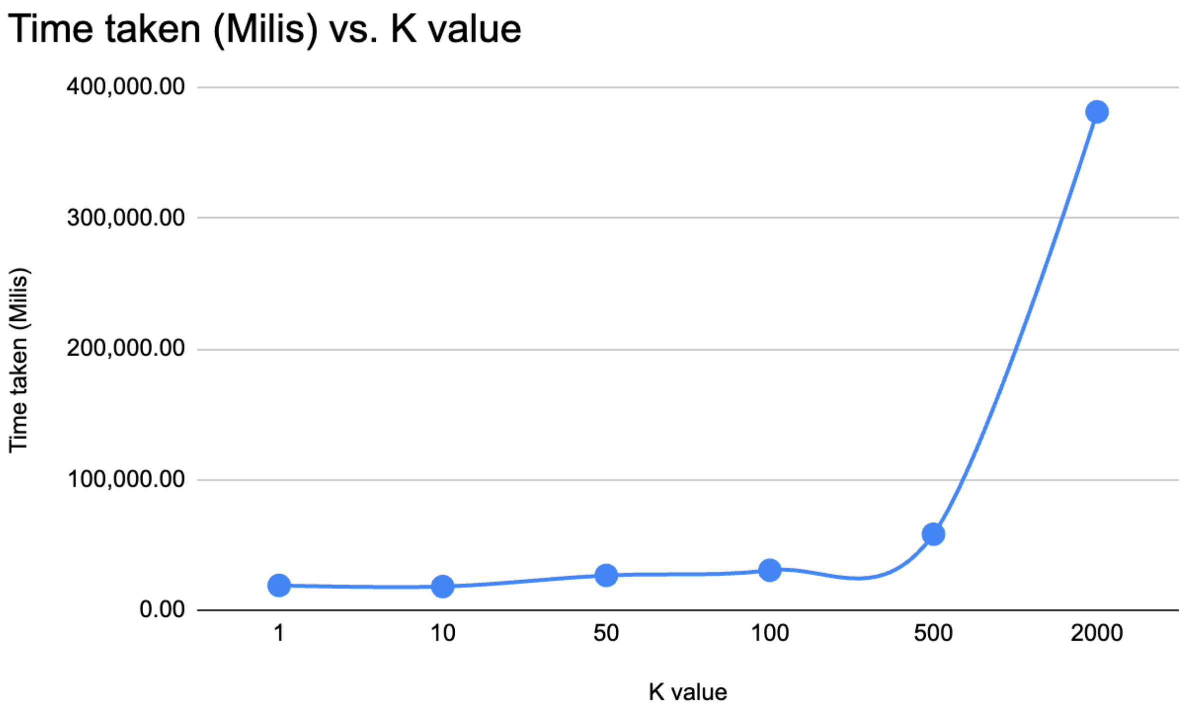
Matin hassanzadeh Mobini
StudentID: 300283864

CSI Assingment 1

Run times:

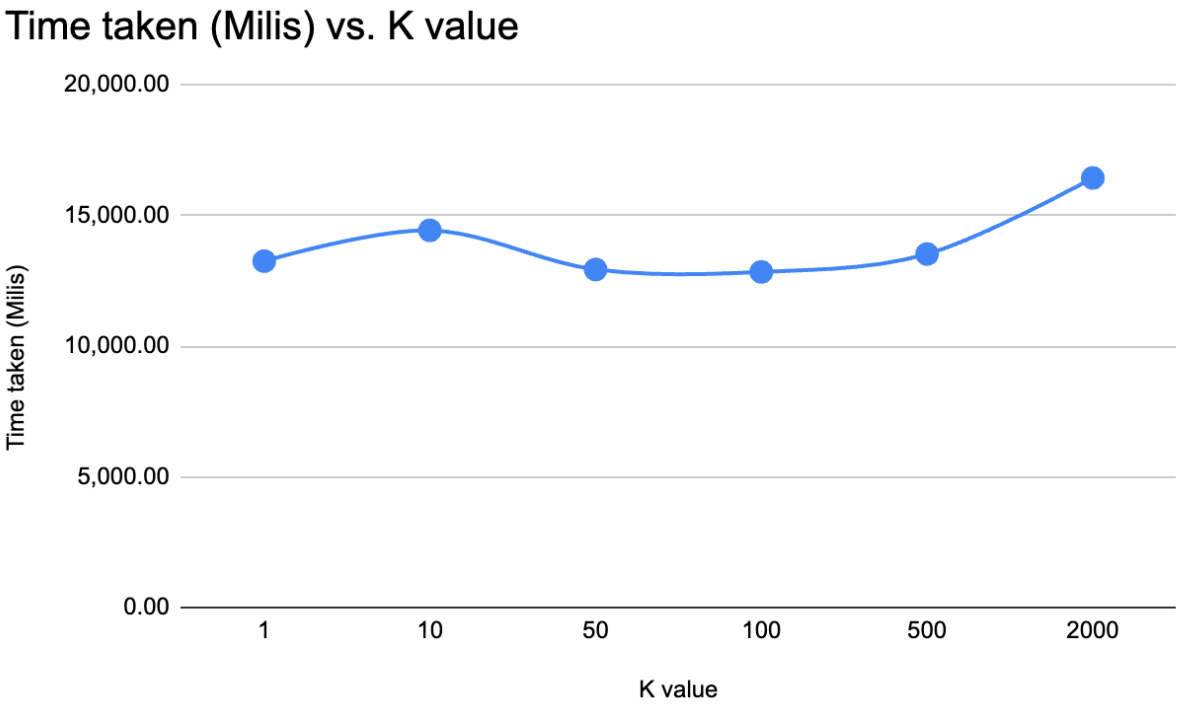
Priority Queue 1:

K value	Time taken (Milis)
1	18,958.0
10	18,089.0
50	26,632.0
100	30,555.0
500	58,184.0
2000	381,431.0



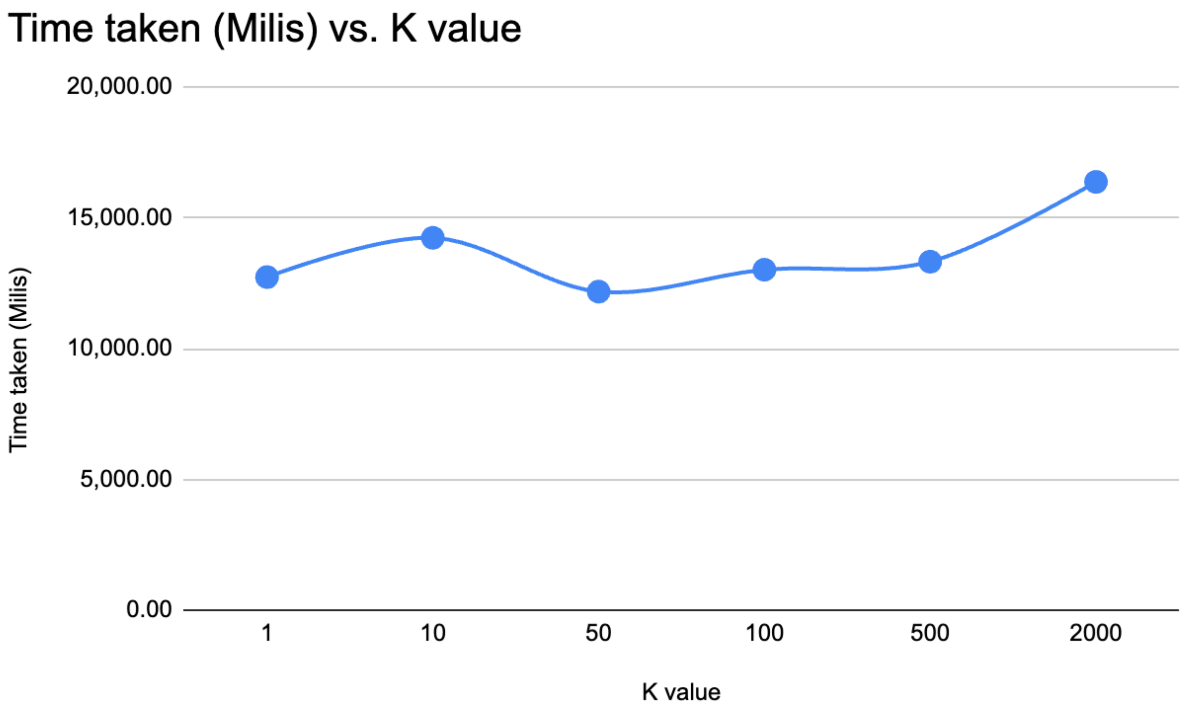
Priority Queue 2:

K value	Time taken
1	13,253.0
10	14,431.0
50	12,934.0
100	12,839.0
500	13,525.0
2000	16,436.0



Priority Queue 3:

K value	Time taken
1	12,736.0
10	14,239.0
50	12,180.0
100	13,018.0
500	13,326.0
2000	16,377.0



Results:

Priority Queue 1: For the Priority Queue 1 i have used an Arraylist to store the query points, the points in the ArrayList are in ascending order(index 0 = lowest distance in the set). With the K value inputted into the command line the program uses that variable to appoint the size of the number of nearest values(distance) to the each query point, as the K value increases so does the output size, seeing how more items are appointed to a certain set of NearestNeighbours for a specific query point. The offer method for this Priority Queue implementaion has a Big Oh of O(K). This implementation is the slowest from the three PriorityQueue implementations.

Priority Queue 2: For the Priority Queue 2 i have also used an Arraylist to store the query points, the points in the ArrayList are in descending order(index 0 = highest distance in the set). With the K value inputted into the command line the program uses that variable to appoint the size of the number of nearest values(distance) to the each query point, as the K value increases so does the output size, seeing how more items are appointed to a certain set of NearestNeighbours for a specific query point. The offer method for this Priority Queue implementaion has a Big Oh of O(log(K)). This implementation is the second fastest from the three PriorityQueue implementations, Priority Queue 2 and 3 share similar execution speeds.

Priority Queue 3: For the Priority Queue 3 i have imported the PriorityQueue library and used a PriorityQueue to store the query points, the points in the PQ are in descending order(index 0 = highest distance in the set). With the K value inputted into the command line the program uses that variable to appoint the size of the number of nearest values(distance) to the each query point, as the K value increases so does the output size, seeing how more items are appointed to a certain set of NearestNeighbours for a specific query point. The offer method for this Priority Queue implementaion has a Big Oh of O(K). This implementation is the fastest from the three PriorityQueue implementations, but Priority Queue 2 and 3 share similar execution speeds.