$\underline{\text{REV}}$ersed $\underline{\text{I}}$ndexes $\approx$ $\underline{\text{VAL}}$ues in Wavelet Trees
REVIVAL in Wavelet Trees

Xiangjun Peng
The Chinese University of Hong Kong

Abstract

Data Compression (or Source Encoding) constructs indexes, by encoding information using fewer bits (than the original representation for values), to reduce the size of data storage. Therefore, if any computation needs to be performed, the compressed data needs to be uncompressed first. Since a limit of lossless data compression is expected to exist (lower-bounded by Shannon 1948), it is expected to expand the functionalities of compressed data under lossless compression. To this end, Succinct Data Structures are proposed (Jacobson 1988) and explored, which enable queries directly on compression.

This work presents a discovery to advance the above wisdom in a particular Succinct Data Structure: Wavelet Tree (Grossi, Gupta, and Vitter 2003). The discovery is first made by showing the feasibility of Reversed Indexes = Values: for integers within $[0, 2^N)$, there exists a Wavelet Tree that its compressed indexes can be equivalent to the Leibniz Binary system (Leibniz 1703), with only the bit reversal. Then we show how to strengthen the discovery by generalizing it into Reversed Indexes $\approx$ Values, by applying a longest common subsequence in bits and its patterns. Finally, we conjuncture potential implications of the above ideas by discussing its benefits, and modifications to the RAM model.

The discovery reveals that: (1) the usability of Succinct Data Structure can be significantly expanded, by enabling Computation Directly on Compression; and (2) near-optimal lossless compression can still yield close connections with the Leibniz Binary System (Leibniz 1703), which breeds polymorphic functionalities within a single piece of the information. This work also provides an initial analysis of the benefits from the method (and potentially other extensions), and suggests potential modifications. We conjecture that: with $\underline{\text{REV}}$ersed $\underline{\text{I}}$ndexes $\approx$ $\underline{\text{VAL}}$ues, everything old can be new now.

Contents

1 Introduction

Data Compression constructs indexes to reduce the elementary size of original values, and therefore the storage cost is saved. Lower-bounded by Shannon (Shannon 1948), a fundamental limit to lossless data compression exists, which is known as the entropy rate. For decades, an extensive amount of efforts are paid on the potential breakthrough of this limit.

This work classifies prior work into two types. One is to improve the efficiency of lossless compression; and the other is to expand the functionalities of lossless compression. The latter is the focus in this work. Succinct Data Structures, originally formalized by Jacobson (Jacobson 1988), are proposed to expand the functionalities of lossless compression. These data structures use an amount of space that is "close" to the entropy rate, while still retaining the functionality for efficient query operations. Therefore, queries on data can be directly performed on compression.

However, to perform any computation, it is still needed for decompression first in Succinct Data Structures. This work centers the focus on a particular Succinct Data Structure - Wavelet Tree (Grossi, Gupta, and Vitter 2003). Wavelet Tree is used to store strings in compressed space. The definition of Wavelet Tree is achieved by recursively partitioning the alphabet (from the string) into pairs of subsets; the leaves correspond to individual symbols of the alphabet, and at each node a bitvector stores whether a symbol of the string belongs to one subset or the other.

This work first breaks the assumption of Wavelet Tree (or all lossless compression methods), by directly taking Leibniz Binary System (Leibniz 1703) as the reference point (rather than character encoding such as ASCII codes). We find that: the encoding patterns of Leibniz Binary System (Leibniz 1703) can be highly correlated with the formalization of Wavelet Tree (Grossi, Gupta, and Vitter 2003), with only the bit reversal. Therefore we derive the discovery of Reversed Indexes = Values, which is described in Section 1.1.1.

This work then rolls back to the formalization of Wavelet Tree (or all lossless compression methods), by refining the reference point back to the character encoding. We find that: the above approach can be generalized to character encoding, by (1) simply accounting for common subsequence(s) in bits; and (2) utilizing these subsequences as patterns, to recover indexes to values. We also discuss extensions to other scenarios (e.g., other data types). It is described in Section 1.1.2.

This work finally conjectures potential implications of the above ideas, by analyzing the benefits (and hypothesizing potential modifications to RAM model) for "Indexes $\approx$ Values" principle. This work considers two viewpoints by leveraging the classification from (Navarro 2014). We first view Wavelet Tree as a compression method, and the benefits can be directly derived. Then we view Wavelet Tree as a data structure, and the design space is discussed and we assume it deserves further investigations. It is described in Section 1.1.3.

