Problem 2.1

a) The source code for this problem is in 2-1.c file.

b) Time complexity comparison for different k

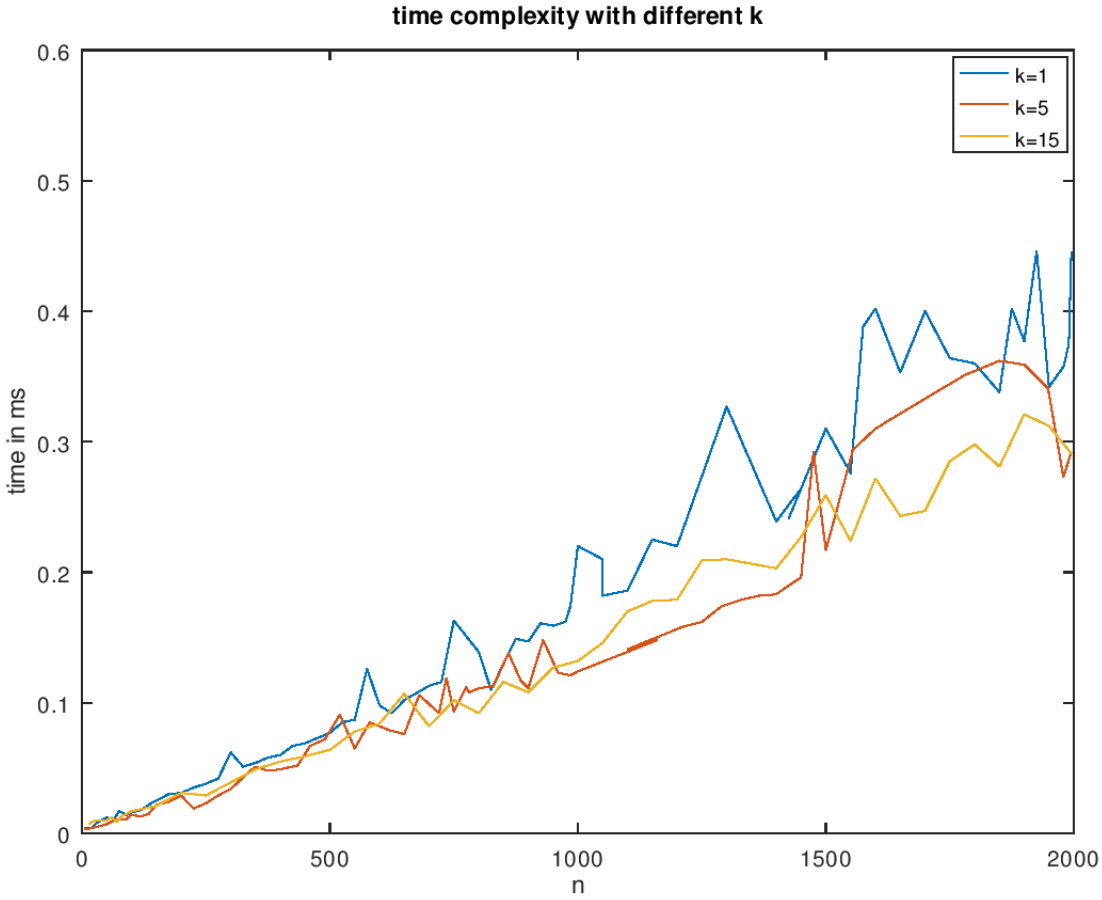


Figure 1

Time taken for different values of n with k=1, k=5 and k=15 are measured. The implementation of the code is in 2-2.c file in the zip.
The random time taken for different values of n and k are measured by several times, and the average time measurement is plotted in the figure above.
According to Figure 1, the algorithm takes less time with greater n and k.

