**Sorts: Bubble vs Selection vs Insertion**

Answer 1:

By first scanning the entire list before locating the exact pair of numbers to swap, only two writes to memory are performed by Selection Sort for each O(n) scan, whereas Bubble Sort does writes on each and every comparison. So Selection Sort does O(n) writes to memory whereas Bubble Sort does O(n^2) writes.  
But this wouldn't likely amount to an appreciable difference in practice, particularly because you really shouldn't use either algorithm in production. I guess this makes Selection Sort trivially better for the lifetime of your RAM?  
  
Insertion Sort actually does see a good amount of use in production in spite of running in O(n^2) time, because its performance trends toward O(n) time for lists which are already mostly-sorted, which happens quite often in real-world cases (and being O(1) in space sure doesn't hurt).  
For instance, Python's default list sorting algorithm is called [Timsort](http://en.wikipedia.org/wiki/Tim_sort), and it's a combination of Insertion Sort and Merge Sort, choosing between them according to their strengths.

Answer 2:

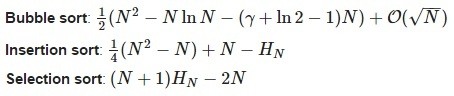
If you have to sort 100 fridges in order of their weight, you’ll prefer less number of moves even if it means more comparisons.  
  
But if you have to sort a group of 1000 people by their racing speed, it will be annoying for everyone, if your strategy is to make every possible couple of those people, race every couple, mark the winner and merge to the aggregate list in order of winning.(If you actually implement like this some of these people may punch your face),you’d better ask them to race together and record the people who reach the finish line , in that order.

Answer 3:

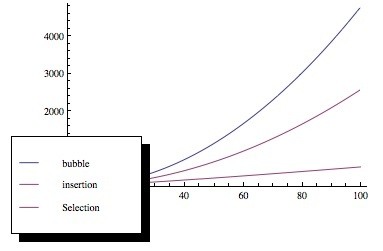
Selection Sort is used normally in cases where *memory writes* are quite expensive than memory reads. It only does *O*(n) memory writes, but Bubble Sort needs at least twice that many memory writes. Experiments by Astrachan sorting strings in Java show bubble sort to be roughly 40% slower than Selection Sort

Answer 4:

Given Big O notation all of its addition and constants are omitted.  
Below are the functions of each sort:



So, when we plot these functions on the graph we can see the difference:



Now, we can see that selection sort is better when the number of elements increases. (Input is random and time complexity omitted)