Analysis and Design of Algorithms

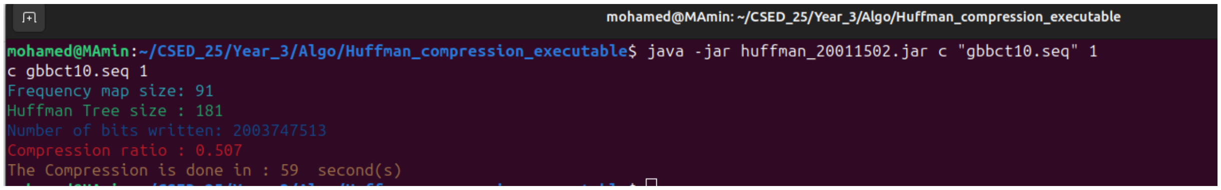
Huffman compression

Name: Mohamed Elsayed Amin ID: 20011502

File1: NIH genetic sequence database

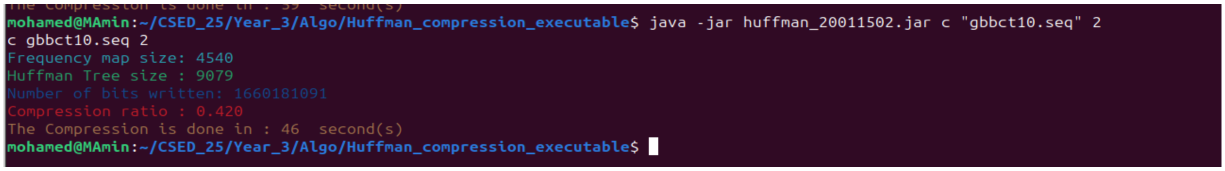
N	Compression time
1	59 seconds
2	46 seconds
3	41 seconds
4	40 seconds
5	49 seconds

N = 1:


mohamed@MAmin: ~/CSED_25/Year_3/Algo/Huffman_compression_executable

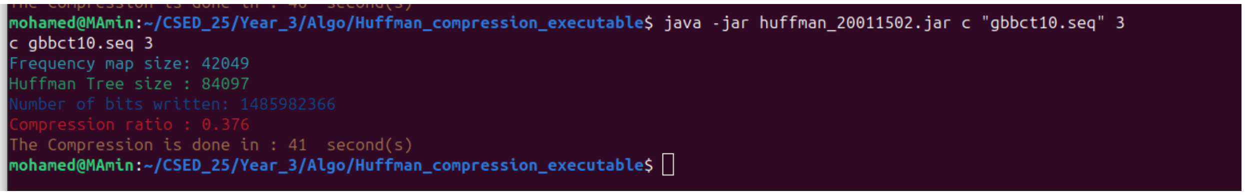
```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "gbbct10.seq" 1
c gbbct10.seq 1
Frequency map size: 91
Huffman Tree size : 181
Number of bits written: 2003747513
Compression ratio : 0.507
The Compression is done in : 59  second(s)
```

N = 2:



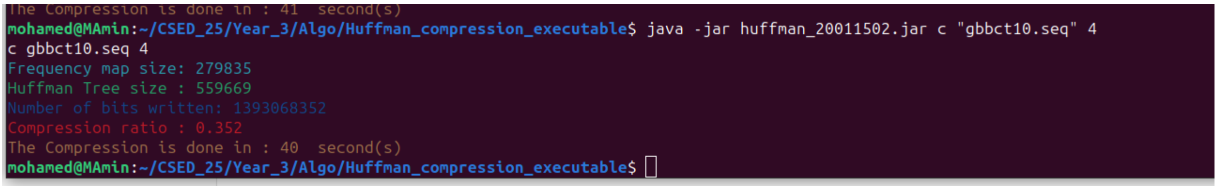
```
The Compression is done in : 59  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "gbbct10.seq" 2
c gbbct10.seq 2
Frequency map size: 4540
Huffman Tree size : 9079
Number of bits written: 1660181091
Compression ratio : 0.420
The Compression is done in : 46  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ █
```

