

Sorting Algorithms: Comparing Between Insertion, Merge, Heap and Quick Sorting Algorithms and Making a Hybrid Sorting

Hakan Aydin

December 21, 2022

1. Introduction

In this paper I am going to implement insertion sort, quick sort, merge sort and heap sort. After creating these algorithms, I am going to compare them. Once I compare the four algorithms, I am to create a hybrid sorting algorithm based on our findings and compare the hybrid sort with the other sorting algorithms.

2. Insertion Sort:

Is a simple, in-place sorting algorithm that has a time complexity of $O(n^2)$ in the worst case. This means that the time taken to sort the array increases quadratically with the size of the array. Insertion Sort is a stable sort, which means that it preserves the order of equal elements. It is not a very efficient algorithm for large arrays, but it can be useful for small arrays or for sorting partially sorted arrays.

3. Merge Sort:

Is a divide and conquer sorting algorithm that has a time complexity of $O(n \log n)$ in the worst case. This means that the time taken to sort the array increases logarithmically with the size of the array. Merge Sort is a stable sort, which means that it preserves the order of equal elements. It is generally more efficient than Insertion Sort, especially for larger arrays. However, it requires additional space to store the subarrays that are being merged, which can be a disadvantage in certain situations.

4. Heap Sort:

Is a comparison-based sorting algorithm that has a time complexity of $O(n \log n)$ in the worst case. This means that the time taken to sort the array increases logarithmically with the size of the array. Heap Sort is not a stable sort, which means that it does not preserve the order of equal elements. It is generally more efficient than Insertion Sort, especially for larger arrays. However, it requires additional space to store the heap, which can be a disadvantage in certain situations.

5. Quick Sort

Is a divide and conquer sorting algorithm that has a time complexity of $O(n^2)$ in the worst case, but an average time complexity of $O(n \log n)$. This means that the time taken to sort the array increases logarithmically with the size of the array on average, but there is a possibility of worst-case performance that is quadratic. Quick Sort is not a stable sort, which means that it does not preserve the order of equal elements. It is generally more efficient than Insertion Sort, especially for larger arrays. However, it requires additional space to store recursive function calls, which can be a disadvantage in certain situations.

6. Methodology

This is a program that compares the performance of four sorting algorithms: merge sort, quick sort, insertion sort, and heap sort. It does this by generating random arrays of different sizes and measuring the time it takes for each algorithm to sort those arrays. The program then uses this data to determine which algorithm is the fastest for a given array size, and stores this information in the 'fastestAlgorithms' vector. Finally, the program defines a function 'hybrid_sort' that uses this information to choose the fastest algorithm for a given array, based on its size.

The program starts by measuring the performance of the sorting algorithms on small arrays (sizes 1 to 100). It does this by generating 'iterations' (100) random arrays of a given size and measuring the time it takes for each algorithm to sort each array. It then calculates the average time for each algorithm by dividing the total time by 'iterations'. This process is repeated for each array size in the range.

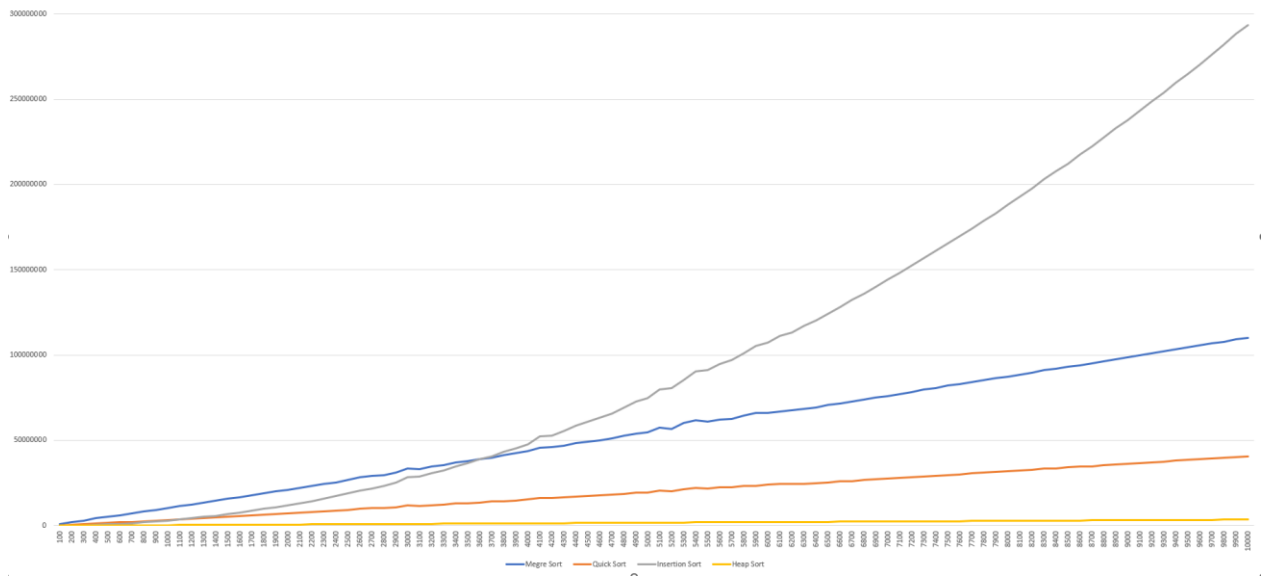
Next, the program measures the performance of the sorting algorithms on large arrays (sizes 100 to 10000, in steps of 100). It does this in the same way as for small arrays, but only measures the performance for a few selected sizes.