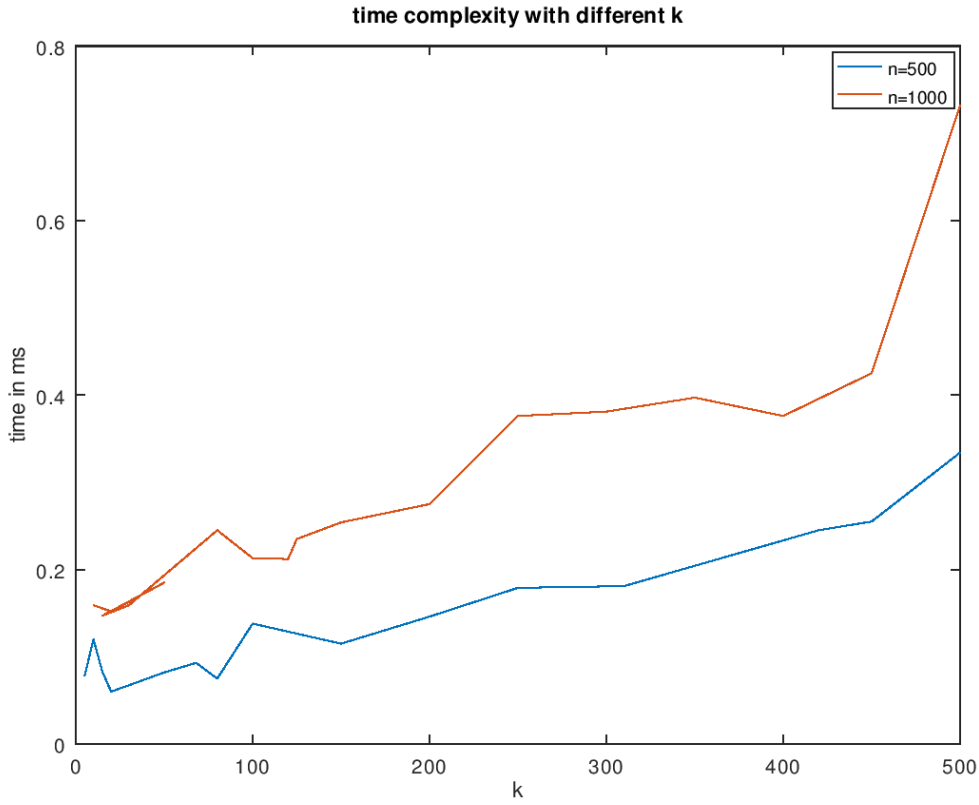


Figure 2

However, in Figure 2, starting from k being around 20, both arrays take longer time to finish sorting for increasing k. Hence, it is good to use large k values for arrays with large sizes for this algorithm.

c) The best case, worst case, average case with different values of n and k are measured in the following figures. The full code implementation is in 2-3.c file.

The random time taken for different values of n and k are measured by several times, and the average, best, and worst case time measurement is plotted in the figure above. The final time measurements for both b) and c) are collected in k1.txt, k5.txt and k15.txt files.

