5.4 The Sum-of-Subsets Problem

Recall our thief and the 0-1 Knapsack problem from Section 4.4.1 problem, there is a set of items the thief can steal, and each item ha weight and profit. The thief's knapsack will break if the total weigh items in it exceeds W. Therefore, the goal is to maximize the total valu stolen items while not making the total weight exceed W. Suppose he the items all have the same profit per unit weight. Then an optimal s for the thief would simply be a set of items that maximized the total subject to the constraint that its total weight did not exceed W. The thief first try to determine whether there was a set whose total weight equa because this would be best. The problem of determining such sets is call Sum-of-Subsets problem.

Specifically, in the Sum-of-Subsets problem, there are n positive inte (weights) w_i and a positive integer W. The goal is to find all subsets o integers that sum to W. As mentioned earlier, we usually state our prob so as to find all solutions. For the purposes of the thief's application, how only one solution need be found.

ample 5.2

Suppose that $n = 5$, $W = 21$, and

$$w_1 = 5 \qquad w_2 = 6 \qquad w_3 = 10 \qquad w_4 = 11 \qquad w_5 = 16.$$

Because

$$\begin{aligned} w_1 + w_2 + w_3 &= 5 + 6 + 10 = 21, \\ w_1 + w_5 &= 5 + 16 = 21, \text{ and} \\ w_3 + w_4 &= 10 + 11 = 21, \end{aligned}$$

the solutions are $\{w_1, w_2, w_3\}$, $\{w_1, w_5\}$, and $\{w_3, w_4\}$.

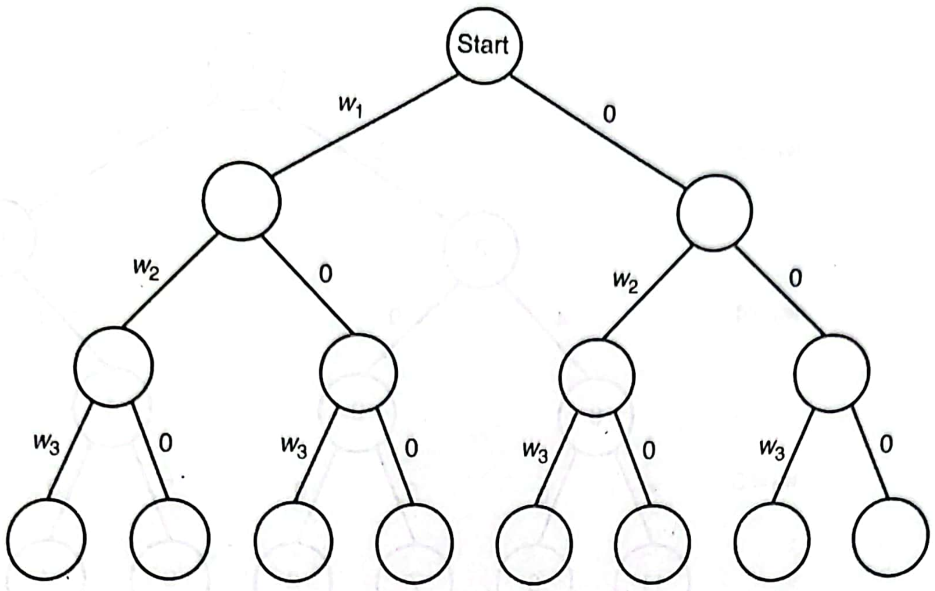

Figure 5.7 • A state space tree for instances of the Sum-of-Subsets problem in which $n = 3$.

This instance can be solved by inspection. For larger values of n, a systematic approach is necessary. One approach is to create a state space tree. A possible way to structure the tree appears in Figure 5.7. For the sake of simplicity, the tree in this figure is for only three weights. We go to the left from the root to include w_1, and we go to the right to exclude w_1. Similarly, we go to the left from a node at level 1 to include w_2, and we go to the right to exclude w_2, etc. Each subset is represented by a path from the root to a leaf. When we include w_i, we write w_i on the edge where we include it. When we do not include w_i, we write 0.

---

le 5.3 Figure 5.8 shows the state space tree for $n = 3$, $W = 6$, and

$$w_1 = 2 \qquad w_2 = 4 \qquad w_3 = 5.$$

At each node, we have written the sum of the weights that have been included up to that point. Therefore, each leaf contains the sum of the weights in the subset leading to that leaf. The second leaf from the left is the only one containing a 6. Because the path to this leaf represents the subset $\{w1,\ w2\}$, this subset is the only solution.

