1. Multiplying two $n \times n$ matrices can be done with $O(n^3)$ multiplications (using the normal method), however a different method called Strassen's algorithm uses asymptotically less multiplications. Strassen's algorithm divides the matrix into four pieces, recurses on those and then uses 7 multiplications (and some additions) to combine these into the product wanted, so it uses
$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 7T\left(\left\lceil \frac{n}{2} \right\rceil\right) + \Theta(n^2) & \end{cases}$$
multiplications. How many multiplications does this algorithm need asymptotically?

2. Solve the recurrence
$$T(n) = \begin{cases} T\left(\sqrt[3]{n}\right) + n & \text{if } n > 1, \\ 1 & \text{if } n = 1. \end{cases}$$

3. Solve the recurrence
$$T(n) = \begin{cases} 3T\left(\frac{n}{4}\right) + n \log n & \text{if } n > 1, \\ 1 & \text{if } n = 1. \end{cases}$$

4. If
$$T(n) = \begin{cases} 2T\left(\frac{n}{7}\right) + 5T\left(\frac{n}{8}\right) + n & \text{if } n > 1, \\ 1 & \text{if } n = 1. \end{cases}$$
Prove that $T(n) = O(n \log n)$.