CSC 421/Applied Algorithms and Structures
Recursion trees

The recurrence relations that arise from analyzing a divide-and-conquer algorithm almost always have the form:

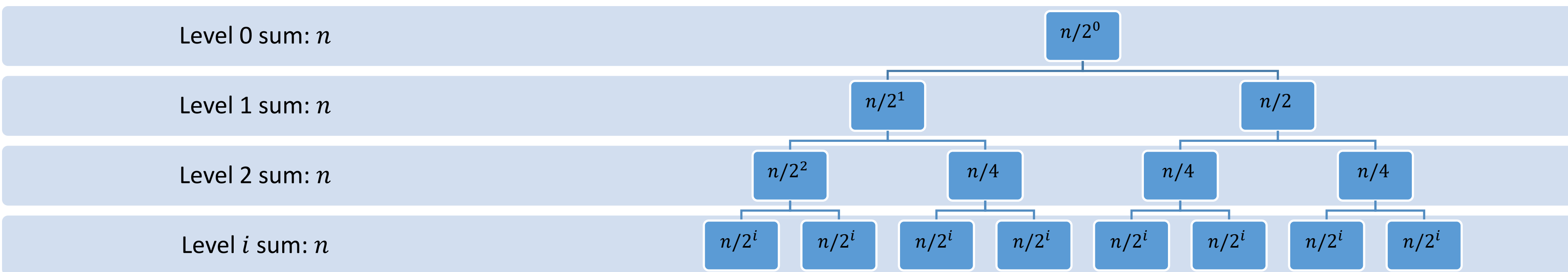
$$T(n) = rT(n/c) + f(n)$$

The variable r is a count of the number of recursive calls, the variable c is the fraction of the input passed to each recursive call, and $f(n)$ is the number of steps performed by the divide and the combine steps.

Examples
Here is the recurrence and the recursion tree for algorithms like merge sort and the average case of quick sort.

$$T(n) = 2T(n/2) + n; T(1) = 0$$

$$r = 2, c = 2, f(n) = n$$



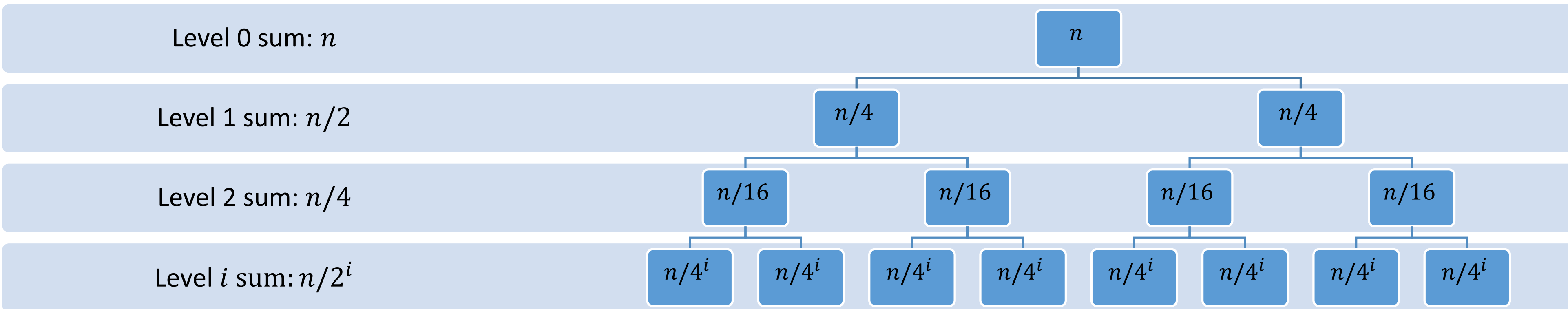
The sum of each level is **equal**. There are $\lg(n) + 1$ levels. The bottom level has a total of 0 so the total running time is:

$$T(n) = \sum_{i=0}^{\lg(n)} n = n \cdot (\lg(n) + 1) = O(n \lg(n))$$

Here is another recurrence and its recursion tree.

