Backtracking

Sum of Subsets

Backtracking

Subset-sum Problem

- **Subset-sum Problem:** The problem is to find a subset of a given set **S = {s_1, s_2,- - -, s_n}** of **'n'** positive integers whose sum is equal to a given positive integer **'d'**.
- **Observation :** It is convenient to sort the set's elements in increasing order, $S_1 \leq S_2 \leq \ldots.. \leq S_n$. And each set of solutions don't need to be necessarily of fixed size.
- **Example :** For S = {3, 5, 6, 7} and d = 15, the solution is shown below :-

  Solution = {3, 5, 7}

Implicit Constraint

- **No two weights should be the same**
- **The sum of the corresponding w_i's be m**

Sum of Subsets Problem-Algorithm

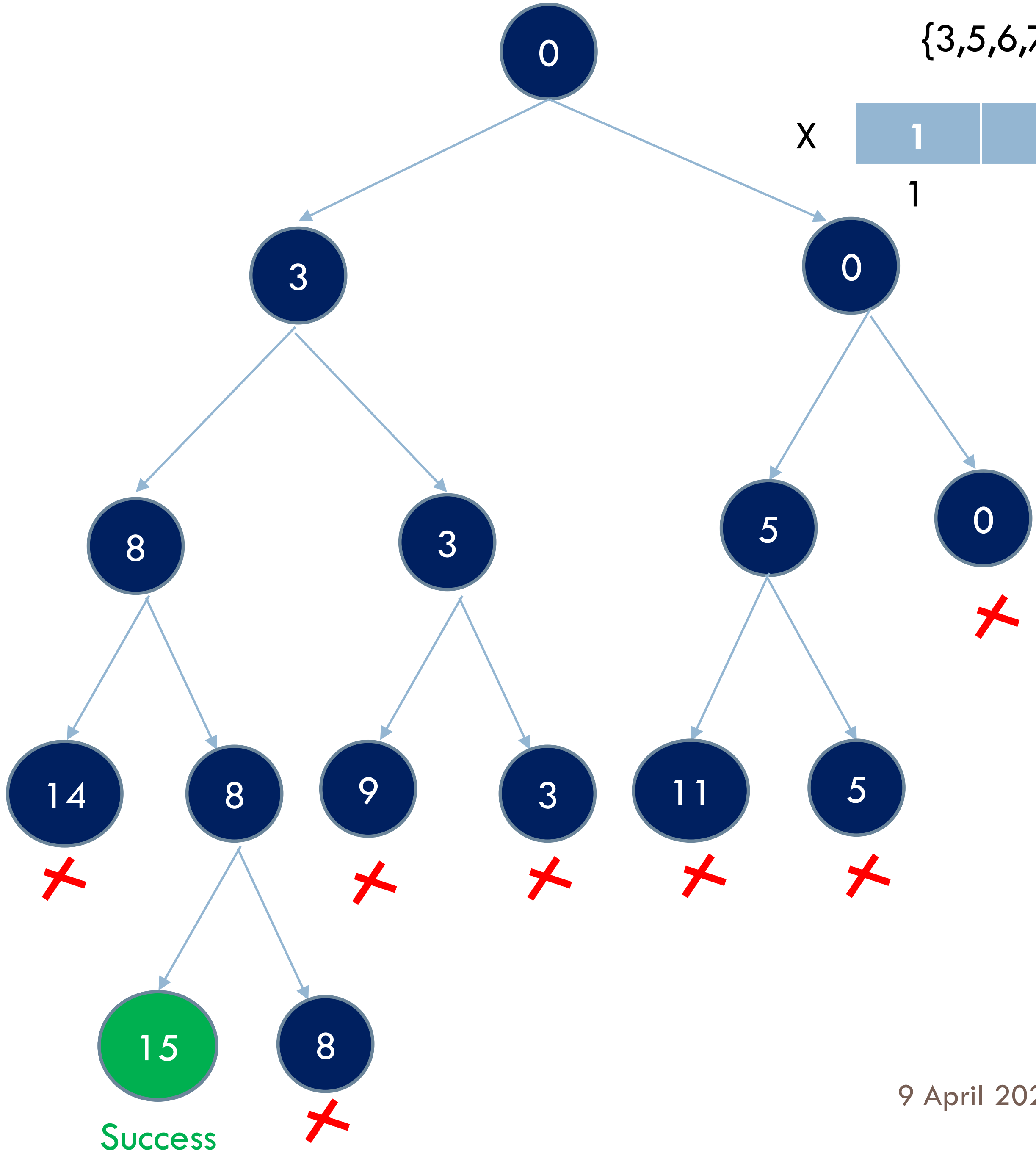
- Start with an empty set
- Add to the subset the next element from the list
- If the subset is having the sum d then stop with that subset as solution
- If the subset is not feasible or we if we have reached the end of the set then backtrack through the subset until we find the most suitable value.
- If the subset is feasible then repeat step 2
- If we have visited all the elements without finding a suitable subset and if no backtracking is possible then stop without solution

