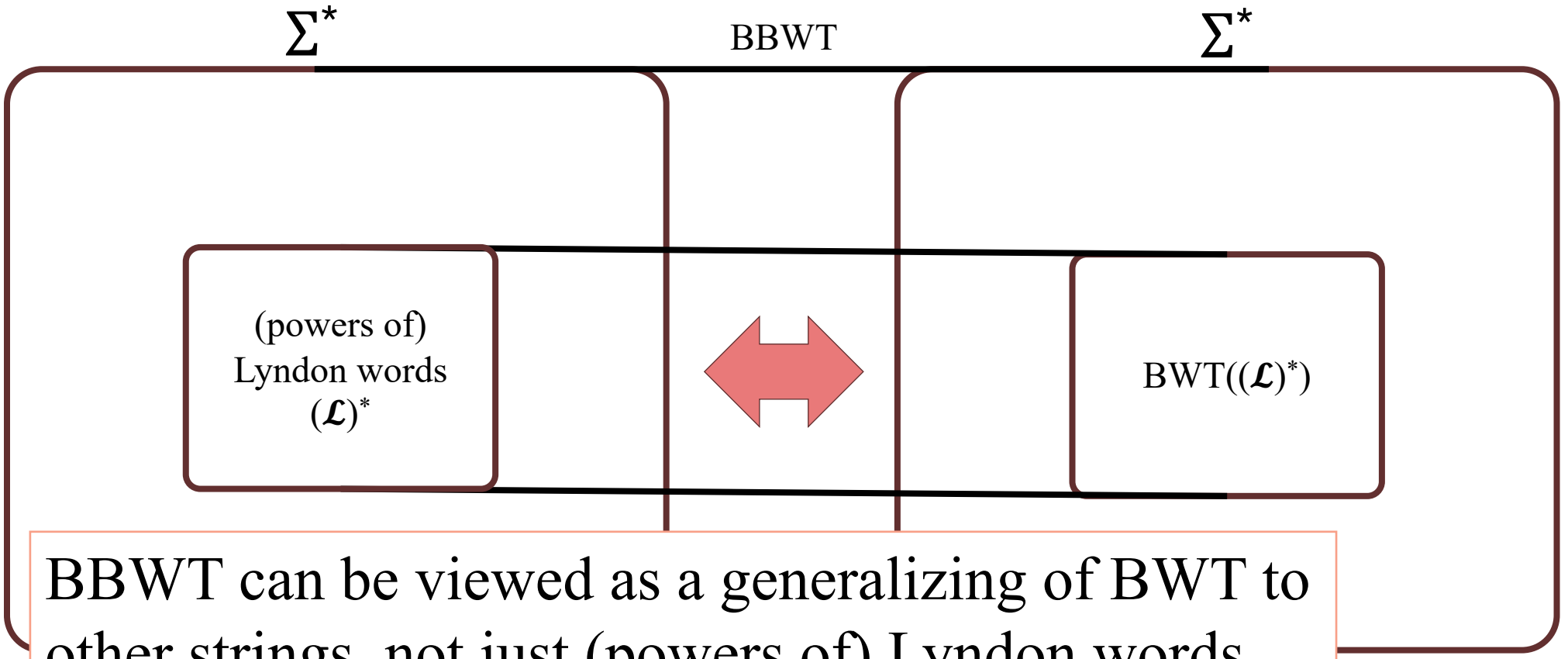
Although BWT is not a bijection when the domain is Σ^*…



Bijective BWT is a generalization of BWT

BWT is a bijection if we restrict domain to (powers of) Lyndon words.

➔ BBWT(T) = BWT(T) for any T = w^k for w: Lyndon, integer $k \geq 0$.



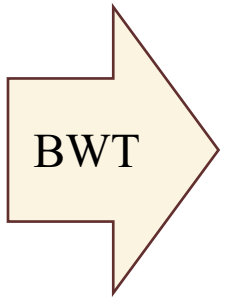
BBWT can be viewed as a generalizing of BWT to other strings, not just (powers of) Lyndon words.

