CONTENTS

1 INTRODUCTION

1.1 WHY COMPACT DATA STRUCTURES?

1.2 SOMETHING TO EXPLAIN WHAT WE HAVE DONE

1.3 STRUCTURE OF THE THESIS

2 COMPRESSION PRINCIPLES AND METHODS

TODO: Some introduction about the concept and idea of entropy, taken from [16, 11, 8].

WORST CASE ENTROPY TODO: Write something about worst case entropy, taken from [11]. Just a paragraph or two.

2.1 SHANNON ENTROPY

Let's introduce the concept of entropy as a measure of uncertainty of a random variable. A deeper explanation can be found in [8, 11, 2]

Definition 2.1 (Entropy of a Random Variable)**.** *Let* X *be a random variable taking values in a finite alphabet* $\mathcal{X}$ *with the probabilistic distribution* $P_X(x) = Pr\{X = x\}$ $(x \in \mathcal{X})$*. Then, the entropy of* X *is defined as*

$$H(X) = H(P_X) \overset{\text{def}}{=} E_{P_x}\{-\log P_X(x)\} = -\sum_{x \in \mathcal{X}} P_X(x) \log P_X(x) \tag{1}$$

Where E_P denotes the expectation with respect to the probability distribution P. The log is taken to the base 2 and the entropy is expressed in bits. It is then clear that the entropy of a discrete random variable will always be nonnegative[1].

Example 2.2 (Toss of a fair coin)**.** *Let* X *be a random variable representing the outcome of a toss of a fair coin. The probability distribution of* X *is* $P_X(0) = P_X(1) = \frac{1}{2}$*. The entropy of* X *is*

$$H(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1 \tag{2}$$

This means that the toss of a fair coin has an entropy of 1 bit.

Remark 2.3. *Due to historical reasons, we are abusing the notation and using* $H(X)$ *to denote the entropy of the random variable* X*. It's important to note that this is not a function of the random variable: it's a functional of the distribution of* X*. It does not depend on the actual values taken by the random variable, but only on the probabilities of these values.*

The concept of entropy, introduced in definition 2.1, helps us quantify the randomness or uncertainty associated with a random variable. It essentially reflects the average amount of information needed to identify a specific value drawn from that variable. Intuitively, we can

1 The entropy is null if and only if $X = c$, where c is a costant with probability one

think of entropy as the average number of digits required to express a sampled value.

However, for continuous random variables (variables with an infinite number of possible values), expressing a single value with perfect accuracy requires an infinite number of digits. This is because any finite number of digits will only represent a range of possible values, not a single precise value. As a result, when we apply the definition of entropy to a continuous variable by dividing its range into increasingly smaller intervals and taking the limit, the entropy diverges to infinity.

2.1.1 *Properties*

In the previous section 2.1, we have introduced the entropy of a single random variable X. What if we have two random variables X and Y? How can we measure the uncertainty of the pair (X, Y)? This is where the concept of joint entropy comes into play. The idea is to consider (X, Y) as a single vector-valued random variable and compute its entropy. This is the joint entropy of X and Y.

Definition 2.4 (Joint Entropy)**.** *Let* (X, Y) *be a pair of discrete random variables* (X, Y) *with a joint distribution* $P_{XY}(x, y) = Pr\{X = x, Y = y\}$*. The joint entropy of* (X, Y) *is defined as*

$$H(X, Y) = H(P_{XY}) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P_{XY}(x, y) \log P_{XY}(x, y) \tag{3}$$

Which we can be extended to the joint entropy of n random variables $(X_1, X_2, \dots, X_n)$ as $H(X_1, \dots, X_n)$.

We also define the conditional entropy of a random variable given another as the expected value of the entropies of the conditional distributions, averaged over the conditioning random variable. Given two random variables X and Y, we can define $W(y|x)$, with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, as the conditional probability of Y given X. The set W of those conditional probabilities is called *channel* with *input alphabet* $\mathcal{X}$ and *output alphabet* $\mathcal{Y}$.

