Assigment d2: Compression Algorithm

Lise Johansen

October 2024

1 Introduction

For this assignment, the task was to implement an alternative compression algorithm. The code originally used Variable Byte Encoding (VBE) to compress the inverted index. I first implemented an Elias Gamma Code (EGC) algorithm using bitarrays. Then made the VBE algorithm work with a bitarray (originally used a bytearray).

File that is edited:

- **postinglist.py**: Changed the codecs used.
- **variablebytecodeec.py**: implementation of the encode and the decode
- **test/assignment.py**: added d-2 as a target which runs the corresponding tests.

2 Experiments

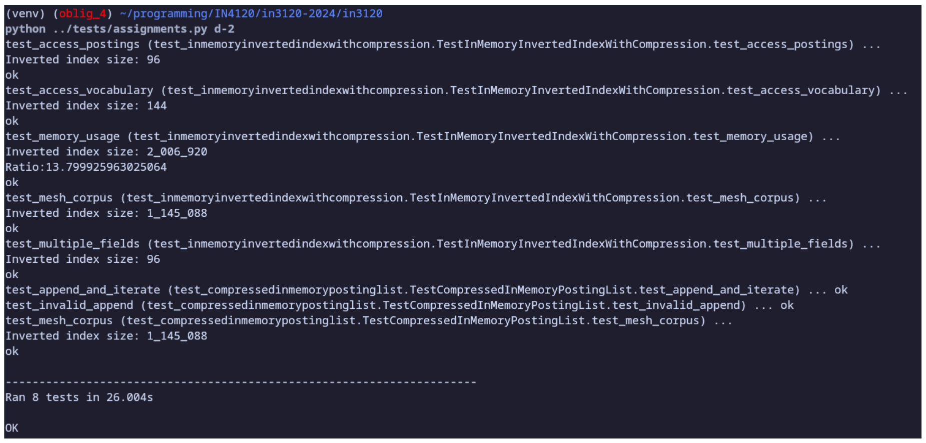
I summed up the size of all the lengths of the bitarrays used for the posting lists. From this sum, I got the combined memory usage of all the posting lists in the inverted index. Then I compared the sum from the VBE algorithm and the EGC algorithm.

Also summed up all the sizes for all the inverted indexes as a class variable to see the average size across the different corpora.

As an additional test, we can measure the time it takes to build a compressed index versus the time it takes to build an uncompressed index.

3 Result

As shown in the picture the Variable Byte Encoding has a memory usage of 2 006 920 bits, and across all tests, the memory usage results in 4 297 432 bits.

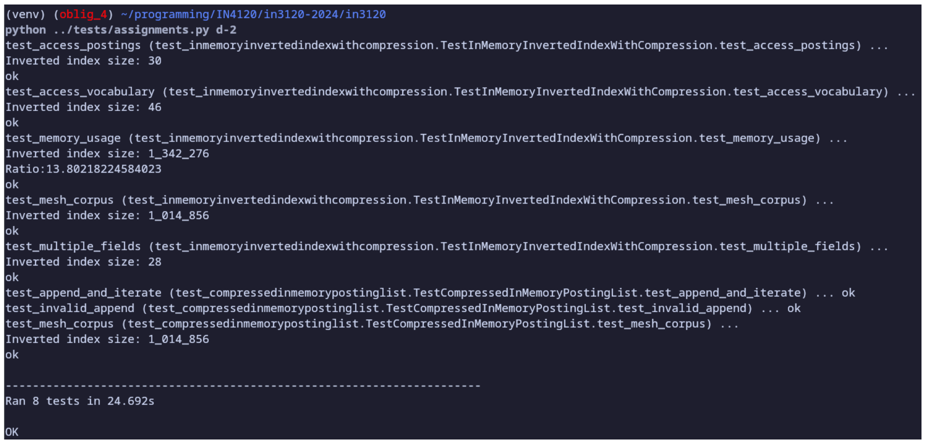

```
(venv) (oblig_4) ~/programming/IN4120/in3120-2024/in3120
python ../tests/assignments.py d-2
test_access_postings (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_access_postings) ...
Inverted index size: 96
ok
test_access_vocabulary (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_access_vocabulary) ...
Inverted index size: 144
ok
test_memory_usage (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_memory_usage) ...
Inverted index size: 2_006_920
Ratio:13.799925963025064
ok
test_mesh_corpus (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_mesh_corpus) ...
Inverted index size: 1_145_088
ok
test_multiple_fields (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_multiple_fields) ...
Inverted index size: 96
ok
test_append_and_iterate (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_append_and_iterate) ... ok
test_invalid_append (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_invalid_append) ... ok
test_mesh_corpus (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_mesh_corpus) ...
Inverted index size: 1_145_088
ok

----------------------------------------------------------------------
Ran 8 tests in 26.004s

OK
```

