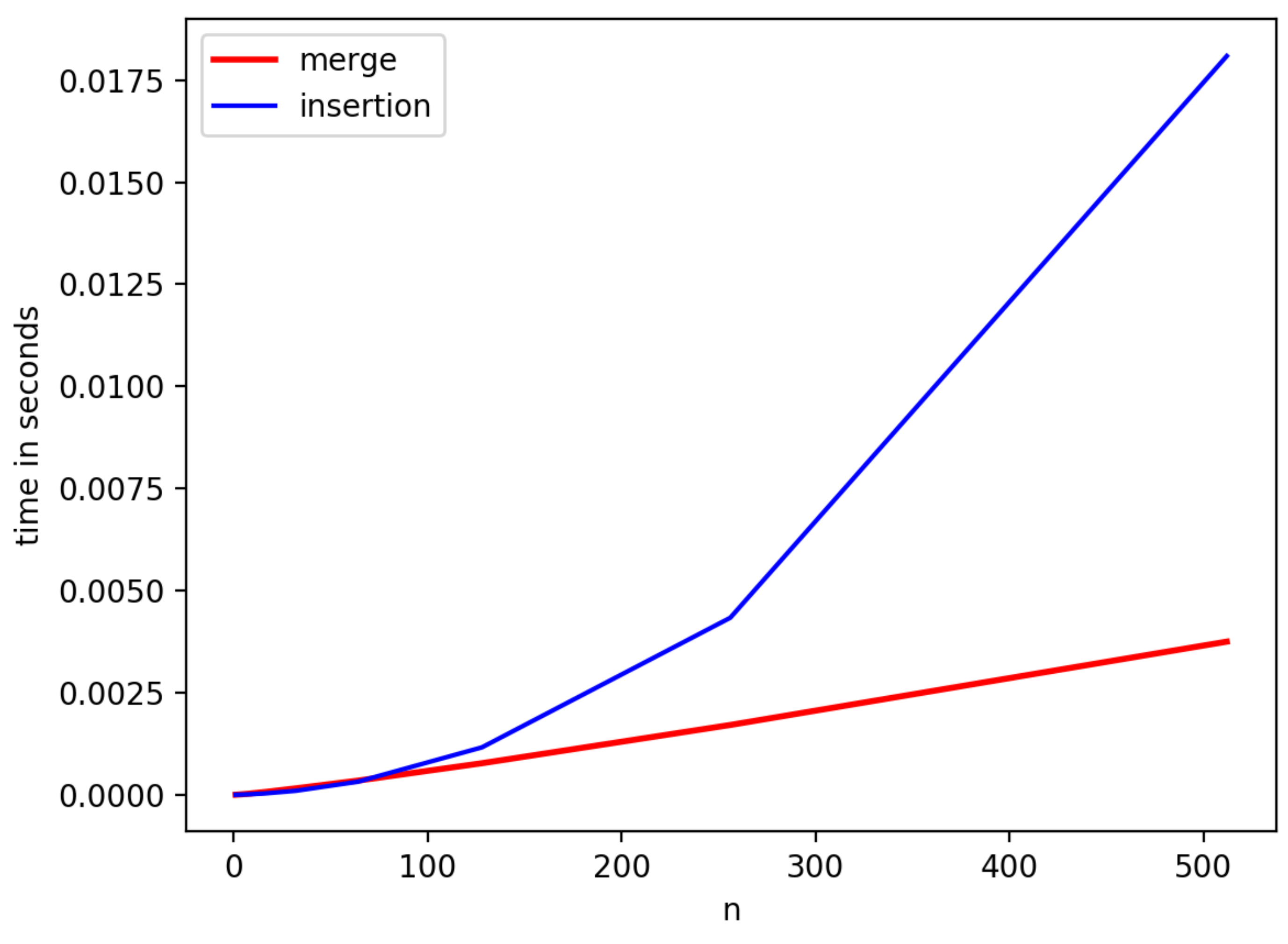
# CS140 Extra Credit

## Gabe Magee

1. For what values of n is insertion sort faster than mergesort?

For these problems, I made my own versions of each sorting algorithm, created random lists in doubling size, and timed them sorting each list. To make sure I was accounting for outliers, I averaged the results over 100 iterations at each size of list. I got this result, showing Iterative search's O(n^2) nature and Merge sorts O(nlog(n)) nature.