Definition 2.5 (Conditional Entropy)**.** *Let* (X, Y) *be a pair of discrete random variables with a joint distribution* $P_{XY}(x, y) = Pr\{X = x, Y = y\}$*. The conditional entropy of* Y *given* X *is defined as*

$$\begin{aligned}
H(Y|X) = H(W|P_X) &\overset{\text{def}}{=} \sum_{x} P_X(x) H(Y|x) &(4)\\
&= \sum_{x \in \mathcal{X}} P_X(x) \Big\{ -\sum_{y \in \mathcal{Y}} W(y|x) \log W(y|x) \Big\} &(5)\\
&= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P_{XY}(x, y) \log W(y|x) &(6)\\
&= E_{P_{XY}}\{-\log W(Y|X)\} &(7)
\end{aligned}$$

Since entropy is always nonnegative, conditional entropy is likewise nonnegative; it has value zero if and only if Y can be entirely determined from X with certainty, meaning there exists a function $f(X)$ such that $Y = f(X)$ with probability one.

The intuitive coherence of the definitions of joint entropy and conditional entropy becomes apparent when considering that the entropy of two random variables is equal to the entropy of one of them plus the conditional entropy of the other. This relationship is formally demonstrated in the following theorem.

Theorem 2.6 (Chain Rule). *Let (X, Y) be a pair of discrete random variables with a joint distribution $P_{XY}(x, y)$. Then, the joint entropy of (X, Y) can be expressed as*

$$H(X, Y) = H(X) + H(Y|X) \tag{8}$$

This is also known as additivity of entropy.

Proof. From the definition of conditional entropy (2.5), we have

$$\begin{aligned} H(X, Y) &= -\sum_{x,y} P_{XY}(x, y) \log W(y|x) \\ &= -\sum_{x,y} P_{XY}(x, y) \log \frac{P_{XY}(x, y)}{P_X(x)} \\ &= -\sum_{x,y} P_{XY}(x, y) \log P_{XY}(x, y) + \sum_{x,y} P_X(x) \log P_X(x) \\ &= H(XY) + H(X) \end{aligned}$$

Where we used the relation

$$W(y|x) = \frac{P_{XY}(x, y)}{P_X(x)} \tag{9}$$

When $P_X(x) \neq 0$. □

Corollary 2.7.

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z) \tag{10}$$

Proof. The proof is analogous to the proof of the chain rule. □

Corollary 2.8.

$$\begin{aligned} H(X_1, X_2, \ldots, X_n) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) \\ + \ldots + H(X_n|X_1, X_2, \ldots, X_{n-1}) \end{aligned} \tag{11}$$

Proof. We can apply the two-variable chain rule in repetition obtain the result. □

2.1.2 *Mutual Information*

Given two random variables X and Y, the mutual information between them quantifies the reduction in uncertainty about one variable due to the knowledge of the other. It is defined as the difference between the entropy and the conditional entropy

Definition 2.9 (Mutual Information)**.** *Let (X, Y) be a pair of discrete random variables with a joint distribution $P_{XY}(x, y)$. The mutual information between X and Y is defined as*

$$I(X;Y) = H(X) - H(X|Y) \tag{12}$$

Using the chain rule (2.6), we can rewrite it as

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(X) + H(Y) - H(X,Y) \qquad (13) \\ &= -\sum_x P_X(x)\log P_X(x) - \sum_y P_Y(y)\log P_Y(y) \\ &\quad + \sum_{x,y} P_{XY}(x,y)\log P_{XY}(x,y) \qquad (14) \\ &= \sum_{x,y} P_{XY}(x,y)\log\frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} \qquad (15) \\ &= E_{P_{XY}}\left\{\log\frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}\right\} \qquad (16) \end{aligned}$$

It immediately that the mutual information is symmetric, $I(X;Y) = I(Y;X)$.

2.1.3 *Fano's inequality*

Information theory serves as a cornerstone for understanding fundamental limits in data transmission and compression. It not only allows us to prove the existence of encoders achieving demonstrably good performance, but also establishes a theoretical barrier against surpassing this performance. The following theorem, known as Fano's inequality, provides a lower bound on the probability of error in guessing a random variable X to it's conditional entropy $H(X|Y)$, where Y is another random variable[2].

