The general formula for recurrences:
Suppose $T(n) = aT(n/b) + O(n^d)$ then
If $a < b^d$ then $T(n) = O(n^d)$
If $a = b^d$ then $T(n) = O(n^d \log n)$
If $a > b^d$ then $T(n) = O(n^{\log_b a})$

Give the order of magnitude of the running time, in the form $T(n) = O(\ldots)$ for the divide and conquer algorithms described below. Read the questions carefully, there are small but important differences.

1. The algorithm solves a problem of size $n$ by solving two problems of size $n/2$ and then doing a constant $[O(1)]$ amount of work.
   \[ T(n) = 2T(n/2) + O(1) = O(n) \]

2. The algorithm solves a problem of size $n$ by solving two problems of size $n/2$ and then doing a linear $[O(n)]$ amount of work.
   \[ T(n) = 2T(n/2) + O(n) = O(n \log n) \]

3. The algorithm solves a problem of size $n$ by solving four problems of size $n/2$ and then doing a quadratic $[O(n^2)]$ amount of work.
   \[ T(n) = 4T(n/2) + O(n^2) = O(n^2 \log n) \]

4. The algorithm solves a problem of size $n$ by solving one problem of size $n/2$ and then doing a linear $[O(n)]$ amount of work.
   \[ T(n) = T(n/2) + O(n) = O(n) \]