Question 2. (30 Points)

If we sort the Y array at each recursive call, we use mergesort at each call, which means adding a time of $n\log(n)$ for each mergesort call. We also assume $T(3)=1$ for the base case.

1) $t(n) = 2T(\frac{n}{2}) + n\log(n) + cn + d$

2) $t(n) = 2(2T(\frac{n}{4}) + \frac{n}{2}\log(\frac{n}{2}) + c(\frac{n}{2}) + d) + n\log(n) + cn + d$

$= 4T(\frac{n}{4}) + n\log(\frac{n}{2}) + 2\cdot cn + d$

3) $t(n) = 4(2T(\frac{n}{8}) + \frac{n}{4}\log(\frac{n}{4}) + c(\frac{n}{4}) + d) + n\log(\frac{n}{2}) + 2cn + 3d$

$= 8T(\frac{n}{8}) + n\log(\frac{n}{4}) + n\log(\frac{n}{2}) + n\log(n) + 3cn + 7d$

$\vdots$

k) $t(n) = 2^k T(\frac{n}{2^k}) + n\sum_{i=1}^{k} n\log(\frac{n}{2^{i-1}}) + kcn + d(2^k - 1)$

Now, we can write the summation $\sum_{i=1}^{k} n\log(\frac{n}{2^{i-1}})$ as:

$$\sum_{i=1}^{k} n\log(\frac{n}{2^{i-1}}) = \sum_{i=1}^{k} n\log(n) - \sum_{i=1}^{k} n\log(2^{i-1})$$

$$= k\log(n) - \log(2)\sum_{i=1}^{k} i-1 = k\log(n) - \frac{\log(2)}{2}(k^2-k)$$

So therefore: $T(n) = 2^k T(\frac{n}{2^k}) + n(k\log(n) - \frac{\log(2)(k^2-k)}{2} + kc) + (2^k-1)d$

Recursion will finish when $\frac{n}{2^k} = 3 \Rightarrow \frac{n}{3} = 2^k \Rightarrow \log_2(\frac{n}{3}) = k$

Now, getting rid of k, we get:

$$T(n) = \frac{n}{3}T(3) + n(\cancel{\log(\frac{n}{3})}\log(n) - \frac{\log(2)(\cancel{2\log(\frac{n}{3}) - \log(\frac{n}{3})})}{2} + c(\cancel{\log(\frac{n}{3})})) + (\frac{n}{3} - 1)d$$

$$= \frac{n}{3} + n\log(\frac{n}{3})(\log(n) - \frac{\log(2)}{2} + c) + (\frac{n}{3} - 1)d$$

$$\approx n + n\log(n)(\log(n))$$

$$\approx n(\log^2(n))$$

$\therefore O(n(\log n)^2)$