$$T(n) = 2T(n/4) + n; T(1) = 0$$

$$r = 2, c = 4, f(n) = n$$



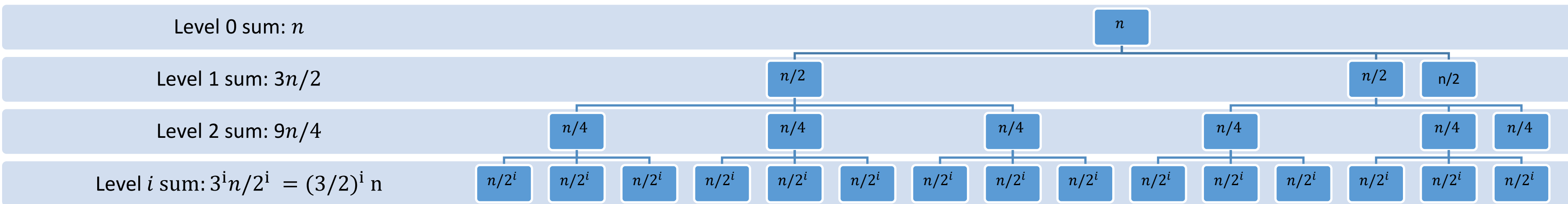
The sum of each level is **decreasing**. There are $\lg(n) + 1$ levels. The bottom level has a total of 0 so the total running time is:

$$\sum_{i=0}^{\lg(n)} \frac{n}{2^i} = \frac{n}{1} + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{n} = n(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{n}) \leq 2n = O(n)$$

Here is another recurrence and its recursion tree.

$$T(n) = 3T(n/2) + n; T(1) = 0$$

$$r = 3, c = 2, f(n) = n$$



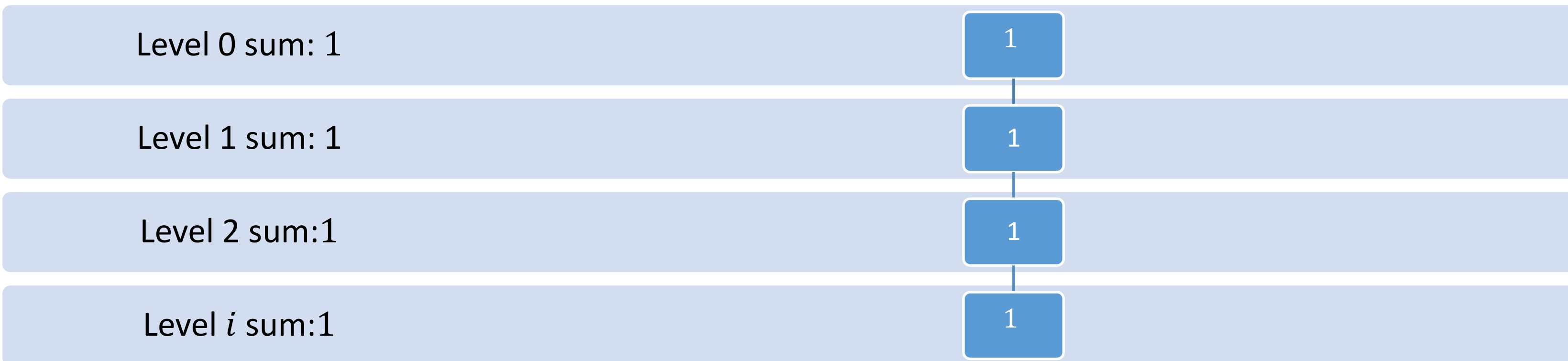
The sum of each level is **increasing**. There are $\lg(n) + 1$ levels. The bottom level has a total of 0 so the total running time is:

$$\sum_{i=0}^{\lg(n)} \left(\frac{3}{2}\right)^i n = n\left(1 + \frac{3}{2} + \frac{9}{4} + \cdots + \frac{3^i}{2^i} + \cdots\right) = O\left(n^{\lg(3)}\right) = O(n^{1.58})$$

Here is the recurrence and the recursion tree for binary search.

$$T(n) = T(n/2) + 1; T(1) = 0$$

$$r = 1, c = 2, f(n) = 1$$



The sum of each level is **equal**. There are $\lg(n) + 1$ levels so the total running time is:

$$\sum_{i=0}^{\lg(n)} 1 = (1 + \lg(n)) = O(\lg n)$$