Bill Zhang
jzhan411@ucsc.edu
4/25/2021

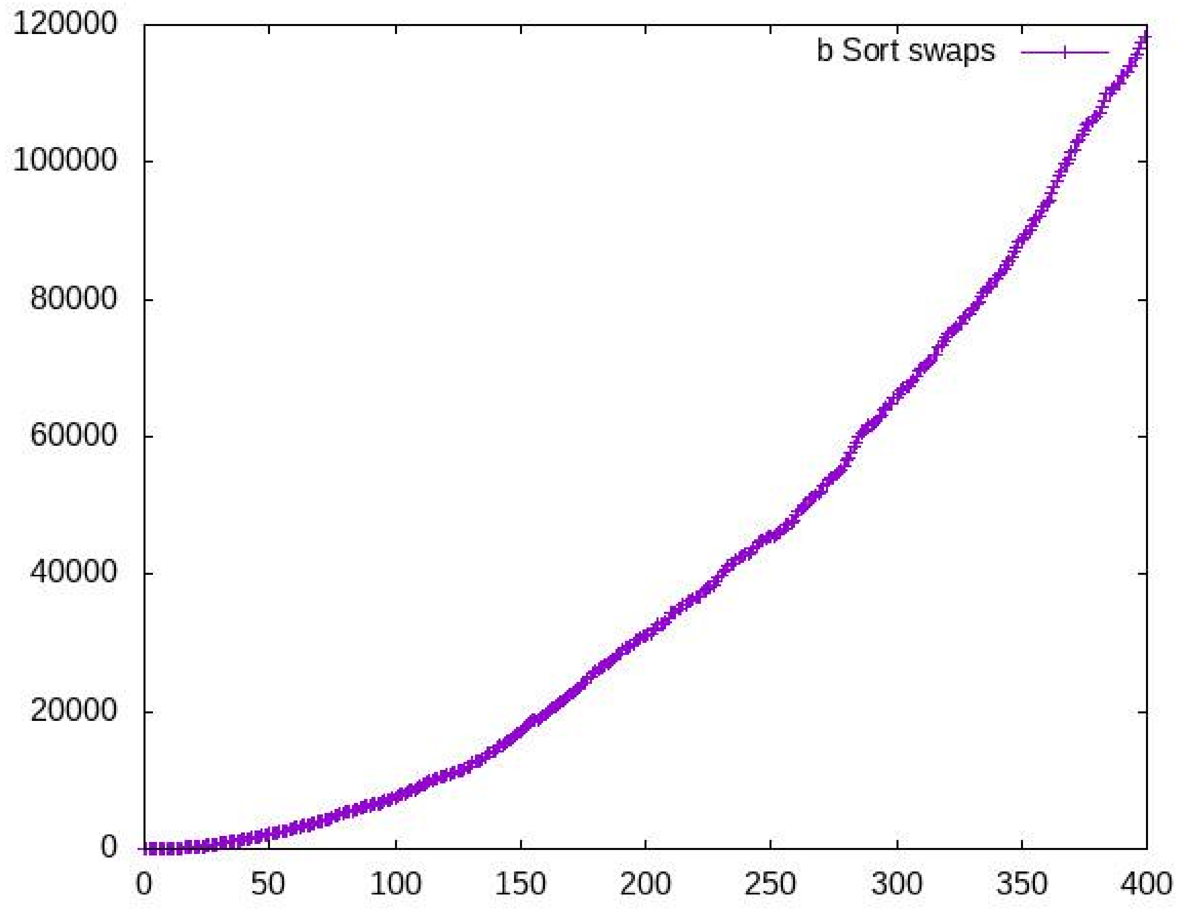
CSE13S Spring 2021
Assignment 3: Sorting: Putting your affairs in order
WriteUp

Bubble Sort:

The time complexity for bubble sort is O(n^2). You can see that the graphs in Swaps and Compares both are similar to the y=x^2 graphs. The main thing bubble sort taught me was that you can't compare different int types. In the for loop where the initial variable is int and the compared variable is uint32_t. I've never had this problem before. After seeing that, I realized that for every for loop the initial variable and compared variable must have the same data type.

