Report

For measuring the success of this program, I chose a sample of six tests from the milestone input zip folder.

The files with the file sizes are as follows:

test1.txt (5 B) | test2.txt (38 B) | test3.txt (862 B) | test4.txt (4010 B) | test5.txt (4040 B) | test6.txt (1,921,525 B)

1) The average compression ratio; in addition to the average, compute the minimum, maximum, and standard deviation of the compression ratio.

	Compression %	
test1	0	
test2	14.28	
test3	43.66	
test4	22.7	
test5	15.2	
test6	12.5	
Avg Compression		18.05667
Max Compression		43.66
Min Compression		0
Standard Dev		14.53571

2) The time to encode each input for each type of sort, i.e., for Insertion and for Merge Sort. Plot the run time as a function of input size.

1		Insertion Sort Encoding Runtime	Merge Sort Encoding Runtime
2	test1	0.000205	0.000195
3	test2	0.01163	0.000568
4	test3	0.0018	0.012437
5	test4	0.00688	0.096332
6	test5	0.009356	0.103491
7	test6	3.618815	39.340331

3) The time to decode each encoded input. Plot the run time as a function of input size.

	Decode Time	Input size
test1	0.000263	5
test2	0.000238	38
test3	0.002519	862
test4	0.012175	4010
test5	0.023287	4040
test6	6.211844	1921525

4) The compression ratio as a function of number of lines encoded. The encoding algorithm has been described as encoding one line at a time. If you instead encode 2, 3, . . . lines at a time, does the compression ratio improve? What do you expect to happen?

	Compression %	Input size (B)
test1	0	5
test2	14.28	38
test3	43.66	862
test4	22.7	4010
test5	15.2	4040
test6	12.5	1921525

It depends on the text, but I think more chances of cluster formation will be available with encoding more lines at a time, and so compression ration should in theory improve.

Can you draw any general conclusions about the compression ratio, about the impact of the sorting algorithm on the run time of the encoding scheme, about the impact of input size on the decoding scheme, about the impact on the compression ratio as a function of the number of lines encoded?

It seems that the insertion sort was running better than the merge sort, I don't know why, maybe because the input is smaller comparatively. The compression ratio goes down with the increasing size of the input, generally. The runtime is proportional

to the size of the input so is the decoding algorithm runtime, pretty obvious.

I could not plot them for some reason (probably last minute hurry, I will take care next time).

Thank you!