Paragraph(for 1, 2 and 3, 4, 5, and 6) (in case it is needed to supplement information not covered in 1-6)

Things I found easy to understand was the selection and insertion sort because it was a review of class lessons. Things not easy understand was how the quicksort worked because it was not covered in class and I didn’t understand the point of a pivot and the recursive usage. It was easy to figure out that the insertion sort was not O(n2) because when I reviewed the implementation of selection sort, I remembered that binary search was a O(log n) algorthim. Something I struggled with was finding a mathematical relationship between run times.

Things I found easy to understand about the radix sort was HOW it worked, but I really struggled with designing the algorithm. The warning about 10 variables was ambiguous because it implied to me to not use any data structures at all so that puzzled me a lot. I eventually figured out that what we weren’t supposed to do was create 10 variable NAMES(like no NAME=data\_structure) and assigned them as data structures when I first created the code, since I had an issue on calling a particular one, when given an index. I realized instead we were supposed to create 10 QUEUES but not assign any names to them so I used a list of Queues instead. Then the manipulation of mod and division operators was the hard part and I keep switching between using powers of 10 OR using an index and then exponenting that particular index. Writing the tests for radix sort was easy to understand, and answering the questions for the lab.

I really had problems with modifying radix sort into negative values, because it was hard to implement it the original set up as the bins becomes easily buggy when I first tried to insert (see commented original \_insert on radix.py) negative values into the base bins. This was causing a lot of problems and the algorithm was VERY hard for me to understand too. So I decided to create more bins to make it a lot more understandable, and the only obstacle was making sure the negative values were being placed in the right bins. The radix sort overall was hard in general because of lack of TA support.

Lower case words became easier once I understood the algorithm and how to write a version of the radix sort. Test cases were pretty easy, I originally decided to start by indexing from theleft and move to the right but this caused a failure in one of my test cases (the first one) so I decided to index from the right to left instead.

Something I didn’t find really easy to understand was why use base 10, because I thought any bases would be the same speed(By ‘shortening’ certain parts of algorithm, other parts would make up for the shortening ), but when I looked at the run times, I think that the processor takes up more time on parts of the algorithm then the other such as removing from a queue takes longer than inserting onto a queue.