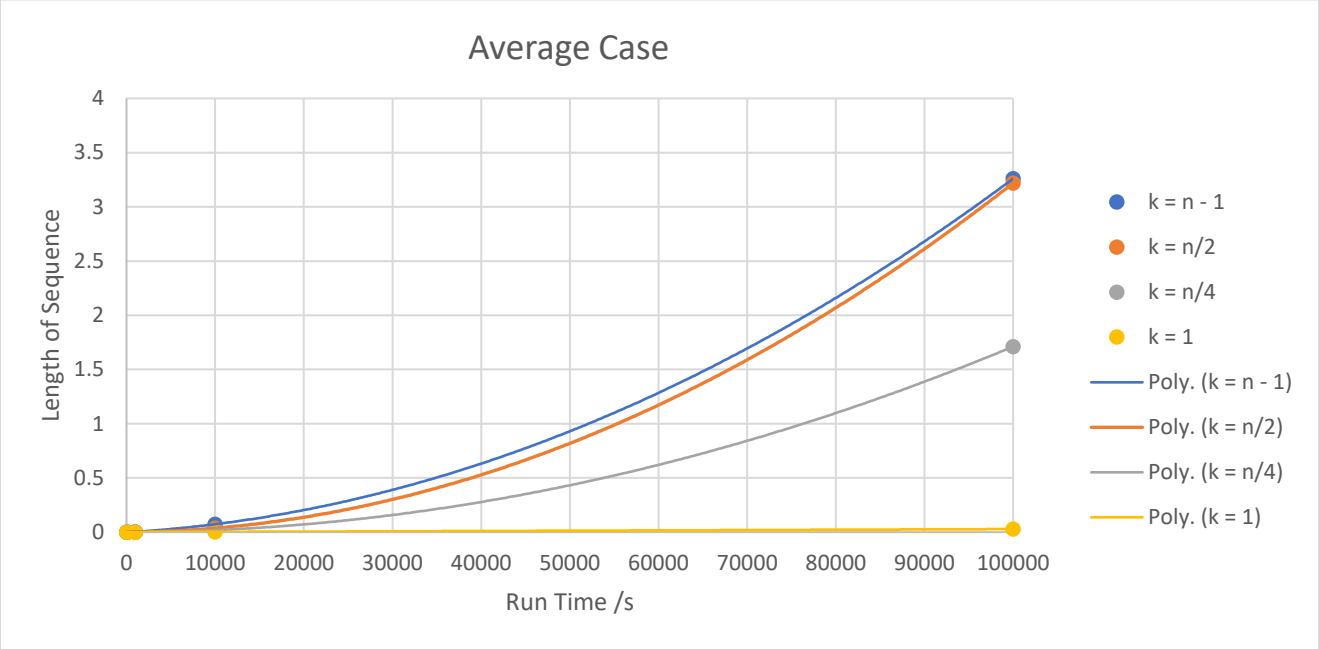
Problem 1: Merge Sort

a) *Please find c file attached for merge sort implementation.*

b) See excel spreadsheet.

c) The sequences that were plotted had lengths of 10, 100, 1000, 10000 and 100000.



Below are the functions for the average case of sequences of varying length.

