Problem 1

1. $n^2 + n \notin O(n \log n)$
   Assume that $n^2 + n \in O(n \log n)$. Then, there exist positive constants c and n_0 such that for all $n \geq n_0$:
   $$n^2 + n \leq c \cdot n \log n$$
   Divide both sides by n:
   $$n + 1 \leq c \log n$$
   As n approaches infinity, $n + 1$ grows linearly, while $\log n$ grows logarithmically. Therefore, for any constant c, there exists an n large enough such that $n + 1 > c \log n$, contradicting our assumption.
2. $n^2 + n \notin o(n^2)$
   Consider the limit:
   $$\lim_{n\to\infty} \frac{n^2 + n}{n^2} = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right) = 1$$
   By definition, $f(n) \in o(g(n))$ if:
   $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$
   But in this case, the limit is 1, not 0.

Problem 2

The total cost of the recursion tree is:

$$T(n) = \sum_{i=0}^{\log_2 n} n = (\log_2 n + 1)n = \frac{n \log n}{\log 2} + n \geq n \log n \text{ (when } n \geq \log 2)$$

Therefore, $T(n) \in \Omega(n \log n)$. Since $T(n) = O(n \log n)$ and $T(n) = \Omega(n \log n)$, it follows that $T(n) = \Theta(n \log n)$.

Problem 3

We need to prove that for all $n \geq n_0$, $T(n) \leq cn \log n$. Assume it holds for $\lfloor n/2 \rfloor + 17$, then we have:

$$T(\lfloor n/2 \rfloor + 17) \leq c \cdot (\lfloor n/2 \rfloor + 17) \cdot \log(\lfloor n/2 \rfloor + 17)$$

Plug in the definition of $T(n)$:

$$\begin{aligned} T(n) &\leq c \cdot (n + 34) \cdot \log(n/2 + 17) + n \\ &\leq c \cdot (n + 34) \cdot \log(n) + n \text{ (when } n \geq 34) \\ &= c \cdot n \log n + n \cdot (1 + 34c\frac{\log n}{n}) \end{aligned}$$

Since $\log n$ grows slower than n, there exists a constant c_1 such that $1 + 34c\frac{\log n}{n} \leq c_1$ for all $n \geq n_1$. Therefore, we have:

$$T(n) \leq c \cdot n \log n + n \cdot c_1 \leq c \cdot n \log n$$

This show that $T(n) \in O(n \log n)$.

Problem 4

The lower bound will be determined by the branch that terminates faster, which is the $T(n/3)$ branch. The total cost of the recursion tree is:

$$T(n) = \sum_{i=0}^{\log_3 n} cn = (\log_3 n + 1)cn = \frac{cn \log n}{\log 3} + cn \geq cn \log n \text{ (when } cn \geq \log 3)$$

Therefore, $T(n) \in \Omega(n \log n)$.

Problem 5

In all of the following cases, $\log_b a = \log_2 4 = 2$

1. Since $f(n) = n = O(n^{2-\epsilon})$, the time complexity is $\Theta(n^2)$.
2. Since $f(n) = n^2 = \Theta(n^2)$, the time complexity is $\Theta(n^2 \log n)$.
3. Since $f(n) = n^3 = \Omega(n^{2+\epsilon})$, the time complexity is $\Theta(n^3)$.

Problem 6

We can implement using Radix Sort. The time complexity is $O(d(n+b))$. Since n^2-1 is the maximum value, $d = O(\log_b(n))$, which makes the time complexity to $O((n + b) \log_b(n))$. To make the time complexity $O(n)$, we need to change the base to n, which makes $O(\log_b(n))$ to $O(1)$. Therefore, the overall time complexity is $O(n)$.