Nout Reusken

Professor Dunne

CSE 13S

15 November 2020

Assignment 4 Write Up

In this assignment, we implemented 4 different sorting algorithms: Bubble Sort, Shell Sort, Quick, and Binary Insertion Sort. In this assignment we found out the difference between those sorting algorithms, and their time complexities. In the data we found from the counter that calculate the moves and comparisons, we can see how complex certain sorting algorithms are and how they speed up the process of sorting a given array.

Each sortin algorithm has different time complexities, starting with the most basic algorithm, Bulle sort. Bubble sort has a time complexity of O(n^2) at its wost O(nlog n) (Wikipedia). Which isn't the worst and least complex of all algorithms. Shell Sort has a time complexity of O(nlog^2n) best known but usually it is O(nlogn)(Wikipedia), which shows more complexity and better than Bubble sort making it often faster to solve depending on the gap. The quick sort algorithm has a time complexity of O(n^2) at its worst or O(nlogn) at its best, making it one of the quickest ways to sort an array. The binary insertion sort algorithm has a time complexity O(logn) making it less complex than all of the other sorting algorithms.

From the assignment, I learned how time complexity affected the amount of compares, the lower the time complexity the less the sorting algorithm has to compare numbers within the array. I did find out that quicksort has the shortest way of sorting any given array. I also learned that even though Bubble seems to be most straight forward, it yet takes the longest to sort a given array.

How I experimented with the sort is by using different seeds to have different random lists and see how it affected the amount of compares and moves, often still showing Binary sort uses the least compares and quick sorting using the least amount of moves. The data also shows that in general bubble sort is the worst to implement as a sort yet it is the most basic to implement. On the page below there is a picture of one example used for data when looking at the different sorting algorithms and shows how each sorting algorithm compares to the other.

In general the assignment has taught me the differences in complexities of the four most used sorting algorithms in coding. Also really how the implementation of each sorting algorithm works and how the fundamentals of each algorithm works and sorts the lists.

```
Bubble Sort
100 elements, 7077 moves, 4950 compares
    11928963    12591398    20021796    21128680    21961325    31822619    34412749
    34886069    49021110    57509491    59851133    68574097    75257048    77144257
    86585853   101301346   114389756   114632963   118852153   127224362   129666022
   134022820   138123204   141594986   143120736   155493230   161637111   170409441
   171807697   174240889   191344001   191929321   198838075   212472681   218109776
   229706589   270359703   292850682   336205321   342217139   345616751   350595885
   351969720   389883213   449398437   461085871   462792178   466359476   501405920
   510106982   513334883   521241737   526542695   557107568   561287884   567873507
   575563805   587056094   588721288   607827590   617793922   619973746   620258073
   620543907   621942163   623607358   627844041   640600366   653191764   654929869
   677513702   684315706   697535498   718744680   725644732   732145788   780416126
   781032961   790225475   792153223   835312041   837534179   838653684   877426267
   897448063   914913962   928880504   942053238   945718706   961852377   965904307
   967680031   993505642  1009063125  1020521027  1020975754  1030583005  1036254129
  1045618677  1065040762
Shell Sort
100 elements, 1101 moves, 6921 compares
    11928963    12591398    20021796    21128680    21961325    31822619    34412749
    34886069    49021110    57509491    59851133    68574097    75257048    77144257
    86585853   101301346   114389756   114632963   118852153   127224362   129666022
   134022820   138123204   141594986   143120736   155493230   161637111   170409441
   171807697   174240889   191344001   191929321   198838075   212472681   218109776
   229706589   270359703   292850682   336205321   342217139   345616751   350595885
   351969720   389883213   449398437   461085871   462792178   466359476   501405920
   510106982   513334883   521241737   526542695   557107568   561287884   567873507
   575563805   587056094   588721288   607827590   617793922   619973746   620258073
   620543907   621942163   623607358   627844041   640600366   653191764   654929869
   677513702   684315706   697535498   718744680   725644732   732145788   780416126
   781032961   790225475   792153223   835312041   837534179   838653684   877426267
   897448063   914913962   928880504   942053238   945718706   961852377   965904307
   967680031   993505642  1009063125  1020521027  1020975754  1030583005  1036254129
  1045618677  1065040762
Quick Sort
100 elements, 465 moves, 902 compares
    11928963    12591398    20021796    21128680    21961325    31822619    34412749
    34886069    49021110    57509491    59851133    68574097    75257048    77144257
    86585853   101301346   114389756   114632963   118852153   127224362   129666022
   134022820   138123204   141594986   143120736   155493230   161637111   170409441
   171807697   174240889   191344001   191929321   198838075   212472681   218109776
   229706589   270359703   292850682   336205321   342217139   345616751   350595885
   351969720   389883213   449398437   461085871   462792178   466359476   501405920
   510106982   513334883   521241737   526542695   557107568   561287884   567873507
   575563805   587056094   588721288   607827590   617793922   619973746   620258073
   620543907   621942163   623607358   627844041   640600366   653191764   654929869
   677513702   684315706   697535498   718744680   725644732   732145788   780416126
   781032961   790225475   792153223   835312041   837534179   838653684   877426267
   897448063   914913962   928880504   942053238   945718706   961852377   965904307
   967680031   993505642  1009063125  1020521027  1020975754  1030583005  1036254129
  1045618677  1065040762
Binary Insertion Sort
100 elements, 7077 moves, 525 compares
    11928963    12591398    20021796    21128680    21961325    31822619    34412749
    34886069    49021110    57509491    59851133    68574097    75257048    77144257
    86585853   101301346   114389756   114632963   118852153   127224362   129666022
   134022820   138123204   141594986   143120736   155493230   161637111   170409441
   171807697   174240889   191344001   191929321   198838075   212472681   218109776
   229706589   270359703   292850682   336205321   342217139   345616751   350595885
   351969720   389883213   449398437   461085871   462792178   466359476   501405920
   510106982   513334883   521241737   526542695   557107568   561287884   567873507
   575563805   587056094   588721288   607827590   617793922   619973746   620258073
   620543907   621942163   623607358   627844041   640600366   653191764   654929869
   677513702   684315706   697535498   718744680   725644732   732145788   780416126
   781032961   790225475   792153223   835312041   837534179   838653684   877426267
   897448063   914913962   928880504   942053238   945718706   961852377   965904307
   967680031   993505642  1009063125  1020521027  1020975754  1030583005  1036254129
  1045618677  1065040762
```

Work Cited

"Bubble Sort." *Wikipedia*, Wikimedia Foundation, 13 Nov. 2020, en.wikipedia.org/wiki/Bubble_sort.

"Quicksort." *Wikipedia*, Wikimedia Foundation, 29 Oct. 2020, en.wikipedia.org/wiki/Quicksort.

"Shellsort." *Wikipedia*, Wikimedia Foundation, 3 Oct. 2020, en.wikipedia.org/wiki/Shellsort.