Theorem 2.10 (Fano's Inequality)**.** *Let X and Y be two discrete random variables with X taking values in some discrete alphabet $\mathcal{X}$, we have*

$$H(X|Y) \leqslant Pr\{X \neq Y\}\log(|\mathcal{X}| - 1) + h(Pr\{X \neq Y\}) \tag{17}$$

where $h(p) = -p\log p - (1-p)\log(1-p)$ is the binary entropy function.

2 We have seen in 2.5 that the conditional entropy of X given Y is zero if and only if X is a deterministic function of Y. Hence, we can estimate X from Y with zero error if and only if $H(X|Y) = 0$.

Proof. Let Z be a random variable defined as follows:

$$Z = \begin{cases} 1 & \text{if } X \neq Y \\ 0 & \text{if } X = Y \end{cases} \tag{18}$$

We can then write

$$\begin{aligned} H(X|Y) &= H(X|Y) + H(Z|XY) = H(XZ|Y) \\ &= H(X|YZ) + H(Z|Y) \\ &\leqslant H(X|YZ) + H(Z) \end{aligned} \tag{19}$$

The last inequality follows from the fact that conditioning reduces entropy. We can then write

$$H(Z) = h(\Pr\{X \neq Y\}) \tag{20}$$

Since $\forall y \in \mathcal{Y}$, we can write

$$H(X|Y = y, Z = 0) = 0 \tag{21}$$

and

$$H(X|Y = y, Z = 1) \leqslant \log(|\mathcal{X}| - 1) \tag{22}$$

Combining these results, we have

$$H(X|YZ) \leqslant \Pr\{X \neq Y\} \log(|\mathcal{X}| - 1) \tag{23}$$

From equations 19, 20 and 23, we have Fano's inequality. □

2.2 SOURCE AND CODE

We enrich the understanding of entropy by establishing its core role in setting the fundamental limit for information compression. This process involves condensing data by assigning shorter descriptions to more frequent outcomes and longer descriptions to less frequent ones. For example, Morse code uses a single dot to represent the most common symbol. Within this chapter, we ascertain the minimum average description length for a random variable. [2]

2.2.1 *Codes*

A source characterized by a random process generates symbols (letters) from a specific alphabet at each time step. The objective is to transform this output sequence into a more concise representation. This data reduction technique, known as *source coding* or *data compression*, utilizes a code to represent the original symbols more efficiently. The device that performs this transformation is termed an *encoder*, and the process itself is referred to as *encoding*. [8]

Definition 2.11 (Source Code)**.** *A source code for a random variable X is a mapping from the set of possible outcomes of X, called $\mathcal{X}$, to $\mathcal{D}^*$, the set of all finite-length strings of symbols from a $\mathcal{D}$-ary alphabet. Let $C(X)$ denote the codeword assigned to x and let $l(x)$ denote length of $C(x)$*

Definition 2.12 (Expected length)**.** *The expected length $L(C)$ of a source code C for a random variable X with probability mass function $P_X(x)$ is defined as*

$$L(C) = \sum_{x \in \mathcal{X}} P_X(x) l(x) \tag{1}$$

where $l(x)$ is the length of the codeword assigned to x.

Let's assume from now for simplicity that the $\mathcal{D}$-ary alphabet is $\mathcal{D} = \{0, 1, \dots, D-1\}$.

Example 2.13. *Let's consider a source code for a random variable X with $\mathcal{X} = \{a, b, c, d\}$ and $P_X(a) = 0.5$, $P_X(b) = 0.25$, $P_X(c) = 0.125$ and $P_X(d) = 0.125$. The code is defined as*

$$C(a) = 0$$
$$C(b) = 10$$
$$C(c) = 110$$
$$C(d) = 111$$

The entropy of X is

$$H(X) = 0.5 \log 2 + 0.25 \log 4 + 0.125 \log 8 + 0.125 \log 8 = 1.75 \textit{ bits}$$

The expected length of this code is also 1.75*:*

$$L(C) = 0.5 \cdot 1 + 0.25 \cdot 2 + 0.125 \cdot 3 + 0.125 \cdot 3 = 1.75 \textit{ bits}$$

In this example we have seen a code that is optimal in the sense that the expected length of the code is equal to the entropy of the random variable.

