Design and Analysis of Algorithms

Lecture-39

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

Sum of Subsets problem

Statement: In this problem, we are given n distinct positive numbers (called weights) and one another number m. We have to find all combinations of these numbers whose sums are m.

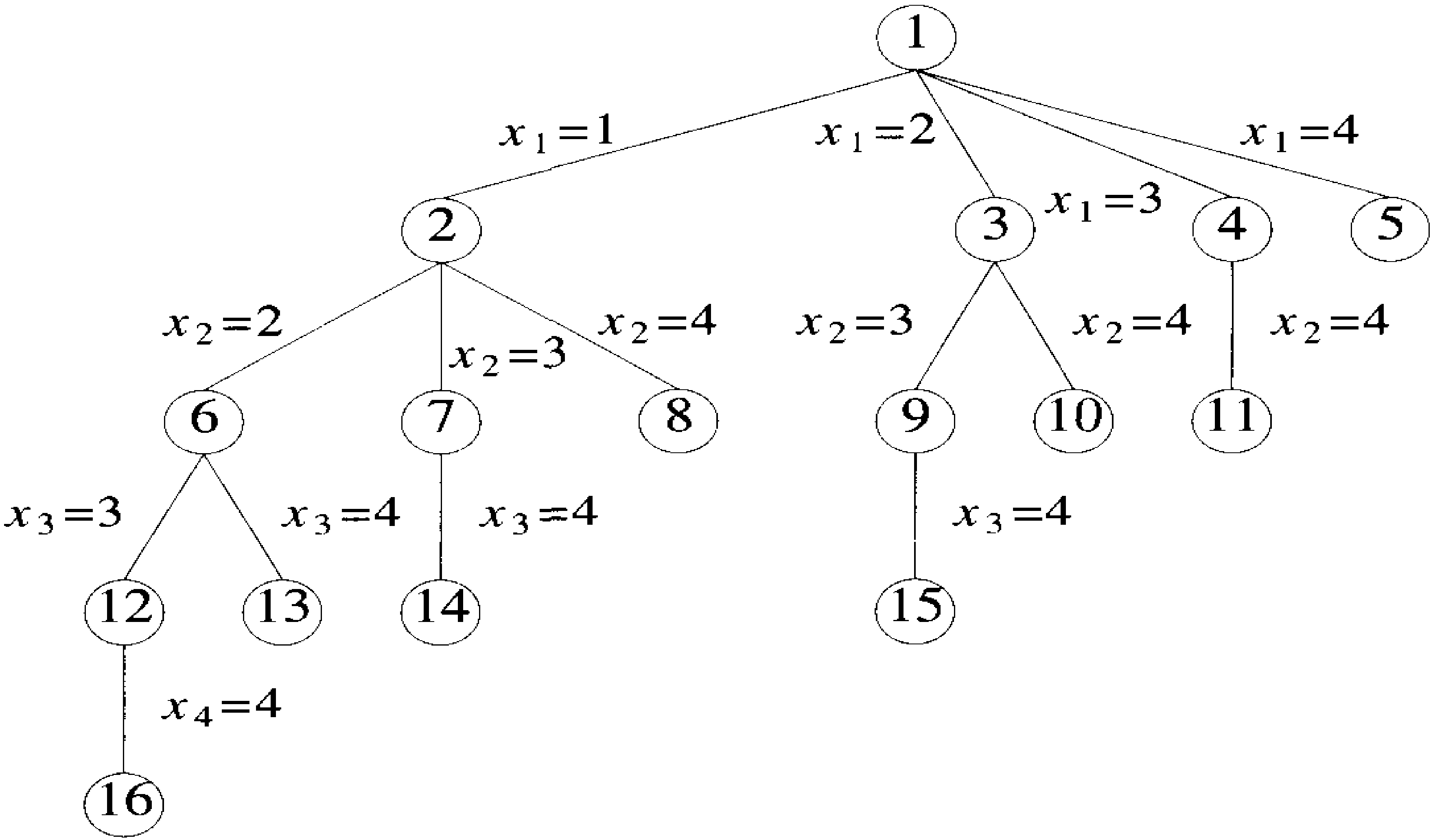
This problem can be formulated using either fixed or variable sized tuples.

Example: Consider $n=4$, $(w1, w2, w3, w4) = (11, 13, 24, 7)$ and $m=31$.

- The desired subsets are (11, 13, 7) and (24, 7).
- The solution in variable-size tuple will be (1, 2, 4) and (3,4).
- The solution in fixed-size tuple will be (1, 1, 0, 1) and (0, 0, 1, 1).

