```
Algorithm Knapsack (S, K) ;
Input: S (an array of size n storing the sizes of the items),
    and K (the size of the knapsack).
Output: P (a two-dimensional array such that P [i, k ].exist = true if there
    exists a solution to the knapsack problem with the first i elements and a
    knapsack of size k, and P [i, k ].belong = true if the ith element belongs
    to that solution).
    { See Exercise 5.15 for suggestions about improving this program. }

begin
    P [0, 0].exist := true ;
    for k := 1 to K do
        P [0, k ].exist := false ;
    { there is no need to initialize P [i, 0] for i ≥ 1, because it will
        be computed from P [0, 0] }
    for i := 1 to n do
        for k := 0 to K do
            P [i, k ].exist := false ; { the default value }
            if P [i − 1, k ].exist then
                P [i, k ].exist := true ;
                P [i, k ].belong := false
            else if k − S [i ] ≥ 0 then
                    if P [i − 1, k − S [i ]].exist then
                        P [i, k ].exist := true ;
                        P [i, k ].belong := true
end
```

Figure 5.10 Algorithm *Knapsack*.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$k_1=2$	O	-	I	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$k_2=3$	O	-	O	I	-	I	-	-	-	-	-	-	-	-	-	-	-
$k_3=5$	O	-	O	O	-	O	-	I	I	-	I	-	-	-	-	-	-
$k_4=6$	O	-	O	O	-	O	I	O	O	I	O	I	-	I	I	-	I

Figure 5.11 An example of the table constructed for the knapsack problem. The input consists of four items of sizes 2, 3, 5, and 6. The symbols in the table are the following: "I": a solution containing this item has been found; "O": a solution without this item has been found; "-": no solution has not yet been found. (If the symbol "-" appears in the last line, then there is no solution for a knapsack of this size.)