Example 2.14 (Morse Code)**.** *TODO from [2]*

Definition 2.15 (Nonsingular Code)**.** *TODO from [2]*

Definition 2.16 (Extension of a Code)**.** *TODO from [2]*

Definition 2.17 (Unique Decodability)**.** *TODO from [2]*

Definition 2.18 (Prefix Code)**.** *TODO from [2]*

2.2.2 *Kraft's Inequality*

Some introduction from [2]

Theorem 2.19 (Kraft's Inequality)**.** *TODO from [2]*

Proof. TODO from [2] □

2.2.3 *Source Coding Theorem*

Some introduction from [2, 16, 1, 8]

Theorem 2.20 (Source Coding Theorem)**.** *TODO from [2, 8]*

Proof. TODO from [2, 8] □

2.3 EMPIRICAL ENTROPY

TODO: Introduction to empirical entropy, from [8] in section 2.6 and in [11] at section 2.3

2.3.1 *Bit Sequences*

Definition 2.21 (zero-order empirical entropy)**.** *TODO*

TODO: Write here about the zero-order empirical entropy and show the connection with the worst case entropy, from [11]

Example 2.22. *TODO: Put here an example of the zero-order empirical entropy, from [11]*

2.3.2 *Entropy of a Text*

TODO: Introduction to the concept of entropy of a text, from [11] in section 2.3.2

Example 2.23. *TODO: Classic example on finding the zero-order empirical entropy of a word like abracadabra or banana*

2.4 INTEGER CODING

Most important references for this section: [4, 14, 11]

TODO: Introduce the following problem: given $S = \{x_1, x_2, \dots, x_n\}$, where $x_i \in \mathbb{N}$, we want to represent the integers of S as a sequence of bits that are self-delimiting. The goal is to minimize the space occupancy of the representation [4]. Add here some examples of where this problem appears in practice [17]

The central concern tackled in this section revolves around formulating an efficient binary representation method for an indefinite sequence of integers. Our objective is to minimize bit usage while ensuring that the encoding remains prefix-free. In simpler terms, we aim to devise a binary format where the codes for individual integers can be concatenated without ambiguity, allowing the decoder to reliably identify the start and end of each integer's representation within the bit stream and thus restore it to its original uncompressed state.

2.4.1 *Unary Code*

TODO: Introduction to unary code, from [4, 14]

Theorem 2.24. *The unary code of a positive integer x takes x bits, and thus it is optimal for the distribution $P(x) = 2^{-x}$. [4]*

Theorem 2.25. *Given a set of S integers, of maximum value M, the fixed-length binary code represents each of them in $\lceil \log_2(M) \rceil$ bits and is optimal for the distribution $P(x) = 1/M$. [4]*

2.4.2 *Elias Codes: γ and δ*

TODO: Introduction and explanation of Elias codes, from [4, 3]

Theorem 2.26. *The γ code of a positive integer x takes $2\lceil \log_2(x+1) \rceil - 1$ bits, and thus it is optimal for the distribution $P(x) = 1/x^2$. This is within a factor of two from the bit length $|B(x)| = \lceil \log_2(x) \rceil$ of the fixed-length binary code. [4]*

TODO: Talk a bit about the inefficiency of the γ code [4]

Theorem 2.27. *The δ code of a positive integer x takes $1 + \log_2 x + 2 \log_2 \log_2 x$ bits, and thus it is optimal for the distribution $P(x) = 1/x() \log x)^2$. This is within a factor of $1 + o(1)$ from the bit length $|B(x)| = \lceil \log_2(x) \rceil$ of the fixed-length binary code. [4]*

2.4.3 *Rice Code*

TODO: Just a brief mention on why we use it when values are concentrated around a certain range [4, 14] and on how it works.

