**Uniform distribution**

**Input file size : 2272 bytes**

**File name : input.txt**

**Table1.1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison Factors** | **Huffman encoding algorithm** | **Shannon fano algorithm** | **Lempel-ziv 77 algorithm** |
| Compression ratio | 0.25080 | 0.481954 | 0.577465 |
| Compression Factor | 3.985965 | 2.074886 | 1.731707 |
| Encoded File size | 570 bytes | 1095 bytes | 1312 bytes |
| Time complexity | 0.000667 | 0.001052 | 0.001119 |
| Length of code | 271 | 116 | 170 |
| Saving percentage | 74.911972 | 51.804577 | 42.253521 |

**Conclusion**

For a uniformly distributed data in a file the compression ratio for the huffman encoding algorithm is least, that is for a uniformly distributed file Huffman encoding algorithm is working best and saving percentage is also high with minimum time complexity also reduces the file size by 75 % (Approx) While Shannonfano reduces it to 52% (Approx) and LZ77 reduces it to 42% (Approx).

**Non Uniform(even) distribution**

**Input file size : 1384 bytes**

**File name : input3.txt**

**Table1.2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison Factors** | **Huffman encoding algorithm** | **Shannon fano algorithm** | **Lempel-ziv 77 algorithm** |
| Compression ratio | 0.251445 | 0.357659 | 0.531792 |
| Compression Factor | 3.977011 | 2.795960 | 1.880435 |
| Encoded File size | 348 bytes | 495 bytes | 736 bytes |
| Time complexity | 0.000526 | 0.000605 | 0.000904 |
| Length of code | 271 | 116 | 170 |
| Saving percentage | 74.855492 | 64.234100 | 46.820812 |

**Conclusion**

For a Non-uniform evenly distributed data in a file the compression ratio for the huffman encoding algorithm is least, that is for a Non-uniform evenly distributed file Huffman encoding algorithm is working best and saving percentage is also high with minimum time complexity also reduces the file size by 75% (Approx) While Shannonfano reduces it to 64% (Approx) and LZ77 reduces it to 46% (Approx).

**Non Uniform distribution**

**Input file size : 4840 bytes**

**File name : input1.txt**

**Table1.3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison Factors** | **Huffman encoding algorithm** | **Shannon fano algorithm** | **Lempel-ziv 77 algorithm** |
| Compression ratio | 0.249174 | 0.429339 | 0.119008 |
| Compression Factor | 4.013267 | 2.329163 | 8.402778 |
| Encoded File size | 1206 bytes | 2078 bytes | 576 bytes |
| Time complexity | 0.001453 | 0.001420 | 0.005355 |
| Length of code | 271 | 116 | 170 |
| Saving percentage | 75.082642 | 57.066113 | 88.099174 |

**Conclusion**

For a Non-uniformly distributed data in a file the compression ratio for the LZ77 algorithm is least, that is for a Non-uniformly distributed file LZ77 algorithm is working best and saving percentage is also high also reduces the file size by 88% (Approx) ,While Shannonfano reduces it to 57% (Approx) and Huffman encoding algorithm reduces it to 75% (Approx).

**Time complexity**

**Table1.4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File name** | **File size**  **(in bytes)** | **Huffman encoding algorithm** | **Shannon-fano algorithm** | **Lempel-ziv 77 algorithm** |
| input8.txt | 448 | 0.000608 | 0.000824 | 0.001272 |
| input10.txt | 1384 | 0.000636 | 0.001064 | 0.001644 |
| input1.txt | 1936 | 0.001065 | 0.001093 | 0.002505 |
| input2.txt | 5264 | 0.000292 | 0.000370 | 0.001395 |
| input4.txt | 6856 | 0.002414 | 0.002788 | 0.012018 |
| input6.txt | 10456 | 0.003693 | 0.003008 | 0.018217 |
| input3.txt | 11776 | 0.004316 | 0.003003 | 0.024441 |
| input5.txt | 19392 | 0.002497 | 0.001584 | 0.011361 |
| input7.txt | 29864 | 0.002680 | 0.006254 | 0.020151 |
| input9.txt | 68152 | 0.004373 | 0.002717 | 0.042222 |

**Conclusion**

* As we increase file size the Time complexity of Huffman Algorithm increases linearly.
* As we increase file size the Time complexity of Shannonfano Algorithm increases or decreases depending on type of data.
* As we increase file size the Time complexity of LZ77 Algorithm increases abruptly.