Institute of SCIENCE TOKYO

BWT/BBWT and (Dictionary) Compression

The “Clustering Effect” of BWT

```
On_31_October_2023,_the_Japanese_cabinet_decid
ed_to_amend_the_National_University_Corporatio
n_Act._The_former_National_University_Corporat
ion_Tokyo_Institute_of_Technology_was_renamed_
the_National_University_Corporation_Institute_
of_Science_Tokyo,_and_the_National_University_
Corporation_Tokyo_Medical_and_Dental_Universit
y_was_dissolved_at_the_same_time._After_being_
passed_by_the_National_Diet_of_Japan,_the_amen
dment_took_effect_on_1_October_2024._The_new_u
niversity's_headquarters_is_located_in_Meguro_
ku,_Tokyo,_where_the_former_Tokyo_Institute_of
_Technology_headquarters_was_located.
```




```
yuo3nodet4223___002_2_____________________.__
_____________nrrnn.yyyydlnoofeoneeere11fff..e,
rnnllllloel,drdetskeesydsosseteeetgse,yddd,ete
dtwysy,ceennnnntcsn__p__pJJuuwwwp_ccrrrrNNNNN_
ooa__iooneeeSAeOOeneeeenene_ie_naamhhhhhchshhht
tthrmhhTTdftvstmdM_MbrimmmDbbmmthttvvvvvvnnino
ooofe__AneoottttttTtTt__wccdccDt_betttttttttt_d
ttttsssssnnnnnno_ooooaaaaaaa__oooiaaaadrraoOo
oioooooooeeaeaeai_iuUUUUUhhIIIeeyyyyytrttll___
_lloTTTTTnns_iiiiiiiiitppppffCCCCaa_rrrrddeeee
eooooe_oouoooooeeeeeeeeeaaa'raiar_esrrrrrrsainnn
ceecannuuuaafrr_________aaaaaaaaasss_cc_iiiiii
iiikqq_gtttliiiiie____tttttgbtgkkkkk
```

Text from: https://en.wikipedia.org/wiki/Institute_of_Science_Tokyo

- Original use of BWT was for compression, because BWT tends to make the string “easier to compress”, even with simple compressors like Run-Length-Encoding.
  - Intuitive reason: rotations with similar context are brought closer together.
- What about BBWT?
- What about other compressors (more powerful than RLE)?

Repetitiveness Measures related to Dictionary Compression

See also survey by [Navarro 2021]

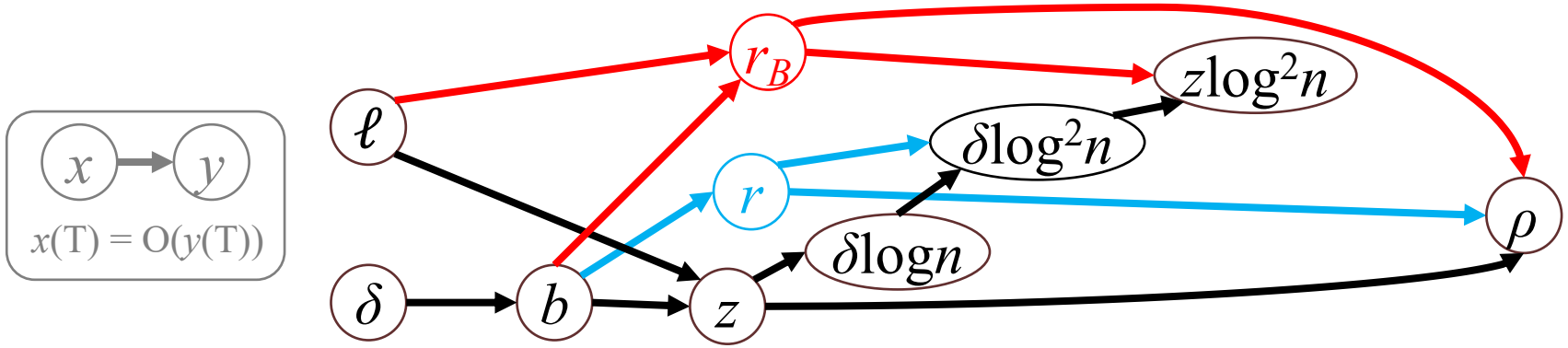
To see how BBWT compares to BWT in terms of compression, we first give results on the analysis of the measure r_B compared to other known repetitiveness measures.

- δ(T): length-normalized substring complexity of T
- ℓ(T): size of Lyndon factorization (# distinct Lyndon factors) of T

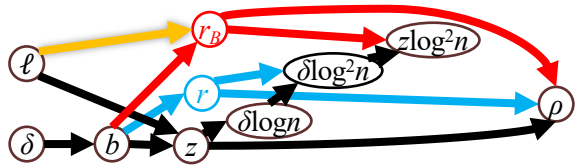
Dictionary Compression: representations based on copy/paste of substrings.

- b(T): size of smallest Bidirectional Macro Scheme (BMS) of T (most powerful in DC)
- z(T): size of LZ77 parsing of T
- ρ(T): size of Run-Length-Encoding (RLE) of T

- **r_B(T): size of RLBBWT (=ρ(BBWT(T)) of T (BBWT+RLE)**
- **r(T): size of RLBWT (=ρ(BWT(T)) of T (BWT+RLE)**



