Ellie Fitzpatrick

Michael Rabinovich

Data Structures

21 April, 2017

Statistical Analysis of Sorting Algorithms

1The log of the mean of time to sort a sorted, reversed, and randomly organized array with heap sort, quick sort, and merge sort.

2 Chart depicting the log of the times for heap quick, and merge sort for a sorted, reversed, and random array. Data collected from one run of the program.

In observing the data, one can immediately see that quicksort is the quickest of all the algorithms for sorted, reversed, and random arrays of both the shortest and largest lengths, 1,000 and 1 mil respectively. This is due to its divide and conquer nature. It is also clear that for every list sorted except for the random quicksort, the order of elements does not matter as arrays get larger. With small arrays, it does matter what order they are in. Quicksort and heapsort both prefer reversed arrays and merge sort prefers random arrays. Another trend that sticks out when looking through the data is how both heap and quicksort have a large variance, but merge sort does not. This is because merge sort algorithmically does the same thing no matter how the data is oriented. It always partitions to size two, then works its way back to the size of the array by combining the partitions. The last interesting data point is that the length of 1000 heap and quick sort both have approximately the same value versus quick sort being less. This must be because quick sort does not have the advantage because of the above reason. This analysis shows that the best method to use is quicksort rather than heap or merge sort in all situations tested.

Raw Data

\*\*\*\* Heap Sort \*\*\*\*

Time in nanoseconds

[1000, 10000, 100000, 1000000]

Sorted: Median Time, Mean Time, Variance Time

[361739, 888579, 8087188, 95031189], [1643285, 930318, 8081623, 93956019], [5333508938212, 5643596711, 708548612779, 74021536381355]

Reverse: Median Time, Mean Time, Variance Time

[77913, 923362, 8118725, 88115478], [81623, 922743, 9053063, 86095149], [46667742, 361837963, 5695219123884, 29573346610063]

Random: Mean Times, Variance Time

[77170, 1030214, 13944208, 141633855], [144113652, 27860174888, 7315412520514, 496858158003948]

\*\*\*\* Quick Sort \*\*\*\*

Time in nanoseconds

[1000, 10000, 100000, 1000000]

Sorted: Median Time, Mean Time, Variance Time

[95536, 140522, 1690435, 18865623], [142222, 161546, 1706048, 18918956], [12234751182, 1385417728, 3679189033, 8909695443]

Reverse: Median Time, Mean Time, Variance Time

[13449, 146087, 1659362, 18927304], [13604, 154744, 1659362, 18941372], [501739, 400328787, 3441025, 32111620808]

Random: Mean Times, Variance Time

[64324, 652985, 6171454, 62210179], [23804237, 572262065, 21079099633, 4333422170782]

\*\*\*\* Merge Sort \*\*\*\*

Time in milliseconds

[1000, 10000, 100000, 1000000]

Sorted: Median Time, Mean Time, Variance Time

[2532, 987, 5702, 362610], [6409, 973, 6029, 366496], [65465029, 62890, 364819, 6075298202]

Reverse: Median Time, Mean Time, Variance Time

[1655, 1001, 4247, 302814], [8857, 1047, 4617, 300468], [173322439, 15090, 1275564, 24019387]

Random: Mean Times, Variance Time

[3122, 527, 4663, 316603], [42061200, 79454, 976113, 2026484585]