2.4.4 *Elias-Fano Code*

TBD if to include this section or not. If included, it should be a brief mention of the Elias-Fano code and its use in integer compression [4].

2.5 STATISTICAL CODING

TODO: Introduction to statistical coding, from [8] and [4]

2.5.1 *Huffman Coding*

TODO: Chapter 4 of [14], chapter 12 of [4], 3.8 from [8], 5.6/7/8 of [2], 2.6 of [11] and the origianl paper [9]

Introduction of the type of source and the situation we are in.

TODO: Talk about construction, encoding and decoding, canonical Huffman codes [15], and the optimality of Huffman codes. [4, 14].

Example 2.28. *TODO: Classic example of Huffman coding with for example* $\mathcal{X} = \{a, b, c, d, e\}$ *and* $P(a) = 0.25$, $P(b) = 0.25$, $P(c) = 0.2$, $P(d) = 0.15$, $P(e) = 0.15$. *Do a nice tree and show the encoding of each symbol.*

There a are a few other theorems and lemmas that can be included in this section, TBD if to include them.

Theorem 2.29. *Let* H *be the entropy of a source emitting the symbols of an alphabet* Σ, *hence* $H = \sum_{\sigma \in \Sigma} P(\sigma) \log_2 \left(\frac{1}{P(\sigma)}\right)$. *Then, the average length of the Huffman code is* $L \leqslant H < L + 1$.

Proof. TODO: from [4], page 215. □

TODO and TBD: There are a lot of comments that can be done after this theorem. Still to decide what to include.

2.5.2 *Arithmetic Coding*

TODO: Do a brief introduction to Arithmetic Coding, explaining the idea behind it (Elias Code from 1960s) and the main differences with Huffman Coding. Section 12.2 from [4], section 4.2 fron [8], chapter 5 from [14].

Example 2.30. *TODO: Begin with an example to underline this differences*

ECODING PROCESS TODO: Talk about the compression algorithm and make an example of encoding a sequence of symbols. Add a pseudo code of the algorithm.

DECODING PROCESS TODO: Talk about the decompression algorithm and make an example of decoding a sequence of symbols. Add a pseudo code of the algorithm.

EFFICIENCY OF ARITHMETIC CODING

Theorem 2.31. *The number of bits emitted by arithmetic coding for a sequence* S *of* n *symbols is at most* $2 + n\mathcal{H}$*, where* $\mathcal{H}$ *is the empirical entropy of the sequence* S*.*

Proof. TODO: from [4], page 228-229. □

NOTE: this theorem requires a lemma and corollary to be proven first.

FURTER COMMENTS ON ARITHMETIC CODING TBD if to include this section. If so, it should show some other techniques such as range coding and prediction by partial matching.

2.6 HIGHER ORDER ENTROPY

TODO: A bit of introduction

Definition 2.32 (Redundacy)**.** *TODO: Give a formal definition of redundacy: informally is a measure of the distance between the source's entropy and the compression ration, and can thereby be seen as a measure of how fast the algorithm reaches the entropy of the source.*

While using measures like 2.32 are certainly intriguing, their feasibility is questionable due to the inherent challenge of determining the entropy of the source generating the string we aim to compress. To address this issue, an alternative empirical approach has been devised, which introduces the concept of the *k-th order empirical entropy* of a string S, denoted as $\mathcal{H}_k(S)$. In the preceding discussion on statistical coding (refer to section 2.5), we delved into the scenario where $k = 0$, relying on symbol frequencies within the string. Now, with $\mathcal{H}_k(S)$, our objective is to enrich the entropy concept by examining the frequencies of k-grams in string S. This entails analyzing subsequences of symbols with a length of k, thereby capturing the inherent *compositional structure* of S. [4]

Let S be a string over the alphabet $\Sigma = \{\sigma_1, \dots, \sigma_n\}$. Denote with n_ω the number of occurrences of the k-gram ω in S. [3]

Definition 2.33 (k-th Order Empirical Entropy)**.** *The* k-th order empirical entropy *of a string* S *is defined as*

$$\mathcal{H}_k(S) = \frac{1}{|S|} \sum_{\omega \in \Sigma^k} \left(\sum_{i=1}^{h} n_{\omega\sigma_i} \log \left(\frac{n_\omega}{n_{\omega\sigma_i}} \right) \right) \tag{1}$$

where $|S|$ *is the length of the string* S.