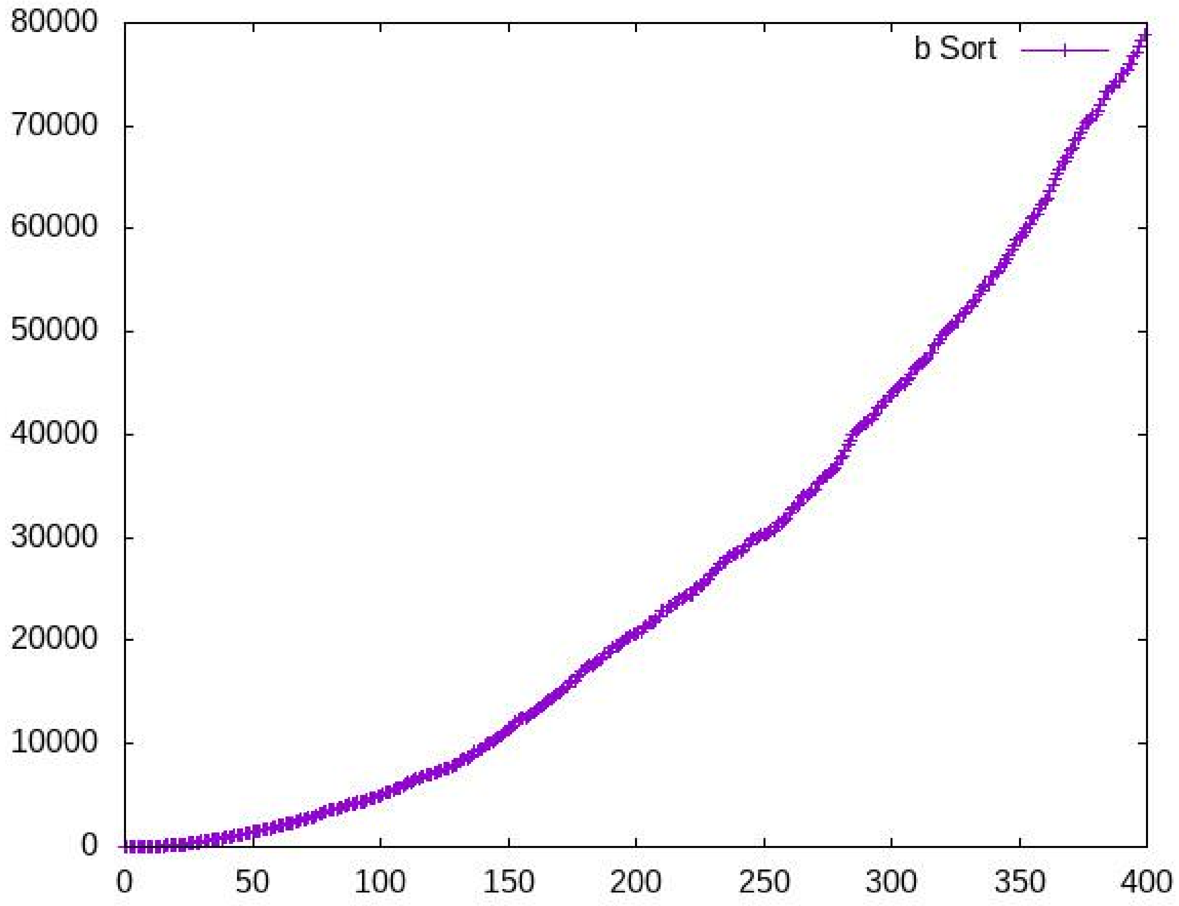
Based on the graphs, it seems like bubble sort is best used for arrays with low amounts of elements. As the amount of elements rises, the time increases massively. At 400 elements there are almost 120000 swaps.

