Lossless Compression based on Bloom Filter

Submitted by:Jionglin Tao
Supervisor:M. R. C. van Dongen

Abstract

A Bloom filter is a space-efficient probabilistic data structure, it is used to test whether an element in a set. Bloom filter uses a small bitmap and a collection of k hash functions to implement the test, but it will cause false positive rate.To eliminate the false positive, this project use a witness to represent the filter's false positive. In this report, the project implement a new lossless compression and decompression algorithm that used bloom filter and witness. The compression ratio of the new lossless compression depend on number of hash functions, size of bloom filter bitmap and 1 value probability of input bitmap. And for video file, this project implement a pre-processing algorithm that can reduce 1 value probability of input bitmap to improve the compression ratio. This report evaluated this algorithm against existing lossless compression algorithms.

Contents

1. Introduction
2. Analysis
3. Design
    3.1 Bloom Filter
    3.2 The Witness Data Structure
    3.3 Algorithm illustration
    3.3 Pre-processing for Video File with frames
    3.3 Tools, technologies,packages and libraries
4. Implementation
    4.1 Algorithm Implementation
        4.1.1 Hash Functions
        4.1.2 Compression Algorithm
        4.1.3 Decompression Algorithm
        4.1.4 Video File Pre-processing Algorithm
    4.2 Proofs of Correctness
    4.3 Performance Analysis
    4.4 Uncompleted Work
5. Evaluation
    5.1 Compression Result
    5.2 Evaluate with Existing Algorithm
6. Conclusion and Future Work

1. Introduction

In an age of limited network bandwidth, data compression is particularly valuable. The prospect of efficient data compression enabling large web applications to be made possible on the move is very attractive. With the advent of the Big Data era, the volume of data and the rate at which it is growing are at an all-time high. With the rapid development of 5G technology, there is an increasing demand for applications and application scenarios such as edge computing and the Internet of Things. With limited transmission networks and storage capacity, data compression technology plays an increasingly important role.

Lossless data compression requires that the compressed and decompressed file must be identical to the original file. In this project, a new lossless compression algorithm is implemented through the use of Bloom filters. A Bloom filter is a data structure used to determine whether an element exists in a set, while the quarries have false positives rate. So to achieve lossless compression, the false positive rate of the Bloom filter needs to be eliminated. This project uses wittiness which is used to represent false positive to eliminate false positives, thus meeting the requirements of a lossless compression algorithm.

This report studied how to use the bloom filter to implement to lossless compression,the analysis and design of algorithm, the process of implementing the algorithm, how to improve the compression ratio of lossless compression and the evaluation of the algorithm, and draws conclusions. The algorithm has good compression performance in specific situations. Many challenges were encountered during the process of the project, but not all of them were solved well.

2. Analysis

The objectives of project is to implement a new lossless compression and decompression algorithm, and evaluate the algorithms against existing algorithms.Lossless compression is a class of data compression that allows the original data to be perfectly reconstructed from the compressed data with no loss of information. Lossless compression is possible because most real-world data exhibits statistical redundancy. The new lossless compression used bloom filter.

A bloom filter consists of a bitmap and k hash functions. To initialize a bloom filter, it need to add all element of the set first. To add an element, feed it to each of the k hash functions to get k array positions. Set the bits at all these positions to 1.To query an element, supply it to each of the k hash functions to obtain the k array positions. If any of the bits in these positions are 0, then the element must not be in the set; if so, then all bits will be set to 1 on insertion. if all are 1, then either the element is in the set, or by chance these bits are set to 1 during insertion of other elements, resulting in a false alarm.This is false positive.

To query a element e whether in a set T will have 4 kinds outcome. True positive, false positive,true negative, false negative. As table1.

True Positive	e ⊆T, H(e) ⊆ H(T)
False Positive	e⊈T,H(e) ⊆H(T)
False Negative	e ⊆T,H(e)⊈H(T)
True Negative	e⊈T,H(e)⊈H(T)