When considering a concrete sequence $S[1, n]$ we can compute the *empirical k-th entropy* of S by considering the frequencies of symbols depending on the k preceding symbols.

$$\mathcal{H}_k(S) = \sum_{\omega \in \Sigma^k} \frac{|S_\omega|}{n} \cdot \mathcal{H}_{\prime}(S_\omega) \tag{2}$$

where S_ω is a string formed by collecting the symbol that follows each occurrence of the k-gram $\omega = \sigma_1 \dots \sigma_k$ in S.

Example 2.34. *TODO: add here a numerical example where the first order empirical entropy is much less then the zero order empirical entropy.*

3 We will use the notation $\omega \in \Sigma^k$ to denote a k-gram, i.e., a subsequence of k symbols in the string S.

Extending the concept of zero-order empirical entropy, $n\mathcal{H}_k(S)$ serves as a lower bound for the minimum number of bits attainable by any encoding of S, under the condition that the encoding of each symbol may rely on itself and the k symbols preceding it in S. Consistently, any compressor that surpasses this threshold would also have the capability to compress symbols originating from the related kth-order source to a level lower than its Shannon entropy.

Remark 2.35. *As k grows large (up to $k = n - 1$, and often sooner), the k-th order empirical entropy of S reaches null, given that each k-gram appears only once. This renders our model ineffective as a lower bound for compressors. Even before reaching the k value where $\mathcal{H}_k(S) = 0$, compressors face practical hurdles in achieving the target of $n\mathcal{H}_k(S)$ bits, particularly for high k values. This is due to the necessity of storing the set of σ^{k+1} probabilities or codes, adding complexity to compression. Likewise, adaptive compressors must incorporate σ^{k+1} escape symbols into the compressed file, further complicating the process. In theory, it is commonly posited that S can be compressed up to $n\mathcal{H}k(S) + o(n)$ bits for any $k + 1 \leqslant \alpha \log \sigma n$ and any constant $0 < \alpha < 1$. In such cases, storing σ^{k+1} numbers within the range $[1, n]$ (such as the frequencies of the k-grams) requires $\sigma^{k+1} \log n \leqslant n^{\alpha} \log n = o(n)$ bits. [11]*

Definition 2.36 (Coarsely Optimal Compression Algorithm)**.** *A compression algorithm is coarsely optimal if, for every value of k, there exists a function $f_k(n)$ that tends to zero as the length of the sequence n approaches infinity, such that for all sequences S of increasing length, the compression ratio achieved by the algorithm remains within $\mathcal{H}_k(S) + f_k(|S|)$.*

The *Lempel-Ziv* algorithm (LZ78) serves as an example of a coarsely optimal compression technique, as outlined by Plotnik et al. in their paper referenced in [13]. This algorithm relies on the idea of dictionary-based compression. However, as highlighted by Manzini and Korařaju [10], the notion of coarse optimality doesn't necessarily guarantee the effectiveness of an algorithm. Even when the entropy of the string is extremely low, the algorithm might still perform inadequately due to the presence of the supplementary term $f_k(|S|)$.

FURTHER COMMENTS ON LZ77 AND LZ78 TBD if to include this section, but I think it's not relevant for the thesis. If included, it should discuss very briefly the LZ77 and LZ78 algorithms, and the differences between them. [4]. And then prove two lemmas: one about the compression ration achieved by LZ78 and the other about LZ77 not being coarsely optimal. [4], end of chapter 13.

2.7 BITVECTORS

TODO: Make an introduction to problem, explaining how we arrive to needing to implement rank and select on bitvectors. Make clear of what are our goals and that rank and select are our solution to the problem.

