|  |  |  |  |
| --- | --- | --- | --- |
| Array Size (in thousands) | Bubble Sort time (s) | Insertion sort time (s) | Selection sort time (s) |
| 10 | 0.348951 | 0.109023 | 0.188616 |
| 25 | 2.10859 | 0.581818 | 0.950661 |
| 50 | 8.39106 | 2.09179 | 3.65253 |
| 75 | 18.6292 | 4.66499 | 8.09621 |
| 100 | 33.2474 | 8.14649 | 14.4736 |
| 125 | 51.9961 | 12.4752 | 22.4505 |
| 150 | 75.2707 | 17.9014 | 32.3819 |
| 175 |  | 24.5653 | 44.1964 |
| 200 |  | 31.2829 | 57.032 |
| 300 |  | 70.6035 | 128.229 |

1a and 1b is in this table

Chart, line chart

Description automatically generated

This line chart is from LibreOffice Calc as done in step c

1d) From the time analysis, we can see that the insertion sort is consistently better and the bubble sort is the worst of the three methods as array sizes increase to very large numbers. The lines do not cross since their exe times differ so much in their runs. Bubble sort seems to be four times slower than insertion sort and roughly two times slower than the selection sort. Insertion sort and selection sort are both pretty similar in their run of array size 10,000.

1e) EXAMPLE: g(x) = 0.6428571428571x^(2) + 0.6428571428571x + 0.8

Chart

Description automatically generated

Equation and fit line for bubble sort:

h(x)=0.0033803769201 x^(2)-0.0068004993026 x+0.1476133007232

Chart, line chart

Description automatically generated

Equation and line for insertion sort:

g(x)=0.0007720449317 x^(2)+0.0035193617891 x+0.0167006105933

Chart, line chart

Description automatically generated

Equation and line for Selection sort:

g(x)=0.0014116239559 x^(2)+0.0041150659886 x-0.0497185642815

Chart, line chart

Description automatically generated

2a and 2b)

|  |  |  |
| --- | --- | --- |
| Array Size (in millions) | Quick Sort (s) | Merge Sort (s) |
| .1 | 0.0247984 | 0.0301854 |
| .2 | 0.0472827 | 0.0549248 |
| .5 | 0.105937 | 0.13626 |
| 1 | 0.191394 | 0.264135 |
| 2 | 0.370604 | 0.546871 |
| 3 | 0.584291 | 0.790418 |
| 10 | 1.9604 | 2.74665 |
| 20 | 4.07862 | 5.65919 |
| 30 | 6.40181 | 8.43686 |
| 50 | 10.6707 | 14.7812 |
| 100 | 21.9419 | 32.1712 |
| 150 | 34.0513 | 51.5883 |
| 200 | 44.6781 | 64.4489 |
| 250 | 56.9538 |  |
| 300 | 68.605 |  |

2c) XY scatter made in Excel

2d) From the time analysis, the quick sort is shown to be consistently faster and better running than the merge sort at larger array sizes (The horizontal axis is array size in millions and the vertical axis is the time in seconds). The lines do not cross, showing that the quick sort is over all better than merge sort. However, they start of VERY similar in the beginning and prove to both be efficient in smaller array sizes, even more so than the three types of sorts we have observed earlier in this experiment. There is no consistency in how much faster quick sorts are than merge sorts. As array size grows, the gap between speeds grows bigger, it is never consistently double the speed, 1.5 the speed, etc. The speed difference between the two sorts grows bigger the more the array size grows.

2e)

Equation and Line for Quick Sort:

g(x)=0.0281320504444 x log(2,x)+1.1595017878484

Chart, line chart

Description automatically generatedEquation and Line for Merge Sort:

h(x)=0.0437162004833 x log(2,x)+1.0872936672917

Chart, line chart

Description automatically generated