Figure 1: Test and the inverted index information for the Variable Byte Encoding

The Elisa Gamma code has a memory usage of 1 342 276 bits, and across all tests a memory usage of 3 372 092 bits, as shown in the picture.

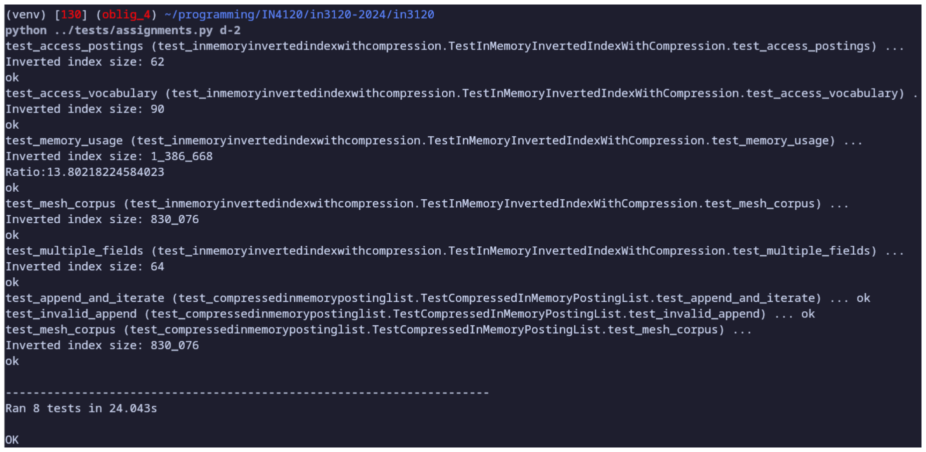

```
(venv) (oblig_4) ~/programming/IN4120/in3120-2024/in3120
python ../tests/assignments.py d-2
test_access_postings (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_access_postings) ...
Inverted index size: 30
ok
test_access_vocabulary (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_access_vocabulary) ...
Inverted index size: 46
ok
test_memory_usage (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_memory_usage) ...
Inverted index size: 1_342_276
Ratio:13.80218224584023
ok
test_mesh_corpus (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_mesh_corpus) ...
Inverted index size: 1_014_856
ok
test_multiple_fields (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_multiple_fields) ...
Inverted index size: 28
ok
test_append_and_iterate (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_append_and_iterate) ... ok
test_invalid_append (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_invalid_append) ... ok
test_mesh_corpus (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_mesh_corpus) ...
Inverted index size: 1_014_856
ok

----------------------------------------------------------------------
Ran 8 tests in 24.692s

OK
```

Figure 2: Information about the Elisa Gamma Code and its test.

When using the Variable byte encoding for the gap and Elias Gamma Code when compressing the term frequency we get a memory usage of 1 386 668 bits. And across all tests, this sums up to 3 047 036 bits.



```
(venv) [130] (oblig_4) ~/programming/IN4120/in3120-2024/in3120
python ../tests/assignments.py d-2
test_access_postings (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_access_postings) ...
Inverted index size: 62
ok
test_access_vocabulary (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_access_vocabulary) .
Inverted index size: 90
ok
test_memory_usage (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_memory_usage) ...
Inverted index size: 1_386_668
Ratio:13.80218224584023
ok
test_mesh_corpus (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_mesh_corpus) ...
Inverted index size: 830_076
ok
test_multiple_fields (test_inmemoryinvertedindexwithcompression.TestInMemoryInvertedIndexWithCompression.test_multiple_fields) ...
Inverted index size: 64
ok
test_append_and_iterate (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_append_and_iterate) ... ok
test_invalid_append (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_invalid_append) ... ok
test_mesh_corpus (test_compressedinmemorypostinglist.TestCompressedInMemoryPostingList.test_mesh_corpus) ...
Inverted index size: 830_076
ok

----------------------------------------------------------------------
Ran 8 tests in 24.043s

OK
```

Figure 3: Shows the use of the combination of both algorithms

In terms of the time measurement, we get a ratio of 0.77, which suggests that there is not much of a difference between compressing the index and not.