$\ell(T) \leq r_B(\mathrm{T})$



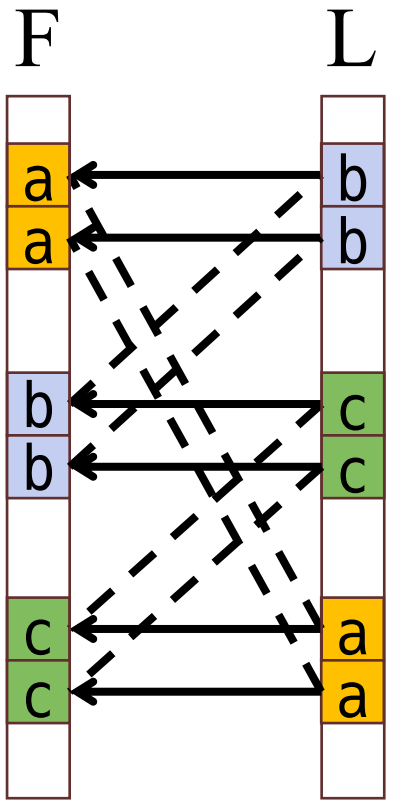
Claim [Badkobeh, B., Köppl 2024]

For any string T, $\ell(\mathrm{T}) \leq r_B(\mathrm{T})$. (Corollary of Corollary by [Boucher et al. 2024])

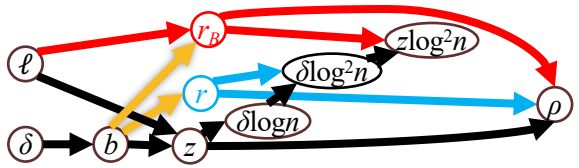
Proof:

Each distinct Lyndon factor induces at least one run-boundary.

- Consider cycle corresponding to a Lyndon factor L_i.
- If the cycle does not go through the beginning of a run in L, this implies that the lex. previous element is (part of) an identical cycle.
- Of the e_i cycles of L_i, the top one cannot have an identical cycle as the previous element, implying a mismatch with the previous element, i.e., a run-boundary.



$b(\mathrm{T}) = \mathrm{O}(r_B(\mathrm{T}))$



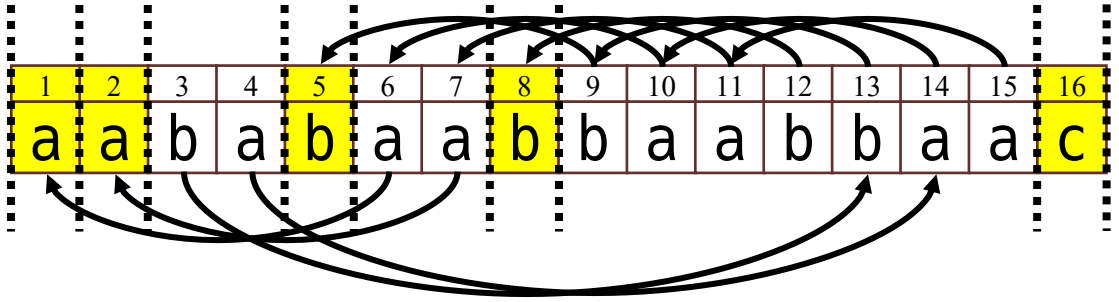
Claim [Badkobeh, B., Köppl 2024]

For any string T, RLBBWT induces an $\mathrm{O}(r_B(\mathrm{T}))$ size BMS, i.e., $b(\mathrm{T}) = \mathrm{O}(r_B(\mathrm{T}))$.

Proof sketch: The proof by [Navarro et al. 2021] for BWT $b(\mathrm{T}) = \mathrm{O}(r(\mathrm{T}))$ can be adapted.

SA[i] – 1	L
16	c
5	b
9	b
13	b
3	b
1	a
6	a
10	a
14	a
4	a
8	b
12	b
2	a
7	a
11	a
15	a

SA[i]	
1	aababaabbaabbaac
6	aabbaabbaacaabab
10	aabbaacaababaabb
14	aacaababaabbaabb
4	abaabbaabbaacaab
2	ababaabbaabbaaca
7	abbaabbaacaababa
11	abbaacaababaabba
15	acaababaabbaabba
5	baabbaabbaacaaba
9	baabbaacaababaab
13	baacaababaabbaab
3	babaabbaabbaacaa
8	bbaabbaacaababaa
12	bbaacaababaabbaa
16	caababaabbaabbaa

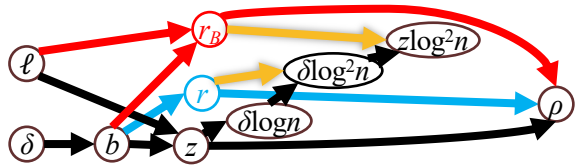


$r = 5$ ➔ BMS of size 8.

Translate copying on BWT/BBWT (lex. order) to text order, from right to left ➔ offset of the copy changes only when encountering run-boundaries (r_B times) or when "wrapping around (adjacent cyclicly, but not linearly)" ($\mathrm{O}(\ell)$ times).

