**Sorting and Complexity**

1. First, implement the [bubblesort](http://en.wikipedia.org/wiki/Bubble_sort" \t "_blank) algorithm. Your function should take a list of numbers as it's single argument and it should return a sorted version of that list. The original list should remain the same. Do the sort 10 times and return the average time taken.
2. Let's make sure that your bubblesort code is correct. Write a function that checks that the function passed to it as an argument correctly sorts a list. You can assume that the built-in sortfunction works correctly, and compare the results of using the function passed in to those obtained by running sort. Or, you can write your own function from scratch.
3. Graph the performance of bubblesort for lists of random numbers of length 100, 200, ... 2000. Remember to give your graph a title and label the axes. Also plot the running time of the built-in function sorted as a comparison.
4. Graph the performance of the built-in sorted function on the same set of data. What do you notice about it, compared to bubblesort?
5. Quicksort is one of the most famous sorting algorithms. Implement it, time it, and graph it.
6. Implement another sorting function, either insertion\_sort or mergesort. Implement it, time it, and graph it