Paul Turner

Github link : <https://github.com/Paul-Kyle-Turner/LinearTimeCounting_and_RandomSelect>

Exercise 8.2-4 :

In file : main\_count\_select.py

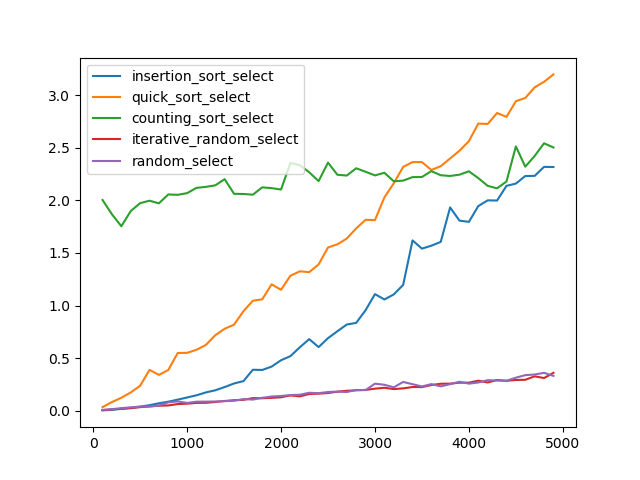
The counting sort algorithm completes the number between range problem in the second loop of the algorithm depicted in the book. This loop sums the amount of objects in each index of the array and therefore delivers the range between [a..b] by number\_of\_elements = c[b] – c[a] where c is the count array. This can be modified slightly to be exclusive by number\_of\_elements = c[b-1] – c[a+1].

Exercise 9.2-3 :

In file : main.py

Five different algorithms were tested for selecting the ith smallest integer in an array. An iterative version of the random select function and the recursive version of the iterative select show a linear time with little overhead. The recursive version only does a single recursive call making it similar to the iterative version and take a similar amount of time. The other three algorithms were sorting algorithms that then chose the ith index in the sorted array. Insertion shows an n^2 operation time over a reasonable n. Counting sort at linear time proves to be similar to the linear time needed with the random selection methods however the overhead of requiring two other arrays and the added processing time k makes it considerably slower. Quick sort takes a longer time than expected being an average n log(n) operation time.

Sources of error have been mitigated by running each algorithm on pythons timeit class. For each n each function was called five times with different n data sets and then run one hundred times for each data set. The times were then averaged for each n to get the general running time for that n.

 Figure one – All five algorithms plotted starting at 100 and repeating by 100 until 5000.

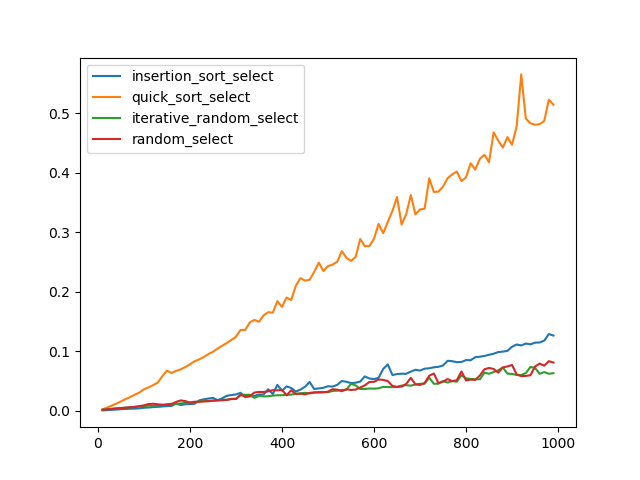
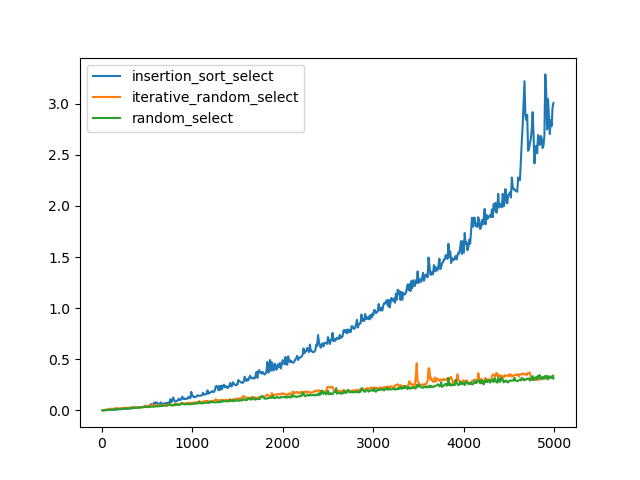
Figure 2 – Iterative random select and Random select plot against insertion sort. Plotted starting at 10 and repeating by 10 until 5000

Figure 3 – Algorithms except counting sort plotted from 10 to 1000 repeating every 10.