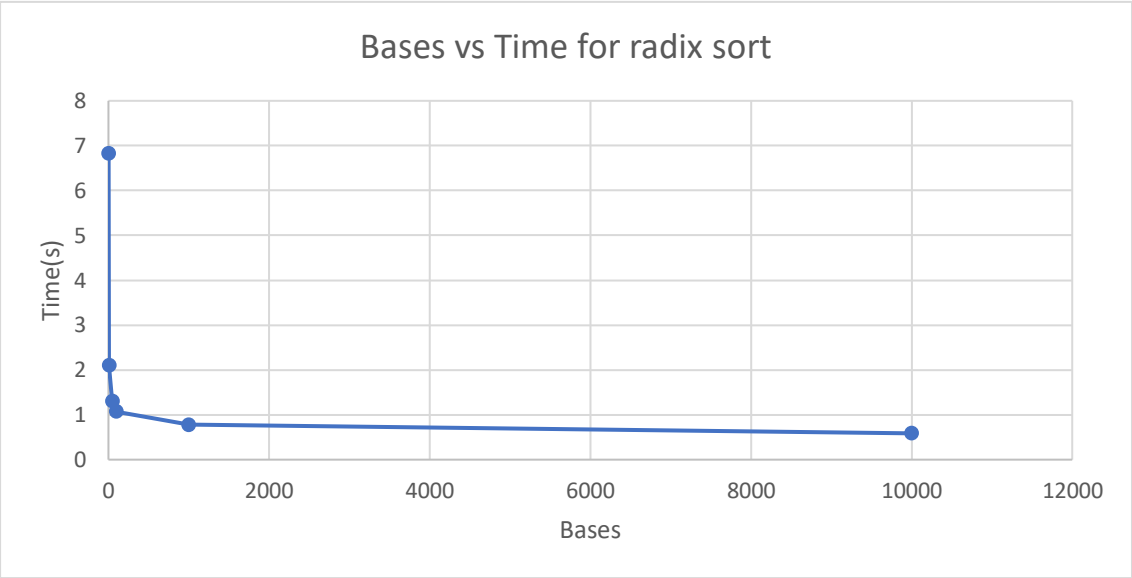
Analysis for radix sort

I chose bases 2,10,50,100,1000,10000 as it covers a variety of different bases from the smallest to a very large one. The first 3-4 bases were chosen in order to see changes in times with less differing bases, while the last 2 bases were to see the overall pattern.

Data obtained: [(2, 6.8325549), (10, 2.100418500000001), (50, 1.3029659999999996), (100, 1.0691301000000006), (1000, 0.7789988000000001), (10000, 0.5862803999999997)]



Firstly, we can see the basic idea of times reducing as the bases increase, though it is not a constant reduction in time. We can see that the reduction in time is a lot more significant during the first few bases when compared to the other bases. The pattern of the graph is logarithmic, meaning that the rate of reduction of time decreases as the bases increase.

The logarithmic pattern is formed due to the number of counting_sort is called, is calculated by math.floor(math.log(max(num_list),b) + 1) or logb(maximum number in the list) rounded to integer. When the base is high, the calculation involving finding the index of count array in counting sort becomes less complicated, when compared to lower bases. Since the index of the count array is determines by ((a_list[i] // b ** place) % b), a larger base means smaller numbers (a_list[i]//b), and less number of steps for the '% b' step. Hence, reducing the overall time taken.