In summary, the discovery of Reversed Indexes $\approx$ Values (REVIVAL) showcases the feasibility to bridge near-optimal lossless compression with the Leibniz Binary System, which makes the following three major contributions.

- The bridge between near-optimal lossless compression and Leibniz Binary System enables Computation Directly on Compression.
- The discovery expands the usability of Succinct Data Structures, and this delivers polymorphic functionalities within a single piece of the information.
- The bridge motivates a revamp of RAM model to support the above idea, and demonstrates potential merits of fine-grained RAM operations.

When is a revival needed? When carelessness and unconcern keep the people asleep.

- Billy Sunday

We first give an overview of the results in this work. Then we summarize the techniques developed in this work.

1.1 Our Results

1.1.1 Reversed Indexes = Values in Wavelet Trees

The discovery is initially made as Reversed Indexes = Values, by extending the usage of Wavelet Tree from strings to integers. By doing so, an accidental connection is observed that: for integers within $[0,2^N)$, there exists a Wavelet Tree that its compressed indexes can be equivalent in the Leibiniz Binary system (Leibniz 1703), with only the bit reversal.

1.1.2 Reversed Indexes ≈ Values in Wavelet Trees

The discovery is then expanded as Reversed Indexes ≈ Values for other types of encoding, by applying (a) common subsequence(s) in bits. Such a supplementary method breeds two opportunities. First, it can allow a more flexible mapping for the range of the to-be-compressed data. This is done by (1) extracting (a) common subsequence(s) in bits as (a) pattern(s); and (2) applying Reversed Indexes = Values in the rest of bits. Second, it can partially break the requirement of the consecutive range for the to-be-compressed data, since the shared common subsequence(s) can have positional variations.

1.1.3 Implications from "Indexes ≈ Values" Principle

Based on the above ideas, this work discusses potential benefits, and suggests a potential revamp of RAM model. Our discussion takes WT as a motivating example, and classifies our analysis into two cases. We first view WT as a compression method, and the benefits can be derived directly. Then we view WT as a data structure, and suggest potential modifications of RAM model: so that we can support both queries and computation directly on compression.

1.2 Technique Outline

Though the discovery of REVIVAL is empirical, there are the following techniques can be derived for further investigations of other instances in the "Indexes ≈ Values" principle.

• **Input-bounded Range of Available Values**: the first step for the usage of REVIVAL (or other techniques) requires the bounded range of all available values, as the preliminary knowledge. This is used to determine how REVIVAL shall play a role in this bounded range, and furthermore how the longest common subsequence in bits shall be extracted (if needed).

• **Conditional Partitions of the Alphabet**: the second step for the usage of REVIVAL (or other techniques) requires a set of conditions to be determined, which are used for encoding all values. The key to decide these conditions shall be closely related to how these values are encoded, and they are expected to be closely correlated.

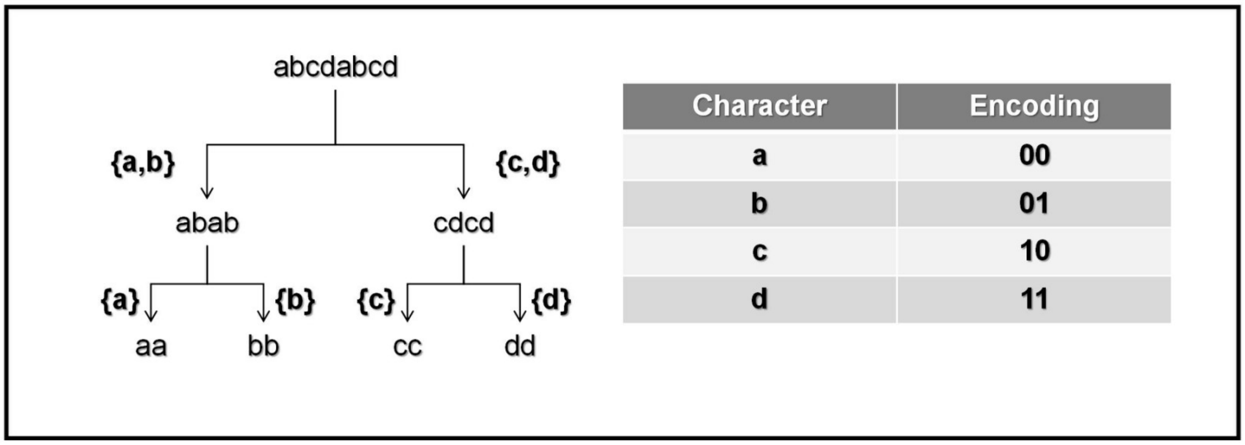
• **Executions on "Indexes ≈ Values" Principle**: the final step for the usage of REVIVAL (or other techniques) requires the abstraction of the above information, for the fine-grained operations over the compressed sequence of bits. This can vary by leveraging different viewpoints, and how different functionalities shall be supported.