Thus, #phrases = $\mathrm{O}(r_B + \ell) = \mathrm{O}(r_B)$

Bounding r_B in terms of z



Theorem [Badkobeh, B., Köppl 2024]

For any string T, $r_B(\mathrm{T}) = \mathrm{O}(z(\mathrm{T})\log^2 n)$

Proof sketch: We adapt the proof for BWT:

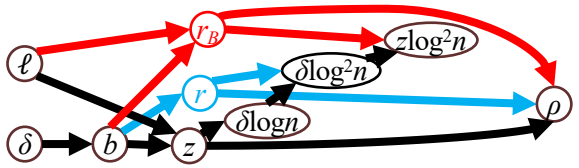
For any string T, $r(\mathrm{T}) = \mathrm{O}(\delta(\mathrm{T})\log^2 n)$ [Kempa & Kociumaka 2020]

$\delta(\mathrm{T}) = \max_k \#(\text{distinct length-}k\text{ substrings of T})/k$

- In the proof for BWT, $\delta(\mathrm{T})$ is used to bound the # of distinct length-k substrings of $\{\ \mathrm{T}^\infty\ \}$ by $\mathrm{O}(\delta(\mathrm{T})\cdot k)$.
- For $r_B(\mathrm{T})$, the proof can be modified to consider the # of distinct length-k substrings of $\{L_1^\infty,\ldots,L_\ell^\infty\}$, where $\mathrm{T} = L_1^{e_1}\cdot\ \ldots\ \cdot L_\ell^{e_\ell}$ is Lyndon factorization of T.
- This can be bounded by $\mathrm{O}((\delta+\ell)\cdot k) = \mathrm{O}(zk)$.

$\ell=\mathrm{O}(z)$ [Urabe, Nakashima, Inenaga, B. Takeda, 2019]

Upper bounds for r/r_B and r_B/r



Theorem [B., I, Nakashima 2025]

For any string T, $r(\mathrm{T})/r_B(\mathrm{T})$, $r_B(\mathrm{T})/r(\mathrm{T}) = \mathrm{O}(\mathrm{polylog}\ n)$

Proof

- $r(\mathrm{T}) = \mathrm{O}(\delta(\mathrm{T})\log^2 n)$
  $= \mathrm{O}(r_B(\mathrm{T})\ \log^2 n)$

- $r_{\mathrm{B}}(\mathrm{T}) = \mathrm{O}(z(\mathrm{T})\log^2 n)$
  $= \mathrm{O}(\delta(\mathrm{T})\log^3 n)$
  $= \mathrm{O}(r(\mathrm{T})\log^3 n)$

$z(\mathrm{T}) = \mathrm{O}(\delta(\mathrm{T})\log n)$ [Raskhodnikova et al. 2013]

Lower bound for r_B/r: String families with $r = o(r_B)$

Theorem [Badkobeh, B., Köppl 2024]

$\exists$infinite string family s.t. $r(\mathrm{T}) = O(1)$ and $r_B(\mathrm{T}) = \Theta(\log n)$.

Proof: Fibonacci words: $F_0 = \mathsf{b}$, $F_1 = \mathsf{a}$, $F_k = F_{k-1} \cdot F_{k-2}$

- $r_B(F_{2k+1}) = \Omega(\ell(F_{2k+1}))$ (Corollary of Corollary by [Boucher et al. 2024])

  $= k+1 = \Theta(\log n)$ [Melançon 1999]

  Lyndon factorizations of F_{2k+1}

  $F_1 = \mathsf{a}$

  $F_3 = \mathsf{ab} \cdot \mathsf{a}$

  $F_5 = \mathsf{ab} \cdot \mathsf{aabab} \cdot \mathsf{a}$

  $F_7 = \mathsf{ab} \cdot \mathsf{aabab} \cdot \mathsf{aabaababaabab} \cdot \mathsf{a}$

  $F_9 = \underbrace{\mathsf{ab}}_{|F_2|=2} \cdot \underbrace{\mathsf{aabab}}_{|F_4|=5} \cdot \underbrace{\mathsf{aabaababaabab}}_{|F_6|=13} \cdot \underbrace{\mathsf{aabaababaabaababaabaababaabaababaabab}}_{|F_8|=34} \cdot \mathsf{a}$

- $r(F_k) = 2$
  - $\mathrm{BWT}(F_k) = \mathsf{b}^{|F_{k-2}|} \cdot \mathsf{a}^{|F_{k-1}|}$ [Mantaci et al. 2003]

Lower bound for r/r_B: String families with $r_B = o(r)$

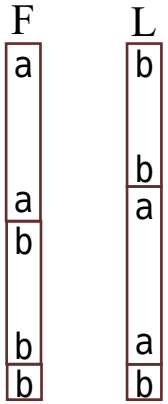
Theorem [B., I, Nakashima 2025]

$^{\exists}$infinite string family s.t. $r_B(\mathrm{T}) = O(1)$ and $r(\mathrm{T}) = \Theta(\log n)$.