4 Discussion

4.1 My version of the VBE algorithm

Rewrote the VBE algorithms since I wanted to work with bits directly. Using bitarrays also made it easier to swap the VBE and EGC and combine them later.

This algorithm splits the number into segments of 7 bits and prefixes each of these segments with a continuation bit. When reading it, we check if the continuation bit is set, and if so, read the next segment as part of the decoded number.

4.2 Gammas Elias Encoding

This algorithm, which is also implemented through the bitarray interface, will prefix the length of the number to itself. This way the decoder knows how many bits it needs to read when figuring out the number stored. This works quite well for small values but gets increasingly worse than the VBE algorithm as we try to store larger values. For instance, values of length 4 or 5 in binary, will consume 8 or 10 bits at encoding, at which point we'd rather just use VBE instead.

4.3 The final compression algorithm

The plot shows the values of the inverted index. The term frequency has a value between 1 to 9. And the gap between listings has a value between 1-4000.

]

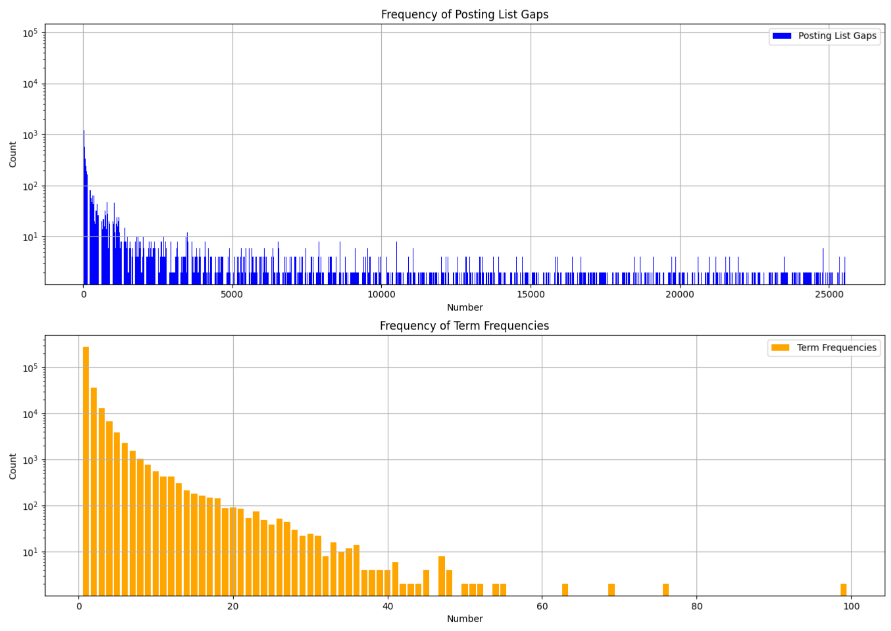


Figure 4: Frequency of data values

The Elias Gamma code works best for small numbers, between 1 to 4. Since every number is doubled in length. The Variable Byte encoding works best for numbers close to the 2**(7n) threshold, but only if it doesn't overshoot it since then it would take a whole other 8 bits to store the value.

Since most of the term frequency has a value of 1, and the gaps have no value over the 2**(7n) value in the inverted index. I could use the EGC for the term frequency and the VBE for the compression of the gaps.

4.4 Result

When looking at memory usage overall the combination of both algorithms seems to be the best. But when only looking at the the specific compression test for the inverted index the EGC algorithm did a little bit better. I think it is because of the sheer amount of ones in the data set.

4.5 Improvements

The improvement I could make is to test other compression algorithms. I could implement Simple9, which is also used when encoding small integers.

5 Conclusion

First I implemented an EGC, and then made a VBE that worked interchangeably with the VBE algorithm. Then I chose to focus on overall compression which led me to use a combination of both EGC and VBE. Knowing information about your data allows us to selectively choose an algorithm that performs better for our specific needs.