N = 3:



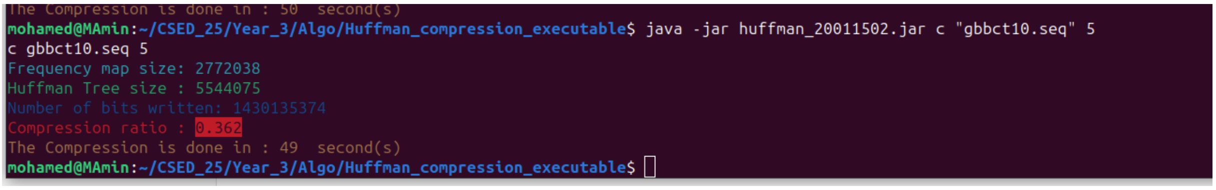
```
The Compression is done in : 46  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "gbbct10.seq" 3
c gbbct10.seq 3
Frequency map size: 42049
Huffman Tree size : 84097
Number of bits written: 1485982366
Compression ratio : 0.376
The Compression is done in : 41  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ ▯
```

N = 4:



```
The Compression is done in : 41  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "gbbct10.seq" 4
c gbbct10.seq 4
Frequency map size: 279835
Huffman Tree size : 559669
Number of bits written: 1393068352
Compression ratio : 0.352
The Compression is done in : 40  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

N = 5:

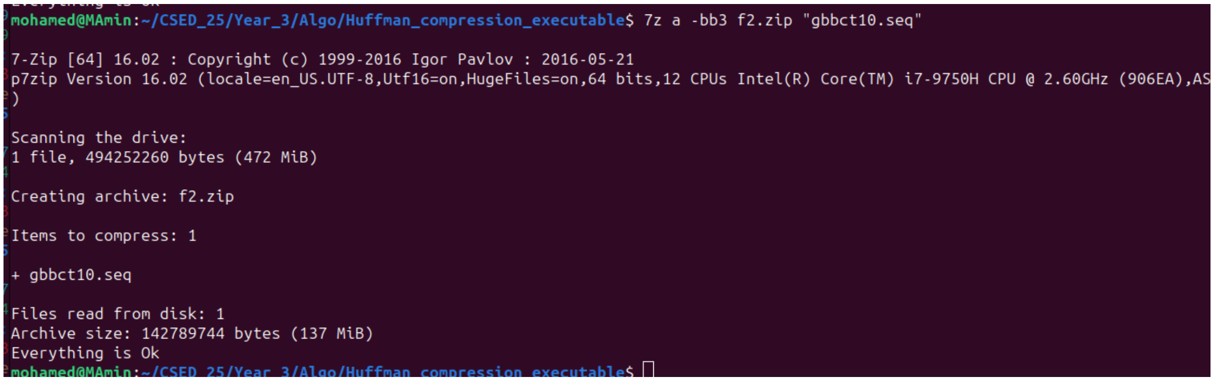


```
The Compression is done in : 50  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "gbbct10.seq" 5
c gbbct10.seq 5
Frequency map size: 2772038
Huffman Tree size : 5544075
Number of bits written: 1430135374
Compression ratio : 0.362
The Compression is done in : 49  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