Table 1.True&False Positive&Negative

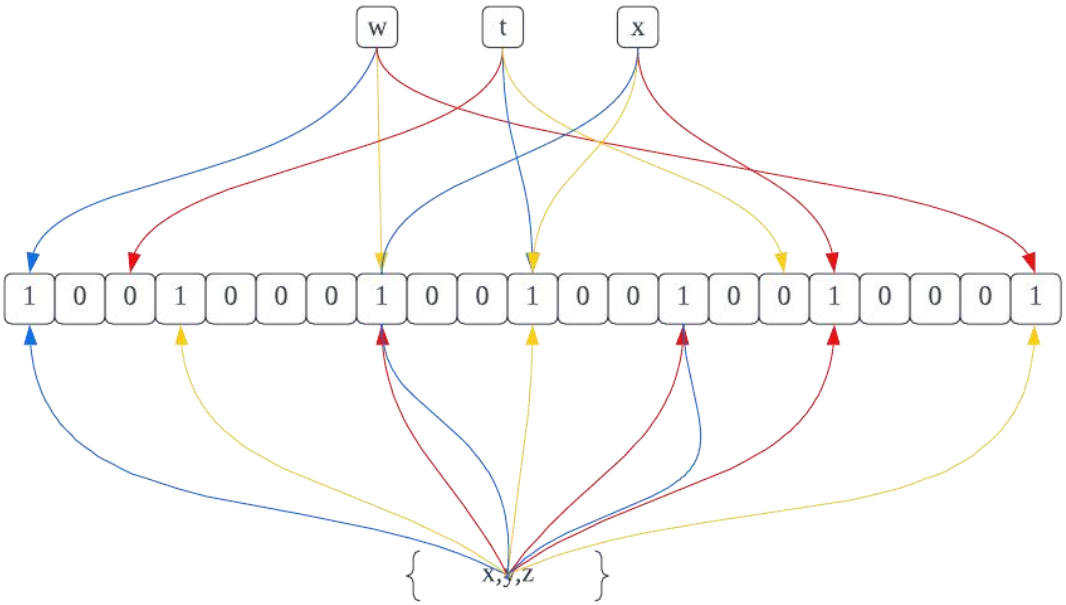

Figure 1. Simple Bloom Filter

As shown in Figure 1, add all the elements in the set {x,y,z} to the bitmap, get different hash function values for each element as the position and point to

the bitmap, set the value of these positions as 1. The different coloured lines in the figure indicate different hash functions, there are k=3 hash functions. Query w, t, x whether them are in set {x,y,z}. w has a hash function value pointing to 0, so it must not be in the set {x,y,z}. t has all hash values pointing to 1, but it is not in the set {x,y,z}, so t is a false positive generated by the Bloom filter. x has all hash function values pointing to 1, and x is in the set {x,y,z}. The Bloom filter has false positives, but querying an element that is not in the set, then that element must not be in the set. That bloom filter do not have false negative.

To implement to new lossless compression algorithm,it needs to eliminate the false positive of bloom filter, but the false positive can not eliminate directly. In this project, it used a witness data structure to record the false positive of bloom filter, the witness is constructed in data compression,and when decompression data, it can use witness to avoid the false positive.

As Figure2. In compression process, if an element passed constructed bloom filter, then check it whether in input set. If it in input set,then it is true positive,add 1 to witness, else it is a false positive, add 0 to witness. After all element queried, append the witness to bloom filter.

