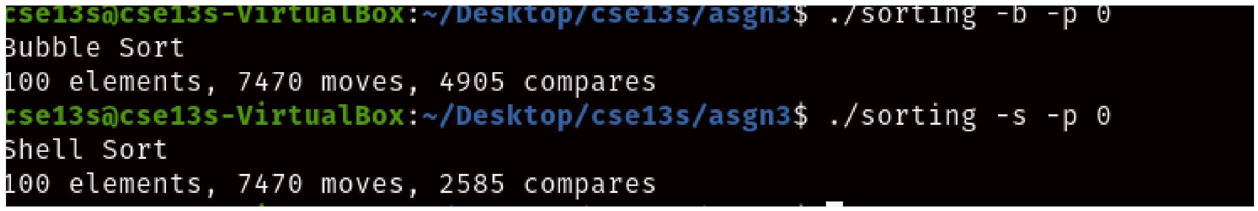
Mattiwos Belachew
mbelache@ucsc.edu
4/25/2021

CSE 13s Spring 2021
Assignment 3:
Sorting: Putting affairs in order
Writeup Document

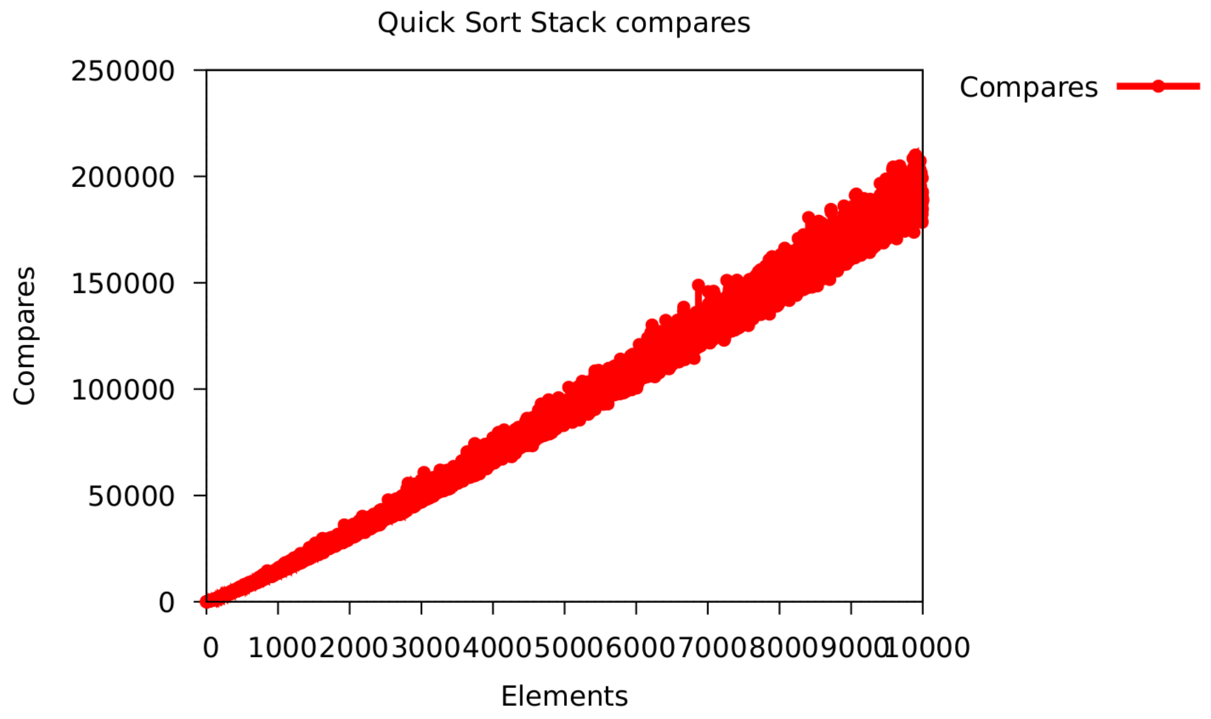
- Identify the respective time complexity for each sort and include what they have to say about constant.
  - The time complexity of bubble sort is O(n) best case for example a sorted array and worst case O(n^2)
  - The time complexity of shell sort is O(nlogn) best case and the worst case is O(n^2) making it basically a bubble sort since the gap will be 1 making it equivalent to running a bubble sort.
  - The time complexity of quick sort is O(nlog(n)) best case and worst case is O(n). Even though the worst case might be every slow it is still quick on average which is O(nlogn). But one of the reasons quicksort is bad is because real world collections are never truly random and there are always ways to create better sorting algorithms than quick sort for different collections making it more efficient than using quick sort.