N	Compression ratio
1	0.507
2	0.420
3	0.376
4	0.352
5	0.362

Compression ratio = size after compression / size before compression

7zip ratio = 0.3

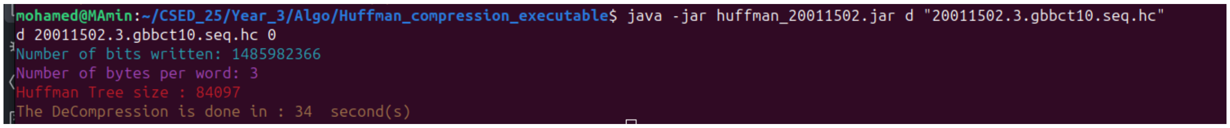


```
Everything is Ok
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 7z a -bb3 f2.zip "gbbct10.seq"

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,12 CPUs Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz (906EA),AS
)

Scanning the drive:
1 file, 494252260 bytes (472 MiB)

Creating archive: f2.zip

Items to compress: 1

+ gbbct10.seq

Files read from disk: 1
Archive size: 142789744 bytes (137 MiB)
Everything is Ok
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

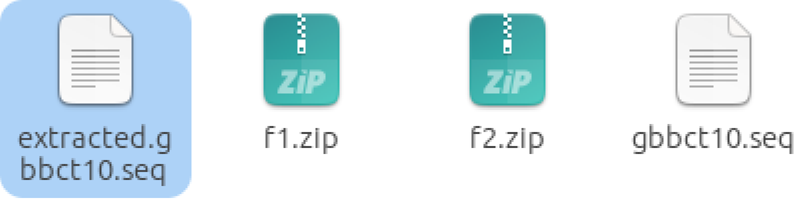
Explanation:

7zip ratio is better as it uses more than Huffman algorithm to compress files. This file is more compressible than the lecture’s file as the binary sequences differ in their frequencies what makes the compression noticeable.

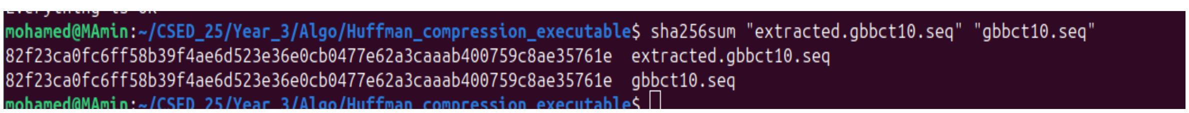
Decompression sample:



```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar d "20011502.3.gbbct10.seq.hc"
d 20011502.3.gbbct10.seq.hc 0
Number of bits written: 1485982366
Number of bytes per word: 3
Huffman Tree size : 84097
The DeCompression is done in : 34  second(s)
```



SHA256 TEST:



```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ sha256sum "extracted.gbbct10.seq" "gbbct10.seq"
82f23ca0fc6ff58b39f4ae6d523e36e0cb0477e62a3caaab400759c8ae35761e  extracted.gbbct10.seq
82f23ca0fc6ff58b39f4ae6d523e36e0cb0477e62a3caaab400759c8ae35761e  gbbct10.seq
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$
```

SHA2 original: 82f23ca0fc6ff58b39f4ae6d523e36e0cb0477e62a3caaab400759c8ae35761e

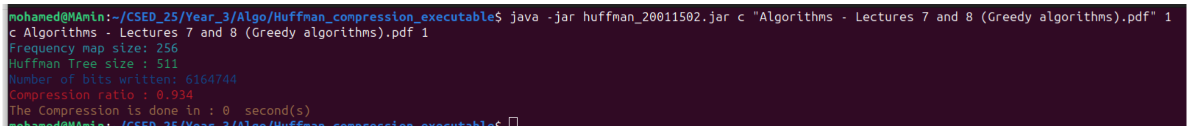
SHA2 extracted:

82f23ca0fc6ff58b39f4ae6d523e36e0cb0477e62a3caaab400759c8ae35761e

File2: Lecture’s file

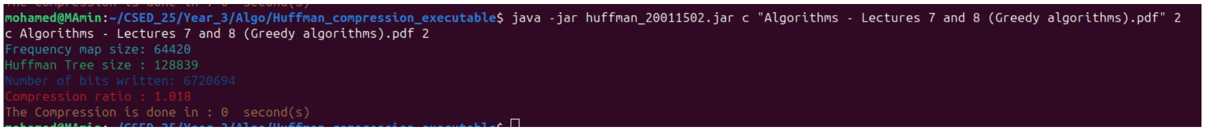
N	Compression time
1	< 1 second
2	< 1 second
3	< 1 second
4	< 1 second
5	< 1 second

N = 1:



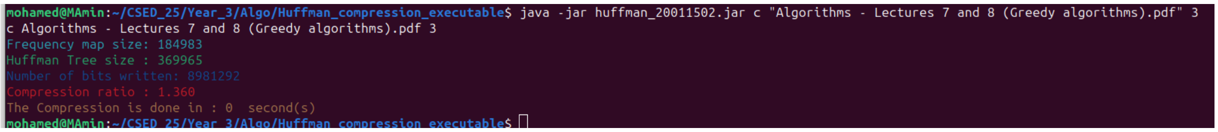
```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf" 1
c Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf 1
Frequency map size: 256
Huffman Tree size : 511
Number of bits written: 6164744
Compression ratio : 0.934
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