Bubble Sort Swaps:



Increases massively at the end

Bubble Sort Comparisons:



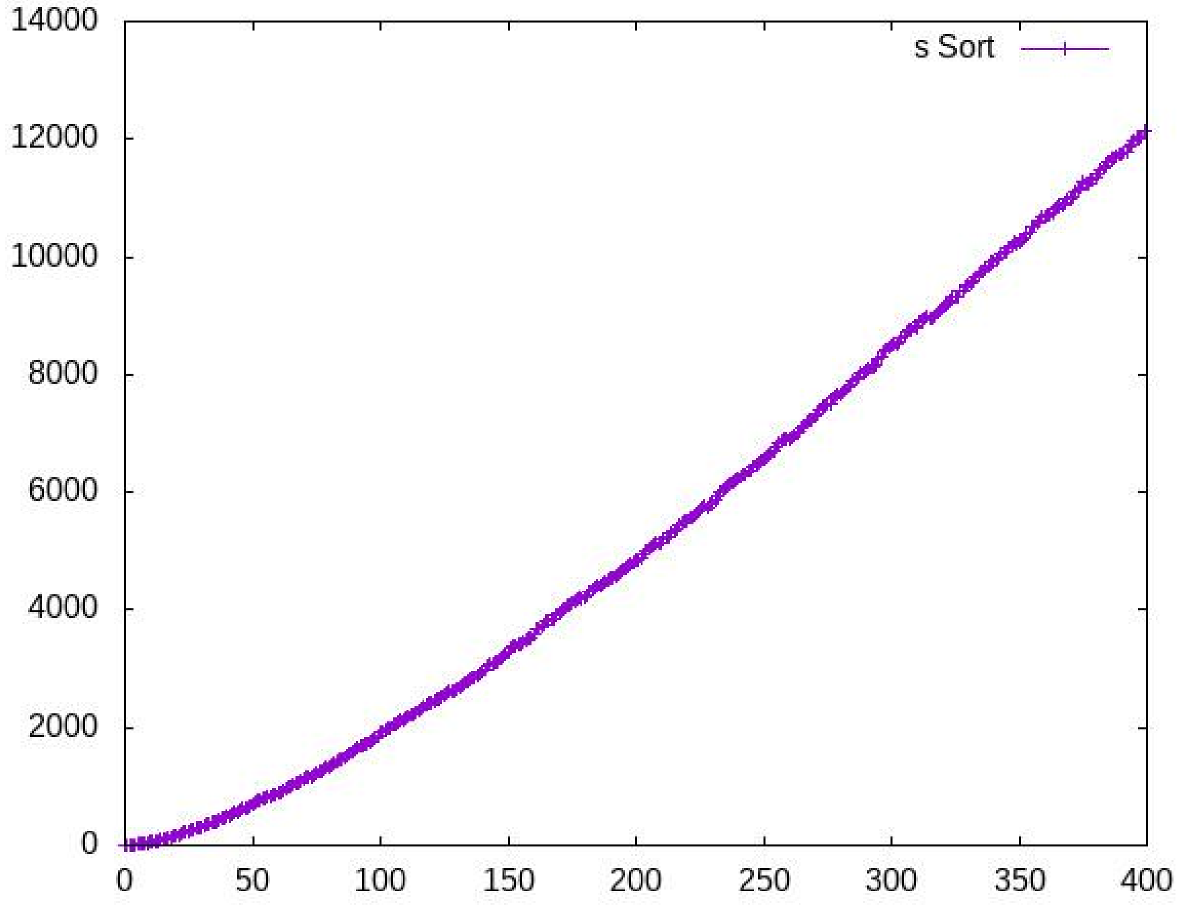
Similar to swaps in shape.

