$q = \max\{p_i + \text{CUT-ROD}(p, n-i) : 1 \le i \le n\}$. Line 6 then returns this value. A simple induction on n proves that this answer is equal to the desired answer r_n, using equation (14.2).

```
CUT-ROD(p, n)
1  if n == 0
2      return 0
3  q = -∞
4  for i = 1 to n
5      q = max {q, p[i] + CUT-ROD(p, n - i)}
6  return q
```

If you code up CUT-ROD in your favorite programming language and run it on your computer, you'll find that once the input size becomes moderately large, your program takes a long time to run. For $n = 40$, your program may take several minutes and possibly more than an hour. For large values of n, you'll also discover that each time you increase n by 1, your program's running time approximately doubles.

Why is CUT-ROD so inefficient? The problem is that CUT-ROD calls itself recursively over and over again with the same parameter values, which means that it solves the same subproblems repeatedly. Figure 14.3 shows a recursion tree demonstrating what happens for $n = 4$: CUT-ROD(p, n) calls CUT-ROD$(p, n-i)$ for $i = 1, 2, \ldots, n$. Equivalently, CUT-ROD(p, n) calls CUT-ROD(p, j) for each $j = 0, 1, \ldots, n-1$. When this process unfolds recursively, the amount of work done, as a function of n, grows explosively.

To analyze the running time of CUT-ROD, let $T(n)$ denote the total number of calls made to CUT-ROD(p, n) for a particular value of n. This expression equals the number of nodes in a subtree whose root is labeled n in the recursion tree. The count includes the initial call at its root. Thus, $T(0) = 1$ and

$$T(n) = 1 + \sum_{j=0}^{n-1} T(j) . \tag{14.3}$$

The initial 1 is for the call at the root, and the term $T(j)$ counts the number of calls (including recursive calls) due to the call CUT-ROD$(p, n-i)$, where $j = n - i$. As Exercise 14.1-1 asks you to show,

$$T(n) = 2^n , \tag{14.4}$$

and so the running time of CUT-ROD is exponential in n.

In retrospect, this exponential running time is not so surprising. CUT-ROD explicitly considers all possible ways of cutting up a rod of length n. How many ways