N = 2:



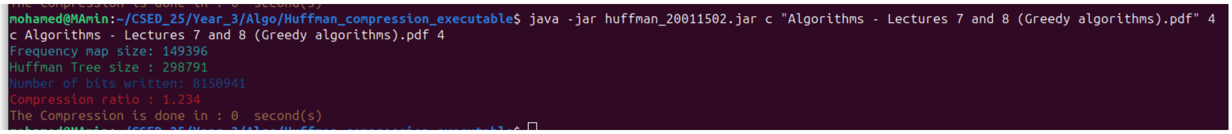
```
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf" 2
c Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf 2
Frequency map size: 64420
Huffman Tree size : 128839
Number of bits written: 6720694
Compression ratio : 1.018
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

N = 3:



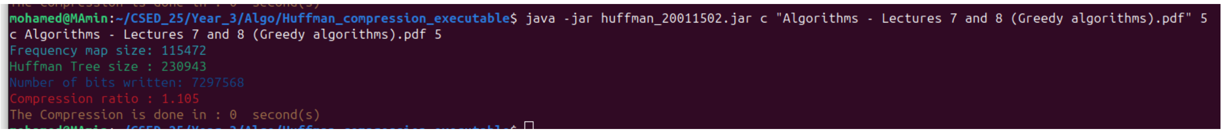
```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf" 3
c Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf 3
Frequency map size: 184983
Huffman Tree size : 369965
Number of bits written: 8981292
Compression ratio : 1.360
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

N = 4:



```
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf" 4
c Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf 4
Frequency map size: 149396
Huffman Tree size : 298791
Number of bits written: 8150941
Compression ratio : 1.234
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

N = 5:

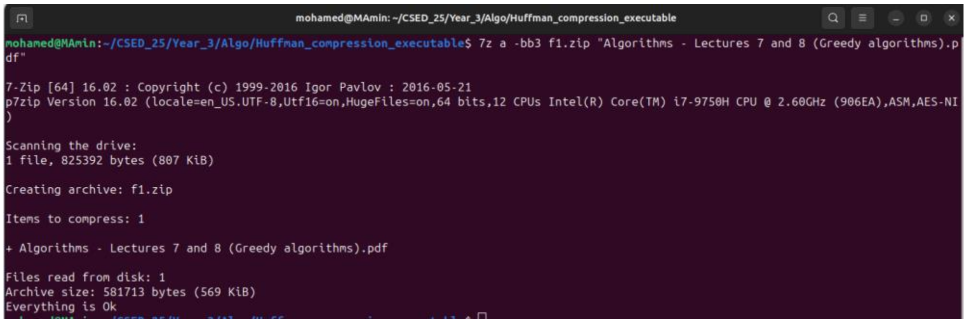


```
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar c "Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf" 5
c Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf 5
Frequency map size: 115472
Huffman Tree size : 230943
Number of bits written: 7297568
Compression ratio : 1.105
The Compression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