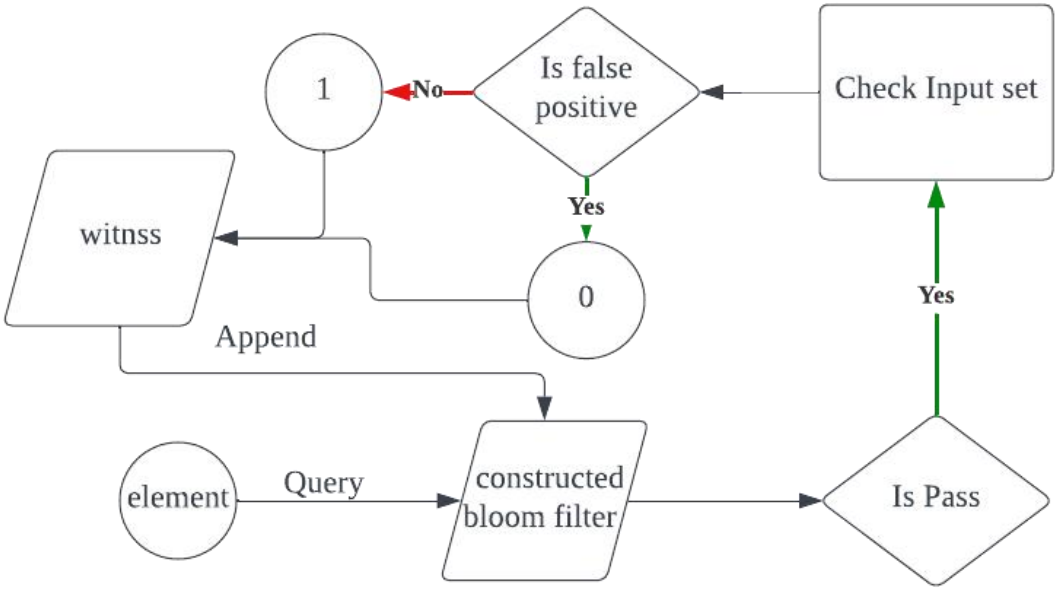

Figure2 Witness in Compression

As Figure 3. In decompression process, if an element passed the bloom filter, check the witness, if it is 1,then it is true positive, add 1 to out put,else it is false positive, add 0 to output. So that the witness can avoid the false positive of bloom filter.

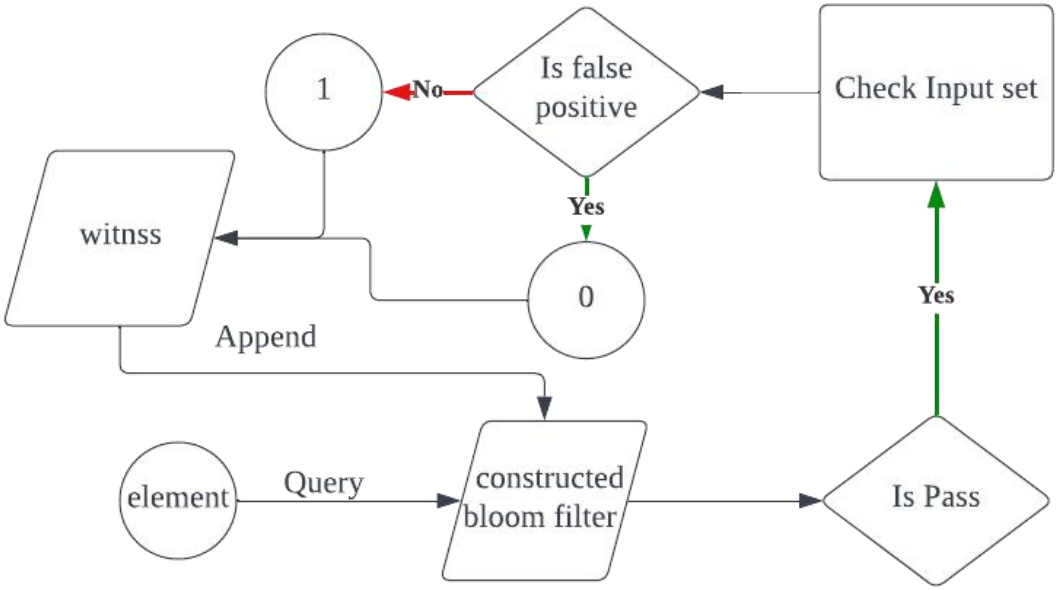

Figure3.Witness in Decompression

Through use bloom filter and witness, this project can implement a new lossless compression algorithm. And this project need to evaluate the new algorithm against existing lossless algorithm. The expected deliverables listed In table 2.

Deliverables	Requirement Detail
Compression&Decompression Application	A software application that can implement a new lossless compression and decompression algorithm. This algorithm used bloom filter and witness data structure and need meet the requirements of lossless compression. The application can compress files in different formats and batch compression.
Project Report	Describe all information relevant to the project. Include: Introduction,Analysis,Design,Implementation,Evaluation,Conclusions and References.
Compression and Decompression Test Result	A file record the compression and decompression test result. Include: Data information, number of hash functions, compression ratio etc.

Table2.Expected Deliverables

3. Design

3.1 Bloom Filter

Bloom filter is a data structure to test a element whether in a set. But in this lossless compression, it used as a container for storing data. Assume that input is a n-bits sequence T and the bloom filter bitmap is a l-bits(0<l<=n) sequence B. Since the bit sequence has only two values of 0 or 1 per bit.It only need to add the indexes of bit with the value of 1 to bloom filter. And the length of bloom filter bitmap need shorter than input bits sequence to achieve a compression effect.