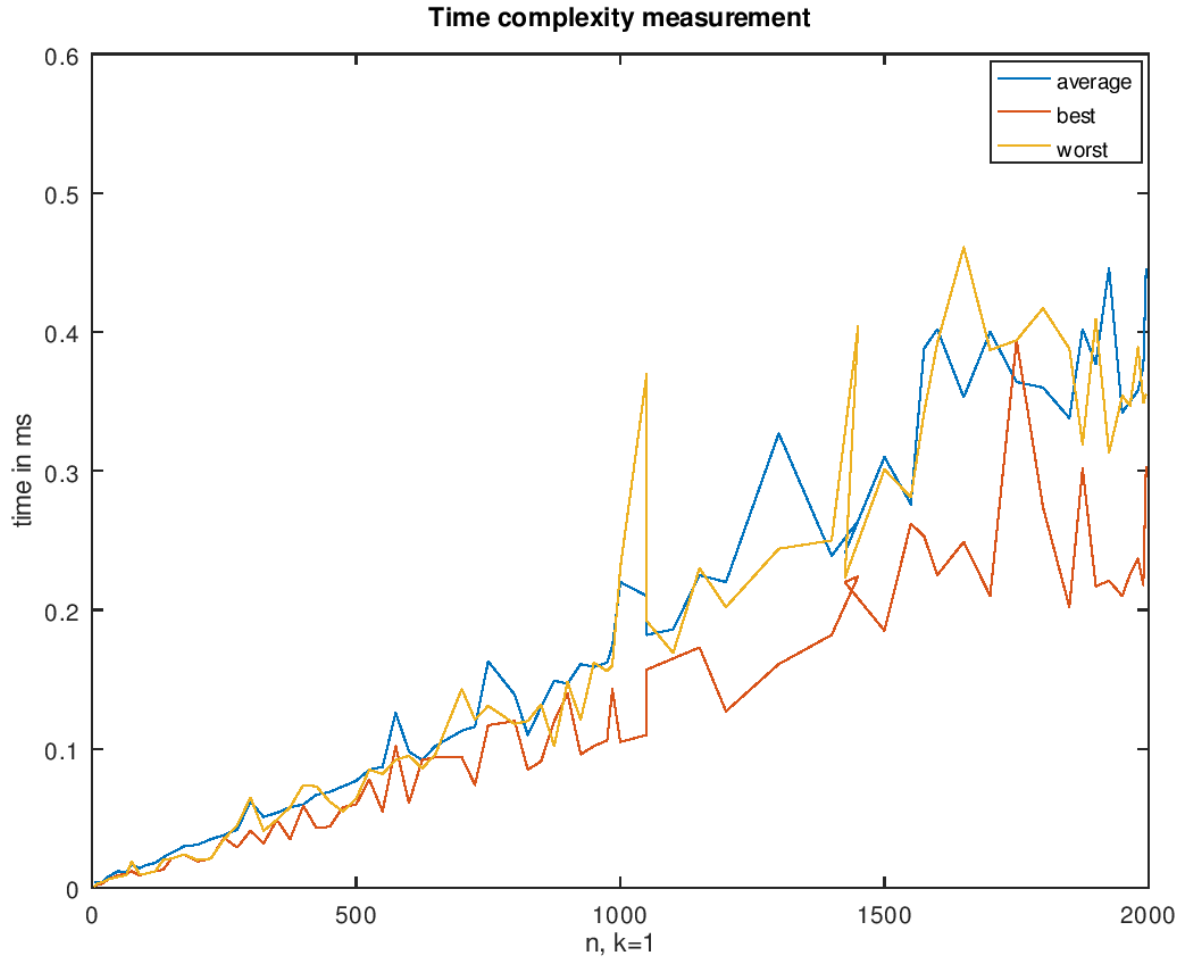


Figure 3

k=1
The full figure is in 1c.png file in the zip.