Shell Sort:

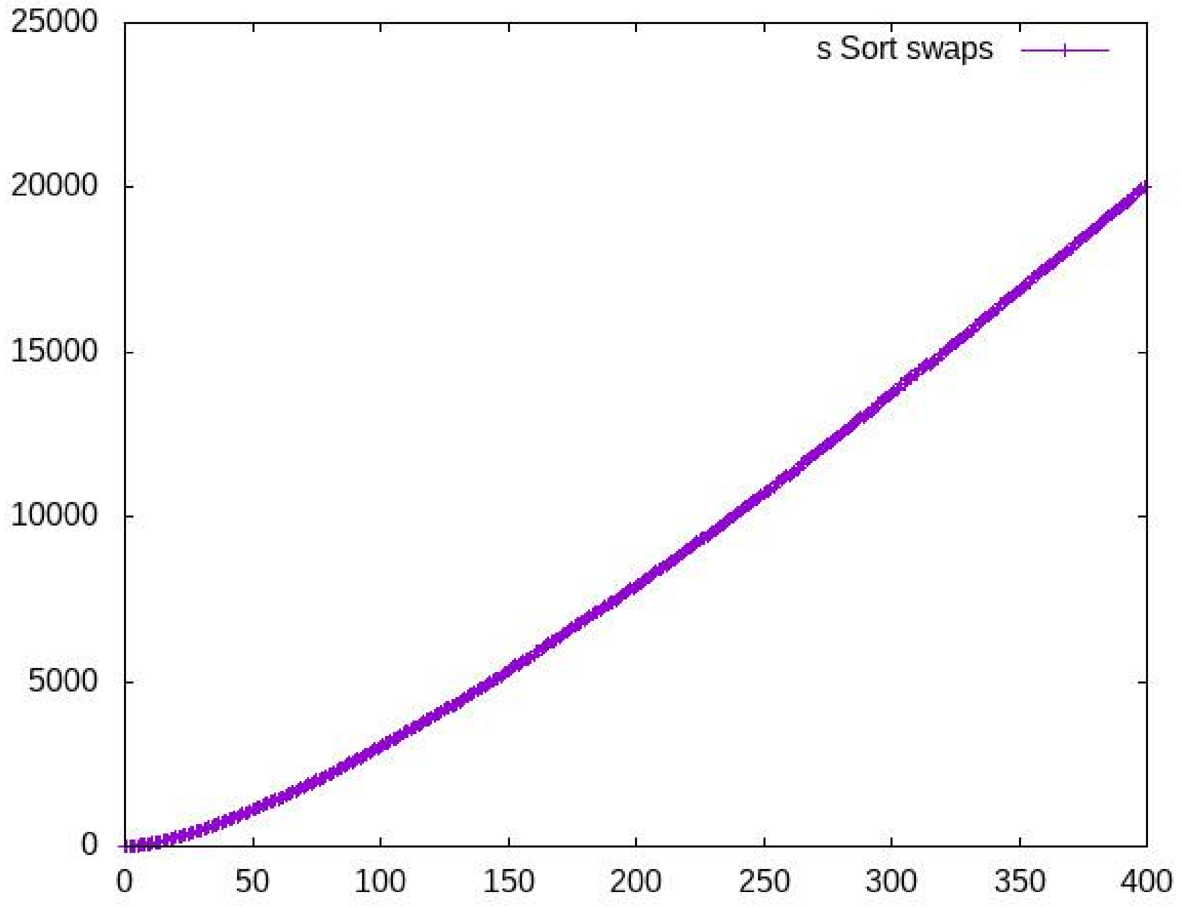
The time complexity for shell sort using Pratt's sequence is nlog^2(n). You can see this from the graph. It doesn't rise as sharply as bubble sort at the end. Though the downside is the time for small length lists is higher than bubble sort. The main thing I learned from Shell Sort is how shell sort works. Before this lab, I pretty much didn't know what shell sort was.