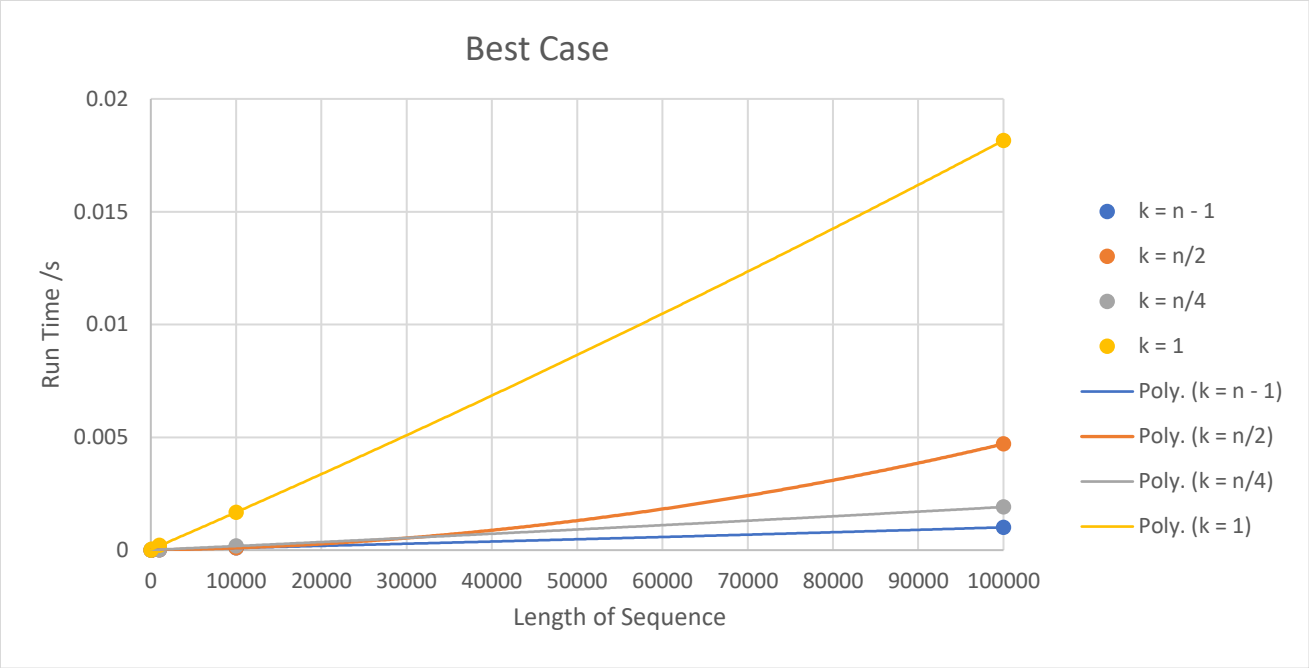
$for\ k = 1, \quad f(n) = 2 \times 10^{-13}n^2 + 3 \times 10^{-7}n - 6 \times 10^{-6}$

$for\ k = \frac{n}{4}, \quad f(n) = 2 \times 10^{-10}n^2 + 2 \times 10^{-7}n - 3 \times 10^{-5}$

$for\ k = \frac{n}{2}, \quad f(n) = 3 \times 10^{-10}n^2 + 6 \times 10^{-7}n - 0.0002$

$for\ k = n - 1, f(n) = 3 \times 10^{-10}n^2 + 5 \times 10^{-6}n - 0.0015$

For the average case, increasing values of k causes an increase in the computation times of the algorithms. This is due to the fact that insertion sort has a heavier cost than merge sort for a randomly arranged array.



Below are the functions for the best case of sequences of varying length.

$for\ k = 1, \quad f(n) = 2 \times 10^{-13}n^2 + 2 \times 10^{-7}n + 3 \times 10^{-5}$

$for\ k = \frac{n}{4}, \quad f(n) = 2 \times 10^{-14}n^2 + 2 \times 10^{-8}n - 7 \times 10^{-6}$

$for\ k = \frac{n}{2}, \quad f(n) = 4 \times 10^{-13}n^2 + 5 \times 10^{-9}n + 5 \times 10^{-6}$

$for\ k = n - 1, f(n) = 1 \times 10^{-14}n^2 + 9 \times 10^{-9}n + 5 \times 10^{-6}$

For the best case (already sorted array), increasing values of k causes a decrease in the computation time of the algorithm. This is because in insertion sort the current element is compared with the previous. The if statement means that if the previous element is smaller the algorithm moves to the next element. So once the insertion sort has traversed the entire array, the only operation that will have been carried out is a comparison of 2 numbers. How ever as merge sort splits the array into subarrays without checking if the sequence is already sorted, the cost is much heavier for merge sort in the best case.

Below are the functions for the worst case of sequences of varying length.