- What you learned from the different sorting algorithms.
  - I learned that if the gap of shell sort is 1 then it is equivalent to bubble sort since the difference between them is that shell sorts likes to move the big numbers from the bottom to the top in order to reduce the number of moves required once the gap gets reduced to 1 for the final sort.
  - Another thing I learned is that even though quick sort is quick it is still very effective for random order numbers.
  - I learned that when the elements are small bubbles are very efficient however when the number increases it becomes a burden and a horrible algorithm compared to the other two.
  - Another thing I learned is that you are able to use both stacks and queues in order to implement quick sort. However the stacks max size is always lower than the queue.
- How you experimented with the sorts.

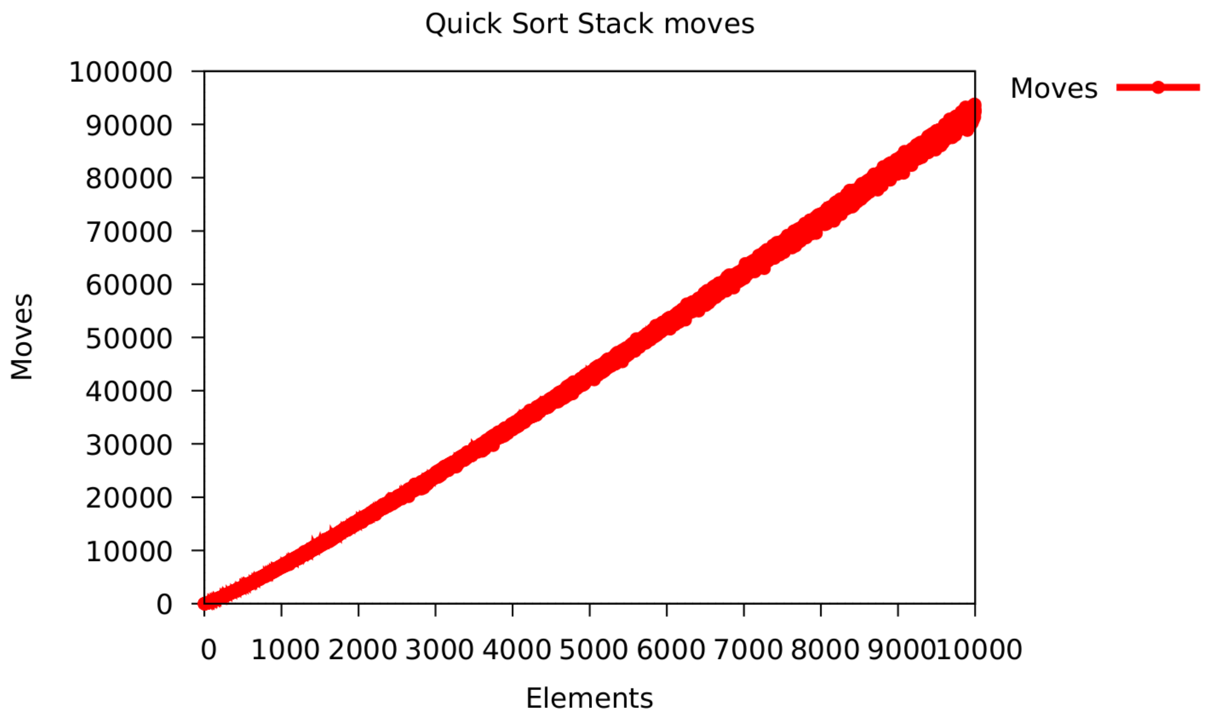


```
cse13s@cse13s-VirtualBox:~/Desktop/cse13s/asgn3$ ./sorting -b -p 0
Bubble Sort
100 elements, 7470 moves, 4905 compares
cse13s@cse13s-VirtualBox:~/Desktop/cse13s/asgn3$ ./sorting -s -p 0
Shell Sort
100 elements, 7470 moves, 2585 compares
```

