Mihir Jham
CS251
Project 3 Report
02/25/2013

CS251 – Project 3 Report

1. I optimized the original QuickSort by the following:
    - Changed the pivot selection from random to a QuickDualPivot Quick Sort.
    - I added a threshold of 30, where the QuickSort stops and a Selection Sort begins sorting the remaining unsorted elements. The reason I selected Selection and not any other sort was because Selection Sort uses the least number of swaps.
    - I added a isSorted() function, which checks if the the array is already sorted, therefore handling one of the worst test cases for a QuickSort algorithm

2. The followiing are the test results for the respective algorithms:

QuickSort.java

	Comparisons	Swaps
Already Sorted	111,187	36,573
Random Set 1	28,552	7,288
Random Set 2	28,350	7,442
Repeated	264,406	87,369
Reverse Sorted	43,754	8,050

*Note: As the algorithm takes a random pivot every time its run, the number varies, but this is an average.

QuickSortOpt.java

	Comparisons	Swaps
Already Sorted	1,999	0
Random Set 1	514,251	4,806
Random Set 2	512,259	4,469
Repeated	517,436	3,750
Reverse Sorted	1,371,561	2,454

As it can be seen, the number of comparisons increases for the optimized algorithm, but the number of swaps greatly reduce, when compared to the original algorithm. This is due to the addition of the Selection Sort, as my main focus was to reduce number of swaps, I took the steps to ensure that, with the trade off being the number of comparisons.

3. A good reason why QuickSort is so fast in practice compared to most is because it is relatively cache-efficient. The reason for this cache efficiency is that it linearly scans the input and linearly partitions the input. This means we can make the most of every cache load we do as we read every number we load into the cache before swapping that cache for another.

Mihir Jham
CS251
Project 3 Report
02/25/2013

4. Merge Sort is better suited to work with Linked Lists, as an item can simply be spliced in and out of a list representing the merged halves without using any extra space, aside from a pointer or two.