Proof: Fibonacci-plus words [Giuliani et al. 2021]

Consider $\mathrm{T} = \mathsf{b}F^*_{2k}$ where $F^*_{2k} = \mathrm{rot}_{\min}(F_{2k})$.

lex. smallest rotation

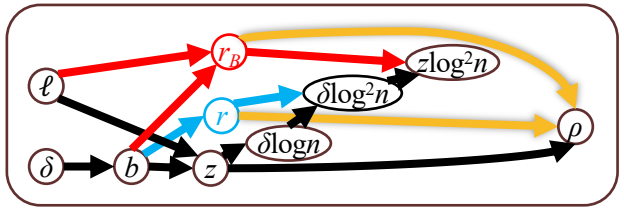


- $r_B(\mathrm{T}) = 3$
  - Lyndon factorization of T: $\mathsf{b} \cdot F^*_{2k}$
  - b: lex largest cycle ➔
    $\mathrm{BBWT}(\mathrm{T}) = \mathrm{BBWT}(F^*_{2k}) \cdot \mathsf{b} = \mathrm{BWT}(F^*_{2k}) \cdot \mathsf{b} = \mathrm{BWT}(F_{2k}) \cdot \mathsf{b} = \mathsf{b}^{|F_{2k-2}|} \cdot \mathsf{a}^{|F_{2k-1}|} \cdot \mathsf{b}$

- $r(\mathrm{T}) = \Theta(\log n)$
  - It turns out that T is a rotation of strings studied previously:
    - v^{rev} of a Fibonacci-plus word $v = F_{2k} \cdot \mathsf{b}$ [Giuliani et al. 2021]
    - inserting b at position $|\mathrm{F}_{2k-1}| - 2$ of F_{2k} [Giuliani et al. 2023,2025]

$r(\mathrm{T}) = r(v^{rev}) = \Theta(\log n)$ [Giuliani et al. 2021][Giuliani et al. 2023,2025]

Bounding r_B by ρ



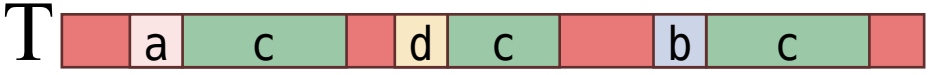
Theorem [B., I, Nakashima 2025]

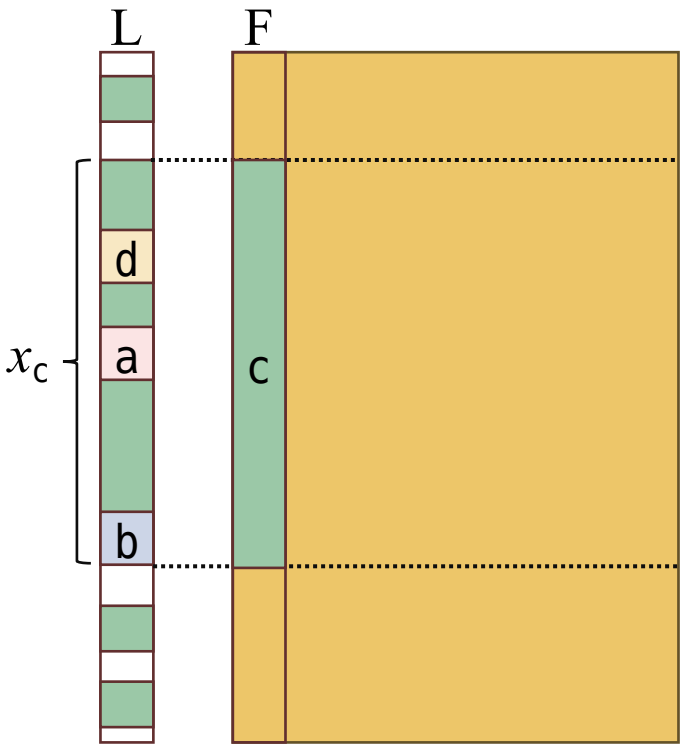
For any string T, $r_B(T) = \rho(BBWT(T)) \leq 2\rho(T)$.

Proof

Adaptation of proof for BWT [Mantaci et al. 2017]

- $\rho_c(T)$: denote # of runs of $c \in \Sigma$ in T.
- Consider substring x_c of BBWT corresponding to rotations starting with c.
- Lyndon factors start/end at RLE-boundaries.
- c is preceded either by c, or (at most) $\rho_c(T)$ other non-c symbols, so $\rho(x_c) \leq 2\rho_c(T)+1$.
- However, equality cannot hold.
- Therefore, $\rho(x_c) \leq 2\rho_c(T)$, and $\rho(BBWT(T)) \leq \sum_c \rho(x_c) \leq \sum_c 2\rho_c(T) = 2\rho(T)$.





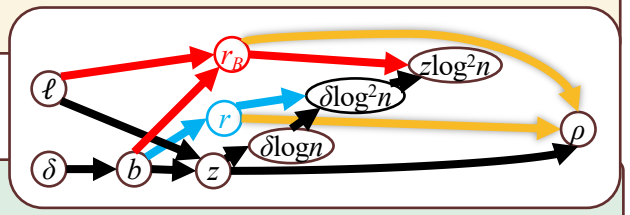
Asymmetry in clustering effect of BWT/BBWT wrt RLE.

