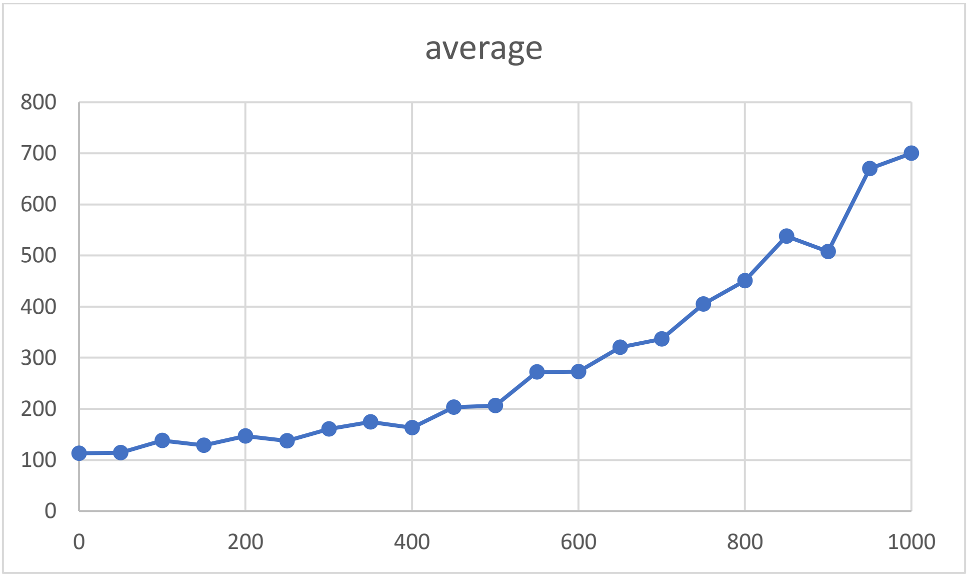
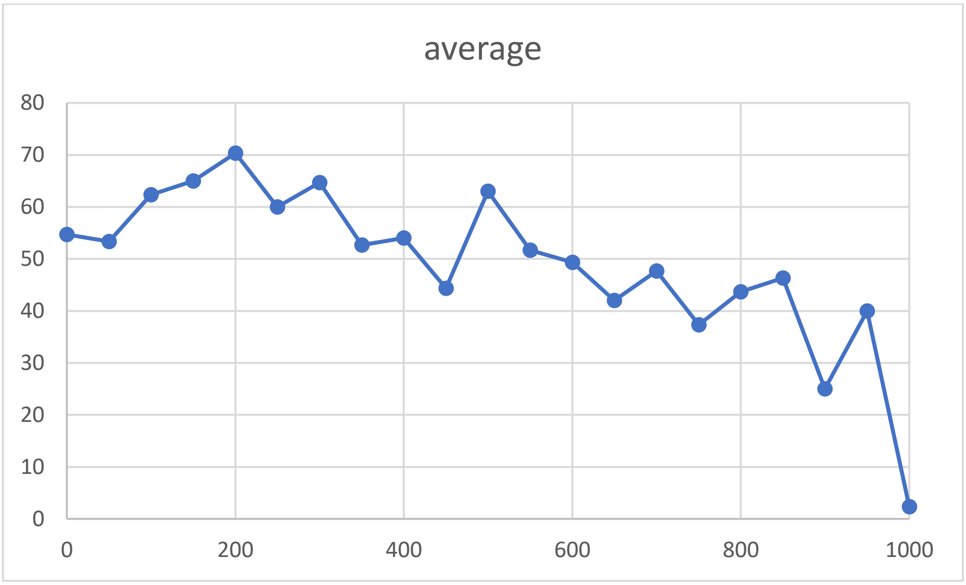
P4.1

b)