2 Background

Wavelet Tree (WT) is a Succinct Data Structure, that can support *rank* and *select* operations efficiently, to store strings in compressed space (Grossi, Gupta, and Vitter 2003).

2.1 Wavelet Tree Definition

A WT is a data structure that recursively partitions a stream of characters into two parts, until homogeneous data are left. The encoding scheme is dependent to the partition of alphabet (and its subsets). Figure 1 gives out an example of WT from the string “abcdabcd".

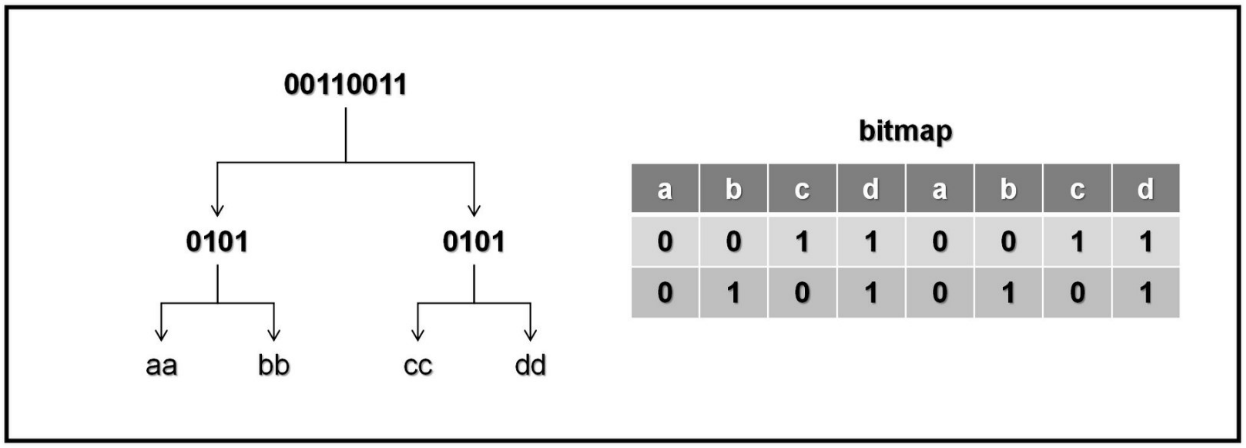


Character	Encoding
a	00
b	01
c	10
d	11

Figure 1: An example Wavelet Tree of the string “abcdabcd".

2.2 Bitmap from Wavelet Tree

A bitmap from WT is to deliver the encoding results level by level, and every index can be viewed vertically (from top to down). Figure 2 gives out the demonstrated bitmap from Figure 1.



bitmap

a	b	c	d	a	b	c	d
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1

Figure 2: The corresponding bitmap from Figure 1.

3 Reversed Indexes = Values

We first introduce Reversed Indexes = Values. We make the discovery by changing the reference point from character encoding into Leibniz Binary System (Leibniz 1703).

3.1 Leibniz Binary System Made WT

The "Reversed Indexes = Values" is described hereby: for integers within $[0, 2^N)$, there exists a Wavelet Tree that its compressed indexes can be equivalent to the Leibniz Binary system (Leibniz 1703), with only the bit reversal. Figure 3 gives an example to demonstrate the idea.