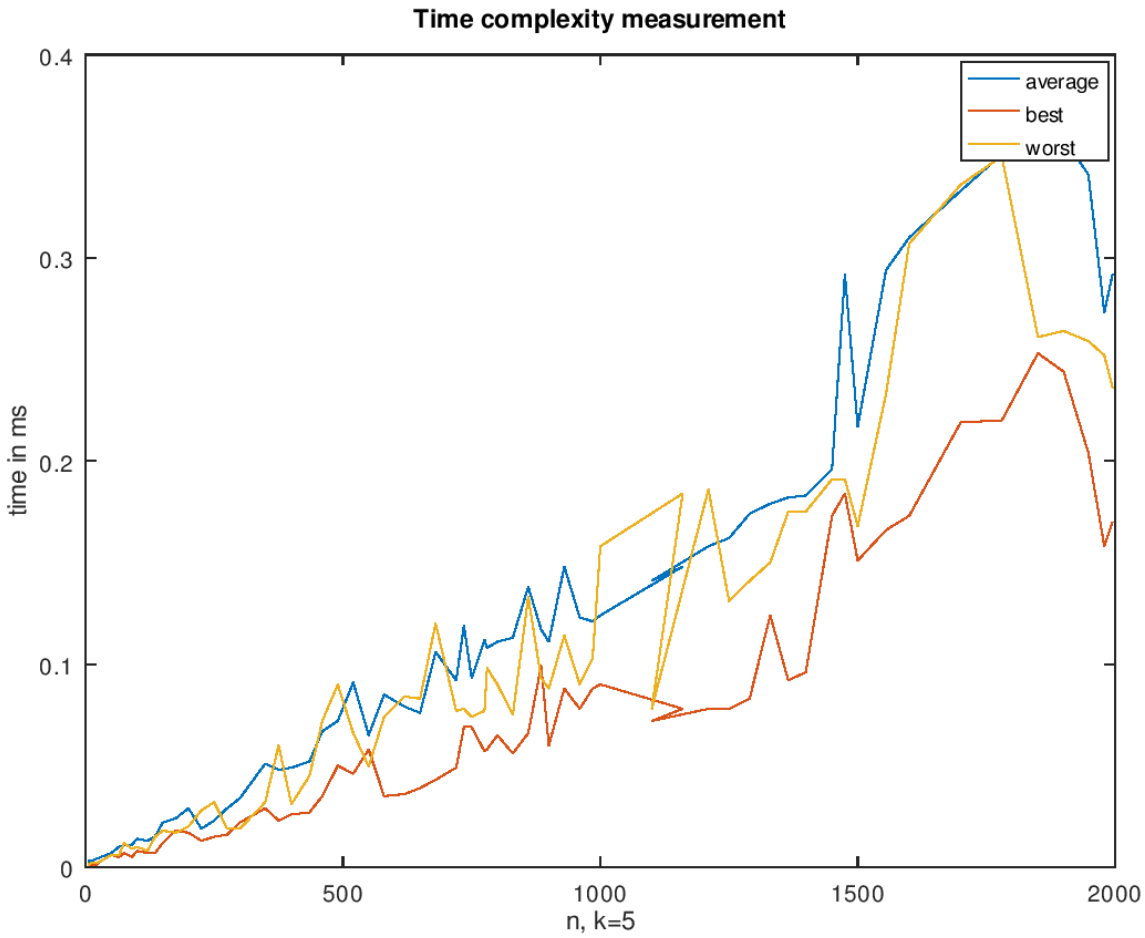


Figure 4

k=5

The full figure is in 5c.png file in the zip.

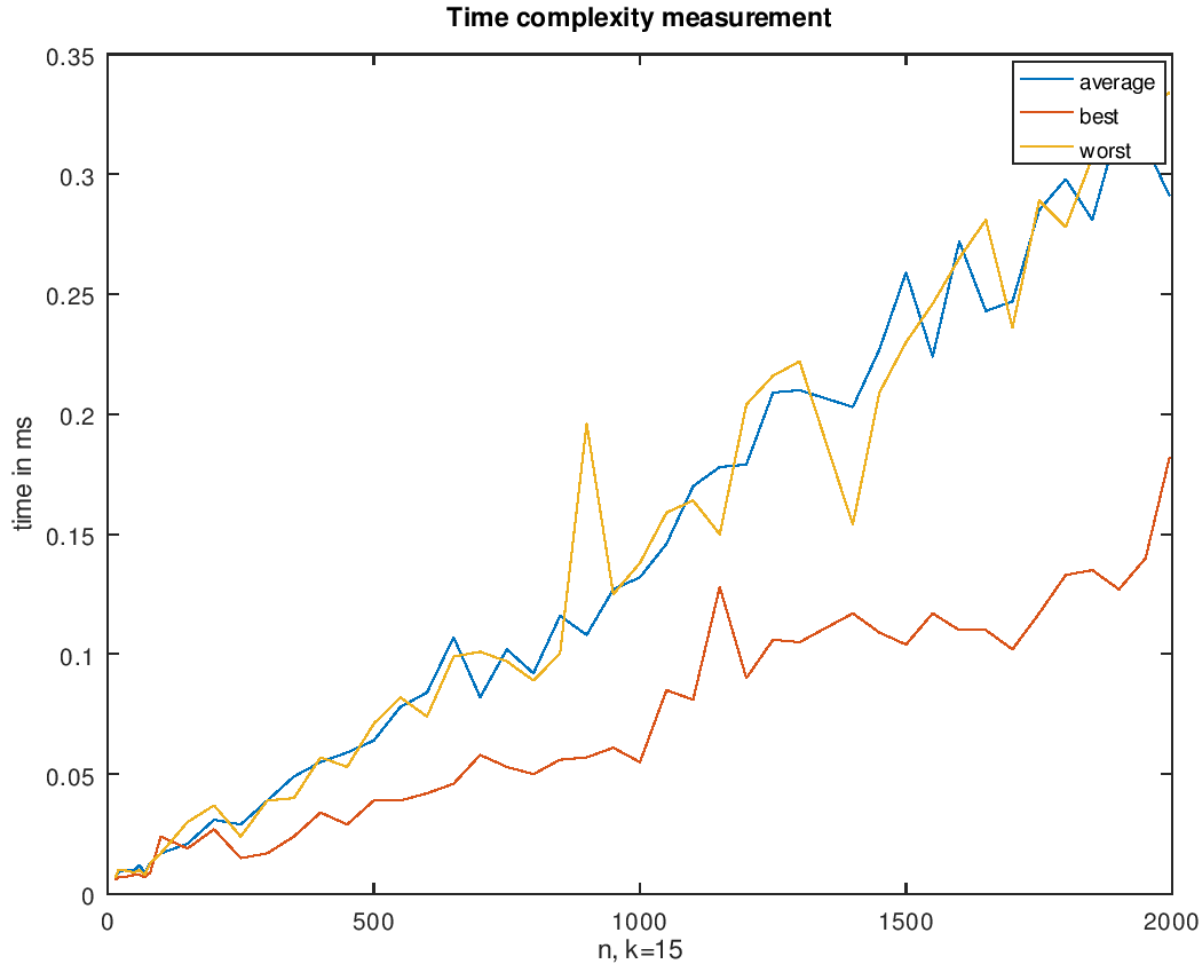


Figure 5

k=15

The full figure is in 15c.png file in the zip.