2.7.1 *Rank*

2.7.2 *Select*

2.7.3 *RRR: A Space-Efficient Rank/Select Structure*

3 WAVELET TREES

3.1 PRELIMINARIES

3.1.1 *Suffix Array*

3.1.2 *Burrows-Wheeler Transform*

3.2 WAVELET TREES

First introduce the following problem: let's consider a sequence $S[1,n]$ as a generalization of bitvectors whose elements $S[i]$ are drawn from an alphabet Σ. The problem is to support the following queries:

- `Access(i)`: return the i-th element of S.
- `Rank(c,i)`: return the number of occurrences of character c in the prefix $S[1,i]$.
- `Select(c,i)`: return the position of the i-th occurrence of character c in S.

Explain the inconvenience of a solution that stores one bitvector per symbol (look from [11] and [4]).

3.2.1 *Structure*

Explain how a wavelet tree is built, from [11] and [6]

3.2.2 *Rank and Select*

3.2.3 *Construction*

3.3 COMPRESSED WAVELET TREES

Compressing the bitvectors using RRR

3.4 APPLICATIONS OF WAVELET TREES

Take from [12, 7, 5]

4 SUBSET WAVELET TREES

4.1 DEGENERATE STRINGS

Brief introduction to degenerate strings

4.1.1 *Subset-Rank and Subset-Select*

4.2 CONSTRUCTION OF THE SUBSET WAVELET TREE

BIBLIOGRAPHY

[1] N. Vereshchagin A. Shen V. A. Uspensky. *Kolmogorov Complexity and Algorithmic Randomness*. Mathematical Surveys and Monographs. Amer Mathematical Society, 2017.

[2] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 2012.

[3] P. Elias. "Universal codeword sets and representations of the integers." In: *IEEE Transactions on Information Theory* 21.2 (1975), pp. 194–203.

[4] P. Ferragina. *Pearls of Algorithm Engineering*. Cambridge University Press, 2023.

[5] Paolo Ferragina, Raffaele Giancarlo, and Giovanni Manzini. "The myriad virtues of Wavelet Trees." In: *Information and Computation* 207.8 (2009), pp. 849–866. ISSN: 0890-5401.

[6] Roberto Grossi, Ankur Gupta, and Jeffrey Vitter. "High-Order Entropy-Compressed Text Indexes." In: *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms* (Nov. 2002).

[7] Roberto Grossi, Jeffrey Scott Vitter, and Bojian Xu. "Wavelet Trees: From Theory to Practice." In: *2011 First International Conference on Data Compression, Communications and Processing*. 2011, pp. 210–221.

[8] T.S. Han and K. Kobayashi. *Mathematics of Information and Coding*. Fields Institute Monographs. American Mathematical Society, 2002.

[9] David A. Huffman. "A Method for the Construction of Minimum-Redundancy Codes." In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101.

[10] S Rao Kosaraju and Giovanni Manzini. "Compression of low entropy strings with Lempel–Ziv algorithms." In: *SIAM Journal on Computing* 29.3 (2000), pp. 893–911.

[11] G. Navarro. *Compact Data Structures: A Practical Approach*. Cambridge University Press, 2016.

[12] Gonzalo Navarro. "Wavelet trees for all." In: *Journal of Discrete Algorithms* 25 (2014). 23rd Annual Symposium on Combinatorial Pattern Matching, pp. 2–20. ISSN: 1570-8667.

[13] Eli Plotnik, Marcelo J Weinberger, and Jacob Ziv. "Upper bounds on the probability of sequences emitted by finite-state sources and on the redundancy of the Lempel-Ziv algorithm." In: *IEEE transactions on information theory* 38.1 (1992), pp. 66–72.

[14] K. Sayood. *Lossless Compression Handbook*. Communications, Networking and Multimedia. Elsevier Science, 2002, pp. 55–64.

[15] Eugene S Schwartz and Bruce Kallick. "Generating a canonical prefix encoding." In: *Communications of the ACM* 7.3 (1964), pp. 166–169.

[16] C. E. Shannon. "A mathematical theory of communication." In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[17] Ian H Witten, Alistair Moffat, and Timothy C Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.