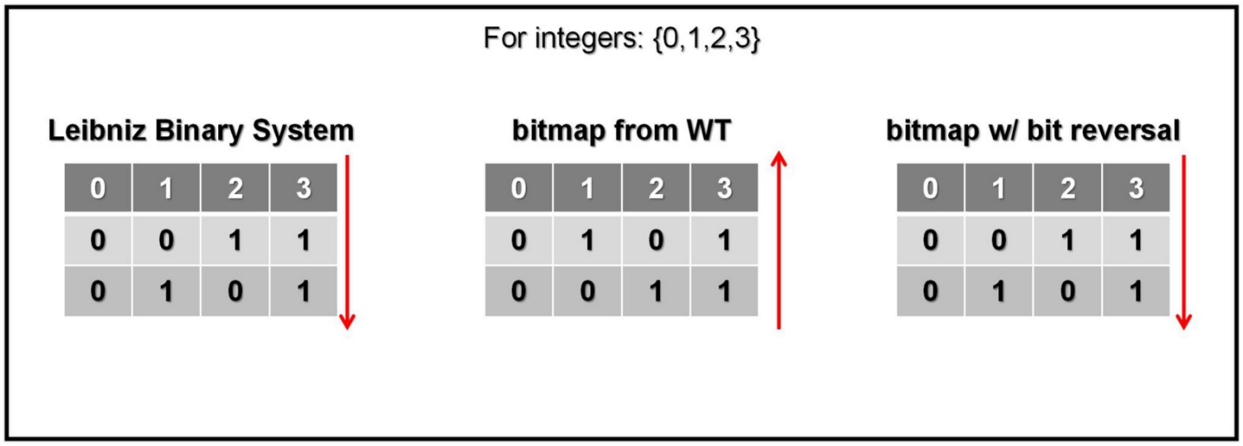


Figure 3: An example of Reversed Indexes = Values using integers within $[0, 2^2)$.

3.2 Generalization to integers within $[0, 2^N)$

The above discovery can be easily generalized. This is because the encoding scheme from Leibniz Binary System (Leibniz 1703) is a natural fit with the definition of WT. To obtain the above results in a generalized form (i.e. for integers within $[0, 2^N)$), it is straightforward to have the following method derived:

> **Sort the alphabet, and then perform the partition for WT by putting smaller ones into a subset, and the rest into the other subset.**

There are a few notes regarding the above generalization of Reversed Indexes = Values.

First, note that this may not be the only method to derive parts of the same results, but the above one is considered as the most general one. This is because the above rule is derived based on binary carry in Leibniz Binary System. More specialized forms are expected to be delivered also.

Second, the lower bound of the input data range has to remain as zero so that the proposed method can function. This is expected since WT encoding requires one of all elements from the alphabet to encoded as “0"s, and therefore enforces the inclusion of zero.

Third, similar results may be derived if we change the overall range of the input data. However, the regulations to partition the alphabet can not be succinct enough, and the derived results can only be an approximation of Reversed Indexes = Values. More instances, from slight changes of the input range, can be derived. We leave this part to the next section.

4 Reversed Indexes ≈ Values

With the strict definition of Reversed Indexes = Values, we present a more general form called Reversed Indexes ≈ Values.

4.1 Common Subsequence in Bits Made Reversed Indexes ≈ Values

The "Reversed Indexes ≈ Values" is described hereby: when leveraging parts of the bit sequence via Reversed Indexes = Values, one or (several patterns) patterns can be used to connect bitmap from WT with the exact values in different encoding schemes. Figure 4 gives an example to demonstrate the idea in ASCII encoding, and we elaborate more on this example.

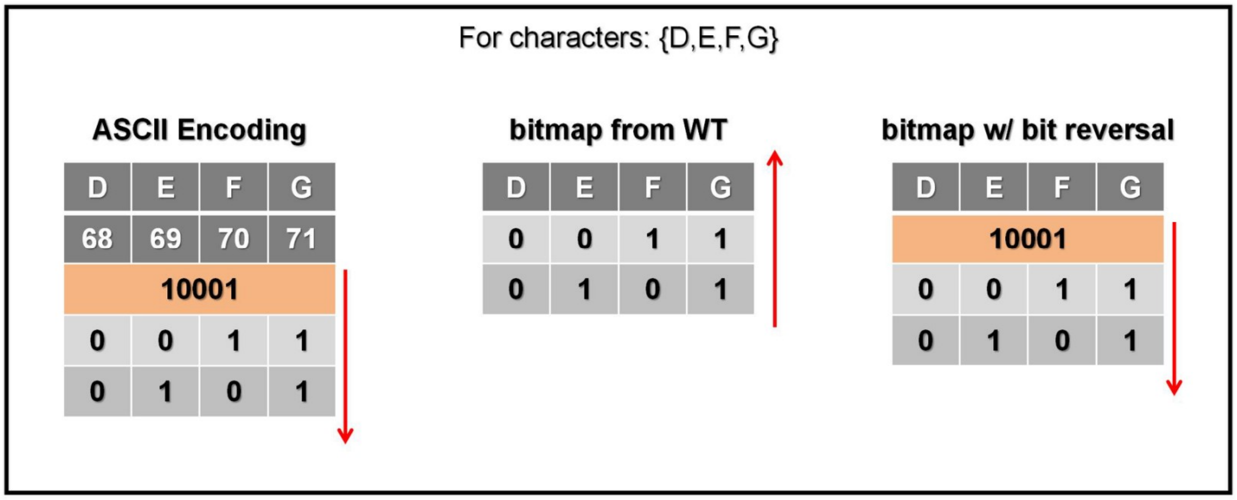


Figure 4: An example of Reversed Indexes ≈ Values using characters within $[D, G]$ in ASCII encoding. The highlighted "10001" is the shared common subsequence of all characters in bits.