Time Complexity	Insertion Sort	Merge Sort
Average Case	$O(n^2)$	O(n log(n))
Worst Case	$O(n^2)$	O(n log(n))
Best Case	O(n)	O(n log(n))

According to Figure 3,4, and 5, the best case graph is below the average case and worst case graphs. However, the time taken for average case are almost the same with the worst case at some values of n, lower than worst case at some n, and becomes much larger than the worst case. The time taken growth rate of the best case tends to become lower as the size of the array (n) becomes larger. However, the time taken for the rest of two cases tend to become higher with higher values of n with same k.

Moreover, according to those figures, if n becomes too large and k is small, the time complexity of average case and worst case will be near to $O(n^2)$ as insertion sort dominates. When both n and k are small, the time complexity and average case and worst case are near to O(n log(n)) as merge sort dominates.

d) According to test cases found, if k is too small and n is too large, the majority of elements in the array list will be sorted by merge sort. Also, if k is too large and n is too small, the majority of elements will be sorted by insertion sort in this algorithm.

According to the figures in c), the time taken for all three cases with small values of n are similar as most of the elements will be sorted by merge sort. When n becomes large, the plot graphs for worst case and average case are near(lower) to normal n^2 graphs. Also, the time taken for all cases with k=15(Figure 5) are lower than those for k=5 (Figure 4).
However, the growth of time taken rate becomes significant if k is around 20. The time taken for same n is in the descending order between k=1 and around 20, and increases afterwards. Hence, it takes the least time if k is around or below 20.

Problem 2.2

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

$T(n) = aT(n/b) + F(n)$

$F(n) = 2n,\ a = 36,\ b = 6$

$n^{\log_b a} = n^{\log_6 36} = n^{\log_6 6^2} = n^{2\log_6 6} = n^2$ <Master Method>

If $\epsilon = 0.5 > 0$, $n^{\log_b a - \epsilon} = n^{1.5}$

$0 \le F(n) \le cn^{1.5}$ if $c = 5 > 0$

$0 \le F(n) \le 5n^{\log_b a - \epsilon}$, $F(n) \in O(n^{\log_b a - \epsilon})$

$\therefore T(n) = \Theta(n^{\log_6 36}) = \Theta(n^2)$ <case 1>

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

$T(n) = aT(n/b) + F(n)$

$f(n) = 17n^{1.2},\ a = 5,\ b = 3$

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

If $\epsilon = 0.2$, $n^{\log_b a - \epsilon} \approx n^{1.26}$

$0 \le 17n^{1.2} \le 20n^{1.26}$, if $c = 20 > 0$

$0 \le F(n) \le 20n^{\log_b a - \epsilon}$, $f(n) = O(n^{\log_b a - \epsilon})$

$\therefore T(n) = \Theta(n^{\log_3 5})$ <Master Method Case 1>

$0 \leq F(n) \leq 20 n^{\log_6 a - \epsilon}$, $F(n) = O(n^{\log_6 a - \epsilon})$