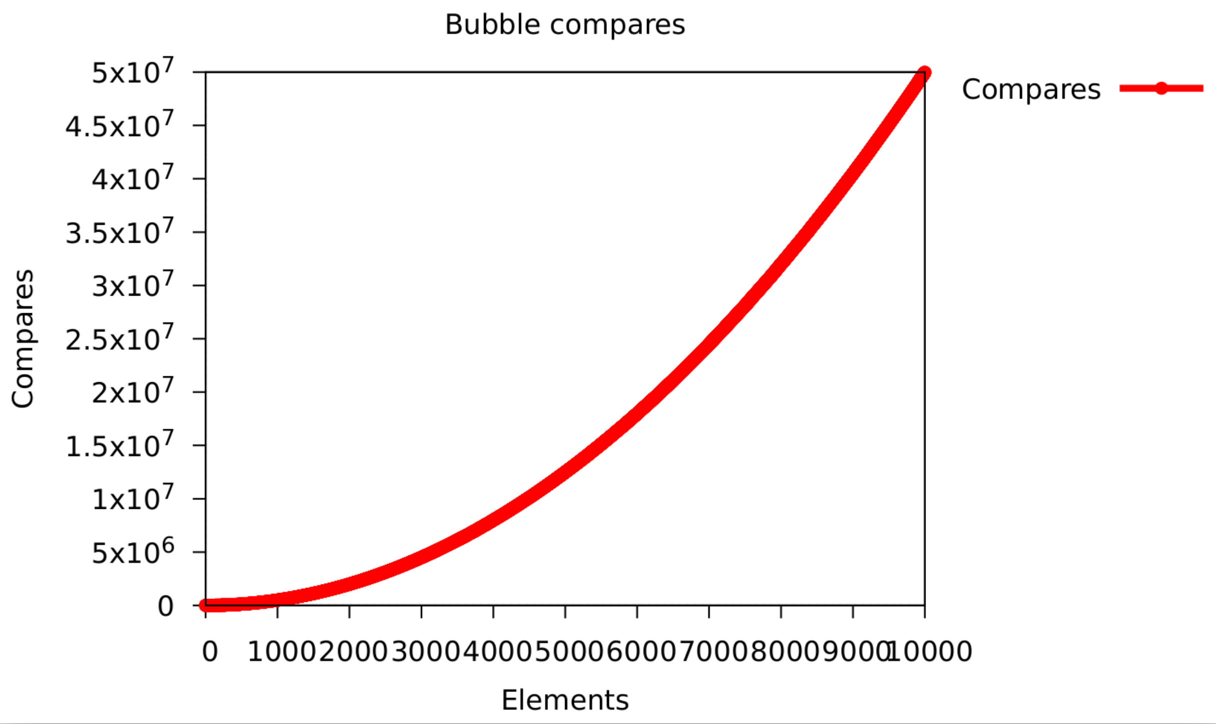
    - One of the things I experimented with was setting the gap to 1 only and see the difference between Bubble sort and shell sort with the gap at 1. I found it interesting that even though the amount of moves is the same. The comparisons differ by a lot, almost 50% which is interesting to think about. This happens as the element increases as well. This might be because of how its comparing the numbers in the while loop which means that it might or might not be the same time complexity.
- How big do the stack and queue set? Does the size relate to the input? How?
    - The highest it got was 86 with 100000000 elements for stacks and the highest that queue got was 163412 for 1000000 elements. The stack max size is related to logn with a constant of about 10 I believe and for a queue I think the queue is n^ a small fraction when related to its size. I believe this since log(100000000) is 8 and with a constant like 10 the result is 80. However for the queue I only have a rough guess since even though it's not logarithmic and nor exponential I thought it might be a stretched paraba or something like that since it increases pretty slow compared to the elements.
- Graphs explaining the performance of the sorts on a variety of inputs, such as arrays in reverse order, arrays with a small number of elements, and arrays with a large number of elements.