Based on the graph, it seems like shell sort is better than bubble sort only for very large lists. Shell sort compares and swaps don't rise as sharply as bubbles at the end. Where Bubble Sort has 140000 swaps at 400 elements, shell only has 20000.

Shell Sort Compares



Shell Sort Swaps



Quick Sort:

The time complexity of quicksort at best is O(nlogn) and at worst is O(n^2). I learned the most in this lab from quick sorts.

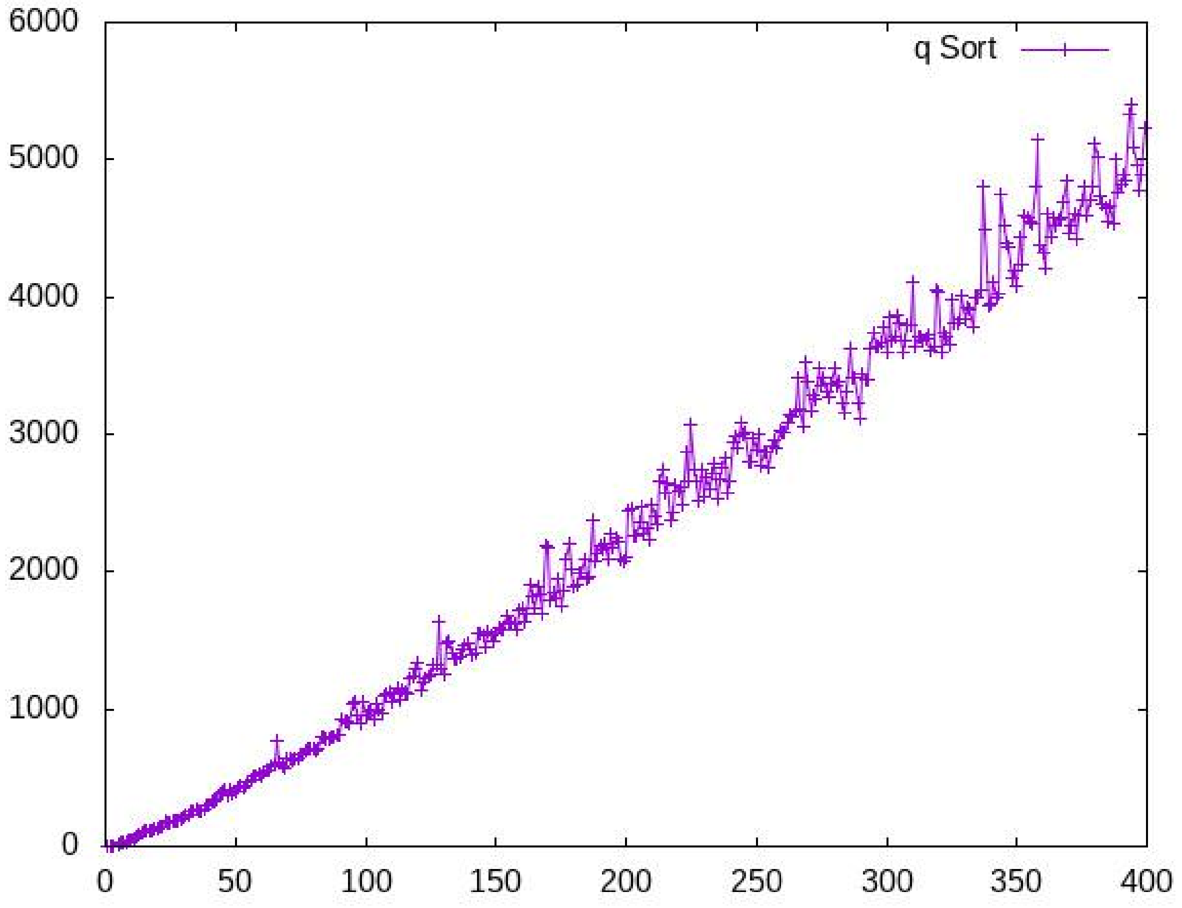
Small list of the stuff I learned:

- Structures in c
- Memory management
- How quick sorts work
- How to make specific variables visible across other files
- And more

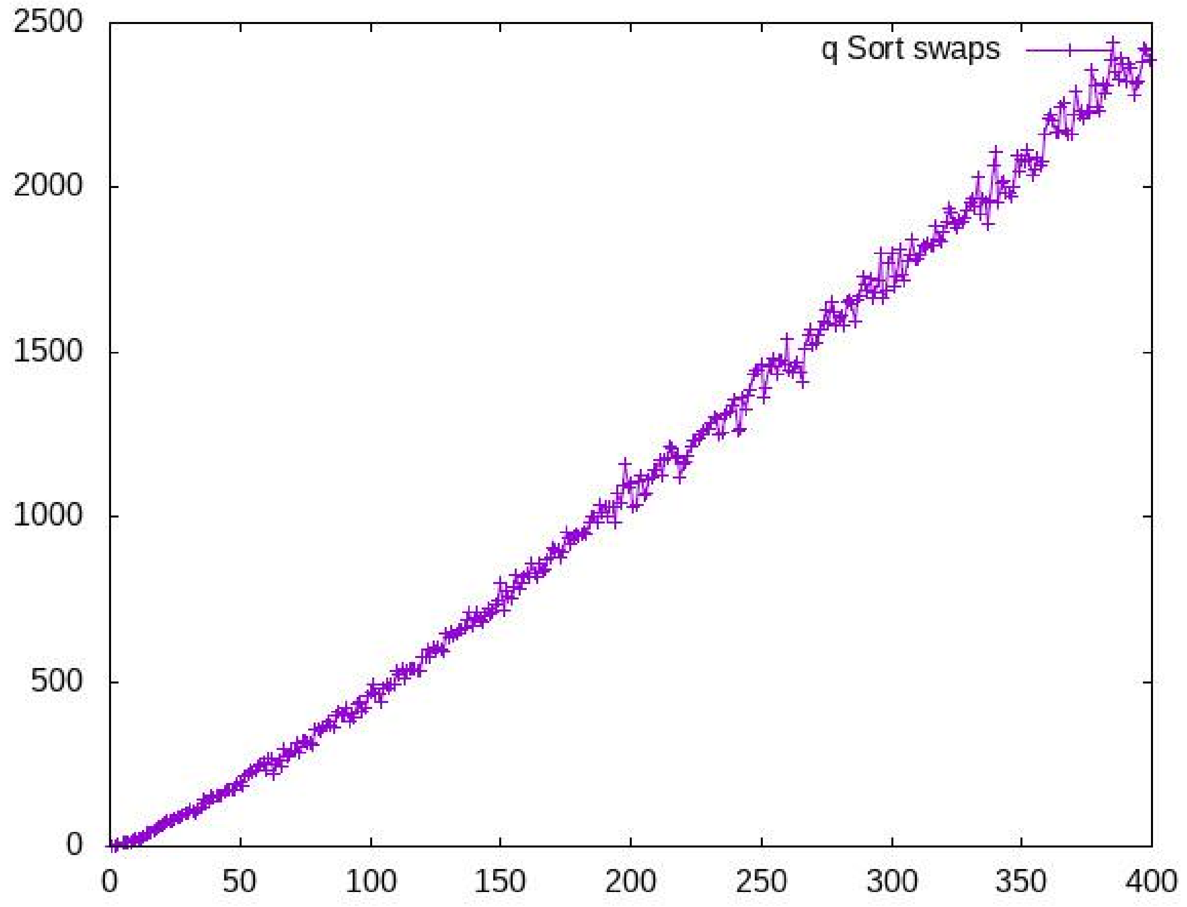
From the graph, you can see that Quick Sort, like the name implies, is the fastest sorting method by a large margin. The max queue for the default values is 36, and the max stack for the default value is 24. It seems like as the input gets larger, queue sizes increase an insane amount compared to stack sizes. For example, at 1 million inputs, queue size is over 150k whereas stack size is only in the double digits. This is most likely because a stack is more efficient at storing values compared to a queue.