If we sort the weights in nondecreasing order before doing the search, there is an obvious sign telling us that a node is nonpromising. If the weights are sorted in this manner, then w_{i+1} is the lightest weight remaining when we are at the ith level. Let ***weight*** be the sum of the weights that have been included up to a node at level i. If w_{i+1} would bring the value of *weight* above W, then so would any other weight following it. Therefore, unless *weight* equals W (which means that there is a solution at the node), a node at the ith level is nonpromising if

$$weight + w_{i+1} > W.$$

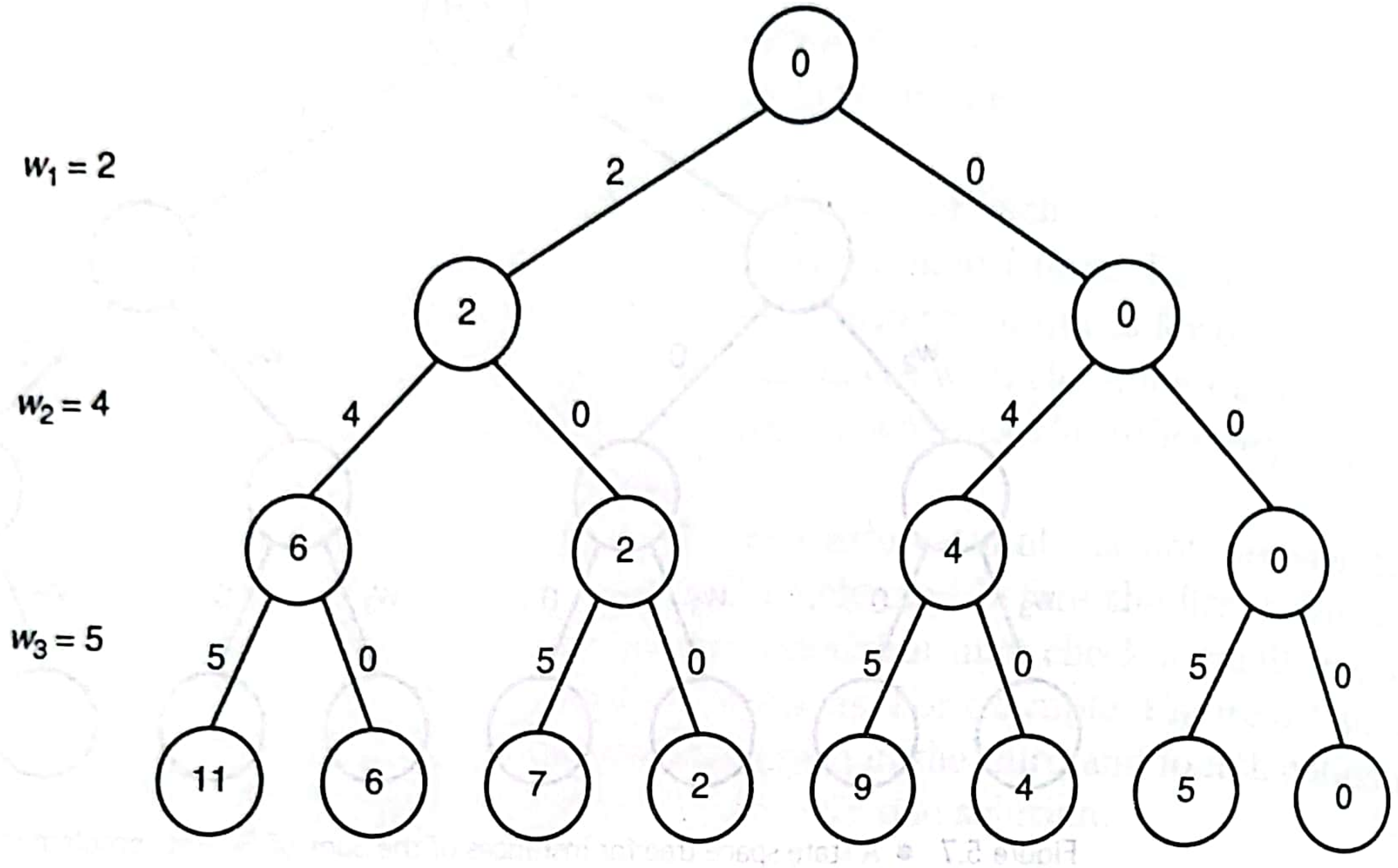

Figure 5.8 • A state space tree for the Sum-of-Subsets problem for the instance in Example 5.3. Stored at each node is the total weight included up to that node.

There is another, less obvious sign telling us that a node is nonpromising. If, at a given node, adding all the weights of the remaining items to *weight* does not make *weight* at least equal to W, then *weight* could never become equal to W by expanding beyond the node. This means that if ***total*** is the total weight of the remaining weights, a node is nonpromising if

$$weight + total < W.$$

The following example illustrates these backtracking strategies.