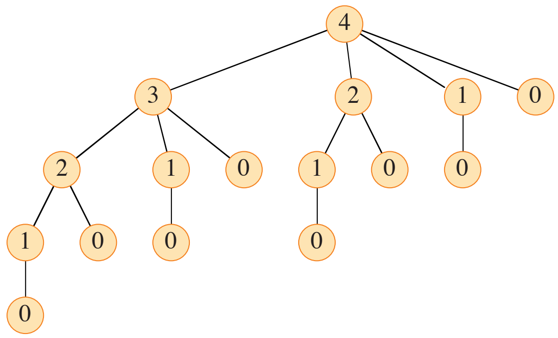


**Figure 14.3** The recursion tree showing recursive calls resulting from a call CUT-ROD$(p, n)$ for $n = 4$. Each node label gives the size $n$ of the corresponding subproblem, so that an edge from a parent with label $s$ to a child with label $t$ corresponds to cutting off an initial piece of size $s - t$ and leaving a remaining subproblem of size $t$. A path from the root to a leaf corresponds to one of the $2^{n-1}$ ways of cutting up a rod of length $n$. In general, this recursion tree has $2^n$ nodes and $2^{n-1}$ leaves.

are there? A rod of length $n$ has $n - 1$ potential locations to cut. Each possible way to cut up the rod makes a cut at some subset of these $n - 1$ locations, including the empty set, which makes for no cuts. Viewing each cut location as a distinct member of a set of $n - 1$ elements, you can see that there are $2^{n-1}$ subsets. Each leaf in the recursion tree of Figure 14.3 corresponds to one possible way to cut up the rod. Hence, the recursion tree has $2^{n-1}$ leaves. The labels on the simple path from the root to a leaf give the sizes of each remaining right-hand piece before making each cut. That is, the labels give the corresponding cut points, measured from the right-hand end of the rod.

### Using dynamic programming for optimal rod cutting

Now, let's see how to use dynamic programming to convert CUT-ROD into an efficient algorithm.

The dynamic-programming method works as follows. Instead of solving the same subproblems repeatedly, as in the naive recursion solution, arrange for each subproblem to be solved *only once*. There's actually an obvious way to do so: the first time you solve a subproblem, *save its solution*. If you need to refer to this subproblem's solution again later, just look it up, rather than recomputing it.

Saving subproblem solutions comes with a cost: the additional memory needed to store solutions. Dynamic programming thus serves as an example of a ***time-memory trade-off***. The savings may be dramatic. For example, we're about to use dynamic programming to go from the exponential-time algorithm for rod cutting