Based on the graph, it seems like Quick Sort is better than both shell and bubble sort. Where Bubble Sort has 140000 swaps at 400 elements and shell has 20000, Quick sort only has a tiny 2500 swaps for stack and 2000 for queue.

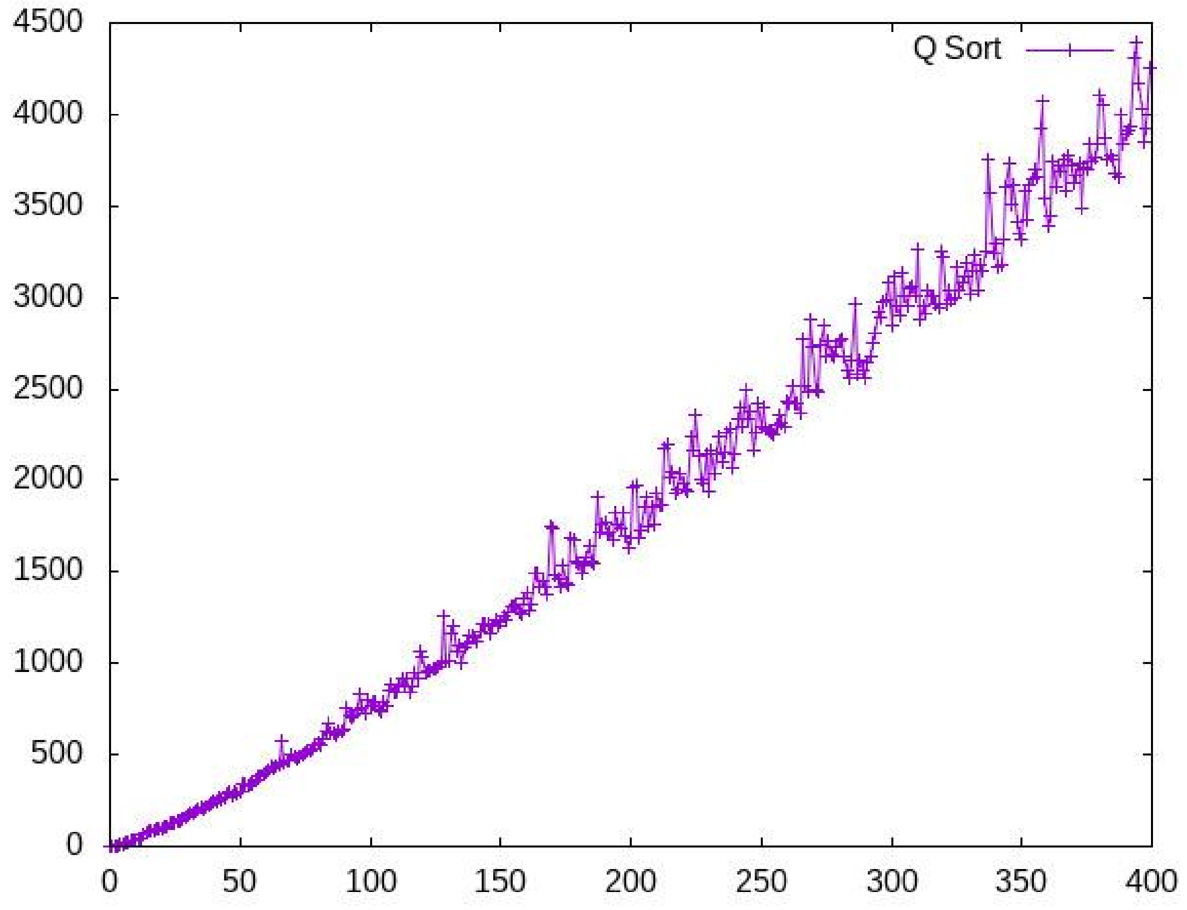
Quick Sort (Stack) Compares



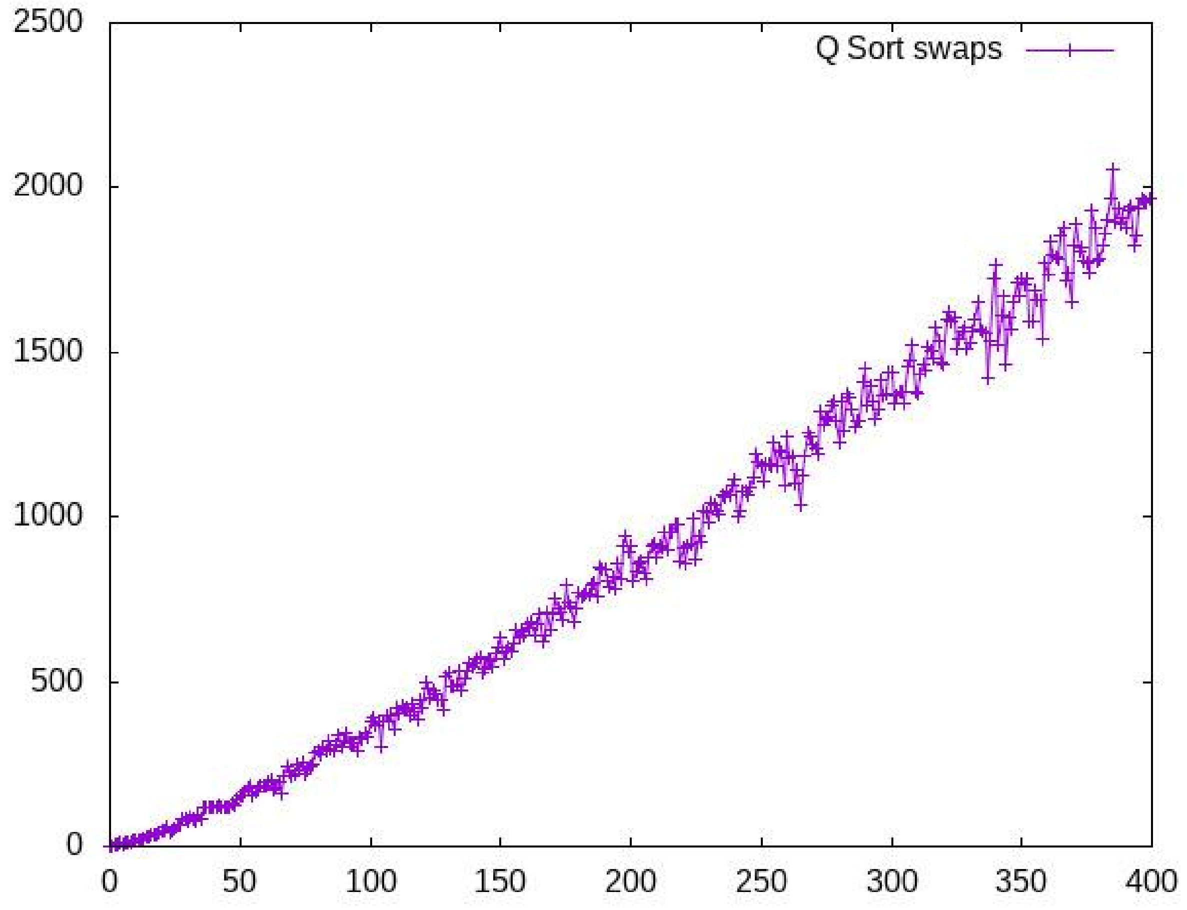
Quick Sort (Stack) Swaps



Quick Sort (Queue) Compares



Quick Sort (Queue) swaps


Q Sort swaps