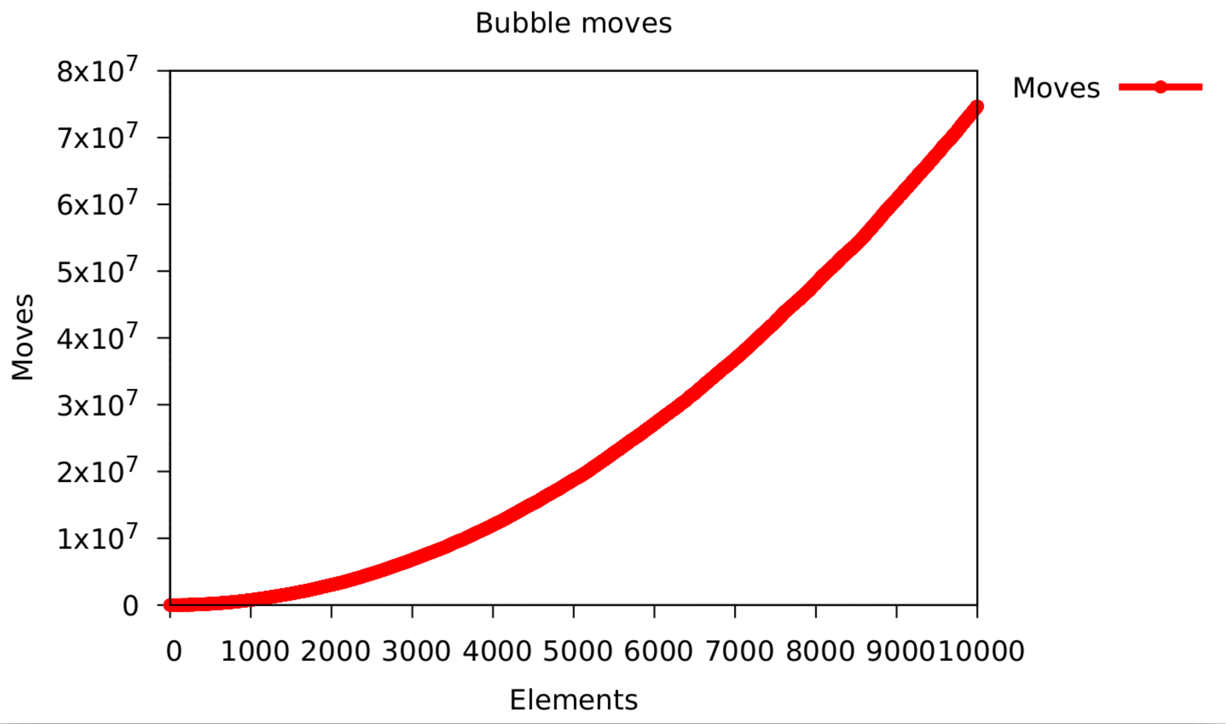


- -

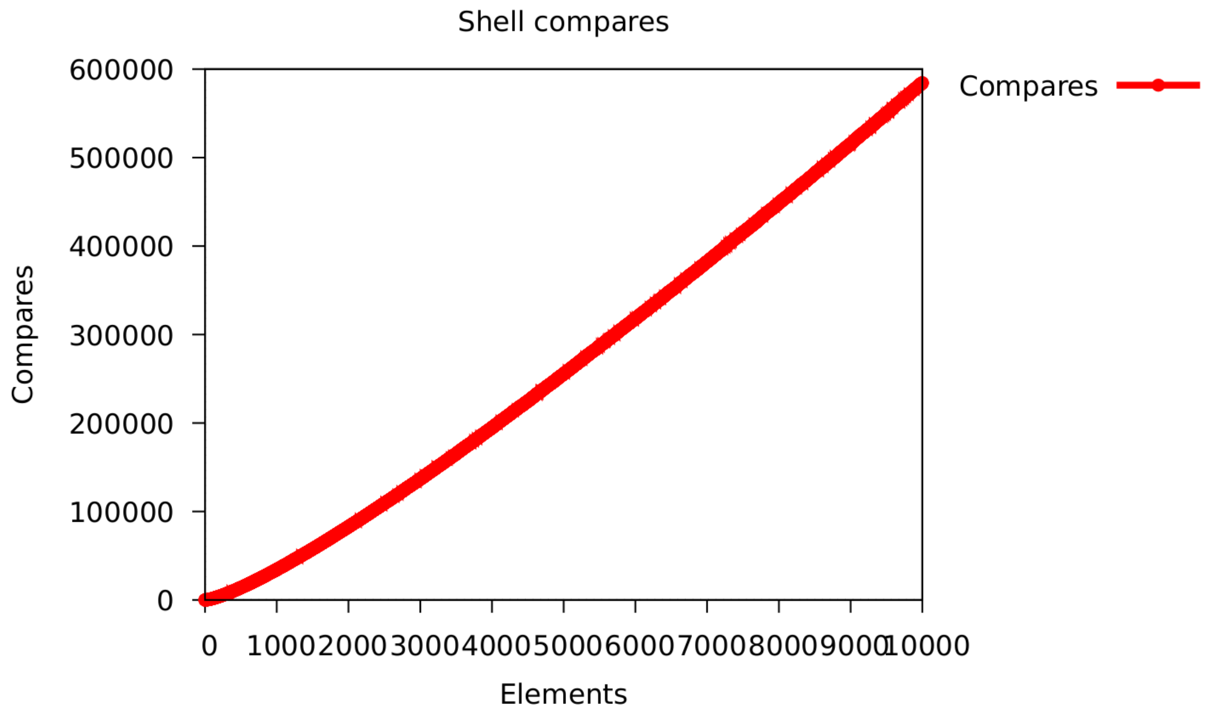


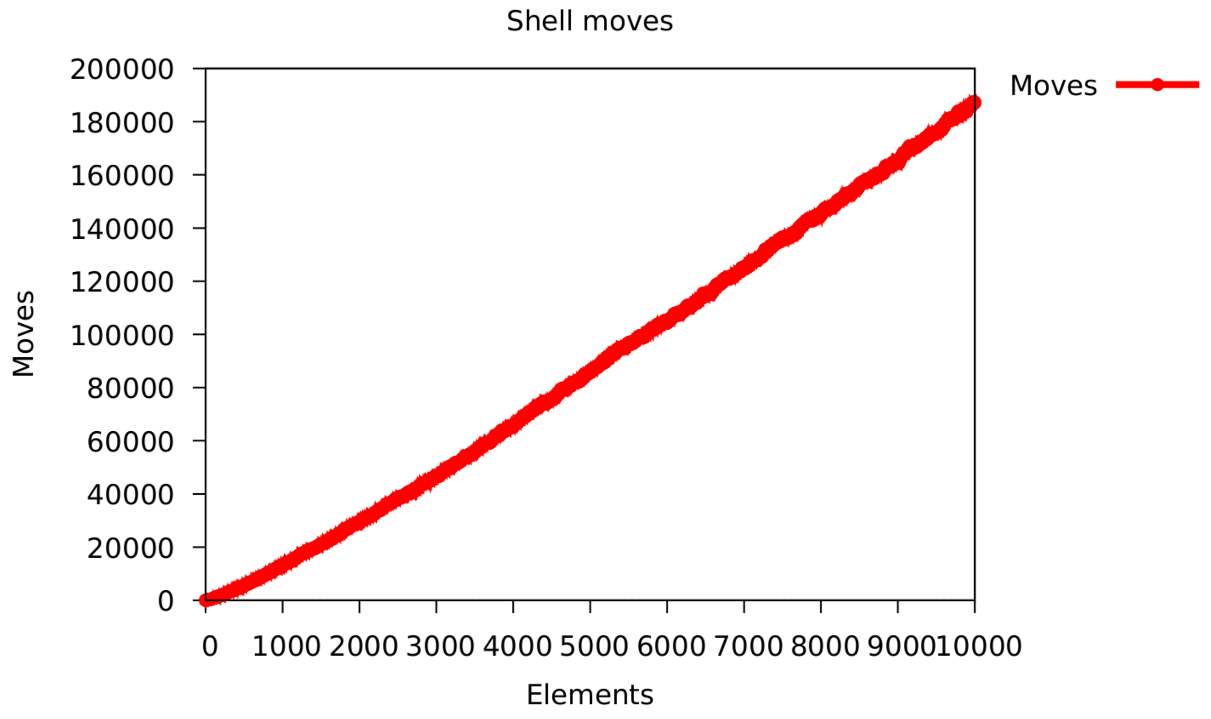
- Quick Sort (Stack) Moves and Compares





- Bubble Sort Moves and Compares





- Shell Sort Moves and Compares

- Analysis of the graphs you produced.

- Compared to the other sorting algorithms bubble sort compares and moves is the largest by far because of its bad time complexity.
- The lowest moves algorithm is Quick sort with stack.
- Bubble sort moves and compares look like they are increasing exponentially.
- My Quick Sort compare standard deviation is very large for some reason. I’m not sure why because the pivot isn’t random so it should have a consistent growth instead of having a large deviation. In addition, the graph y-axis scale isn’t small in order to account to that large of a standard deviation.