For $[D, G]$ in ASCII encoding, all values share the common bit subsequence "10001" in binary. Therefore, after following the principle of Reversed Indexes = Values, we can obtain the bitmap and the reversed bits can be equalized with parts of the exact values. To fully recover the values, the only job is to supplement the common subsequence (i.e. "10001").

4.2 Extensions in Reversed Indexes ≈ Values

It is expected that Reversed Indexes ≈ Values can be extended in a variety of aspects.

First, note that the usage of common subsequences can be generalized, since there can be several common subsequences within the input data range. Hence, the number of these subsequences determine the number of bit patterns, and these patterns are used to isolate so that the rest of bits can be used via Reversed Indexes = Values.

Second, though the demonstrated example only covers character encoding in ASCII encoding, it is expected to be generalized to other types of value encoding. Particularly, for floating-point numbers, we assume dyadic scaling is more suitable one for the discovery.

Third, the discussion so far does not cover the impacts of the sign system, but the inclusion of a sign system is expected not to impact the correctness of our method for Reversed Indexes = Values, as long as the value of zero is included.

5 Implications from “Indexes $\approx$ Values" Principle

We discuss benefits from (and potential modifications to) RAM, so that we can support the usage of Reversed Indexes $\approx$ Values (and their potential variants). The key to support the polymorphic functionalities is to enable the retrieval of these values efficiently. We leverage Reversed Indexes $\approx$ Values as an example: based on the usage of WT, we analyze (and provide potential modifications) using RAM for Reversed Indexes $\approx$ Values. We first view WT as a compression method; and then we view WT as a data structure.

5.1 Viewpoint as a Compression Method

When taking WT as a compression method, the benefits are straightforward and two-fold. First, every index is directly packed together, which reduces the overhead in terms of data transfer. Second, the processor only requires bit manipulation to conclude the de-compression (in Reversed Indexes $\approx$ Values, the bit reversal or constant values for recovering the values based on bit patterns): it saves costs in common de-compression (e.g., lookup costs).

5.2 Viewpoint as a Data Structure

Taking WT as a data structure requires decent modifications to RAM for the usage of Reversed Indexes $\approx$ Values, and we conjecture that there are three parts. First, it requires dual-address modes for indexes and values respectively. Second, it requires level-oriented gather supports by using levels of WT as breakpoints, so that values can be retrieved. Third, though bit manipulation (and constant values for recovering the values based on bit patterns) can be performed within the processor, there can be benefits by integrating these functionalities in some cases (e.g., Processing-In-Memory Paradigm). We leave this to the future investigations.

6 Conclusions

This work describes a discovery to bridge near-optimal lossless compression with Leibniz Binary System. It (1) makes Computation Directly on Compression feasible; and (2) enables polymorphic functionalities (i.e., efficient queries and computation) within a single piece of the information. This work also provides an initial analysis of the benefits from the method (and potentially other extensions), and suggests potential modifications. We conjecture that: with Reversed Indexes $\approx$ Values, everything old can be new now.

References

Grossi, Roberto, Ankur Gupta, and Jeffrey Scott Vitter (2003). “High-order Entropy-compressed Text Indexes”. In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*. ACM/SIAM, pp. 841–850. URL: `http://dl.acm.org/citation.cfm?id=644108.644250`.

Jacobson, Guy Joseph (1988). “Succinct Static Data Structures”. AAI8918056. PhD thesis. USA.

Leibniz, Gottfried Wilhelm (1703). “Explication de l’arithmetique binaire, qui se sert des seuls caracteres O et I avec des remarques sur son utilite et sur ce qu’elle donne le sens des anciennes figures chinoises de Fohy”. In: *Memoires de l’Académie Royale des Science* 3, pp. 85–89.

Navarro, Gonzalo (2014). “Wavelet trees for all”. In: *J. Discrete Algorithms* 25, pp. 2–20. DOI: `10.1016/J.JDA.2013.07.004`. URL: `https://doi.org/10.1016/j.jda.2013.07.004`.

Shannon, Claude Elwood (1948). “A Mathematical Theory of Communication”. In: *The Bell System Technical Journal* 27.3, pp. 379–423.