Example

- If n=4 (w1,w2,w3,w4)=(11,13,24,7) and m=31 then the desired subsets are (11,13,7) and (24,7)
- In general all solutions are k-tuples (x1,x2,…...xk), 1<=k<=n and different solutions may have different sized tuples
- Each solution subset is represented by an n-tuple(x1,x2….xn) such that xiЄ{0,1} , 1<=i<=n
- The solutions to the above instance are {1,1,0,1} and {0,0,1,1}

{3,5,6,7}, d=15

	1	2	3	4
X	1	1	0	1



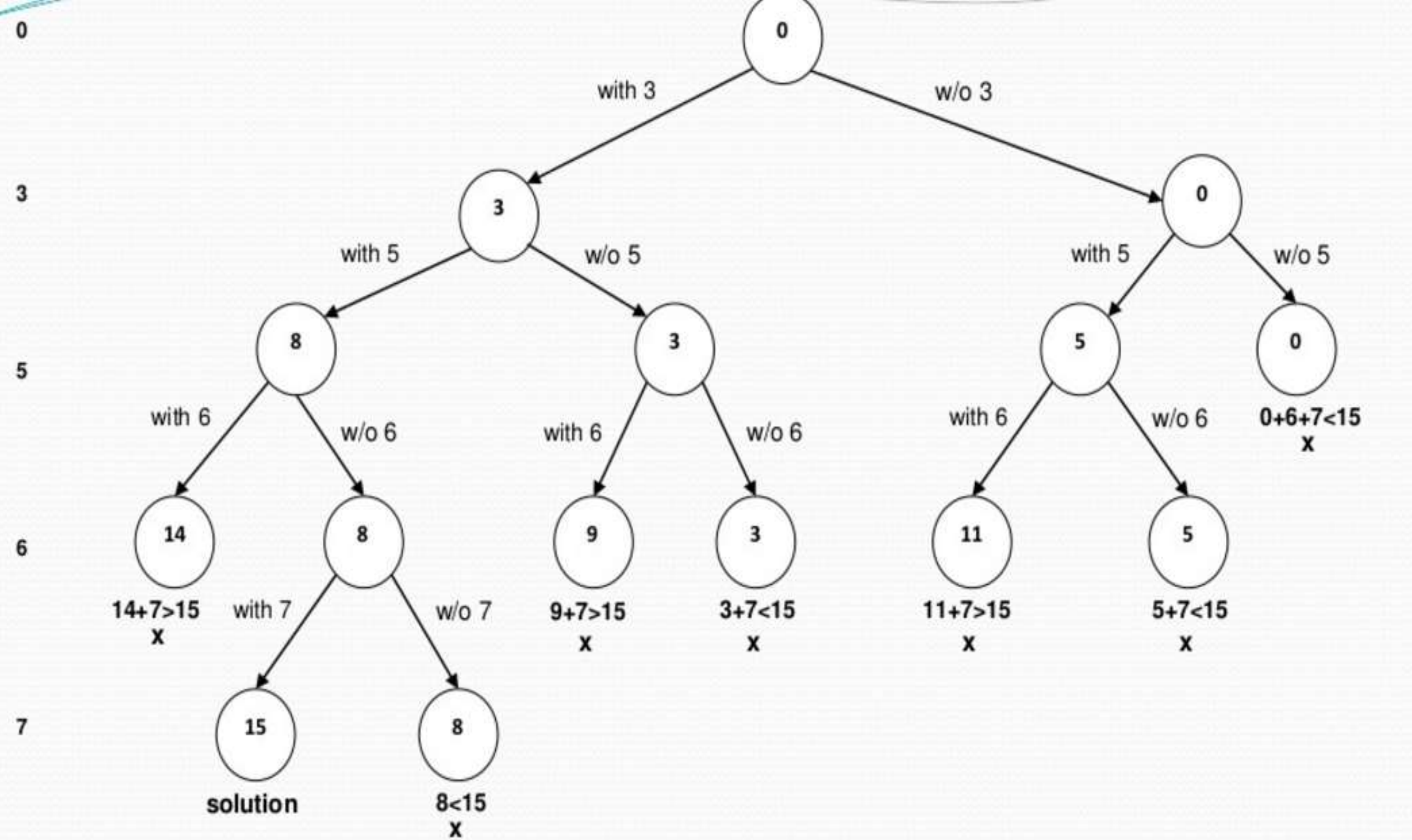
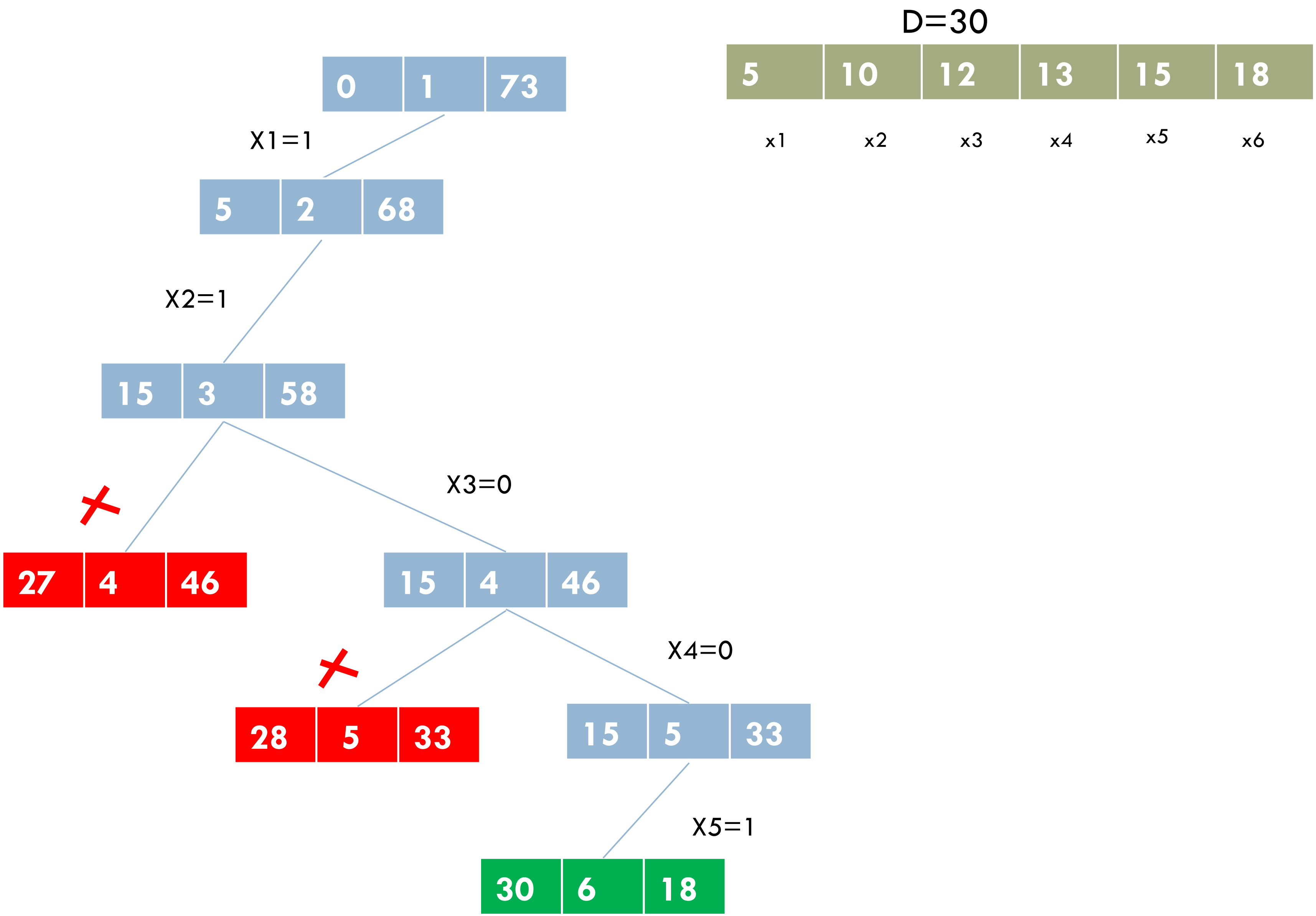


