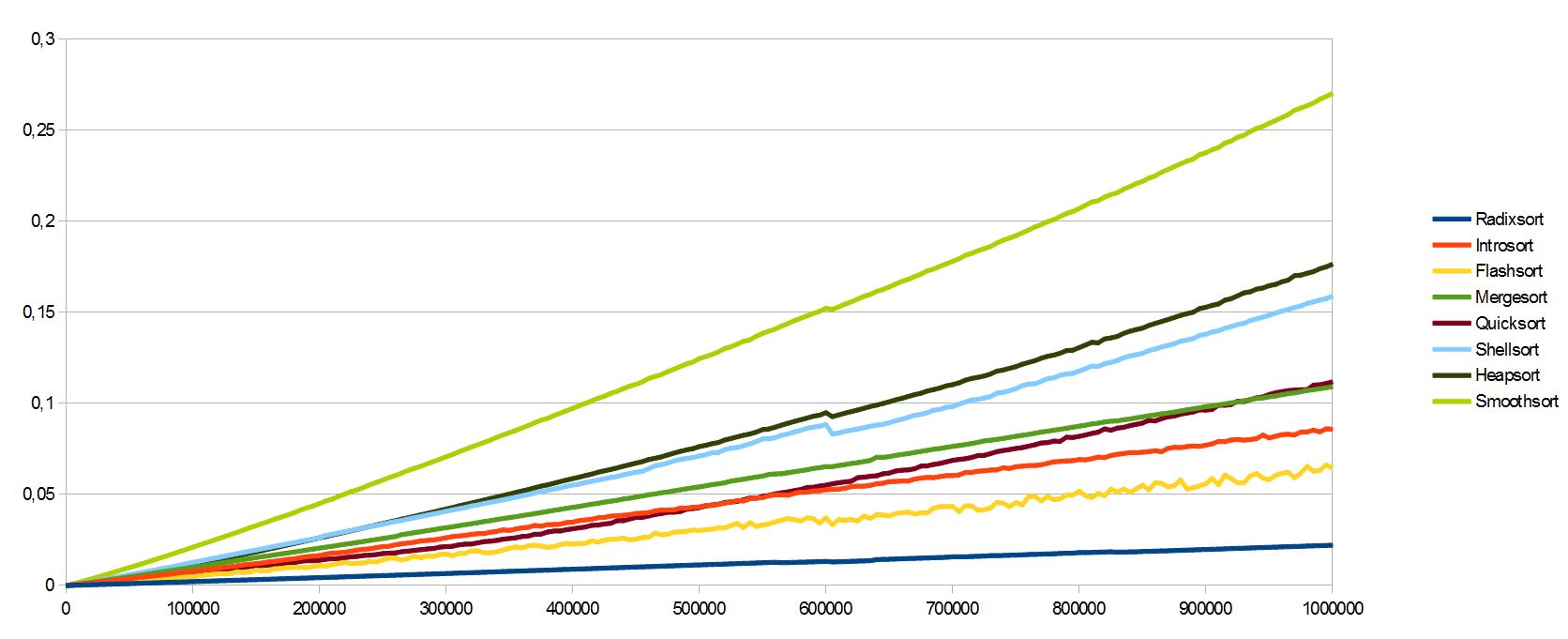
Name: Mena Sergeyous

Date: 11/10/15

This graph show the actual Big-O for each soring Algorithms.



This is my graph show the Big-O for each soring Algorithms.

Insertion sort

This graph shows that insertion sort toke long time because the Big-O is

There are 2 loops that require n iterations = O (N \* N) = O (N^2)

Best Case Average Case Worst Case

O (n) O (n^2) O (n^2)

Bubble Sort

There are 2 loops that require n iteration = O (N \* N) = O (N^2)

Best Case Average Case Worst Case

O (n) O (n^2) O (n^2)

Heapsort

To build heap is O (n) + heapify which is log n = O (n log n)

Best Case Average Case Worst Case

O (n log n) O (n log n) O (n log n)

Merge Sort

Merge sort uses recursive calls to divide the array into smaller array

Call to merge takes merge \*2 long n calls to merge sort () and merge () takes O (2N) time

O (2 log n\* 2n) = O (n log n)

Best Case Average Case Worst Case

O (n log n) O (n log n) O (n log n)

Quicksort

Makes recursive calls like merge sort. Quick sort use recursive calls on the left and the right side

Takes 2\* log n call to quicksort () and for each portion takes 0(n)

= O (2\*log n \*n) = O (n log n)

Best Case Average Case Worst Case

O (n log n) O (n log n) O (n^2)

Final Result

Each sorting Algorithm matches the Big O notation, but bubble sort took more time then insertion sort which has the same Big-O notation O(n^2). Insertion sort outperformed bubble sort because the shifting of indices. The best timing was quick sort not like marge sort which require Big O (n) space. Heap sort performed in between because each time it needs to build max heap to get the larger value and swap it to the last index in the array.