Theorem [Mantaci et al. 2017]

For any string T, $r(\mathrm{T}) = \rho(\mathrm{BWT}(\mathrm{T})) \leq 2\rho(\mathrm{T})$.

Theorem [B., I, Nakashima 2025]

For any string T, $r_B(\mathrm{T}) = \rho(\mathrm{BBWT}(\mathrm{T})) \leq 2\rho(\mathrm{T})$.



- For any string T, BWT/BBWT never increases the measure ρ by a factor of more than 2.
- On the other hand, there are strings where the measure ρ is decreased by a factor of $\Theta(n)$: e.g. Fibonacci words.
  - $\rho(F_k) = \Theta(n)$, $\rho(\mathrm{BWT}(F_k)) = \Theta(1)$
  - $\rho(\mathrm{rot}_{\min}(F_k)) = \Theta(n)$, $\rho(\mathrm{BBWT}(\mathrm{rot}_{\min}(F_k))) = \Theta(1)$

Asymmetry in clustering effect wrt μ (upperbound for increase)

Theorem [B., I, Nakashima 2025]

$$\mu(\mathrm{BWT}(\mathrm{T})), \mu(\mathrm{BBWT}(\mathrm{T})) = \mathrm{O}(\mu(\mathrm{T})\mathrm{polylog}(n))$$

holds for any μ s.t. $\delta(\mathrm{T}) = \mathrm{O}(\mu(\mathrm{T}))$ and $\mu(\mathrm{T}) = \mathrm{O}(\rho(\mathrm{T}))$.

For any string, BWT/BBWT never increases measure μ by a factor of more than polylog(n).

Proof

$\mu(\mathrm{BWT}(\mathrm{T})) = \mathrm{O}(\rho(\mathrm{BWT}(\mathrm{T}))) = \mathrm{O}(r(\mathrm{T})) = \mathrm{O}(\delta(\mathrm{T})\log^2 n) = \mathrm{O}(\mu(\mathrm{T})\log^2 n)$.

[Kempa&Kociumaka 2020, 2022]

$\mu(\mathrm{BBWT}(\mathrm{T})) = \mathrm{O}(\rho(\mathrm{BBWT}(\mathrm{T}))) = \mathrm{O}(r_B(\mathrm{T})) = \mathrm{O}(z(\mathrm{T})\log^2 n)$

$= \mathrm{O}(\delta(\mathrm{T})\log^3 n) = \mathrm{O}(\mu(\mathrm{T})\log^3 n)$.

[Badkobeh, B., Köppl 2024]

[Raskhodnikova et al. 2013]

Asymmetry in clustering effect wrt μ (significant decrease)

Theorem [B., I, Nakashima 2025]

There exists infinite family of strings s.t.

$\delta(T) = \Omega(n/\log n)$, $r(\mathrm{BBWT}(T)) = O(1)$.

$\boldsymbol{\delta(\mathrm{BBWT}^{-1}(F_k)) = \Omega(n/\log n)}$, $r(F_k) = 2$.

Proof Sketch

Fibonacci words (again)!

- $\delta(\mathrm{BBWT}^{-1}(F_k)) = \Omega(n/\log n)$
  - Key observation: If positions of F_k are represented by k-bit Zeckendorf representations, the LF-mapping of a position corresponds to a 1- or 2-bit rotation of the Zeckendorf representation.
  - Thus, each cycle in LF-mapping corresponds to a conjugacy (i.e., cyclically equivalent) class of k-bit Zeckendorf representations of the positions of L.
  - Lyndon words corresponding to each of these cycles have length at most k, and can be shown to occur uniquely and non-overlappingly in $\mathrm{BBWT}^{-1}(F_k)$.

    ➔ Any length-$2k$ substring of T is unique.
  - Result follows from $k = \Theta(\log n)$ and $\delta(T) \geq (n - 2k)/2k = \Theta(n/\log n)$

Implications of $\delta(\text{BBWT}^{-1}(F_k)) = \Omega(n/\log n)$, $r(F_k) = 2$

For $T = \text{BBWT}^{-1}(F_k)$:

- $\delta(T) = \Omega(n/\log n)$ implies that T is <u>incompressible</u> by dictionary compression. ($r(T) \geq b(T) \geq \delta(T) = \Omega(n/\log n)$)
- On the other hand, BBWT(T) transforms T to F_k, which is <u>very compressible</u> by dictionary compression. ($b(F_k) \leq r(F_k) = 2$)

BBWT can change the compressibility of the string not only wrt RLE, but wrt dictionary compression.

➔ combining BBWT + dictionary compression can be **<u>strictly more powerful</u>** than any dictionary compressor

New compression schemes using BBWT

Rotations and RLBWT / RLBBWT

Observations:

- By definition, r is **not** sensitive to rotation, while r_B is.
- $r(\mathrm{T}) = r_B(\mathrm{rot}_{\min}(\mathrm{T})) \geq r_B(\mathrm{rot}_{\mathrm{OPT}}(\mathrm{T}))$

lex. smallest rotation

rotation with smallest r_B

We can compute in linear time, a rotation + RLBBWT that achieves *at least the same compression* as RLBWT, and possibly better. (we previously saw examples with log n factor improvement)

- $r(\mathrm{T}) = \mathrm{rot}_{\min}(\mathrm{T})$ can be computed in linear time.
- can check constant # of other rotations in linear time.
- rotation can be encoded as log n bit integer.
  (instead of the index required for standard BWT)

Why stop at just one rotation and BBWT?