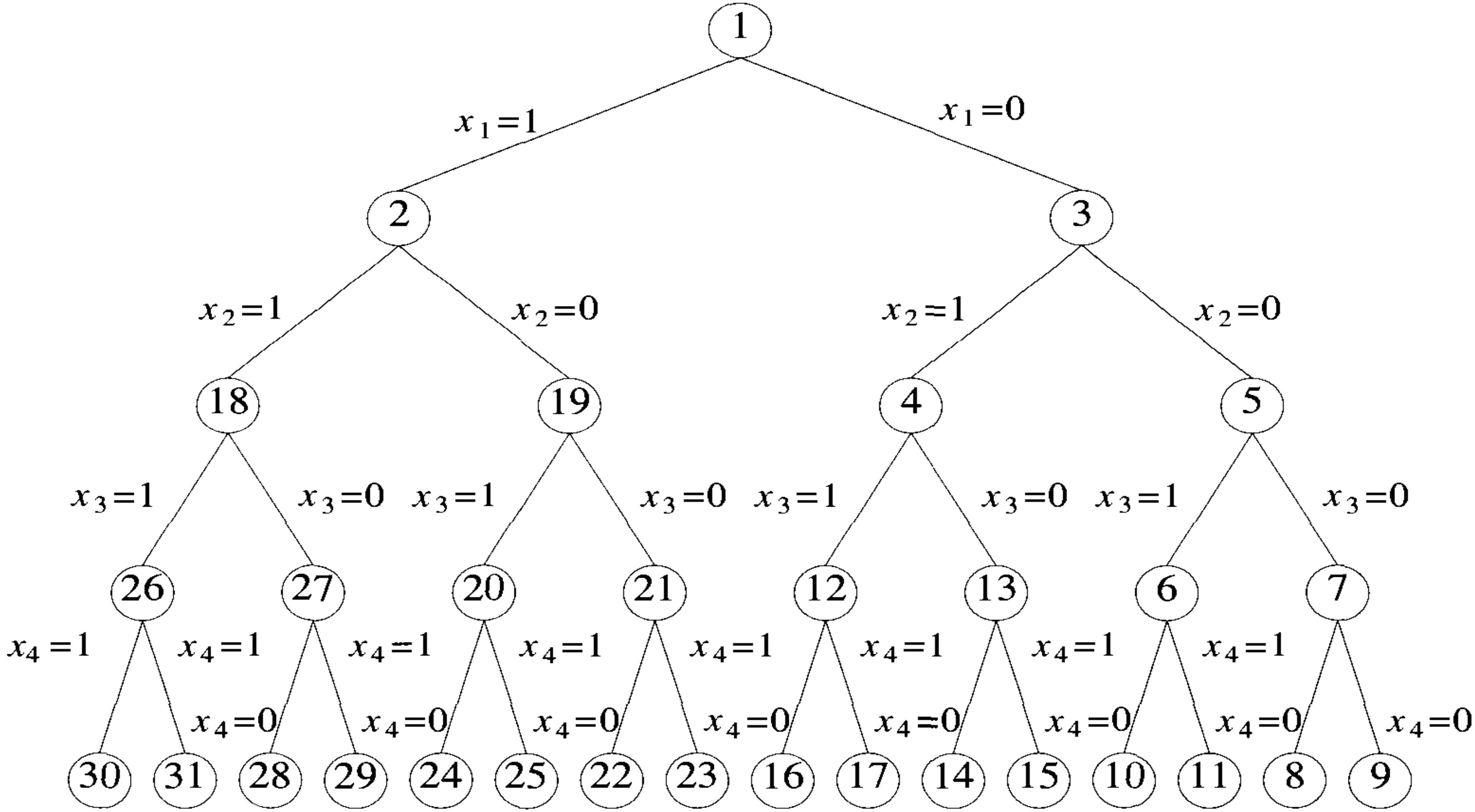
Sum of Subsets problem

The state space tree for this question in case of variable-size tuple is the following:-



Sum of Subsets problem

The state space tree for this question in case of fixed-size tuple is the following:-



Backtracking approach for Sum of Subsets problem

Backtracking solution for this problem uses fixed-sized tuple strategy.

In this case, the value of x_i in the solution vector will be either 1 or 0 depending on whether weight w_i is included or not.

The bounding function is $B_k(x_1, x_2, \ldots\ldots, x_k)$ = true iff

$$\sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i \geq m$$

Clearly $(x_1, x_2, \ldots\ldots, x_k)$ can not lead to an answer node if this condition is not satisfied.

If all the weights are initially in ascending order then $(x_1, x_2, \ldots\ldots, x_k)$ can not lead to an answer node if

$$\sum_{i=1}^{k} w_i x_i + w_{k+1} > m$$

Backtracking approach for Sum of Subsets problem

Therefore, we use the following bounding function

The bounding function is $B_k(x_1, x_2, \ldots\ldots, x_k)$ = true iff

$\sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i \geq m$

and

$\sum_{i=1}^{k} w_i x_i + w_{k+1} \leq m$

Note: Here, we have assume all the weights are in ascending order.

Backtracking approach for Sum of Subsets problem

SumOfSub(s,k,r)

```
1.x[k] ← 1
2.if (s+w[k] = m)
3.      for j←1 to k
4.            print x[j]
5.else if (s+w[k]+w[k+1] ≤ m)
6.SumOfSub(s+w[k], k+1, r-w[k])
7.if (( s+r-w[k] ≥ m) and (s+w[k+1] ≤ m))
8.x[k] ← 0
9.            SumOfSub(s, k+1, r-w[k])
```

The initial call is SumOfSub(0, 1, $\sum_{i=1}^{n} w_i$)

Backtracking approach for Sum of Subsets problem

Time complexity of this algorithm is $O(2^n)$.

Because

Total number of nodes $= 1+2+2^2+2^3+\ldots\ldots.+2^n$

$$= 2^{n+1}-1$$

Example: Let w = {5, 10, 12, 13, 15, 18} and m = 30. Find all possible subsets of w that sum to m. Do this using SumOfSub. Draw the portion of the state space tree that is generated.

Solution: