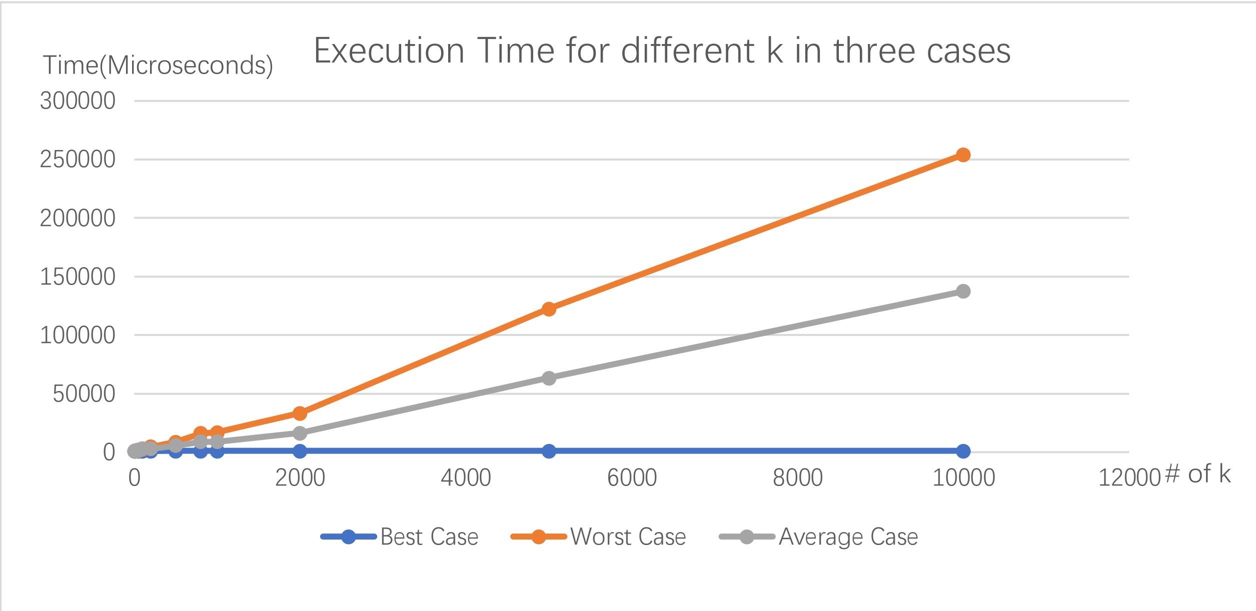
Problem 4.1 Merge Sort

a)

merge sort.cpp

b)

	Time in Microseconds												
n=10000	Best Case				Worst Case				Average Case				
k	trial 1	trial 2	trail 3	Average	trial 1	trial 2	trail 3	Average	trial 1	trial 2	trail 3	Average
5	1000	1002	1000	1000.667	1001	1000	999	1000	1000	999	1000	999.6667
10	1000	999	999	999.3333	1001	999	1001	1000.333	999	1000	1999	1332.667
50	1000	999	999	999.3333	2001	2000	2001	2000.667	2998	999	1001	1666
100	999	998	997	998	3000	2000	3002	2667.333	3000	4000	1999	2999.667
200	994	980	999	991	5002	4001	5001	4668	2000	3000	4000	3000
500	999	999	999	999	9000	9002	8001	8667.667	5000	5002	7001	5667.667
800	999	1000	1002	1000.333	16001	16000	16002	16001	9001	10000	8001	9000.667
1000	1001	999	1000	1000	18002	15999	17002	17001	11003	7999	8002	9001.333
2000	999	1000	999	999.3333	31003	36002	33002	33335.67	16002	17002	16000	16334.67
5000	1000	1001	998	999.6667	121011	124009	122025	122348.3	62007	62005	66004	63338.67
10000	986	1005	1000	997	270311	251019	241020	254116.7	123008	162001	127011	137340



c)

The variant k represents how many elements need to be sorted using Insertion Sort before Merge Sort.

For the Best Case, whatever k is changed into, the time complexity does not change because it is sorted already, no need to go through any sort process.

For the Worst and Average cases, the higher k is, the more time it takes for the algorithm to execute. This is due to the number of elements being sorted using Insertion Sort, which is slower than Merge Sort, therefore, as k increases, less elements are sorted using Merge Sort which leads to more time.

Thus, as k goes bigger, the asymptotic time complexity is going to be more like Insertion Sort instead of Merge Sort.

Problem 4.2

a)

$T(n) = 36T(n/6) + 2n$

$n^{\log_b a} = n^{\log_6 36} = n^2$

$f(n) = 2n$

Case 1: $f(n) = O(n^{2-\epsilon})$ for $\epsilon = 1$

Thus, $T(n) = \Theta(n^2)$

b)

$T(n) = 5T(n/3) + 17n^{1.2}$

$n^{\log_b a} = n^{\log_3 5} = n^{1.46}$

$f(n) = 17n^{1.2}$

~~Case 3:~~ $\cancel{f(n) = \Omega(n^{1.46+\epsilon})}$ ~~for~~

Case 1: $f(n) = O(n^{1.46-\epsilon})$ for $\epsilon = 0.26$

Thus $T(n) = \cancel{\Theta(n^{1.46})}\ \Theta(n^{\log_3 5})$

c)

$T(n) = 12T(n/2) + n^2 \lg n$

$n^{\log_b a} = n^{\log_2 12} = n^{3.58}$

$f(n) = n^2 \lg n$

Case 1: $f(n) = O(n^{3.58-\epsilon})$ for some ϵ (at least 1.38)

Thus, $T(n) = \Theta(n^{\log_2 12})$

d)

According to the recursion tree, 2^n has occupied the entire sequence.

Thus this answer is T(n)= θ(2^n)