N	Compression ratio
1	0.934
2	1.018
3	1.360
4	1.234
5	1.105

Compression ratio = size after compression / size before compression

7zip ratio = 0.71



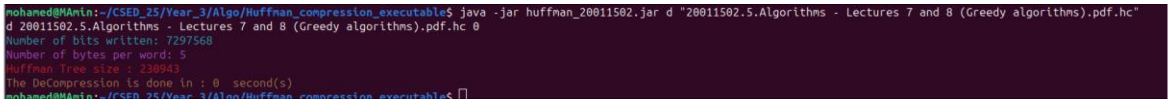
mohamed@MAmin: ~/CSED_25/Year_3/Algo/Huffman_compression_executable

```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 7z a -bb3 f1.zip "Algorithms - Lectures 7 and 8 (Greedy algorithms).p
df"

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,12 CPUs Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz (906EA),ASM,AES-NI
)

Scanning the drive:
1 file, 825392 bytes (807 KiB)

Creating archive: f1.zip

Items to compress: 1

+ Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf

Files read from disk: 1
Archive size: 581713 bytes (569 KiB)
Everything is Ok
```

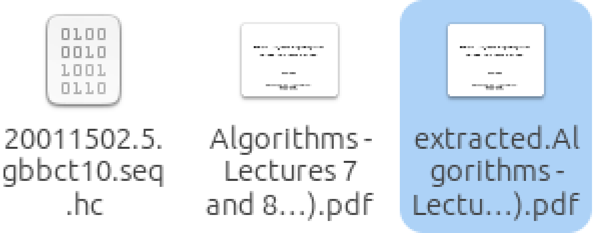
Explanation:

7zip ratio is better as it uses more than Huffman algorithm to compress files. Pdf files can have a frequent bit sequence that make the Huffman tree looks like a perfect tree, that implies that codes almost have the same length as the original sequence, so no improvement happens but instead the header data is added what makes the compressed file larger than the original file.

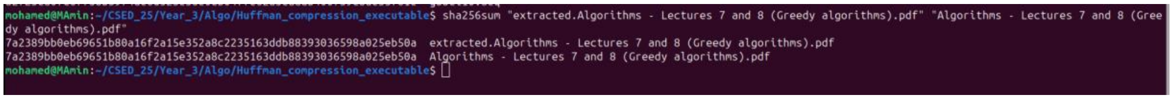
Decompression sample:



```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ java -jar huffman_20011502.jar d "20011502.5.Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf.hc"
d 20011502.5.Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf.hc 0
Number of bits written: 7297568
Number of bytes per word: 5
Huffman Tree size : 230943
The DeCompression is done in : 0  second(s)
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```


20011502.5.gbbct10.seq.hc
Algorithms - Lectures 7 and 8...).pdf
extracted.Algorithms - Lectu...).pdf

SHA256 TEST:



```
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ sha256sum "extracted.Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf" "Algorithms - Lectures 7 and 8 (Gree
dy algorithms).pdf"
7a2389bb0eb69651b80a16f2a15e352a8c2235163ddb88393036598a025eb50a  extracted.Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf
7a2389bb0eb69651b80a16f2a15e352a8c2235163ddb88393036598a025eb50a  Algorithms - Lectures 7 and 8 (Greedy algorithms).pdf
mohamed@MAmin:~/CSED_25/Year_3/Algo/Huffman_compression_executable$ 
```

SHA2 original:

7a2389bb0eb69651b80a16f2a15e352a8c2235163ddb88393036598a025eb50a

SHA2 compressed:

7a2389bb0eb69651b80a16f2a15e352a8c2235163ddb88393036598a025eb50a

NOTES:

In the previous samples I used relative paths despite the required path is absolute path as when parsing it will parse the path ordinary as an absolute path and will put the compressed file in the current directory.

But the recommended way is to enter the absolute path.