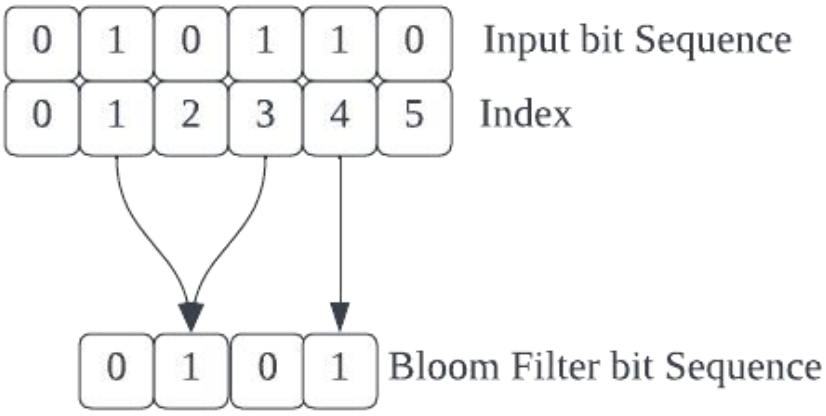

Figure4.Construct Bloom Filter

As the Figure4, it added the indexes of input bits sequence bit with the value of 1, instead add the bit. To construct the bloom filter needs to get the indexes {i,i2,i3,i4} and get the hash functions value of them, then set the position indicated by the hash value in the Bloom filter bits sequence to 1.

Hash functions are very important for a bloom filter. A bloom filter has k different hash functions. If the length of bloom filter have no limit. Then the k is greater the false positive rate is lower. But in this algorithm, length of bloom filter bit sequence l is limited that 0<l<=n. So greater k values are not better. It need to get appropriate k and l for bloom filter. And the choice of hash function type is also important.

In decompression, it need to construct an empty bit sequence first, Then set the position at which the index value can pass through the Bloom filter to 1. The decompression process is a reversal of the compression process.

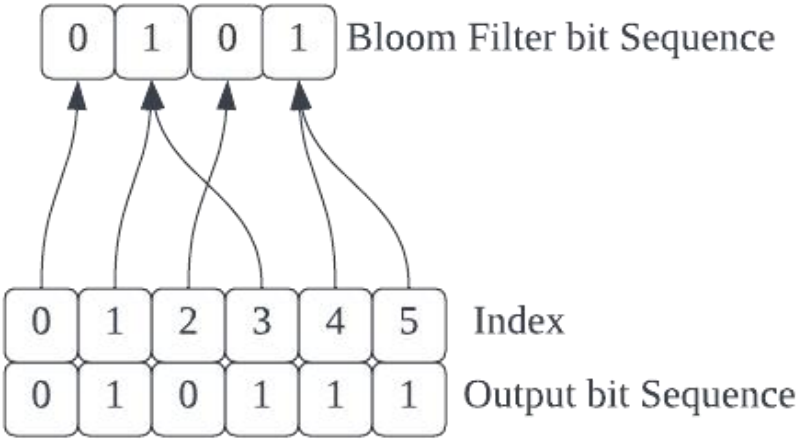

Figure5.Decompression without witness

As the Figure5, The output bit sequence is not as same as input bit sequence. Because the hash value of index value 5 points to 1 in bloom filter.So the bit with the index value of 5 is set to 1 instead of the original 0. This is false positive of bloom filter. If only use bloom filter can not implement lossless compression. In this project, it add witness data structure to solve the problem.

3.2 The Witness Data Structure

The witness is used to record the false positive in compression process and avoid the false positive in decompression process. The witness is a bit sequence too. As the analysis section, witness will check a element that pass a bloom filter whether in input and record it,and in decompression, if hash values of a index pass the bloom filter,then need to check in witness whether it is false positive.

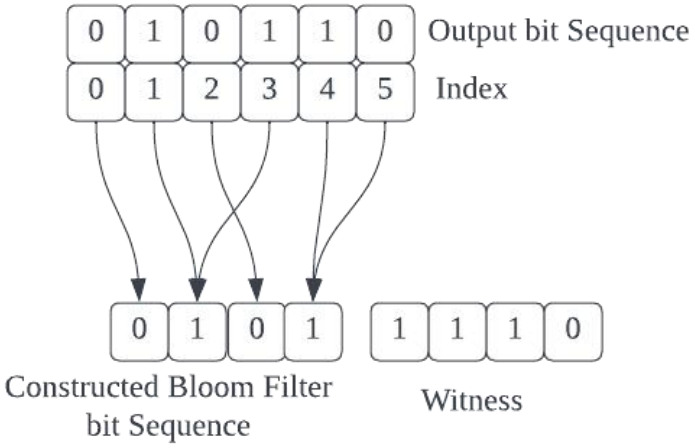

Figure6.Witness in Compression

As the Figure6.After construct the bloom filter as Figure4, it need to double check every index of input bit sequence.When a index passed constructed bloom filter,then check it whether in input bits sequence.If true,add 1 to witness, else add 0 to witness. So that witness can record the false positive.

