```
function CYK(words,grammar) returns The most probable parse
                                    and its probability

  Create and clear π[num_words, num_words, num_nonterminals]

  # base case
  for i ← 1 to num_words
    for A ← 1 to num_nonterminals
      if (A → w_i) is in grammar then
        π [i, i, A] ← P(A → w_i)

  # recursive case
  for span ← 2 to num_words
    for begin ← 1 to num_words − span + 1
      end ← begin + span − 1
      for m = begin to end − 1
        for A = 1 to num_nonterminals
        for B = 1 to num_nonterminals
        for C = 1 to num_nonterminals
          prob = π [begin, m, B] × π [m + 1, end, C] × P(A→B C)
          if (prob > π [begin, end, A]) then
            π [begin, end, A] = prob
            back[begin, end, A] = {m, B, C}
  return build_tree(back[1, num_words, 1]), π [1, num_words, 1]
```

**Figure 12.3** The Probabilistic CYK algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky Normal Form (after Collins (1999) and Aho and Ullman (1972).) *back* is an array of back-pointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

When a treebank is unavailable, the counts needed for computing PCFG probabilities can be generated by first parsing a corpus. If sentences were unambiguous, it would be as simple as this: parse the corpus, increment a counter for every rule in the parse, and then normalize to get probabilities. However, since most sentences are ambiguous, in practice we need to keep a separate count for each parse of a sentence and weight each partial count by the probability of the parse it appears in. The standard algorithm for computing this is called the **Inside-Outside** algorithm, and was proposed by Baker (1979) as a generalization of the forward-backward algorithm of Chapter 7. See Manning and Schütze (1999) for a complete description of

INSIDE-OUTSIDE