Alexander Chatron-Michaud

260611509

5a:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RAW DATA POINTS** | | | | | | |
| Iterations | # of Students | List generation (ms) | Nested loops (ms) | Binary search (ms) | Sorting & Parallel pointers (ms) | Merge and sort (ms) |
| 1024 | 1000 | 0.068 | 0.274 | 0.223 | 0.272 | 0.283 |
| 512 | 2000 | 0.126 | 0.907 | 0.422 | 0.510 | 0.544 |
| 256 | 4000 | 0.236 | 3.333 | 0.917 | 1.119 | 1.124 |
| 128 | 8000 | 0.470 | 12.747 | 1.851 | 2.222 | 2.416 |
| 64 | 16000 | 0.950 | 50.014 | 3.981 | 4.778 | 4.821 |
| 32 | 32000 | 1.929 | 197.993 | 8.299 | 9.755 | 10.118 |
| 16 | 64000 | 3.903 | 775.546 | 17.673 | 20.608 | 21.853 |
| 8 | 128000 | 8.120 | 3123.356 | 36.797 | 42.237 | 43.774 |
| 4 | 256000 | 18.367 | 12352.594 | 81.887 | 92.220 | 91.591 |
| 2 | 512000 | 38.835 | 50286.640 | 177.829 | 190.484 | 199.402 |
| 1 | 1024000 | 81.386 | 198959.610 | 370.477 | 402.082 | 398.836 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DATA AFTER SUBTRACTING LIST GENERATION TIME** | | | | |
| # of Students | Nested loops (ms) | Binary search (ms) | Sorting & Parallel pointers (ms) | Merge and sort (ms) |
| 1000 | 0.21 | 0.15 | 0.20 | 0.21 |
| 2000 | 0.78 | 0.30 | 0.38 | 0.42 |
| 4000 | 3.10 | 0.68 | 0.88 | 0.89 |
| 8000 | 12.28 | 1.38 | 1.75 | 1.95 |
| 16000 | 49.06 | 3.03 | 3.83 | 3.87 |
| 32000 | 196.06 | 6.37 | 7.83 | 8.19 |
| 64000 | 771.64 | 13.77 | 16.70 | 17.95 |
| 128000 | 3,115 | 28.68 | 34.12 | 35.65 |
| 256000 | 12,334 | 63.52 | 73.85 | 73.22 |
| 512000 | 50,248 | 138.99 | 151.65 | 160.57 |
| 1024000 | 198,878 | 289.09 | 320.70 | 317.45 |

5b:

The nested loops, based on the data given, has a function as follows:

T(n) = 0.0000001884n2 + 0.6160 milliseconds   
*(exponential fit provided by LoggerPro with RMSE of 1.264)*

The binary search, based on the data given, has a function as follows:

T(n) = 4.700\*10-5(nlog(n))

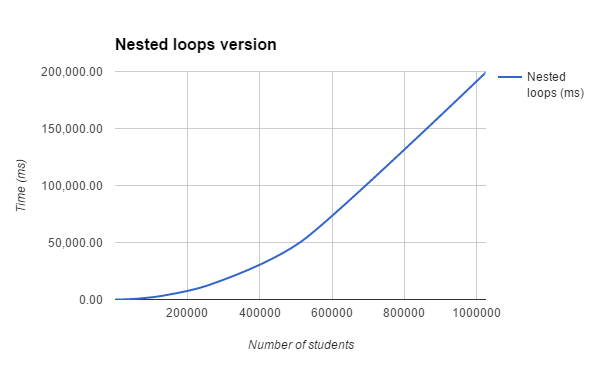
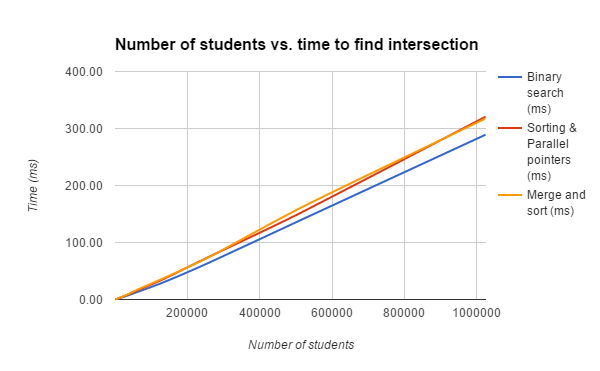
The sorting and parallel pointers, based on the data given, has a function as follows:

T(n) = 5.212\*10-5(nlog(n))

The merge and sort, based on the data given, has a function as follows:

T(n) = 5.226\*10-5(nlog(n))

The given functions were found using Logger Pro. Constants were not added as the graph intersected the y axis at a negative number, which would be more incorrect than not including an added constant.



5c:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L1 | L2 | List Generation Time (ms) | Nested loops (ms) | Binary search (ms) | Sorting & Parallel pointers (ms) | Merge and sort (ms) |
| 32000 | 1024000 | 43.007 | 13,319.053 | 164.942 | 165.011 | 164.661 |
| 1024000 | 32000 | 42.119 | 12,998.070 | 32.878 | 166.555 | 169.374 |

For nested loops, the time doesn’t change much as it must do n\*m comparisons regardless of which list is nested within another loop. For both tries, 32000\*1024000 comparisons must be made and hence it takes the same time.

For binary search, the time taken is far less if the array that has to be sorted is smaller (when L2 is smaller). This is because not only does it take more time to sort a larger array, it takes more time to search through a large array, which means that the searches and sorting time is much faster when L2 (the array being sorted) is smaller than L1.

For sorting and parallel pointers, the time taken is the same as the pointers are moving down both sorted lists. This is a symmetrical process irrespective of which list is larger, as it is going down the lists symmetrically, and hence the time is the same in both cases shown above.

For merge and sort, the exact same operation is performed and hence the time taken is the same irrespective of which list is larger, as in both cases illustrated above, the lists are merged to form a combined list of 1056000 student IDs which is then sorted. The merged list is the same in both cases and the operation done is hence the same.