$\therefore T(n) = \Theta(n^{\log_3 5})$ < Master Method Case 1 >

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

$T(n) = aT(n/b) + F(n)$

$F(n) = n^2 \log_2 n$, $a = 12, b = 2$

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

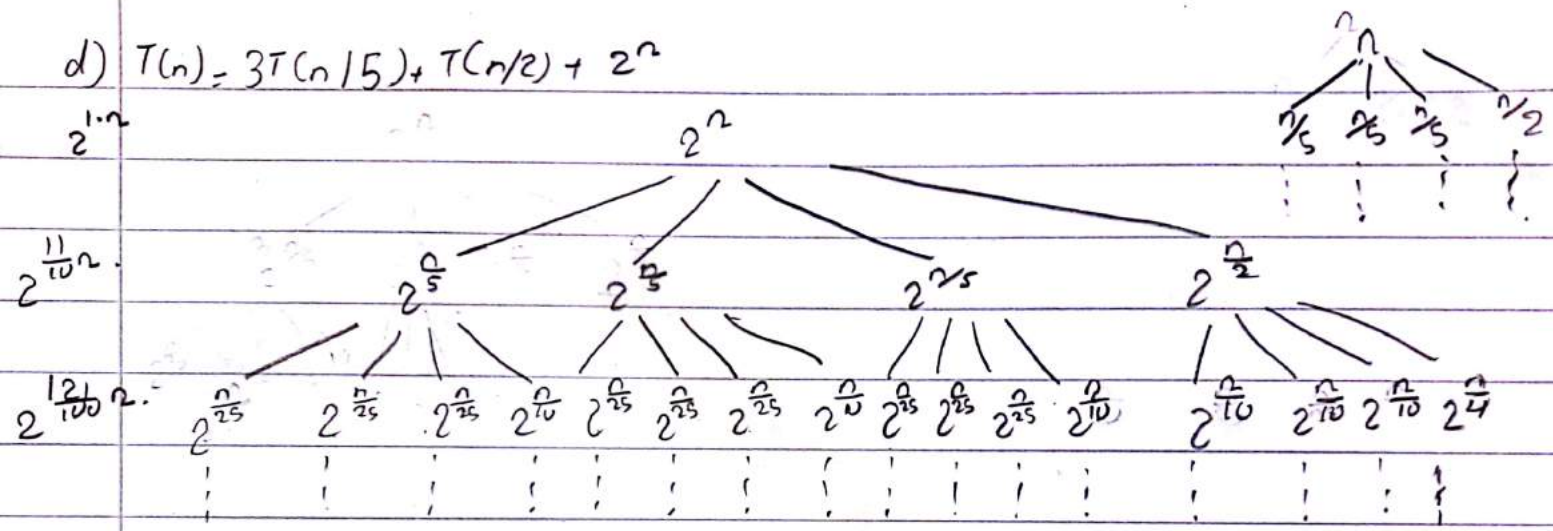
IF $\epsilon = 0.58$, $n^{\log_b a - \epsilon} = n^3$

$$\lim_{n \to \infty} \frac{n^2 \lg n}{n^{\log_b a - \epsilon}} = \lim_{n \to \infty} \frac{n^2 \lg n}{n^3} = \lim_{n \to \infty} \frac{\lg n}{n} = 0$$

$F(n) = O(n^{\log_b a - \epsilon}) \rightarrow F(n) = O(n^{\log_b a - \epsilon})$ < Master method case 1)

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

d) $T(n) = 3T(n/5) + T(n/2) + 2^n$



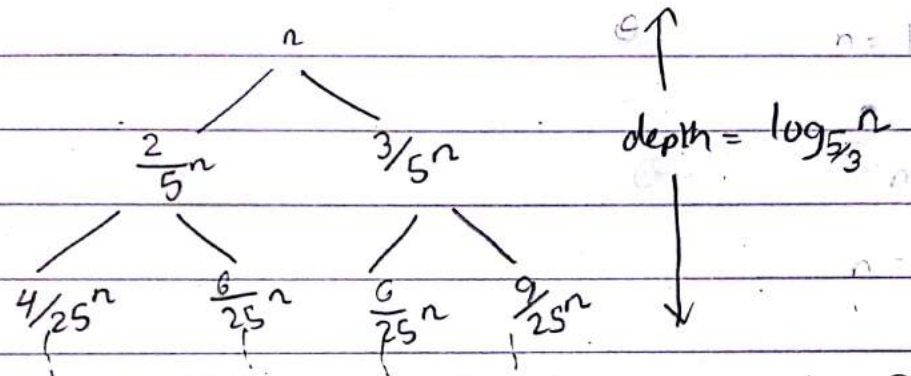
In this function, 2^n dominates, and $2^{n/2}$ is the largest value

$$\Sigma = 2^n + 2^{\frac{11}{10}n} + 2^{\frac{121}{100}n} + \cdots$$

$$\Sigma = n + \frac{11}{10}n + \frac{121}{100}n + \cdots = n\left[1 + \frac{11}{10} + \frac{121}{100} + \cdots\right] = n\left[1 + \left(\frac{11}{10}\right)^1 + \left(\frac{11}{10}\right)^2 + \cdots\right]$$

Thus, the total sum is in the geometric series. So, $T(n) = \Theta(n^2)$

e) $T(n) = T(2n/5) + T(3n/5) + \Theta(n)$



At each level, there are n operations, and $\frac{3}{5}$ is the largest number to be found in the tree. Thus, $T(n) = \Theta(n \log n)$