In decompression process, in Figure 5, the index 5 is the last one passed the bloom filter,but the last bit in witness is 0, so it can know that index 5 is a false positive and set it to 0. The other three index are all true positive. Then the output sequence is as same as the input sequence that lossless compression is achieved.

3.3 Algorithm illustration

The main idea of the new lossless compression technique is to use bloom filters and witnesses to achieve this. The whole process as the Figure7.

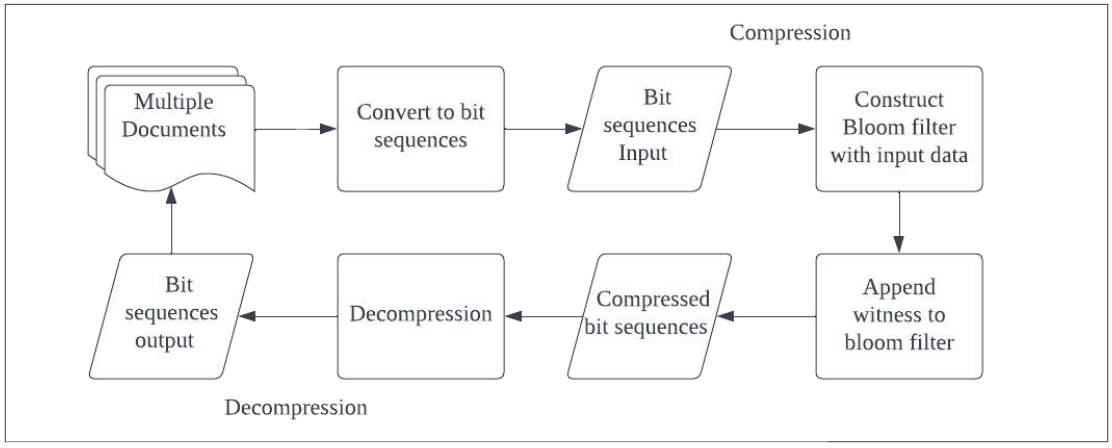

Figure6.Compression and Decompression Process

To compress multiple documents,need to convert them to bits sequences, for each bits sequence as an input,then construct the bloom filter and append the witness to bloom filter bits sequence. These steps make up the compression process. And the algorithm returns a compressed bit sequence including bloom filter and witness. In decompression process, the compressed bit sequence as a input, and return a output bit sequence that as same as the original input bit sequence. The output bits sequences can convert to original documents.

Assuming that the input is a n-bits sequence, the 1 value probability of input bits sequence is p(0<=p<=1) .the bloom filter have l-bits sequence,k hash functions defined and false positive rate is fpr.Following Mitzenmacher and Upfal[2005],Under the assumption that the hash functions are all random, the false positive rate for elements that are not in the set can be calculated

directly, and the probability that a random element input will result in 0 after Bloom's filter is constructed is

$$(1 - 1/l)^{kpn} \approx e^{-kpn/l}$$

And the false positive rate fpr is:

$$\mathrm{fpr} = (1 - e^{-\mathrm{kpn/l}})^{\mathrm{k}}$$

3.4 Pre-processing for Video File with frames

For video compression use this new algorithm, it can do a pre-processing for video. The pre-processing use frames of video to minimize the p of input bits sequence, then the bits sequence is more sparse. Three types of frames are used in video compression: I, P, and B frames. As table2.

I-frame (Intra-coded picture)	A complete image, like a JPG or BMP image file.
P-frame (Predicted picture)	holds only the changes in the image from the previous frame.
B-frame (Bidirectional predicted picture)	saves even more space by using differences between the current frame and both the preceding and following frames to specify its content.

Table2.I,P,B frame

The purpose of pre-processing is to make the input bit sequence more sparse, without directly compressing the video. The method uses the idea of I-frames and P-frames. By recording only one I-frame and encoding all other frames as p-frames, the input bit sequence will be sparser and then compressed by means of a Bloom filter, which will result in much better compression effect.