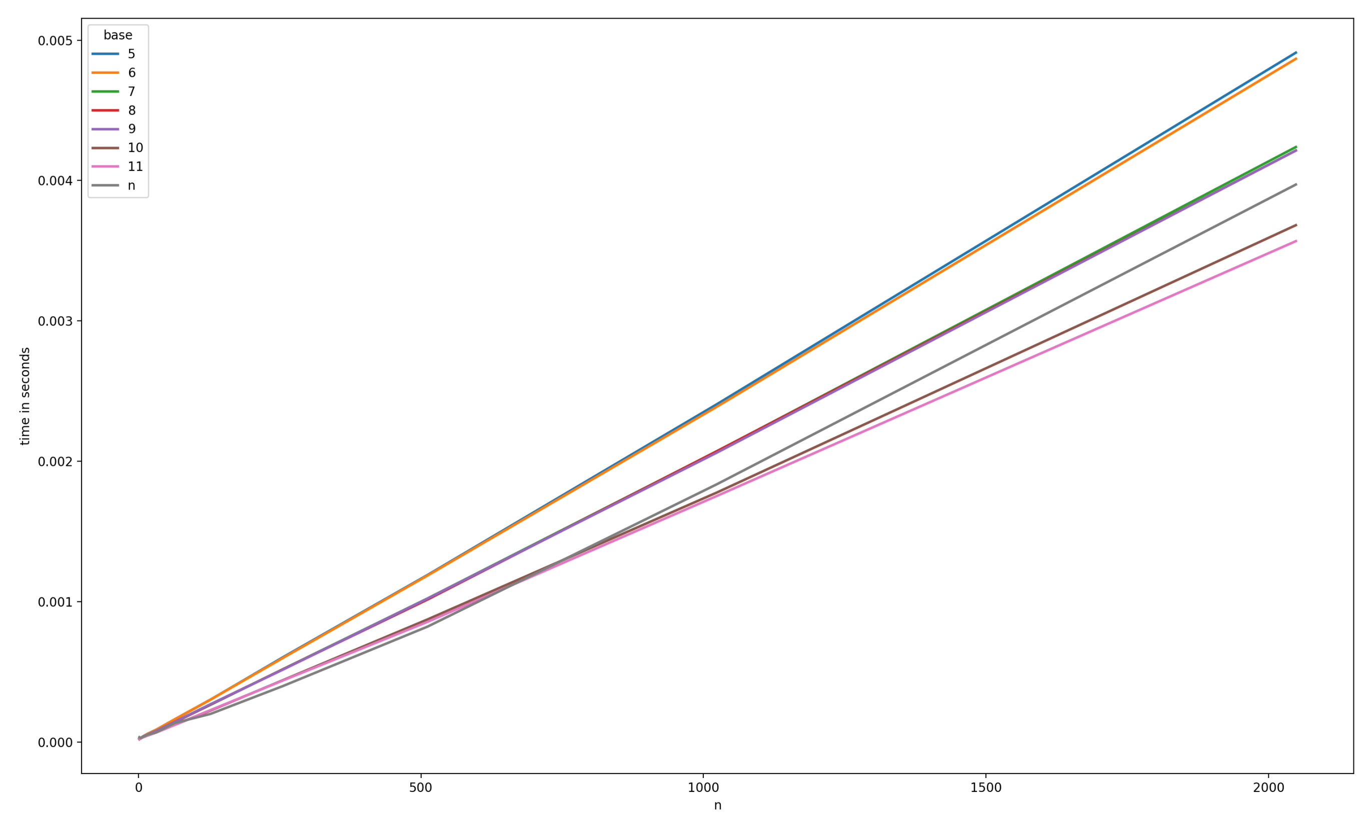
The actual mean times are as follows:

## Merge v Insertion

| function | 2 | 4 | 8 | 16 | 3 |
|---|---|---|---|---|---|
| insertion | 0.000001 | 0.000003 | 0.000006 | 0.000017 | 0.00005 |
| merge | 0.000002 | 0.000006 | 0.000015 | 0.000036 | 0.00007 |

2. What base is best to use with Radix sort when sorting lists with random ints?

I implemented Radix sort using the recomendations I found online at Geeksforgeeks.com. Using this, I used pyplot to make a graph comparing the runtimes of various bases, including the length of the list itself.Similar to the other, I iterated over various lengths of random lists and did multiple trials



If you are interested in the raw averages, here they are:

## Radix Base

| base | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| 5 | 0.000020 | 0.000029 | 0.000041 | 0.000074 | 0.000120 |
| 6 | 0.000020 | 0.000028 | 0.000041 | 0.000073 | 0.000121 |
| 7 | 0.000019 | 0.000026 | 0.000037 | 0.000064 | 0.000104 |
| 8 | 0.000020 | 0.000027 | 0.000039 | 0.000066 | 0.000106 |
| 9 | 0.000020 | 0.000029 | 0.000040 | 0.000068 | 0.000110 |
| 10 | 0.000020 | 0.000026 | 0.000035 | 0.000058 | 0.000092 |
| 11 | 0.000020 | 0.000028 | 0.000036 | 0.000060 | 0.000095 |
| n | 0.000032 | 0.000031 | 0.000040 | 0.000072 | 0.000097 |

It seems the higher we go, the better the base. Some bases are better, like 5,8,10 (possibly due to their nature interacting with different numbers). Theoretically, n should be the best. But with higher max numbers present in the list it becomes more and more of a burden.