SUBSET SUMPROBLEM

The ***subset-sum problem*** finds a subset of a given set $A = \{a1, \ldots, an\}$ of n positive integers whose sum is equal to a given positive integer d. For example, for $A = \{1, 2, 5, 6, 8\}$ and $d = 9$, there are two solutions: {1, 2, 6} and {1, 8}. Of course, some instances of this problem may have no solutions.

It is convenient to sort the set's elements in increasing order. So, we will assume that $a1 < a2 < \ldots < an$.

A = {3, 5, 6, 7} and d = 15 of the subset-sum problem. The number inside a node is the sum of the elements already included in the subsets represented by the node. The inequality below a leaf indicates the reason for its termination.

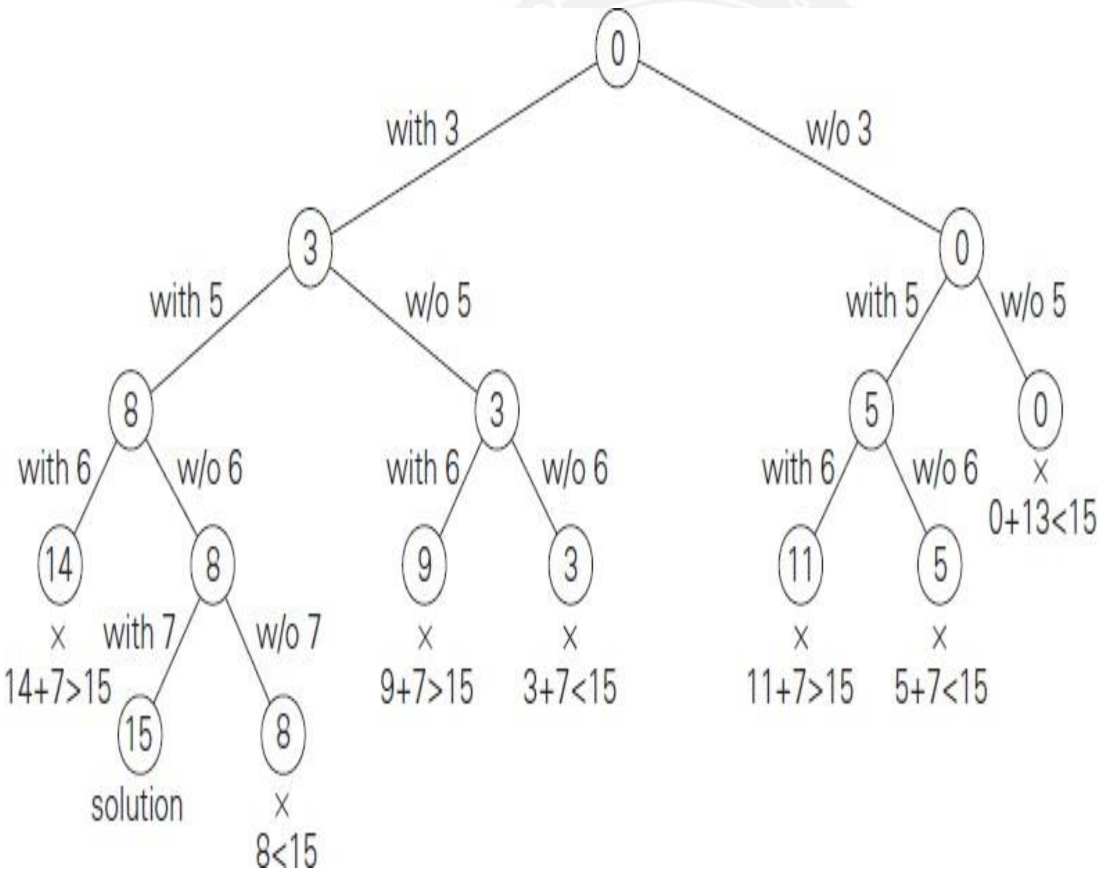

FIGURE Complete state-space tree of the backtracking algorithm applied to the instance

ALGORITHM *Backtrack*(*X* [1..*i*])

//Gives a template of a generic backtracking algorithm
//Input: *X*[1..*i*] specifies first *i* promising components of a solution
//Output: All the tuples representing the problem's solutions
if *X*[1..*i*] is a solution **write** *X*[1..*i*]

else //see Problem thissection
for each element $x \in Si+1$ consistent with *X*[1..*i*] and the constraints **do**
X[*i* + 1] ←
x
Backtrack(
X[1..*i* + 1])

General Remarks

From a more general perspective, most backtracking algorithms fit the following escription. An output of a backtracking algorithm can be thought of as an *n*-tuple $(x_1, x_2, . . . , x_n)$ where each coordinate *xi* is an element of some finite lin early ordered set *Si* . For example, for the *n*-queens problem, each *Si* is the set of integers (column numbers) 1 through *n*.

A backtracking algorithm generates, explicitly or implicitly, a state-space tree; its nodes represent partially constructed tuples with the first i coordinates defined by the earlier actions of the algorithm. If such a tuple $(x_1, x_2, . . . , x_i)$ is not a solution, the algorithm finds the next element in S_{i+1} that is consistent with the values of $((x_1, x_2, . . . , x_i)$ and the problem's constraints, and adds it to the tuple as its (i + 1)st coordinate. If such an element does not exist, the algorithm backtracks to consider the next value of xi, and soon.