Figure : Compete state-space tree of the backtracking algorithm applied to the instance **S = {3, 5, 6, 7}** and **d = 15** of the subset-sum problem. The number inside a node is the sum of the elements already included in subsets represented by the node. The inequality below a leaf indicates the reason for its termination.

- S={5,10,12,13,15,18}
- D=30
- Solve for obtaining sum of Subset.

Algorithm

Algorithm SumOfSub(s, k, r)
// Find all subsets of $w[1:n]$ that sum to m. The values of $x[j]$,
// $1 \leq j < k$, have already been determined. $s = \sum_{j=1}^{k-1} w[j] * x[j]$
// and $r = \sum_{j=k}^{n} w[j]$. The $w[j]$'s are in nondecreasing order.
// It is assumed that $w[1] \leq m$ and $\sum_{i=1}^{n} w[i] \geq m$.
{
    // Generate left child. Note: $s + w[k] \leq m$ since B_{k-1} is true.
    $x[k] := 1;$
    if $(s + w[k] = m)$ **then write** $(x[1:k])$; // Subset found
        // There is no recursive call here as $w[j] > 0,\ 1 \leq j \leq n$.
    else **if** $(s + w[k] + w[k+1] \leq m)$
        then SumOfSub$(s + w[k], k+1, r - w[k])$;
    // Generate right child and evaluate B_k.
    if $((s + r - w[k] \geq m)$ **and** $(s + w[k+1] \leq m))$ **then**
    {
        $x[k] := 0;$
        SumOfSub$(s, k+1, r - w[k])$;
    }
}


D=30
5 10 12 13 15 18
x1 x2 x3 x4 x5 x6
0 1 73
X1=1
5 2 68
X2=1
15 3 58
27 4 46
X3=0
15 4 46
28 5 33
X4=0
15 5 33
X5=1
30 6 18