CSE 310

Full Project 1

Raghav Aggarwal

Experimentation and Report:

Experiment Runs

- The experiments for both encoding and decoding ran well on all the 3(small, medium and large) kind of sample input given.
- Encoding decoding large files sometimes took longer than expected but they were correct.
- Decoded files were exactly same as original files just didn’t had the blank lines.
- The encoding algorithm works great at all times.
- Decoding algorithm gives some null values here and there.

Average values of run time

Small Input		
	Encoding	Decoding
Insertion Sort	0.000321974	0.000316253
Quick Sort	0.000376824	0.000362973
Medium Size Input		
Insertion Sort	0.328625	0.319458
Quick Sort	0.0724413	0.063976
Large Size input		
Insertion Sort	0.78492	0.754592
Quick Sort	0.20418	0.193861

Standard deviation insertion sort = **0.3217283302396**

Standard deviation quick sort = **0.084382797303276**

Minimum values of run time

Small Input		
	Encoding	Decoding
Insertion Sort	0.000290854	0.000285309
Quick Sort	0.000347829	0.000338042
Medium Size Input		
Insertion Sort	0.3179432	0.303390
Quick Sort	0.0717824	0.062865
Large Size input		
Insertion Sort	0.77591	0.747426
Quick Sort	0.19527	0.185409

Maximum values of run time

Small Input		
	Encoding	Decoding
Insertion Sort	0.000336247	0.000330493
Quick Sort	0.000387282	0.000373941
Medium Size Input		
Insertion Sort	0.3362901	0.320873
Quick Sort	0.0749462	0.064928
Large Size input		
Insertion Sort	0.79648	0.769632
Quick Sort	0.21472	0.209437

Final Result - >

Standard deviation insertion sort Run time = **0.29902706309289**

Standard deviation quick sort Run Time= **0.084705084255142**

Conclusion –

Insertion sort is faster for **smaller inputs** than insertion sort

But as the input size increases the time taken by insertion sort increase significantly fast, whereas it doesn’t increase as much for Quick Sort.

In our case it the difference between time taken by **insertion sort exceeded time taken by quick sort by 112%.**

The compression ratio as a function of number of lines encoded

Quick Sort -> $f(n) = 0.0000873477(n-1)^2 + 0.000203352$

Insertion sort -> $f(n) = 0.00066522958(2n-1) +0.000313162$

Where n is number of lines.

Interpretation of the result

- If you want to do sorting of small text which roughly is **<400 lines** use **insertion sort**. It will be faster and more stable.

- If you are sorting 300-500 lines than you can use any of them will roughly take same time.

- If you want to do sorting of **>=500** line use **quick sort** because it reduces time significantly in comparison to insertion sort as the input increases.

- The only advantage of **insertion sort is that it is more stable than quicksort.**
- **Insertion sort is faster for smaller inputs** than quick sort, but **Quick sort is significantly faster for larger inputs.**