$for\ k = 1, \quad f(n) = 4 \times 10^{-13}n^2 + 2 \times 10^{-7}n + 4 \times 10^{-5}$

$for\ k = \frac{n}{4}, \quad f(n) = 3 \times 10^{-10}n^2 + 4 \times 10^{-7}n + 0.0001$

$for\ k = \frac{n}{2}, \quad f(n) = 6 \times 10^{-10}n^2 + 6 \times 10^{-7}n - 0.0002$

$for\ k = n - 1, f(n) = 6 \times 10^{-10}n^2 + 6 \times 10^{-7}n + 0.0003$

For the worst case(sorted in decreasing order), an increase in the value of k increases the run time of the algorithm. The same logic holds for the average case, as insertion sort has a higher cost than merge sort

d) In practice I would chose a large k when the sequence to be sorted is generally in increasing order as less operations are performed when comparisons stop the algorithm from entering a loop in the first place. Otherwise, small values of k are always preferred.

Problem 2

$a)\ T(n) = 36T\left(\frac{n}{6}\right) + 2n$

$Using\ the\ master\ method$:
$a = 36, b = 6$
$n^{\log_b a} = n^2$
$f(n) = 2n$
$\therefore Case\ 1: f(n) = O(n^{2-\varepsilon})\ for\ \varepsilon = 1$
$\therefore T(n) = \Theta(n^2)$

$b)\ T(n) = 5T\left(\frac{n}{3}\right) + 17n^{1.2}$

$Using\ master\ method$:
$a = 5, b = 3$
$n^{\log_b a} = n^{1.4649735\ldots}$
$f(n) = 17n^{1.2}$
$\therefore Case\ 1: f(n) = O(n^{1.464\ldots-\varepsilon})\ for\ \varepsilon = 0.26497\ \ldots$
$\therefore T(n) = \Theta\left(n^{\log_3 5}\right)$

$c)\ T(n) = 12T\left(\frac{n}{2}\right) + n^2 logn$

$Using\ master\ method$:

$a = 12, b = 2$

$n^{\log_b a} = n^{3.5849625\ldots}$

$f(n) = n^2 logn$

$\therefore Case\ 3: f(n) = \Omega\left(n^{\log_2 12-\varepsilon}\right)\ for\ \varepsilon =$

$$T(n) = 12T\left(\frac{n}{2}\right) + n^2 logn$$

$$= 144T\left(\frac{n}{2^2}\right) + 12\frac{n^{\,2}}{2^2}\log\left(\frac{n}{2}\right) + n^2 \log n$$

$$= 144T\left(\frac{n}{2^2}\right) + 3n^2 \log\left(\frac{n}{2}\right) + n^2 logn$$

$$\ldots$$

$$= n^2 logn + 3n^2 \log\left(\frac{n}{2}\right) + 3n^2 \log\left(\frac{n}{4}\right) + \cdots + 3n^2 \log\left(\frac{n}{2^{logn}}\right)$$

$$= n^2 + \frac{n^2(logn+1)}{2} logn$$

$$T(n) = \Theta(n^2 \log^2 n)$$

$d)\ T(n) = 3T\left(\frac{n}{5}\right) + T\left(\frac{n}{2}\right) + 2^n$

$Using\ master's\ theorem$:

$Case\ 3: (regularity\ condition\ holds)$

$\therefore T(n) = \Theta(2^n)$

$e)\ T(n) = T\left(\frac{2n}{5}\right) + T\left(\frac{3n}{5}\right) + \Theta(n)$

$Master\ method\ does\ not\ apply. Using\ a\ recursion\ tree. At\ level\ each\ level, the\ weight\ of\ computation\ is\ \theta(n). The\ number\ of\ levels\ is\ \log_{3/5} n = \theta(logn)$. Guess $T(n) = \theta(nlogn)$ and use substitute method to confirm the estimate.

- Show $T(n) = \theta(n)$.
  Base case: choose large d_1 such that $T(n) < d_1 nlogn$.
  Induction step: assume for all $k < n, T(k) < d_1 nlogn$
  $T(n) \leq T\left(\frac{2n}{5}\right) + T\left(\frac{3n}{5}\right) + c_1 n$
  $\leq d_1 \frac{2n}{5} \log\left(\frac{2n}{5}\right) + d_1 \frac{3n}{5} \log\left(\frac{3n}{5}\right) + c_1 n$
  For $d_1 > \frac{5c_1}{2log5 + 3\log\left(\frac{5}{3}\right)}$,we have shown by induction that $T(n) = O(nlogn)$.
  $\therefore T(n) = \Omega(nlogn)$