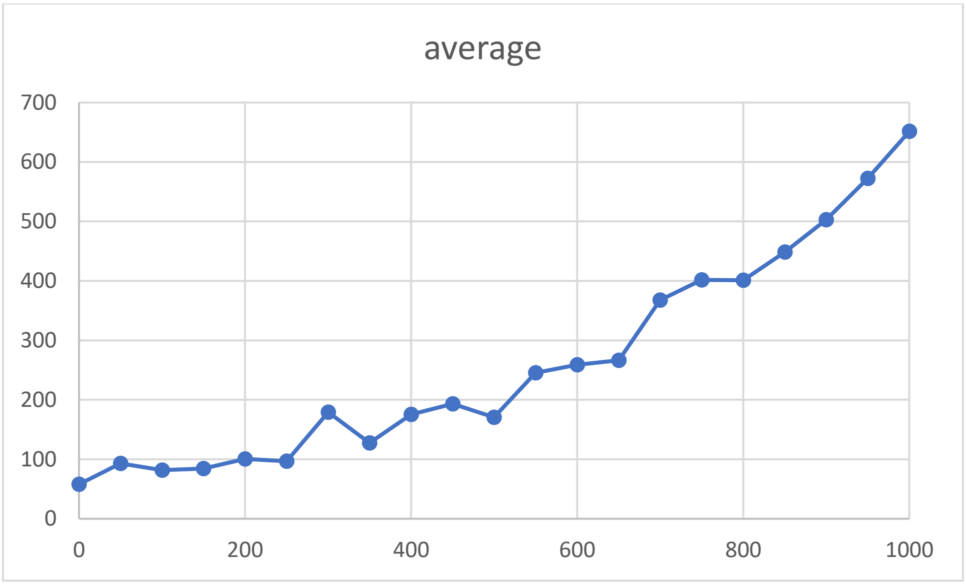
Random order set of 1000 integers with k going from 0 to 1000 in steps of 50(average of 3 different sets):



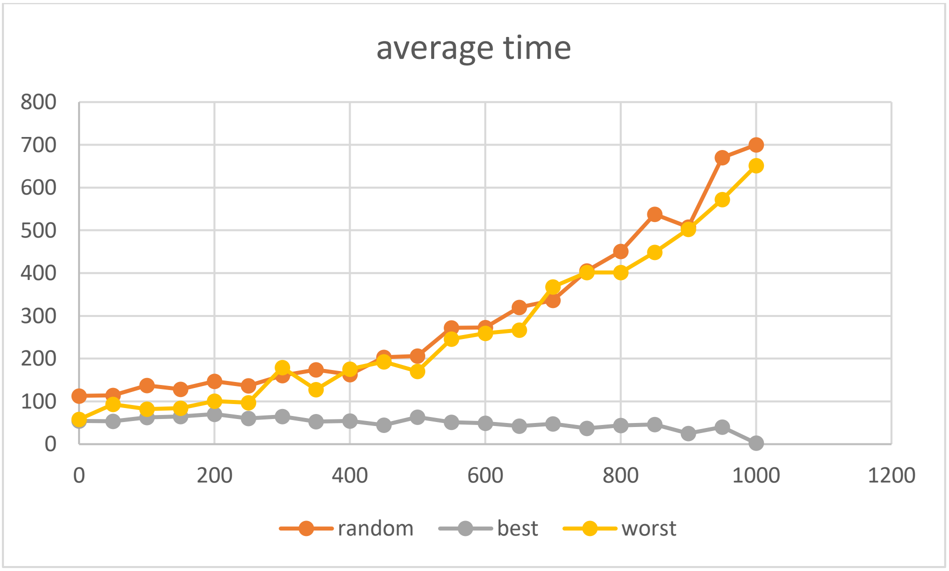
Already sorted array of 1000 integers:



Worst case array for merge sort of 1000 integers (generated with code from here https://www.geeksforgeeks.org/find-a-permutation-that-causes-worst-case-of-merge-sort/)



All in one plot:



C and D)

From my data I cannot really see any benefit in using this method. For the best case, the time it takes to sort the array is basically constant. For the worst and random case, the time seems to increase exponentially with larger k values. In general, this makes sense since the average and the worst-case time complexity of merge sort are O(log(n)). Therefore, it should be faster for most cases. Since this means using this method has a negative impact on the time it takes to sort the array for most cases, I would choose k to be 0. This means I would just use the normal merge sort algorithm.