If $T = \text{BBWT}^{-1}(F_k)$, we saw that $\rho(\text{BBWT}(\text{rot}_{\text{min}}(\text{BBWT}(T)))) = O(1)$.

Conjecture (BBWT Reachability) **[Badkobeh, B., Köppl 2024]**

Any permutation of a given string can be computed by applying a sequence of BBWT and rotation operations.

➔ Any string can be encoded by its Parikh vector (# of each symbol of alphabet), and a sequence of BBWT and rotation operations.

Tested for all strings up to length 17.

Theorem **[Badkobeh, B., Köppl 2024]**

The conjecture holds for $\sigma = 2$ and $\sigma = n$.

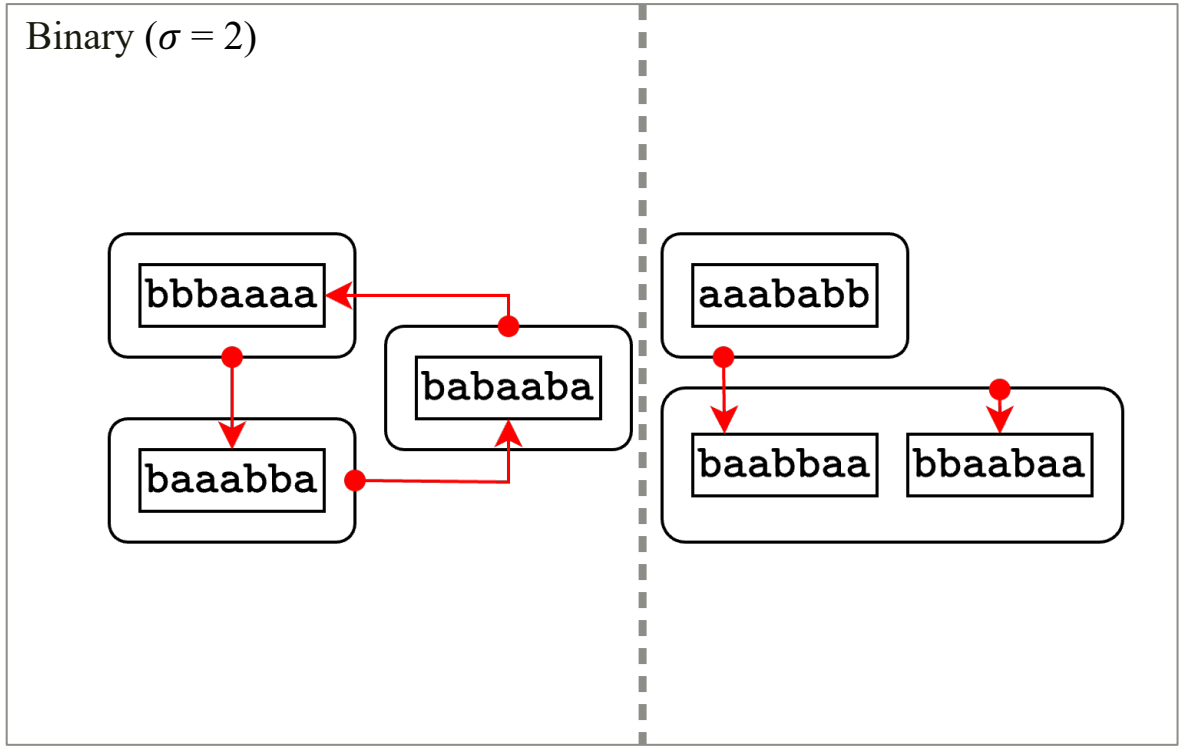
Key Lemma

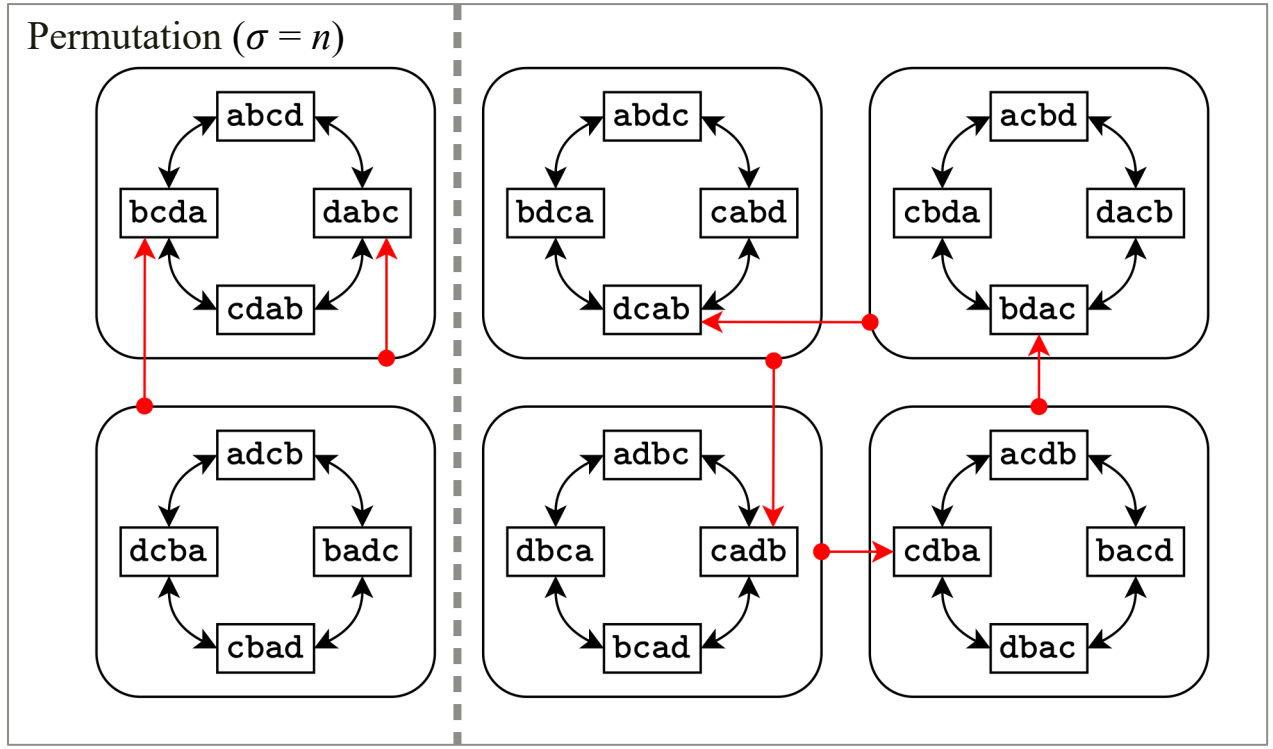
Let $x = \text{rot}_{\text{min}}(x)$, $y = \text{sort}(x)$. If $x[\text{lcp}(x,y)] \neq x[n]$ then, $\text{rot}_{\text{min}}(\text{BBWT}^{-1}(\underbrace{x[n]x[1..n-1]}_{\text{right rotation}})) < x$.

$x[\text{lcp}(x,y)] \neq x[n]$ always holds for $x \neq y$ when $\sigma = 2$ and $\sigma = n$.
Since BBWT and rotations are bijections, any two words can be reached via y.

Unreachable examples by BWT/rotation

Analogous results do NOT hold for BWT





rounded-edge rectangle: conjugacy class

red arrows: BWT (conjugacy class → string)

black arrows: rotation (by 1 position)

Summary and Open Problems

- Revisited BWT from the perspective of bijection and compression
  - Showed that BBWT have similar properties as BWT
  - Bijectivity of BBWT made it (mathematically) easier/possible to