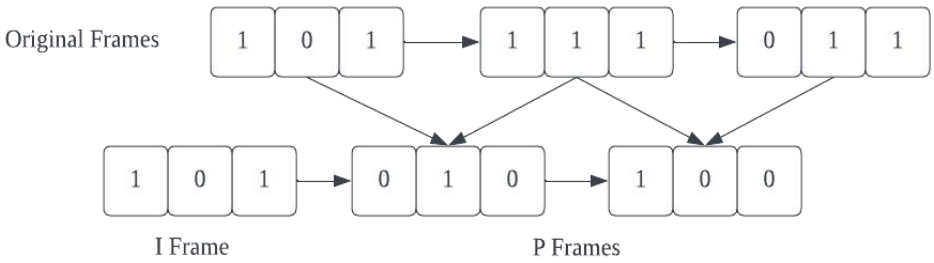

Figure7.Pre-processing Example

As Figure7, it is an pre-processing example. The original input bit sequence have three frame, there are 7 values of 1. And after use P frame, the whole bits sequence have 4 values of 1 that is more sparse than original bits sequence. Then in compression process using bloom filter, p of input bits

sequence is smaller. As Figure8.

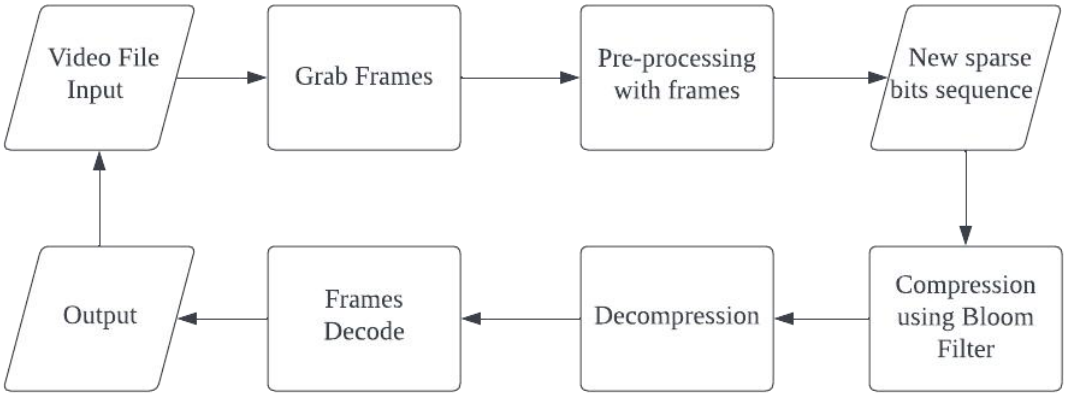

Figure8. Video Compression

3.5 Tools, technologies,packages and libraries

The application is a pure Java Maven project. It used the IntelliJ IDEA to build the application. IntelliJ IDEA is an integrated development environment written in Java for developing computer software written in Java, Kotlin, Groovy, and other JAR based languages.

The project import a BitSequence package to construct the bloom filter. This package is a Java bit sequence utility.It has the following functions:

1. Create byte sequence required length from byte array or number
2. Convert back to byte array or number
3. SubSequence, split, concat, insert, extract of bit sequences
4. Shift, rotate
5. Bitwise and, or, xor, not
6. Set, clear bits
7. Iterate over bit sequence, boolean or int, reverse iterator
8. Get binary string, continuous or grouped to byte, half-byte or any group size

In video pre-processing, the project import Jcodec library and add it in maven. It is a pure java implementation of video/audio codecs. Currently JCodec supports:

1. Video codecs
2. Audio codecs
3. Formats: MP4 (MOV) demuxer / muxer, MKV (Matroska) demuxer / muxer, MPEG PS demuxer, MPEG TS demuxer, WAV demuxer/muxer, MPEG Audio (MP3) demuxer, ADTS demuxer, DPX parser.

4. Implementation

4.1 Algorithm Implementation

4.1.1 Hash Functions

4.1.2 Compression Algorithm

4.1.3 Decompression Algorithm

4.1.4 Video File Pre-processing Algorithm

4.2 Proofs of Correctness

4.3 Performance Analysis

4.4 Uncompleted Work

5. Evaluation

5.1 Compression Result

5.2 Evaluate with Existing Algorithm

6. Conclusion and Future Work