**Compression ratio**

**Table1.5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File name** | **File size**  **(in bytes)** | **Huffman encoding algorithm** | **Shannon fano algorithm** | **Lempel-Ziv 77 algorithm** |
| input8.txt | 448 | 0.254464 | 0.392857 | 1.589286 |
| input10.txt | 1384 | 0.251445 | 0.393064 | 1.208092 |
| input1.txt | 1936 | 0.251033 | 0.421488 | 0.252066 |
| input2.txt | 5264 | 0.242021 | 0.484992 | 0.278116 |
| input4.txt | 6856 | 0.248541 | 0.441219 | 0.072345 |
| input6.txt | 10456 | 0.249809 | 0.436400 | 0.005356 |
| input3.txt | 11776 | 0.247113 | 0.467221 | 0.151495 |
| input5.txt | 19392 | 0.250103 | 0.467512 | 0.030116 |
| input7.txt | 29864 | 0.249933 | 0.468624 | 0.013126 |
| input9.txt | 68152 | 0.249677 | 0.486031 | 0.000704 |

**Conclusion**

* As we increase file size the Compression ratio of Huffman encoding Algorithm remains almost Constant .Average Compression ratio of Huffman encoding algorithm is 0.249414.
* As we increase file size the Compression ratio of Shannonfano Algorithm remains almost Constant .Average Compression ratio of Shannonfano algorithm is 0.3991896
* As we increase file size the Compression ratio of LZ77 Algorithm decreases abruptly.

**Compression factor**

**Table1.6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File name** | **File size**  **(in bytes)** | **Huffman encoding algorithm** | **Shannon fano algorithm** | **Lempel-Ziv 77 algorithm** |
| input8.txt | 448 | 3.929825 | 2.545455 | 0.629214 |
| input1.txt | 1936 | 3.983539 | 2.372549 | 3.967213 |
| input10.txt | 1384 | 3.977011 | 2.544118 | 0.827751 |
| input2.txt | 5264 | 4.131866 | 2.061888 | 3.595628 |
| input4.txt | 6856 | 4.023474 | 2.266446 | 13.822580 |
| input6.txt | 10456 | 4.003063 | 2.291475 | 186.714279 |
| input3.txt | 11776 | 4.046735 | 2.140313 | 6.600897 |
| input5.txt | 19392 | 3.998351 | 2.138981 | 33.205479 |
| input7.txt | 29864 | 4.001072 | 2.133905 | 76.183670 |
| input9.txt | 68152 | 4.005172 | 2.057481 | 1419.83374 |

**Conclusion**

* As we increase file size the Compression factor of Huffman encoding Algorithm remains almost Constant .Average Compression factor of Huffman encoding algorithm is 4.0100108.
* As we increase file size the Compression factor of Shannonfano Algorithm remains almost Constant .Average Compression factor of Shannonfano algorithm is 2.2552611
* As we increase file size the Compression factor of LZ77 Algorithm increases abruptly.

**Encoded file size:**

**Table1.7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File name** | **File size**  **(in bytes)** | **Huffman encoding algorithm** | **Shannon fano algorithm** | **Lempel-ziv 77 algorithm** |
| input8.txt | 448 | 114 | 176 | 712 |
| input10.txt | 1384 | 348 | 544 | 1672 |
| input1.txt | 1936 | 486 | 816 | 488 |
| input2.txt | 5264 | 1274 | 2553 | 1464 |
| input4.txt | 6856 | 1704 | 3025 | 496 |
| input6.txt | 10456 | 2612 | 4563 | 56 |
| input3.txt | 11776 | 2910 | 5502 | 1784 |
| input5.txt | 19392 | 4050 | 9066 | 584 |
| input7.txt | 29864 | 7464 | 13995 | 392 |
| input9.txt | 68152 | 17016 | 33124 | 48 |

**Conclusion**

* For Huffman encoding algorithm the file size is always reduced by almost 74%-75%.
* For Shannonfano algorithm the file size is always reduced by around 55%-60%.
* As We Mentioned that Lempel-ziv 77 algorithm is Dictionary based Algorithm, the reduction of file size is highly dependent on data, if a word repeats several times in our Data File it reduces our file Size to 95%-98**%** ,otherwise for a very small file it sometimes even increases the file size.

**Saving percentage:**

**Table1.8**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File name** | **File size**  **(in bytes)** | **Huffman encoding algorithm** | **Shannon fano algorithm** | **Lempel-ziv 77 algorithm** |
| input8.txt | 448 | 74.553574 | 60.714287 | -58.928574 |
| input10.txt | 1384 | 74.855492 | 60.693638 | -20.89248 |
| input1.txt | 1936 | 74.896690 | 57.851238 | 74.793388 |
| input2.txt | 5264 | 75.797874 | 51.500763 | 72.188446 |
| input4.txt | 6856 | 75.145859 | 55.878059 | 92.765465 |
| input6.txt | 10456 | 75.019127 | 56.359982 | 99.464424 |
| input3.txt | 11776 | 75.288727 | 53.277855 | 84.850540 |
| input5.txt | 19392 | 74.989685 | 53.248764 | 96.988449 |
| input7.txt | 29864 | 75.006699 | 53.137558 | 98.687378 |
| input9.txt | 68152 | 75.032280 | 51.396877 | 99.929573 |

**Conclusion:**

* For Huffman encoding algorithm the saving percentage is almost 74%-75%.
* For Shannonfano algorithm the saving percentage is almost 55%-60%.
* As We Mentioned that Lempel-ziv 77 algorithm is Dictionary based Algorithm, saving percentage is highly dependent on data, if a word repeats several times in our Data File it reduces our file Size to 95%-98**%** ,for a very small file size Sometimes it goes negative.