    - Show that BBWT + dictionary compression can be strictly stronger than dictionary compression (proof for BWT is almost there but not yet complete)
    - Devise new compression schemes by applying BBWT/rotation multiple times

Questions

- Sub-quadratic algorithm for computing rotation that minimizes r_B?
  - $\tilde{O}(nh)$ time (h: height of left and right Lyndon trees) [Badkobeh, B., I, Köppl in press]
- Reachability of BBWT/Rotation for alphabet size $2 < \sigma < n$?
- Tightness of bounds of compressibility using BBWT?
  - $r = O(\delta \log^2 n)$ while $r_B = O((\ell+\delta) \log^2 n) = O(z \log^2 n)$
- Other bijective variants of BWT?

References

- Hideo Bannai, Dominik Köppl, Zsuzsanna Lipták:
  A Survey of the Bijective Burrows-Wheeler Transform.
  The Expanding World of Compressed Data: A Festschrift for Giovanni Manzini's 60th Birthday: 2:1-2:26, 2025.
- Golnaz Badkobeh, Hideo Bannai, Tomohiro I, Dominik Köppl:
  Bijective BWT Based Compression Schemes. Theory Comput. Syst., in press.
- Hideo Bannai, Tomohiro I, Yuto Nakashima:
  On the Compressiveness of the Burrows-Wheeler Transform.
  CPM 2025: 17:1-17:15, 2025.
- Golnaz Badkobeh, Hideo Bannai, Dominik Köppl:
  Bijective BWT Based Compression Schemes. SPIRE 2024: 16-25, 2024.
- Hideo Bannai, Juha Kärkkäinen, Dominik Köppl, Marcin Piatkowski:
  Constructing and indexing the bijective and extended Burrows-Wheeler transform. Inf. Comput. 297: 105153 (2024)