|  |  |  |
| --- | --- | --- |
| Chart of Predictions – Adam Ryder | | |
| pair or group comparison | prediction | actual results |
| bubble, plain vs.  bubble w/ early stop | Early stop will be faster | Bubble plain was faster. |
| This was odd to me. I thought early stop would be faster. It appears that early stop was faster for 1,000 or less and then plain bubble is faster. I think this is due to all the checks to see if there were any swaps. These checks add up the bigger the data set gets. | | |
| all n2 vs.  all n log n | All the nLogn algorithms will be much faster | nLogn algorithms were much faster |
| This is obvious. | | |
| selection vs.  insertion | Insertion will be faster | Insertion with shifts was faster, but swaps was slower than selection. |
| The way that insertion sort finds the spot where it needs to be inserted is much faster than what selection does. Selection just goes through the array finding the minimum every time. | | |
| Quicksort with few duplicates vs.  quicksort with many duplicates | Quicksort should get better the more spread out the numbers are. | Confirmed prediction. |
| The fewer the duplicates, the less sorted a dataset is likely to be and and quicksort does not do well with sorted data. | | |
| All quicksorts types: few duplicates vs many duplicates | Quicksort with median should be better the more spread out the numbers are. | Confirmed prediction. |
| Quicksort with a median is better the larger your range of numbers are because it is more likely to find the middle of the array and therefore the partitions will be larger. | | |
| Which quicksort is best? | Regular quicksort | Quicksort with Median is best with a large range of numbers. Quicksort with insertion is best for sets with many duplicates. |
| Quicksort with median is better for datasets with a larger range of numbers for the above reason. Quicksort with insertion was surprisingly good and right behind quicksort with median for any size data set. This is true because instead of partitioning on smaller sets of 20 or less, it uses insertion sort which is very efficient on smaller data sets. | | |
| Which insertion is best | Insertion with binary search. | Insertion with binary is marginally better for any range of numbers. |
| Insertion with shifts and binary are very close with binary being just slightly better. I think the reason for this is because I implemented my insertion with binary to also do a shift when it finds the insert spot. So it’s using shifts plus the added quickness of binary search to find the insertion spot. | | |
| The best nLogn algorithm for large datasets (1mil+) | Merge sort | Quicksort. |
| I thought merge sort would be faster for large sets because the recursive splitting would be have more impact on time, but all the quick sorts are more efficient, but not by much. | | |
| The best nLogn algorithm for small datasets (<1mil) | Quicksort | Confirmed prediction. |
| Quicksort with insertion seems to be the best for these smaller data sets. | | |
| Best n^2 algorithm on small datasets (<10,000) | Insertion with shifts | Confirmed prediction |
| Insertion with shifts is great because shifts have less overhead than swaps. Insertion with binary is good too for very large range of numbers. Bubble sort and selection of are not going to be good because of they are more brute force. | | |
| Best n^2 algorithm on large datasets (10,000+) | Insertion with binary | Confirmed prediction |
| Insertion with binary just beats out shifts on larger sets because larger sets tend to have a larger range of numbers which is good for binary searching. Shifting tends to get less efficient the bigger N is. | | |
| Best n^2 algorithm for few duplicates | Insertion with binary for large sets, and insertion with shifts for smaller sets | Confirmed prediction |
| Insertion with binary is great for this because it finds the insertion index much faster on larger range of numbers. Shifting is better on smaller sets. | | |
| Best n^2 algorithm for many duplicates | Insertion with binary for large sets, insertion with shifts for small sets | Confirmed prediction. |
| The improved insertion sorts are all close on this one. | | |
| Best algorithm for sorted arrays | Stop early bubble | Confirmed prediction. |
| If it’s already sorted, bubble sort with stop early will only be O(n). | | |
| Worst algorithm for sorted arrays. | Quick sort | Confirmed prediction. |
| Quick sort is horrible for sorted arrays because it has to go through every element and see if it’s larger or smaller than the pivot and there is no check to see if they’re already sorted. | | |