Finally, the program determines which algorithm is the fastest for each array size by finding the algorithm with the lowest average time for that size. It stores this information in the 'fastestAlgorithms' vector. The 'hybrid_sort' function can then use this information to choose the fastest algorithm for a given array.

7. Results

Based on my findings when I compare these 4 algorithms Heap Sorting is the fastest and when the larger arrays are the case Insertion Sort is the slowest. When I include Hybrid Sort to the comparison that I created comparison shows that it is slower than Heap Sorting, Merge Sorting and Quick Sorting but faster than Insertion Sorting. (Unfortunately, Hybrid Sorting is not on the graph)

Chart Area



```

1600 16714613 5664686 7654931 511927
1700 17932728 6089430 8624049 532543
1800 18944766 6448902 9745912 575085
1900 20148442 6819471 10893942 601267
2000 20982926 7220949 11863544 645346
2100 22183800 7546656 13262762 670532
2200 23283854 8002297 14390756 718263
2300 24573990 8520516 15866672 755625
2400 25400459 8881699 17258601 773295
2500 26906129 9183432 18834466 822775
2600 28455021 9789006 20622145 875075
2700 29099028 10401430 21899572 900269
2800 29765410 10334965 23430854 923845
2900 31107565 10837898 25157145 954245
3000 33445995 11882472 28223747 1030227
3100 33130427 11509434 28689987 1025755
3200 34602759 11958906 30641458 1062879
3300 35293629 12339771 32377423 1121132
3400 36990806 12920937 34574767 1135660
3500 37660494 13168196 36619352 1164804
3600 38982769 13592506 39037132 1229525
3700 39974029 14150693 40726965 1228462
3800 41173759 14321564 43168482 1288205
3900 42394995 14838676 45207983 1330618
4000 43606086 15398304 47830820 1344008
4100 45840383 16265199 52153520 1402040
4200 45876352 16301837 52929484 1435180
4300 46873036 16502849 55548434 1489308
4400 48241429 17157821 58633253 1514912
4500 49269625 17419164 60844875 1525180
4600 50021648 17933181 63426595 1572359
4700 51243250 18037086 65712768 1619668
4800 52774626 18097645 69256702 1726802
4900 53932446 19351103 72790430 1668000
5000 54620329 19413139 74835109 1712117
5100 57519559 20696198 79660433 1791485
5200 56861567 20301763 80644702 1760266
5300 60353229 21490848 85217053 1811928
5400 61661001 22160752 90305013 1896554
5500 61178021 21772624 91092534 1924698
5600 61007880 22369782 94904046 1981591
5700 62582337 22509074 96994544 1997528
5800 64532624 23297191 100802413 1997667
5900 66017828 23446430 105391232 2076016
6000 66193673 24157097 107436668 2083991
6100 66890045 24277576 111440390 2110425
6200 67627841 24655897 113324470 2140170
6300 68512480 24622144 116968424 2185003
6400 69392920 25040324 120343566 2207463
6500 70627679 25407783 124193636 2251594
6600 71761664 25961391 128128094 2285328
6700 72806970 26179472 132296031 2328185
6800 73960670 26855721 135811078 2353963
6900 75037046 27380223 139965663 2394391
7000 76097893 27683913 144092359 2436518
7100 77274668 27912900 148235472 2481672
7200 78421868 28591450 152348404 2524700
7300 79657221 28858602 156642520 2542571
7400 80690797 29319126 160930502 2595698
7500 82008917 29517656 165477060 2623035
7600 82888760 30129492 169692929 2658986
7700 84098268 30601935 174186370 2688752
7800 85196820 31210450 178869668 2732938
7900 86537404 31640289 183290193 2771808
8000 87281695 31811538 188067734 2798122
8100 88641866 32335243 192736473 2843891
8200 89734922 32740450 197481546 2883541
8300 91245035 33481467 203071979 2945545
8400 91974477 33663668 207922504 2958407
8500 93237712 34228612 212229353 2995762
8600 94126505 34632254 217596564 3036598
8700 95332095 34849287 222343325 3069766
8800 96415331 35324411 227386812 3100598
8900 97667915 35885120 232780370 3155239
9000 98816651 36378014 237528872 3189447
9100 99951604 36680676 243070694 3233567
9200 101046729 37160508 248649508 3255287
9300 102193171 37554266 253967735 3316568
9400 103406347 38169903 259499275 3337541
9500 104534991 38549318 264739128 3370240
9600 105634209 38894057 270309641 3422789
9700 106784893 39200903 276103559 3455934
9800 107815635 39896206 282236233 3495571
9900 109241960 40308102 288456535 3527140
10000 110188539 40580282 293615404 3560271
Hybrid sorting algorithm time: 292615400 ns

```

8. Conclusions

Heap sorting algorithm is best all-around and then second fastest is Quick sort algorithm. Using the Hybrid sorting algorithm its closer to Insertion but not even close to heap sorting.

So I conclude that Heap sorting, Quick sorting, Hybrid sorting, Merge sorting and Insertion sorting algorithm is the order of fastest to slowest.