Week 7:
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. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1, 2, 6} and {1, 8}. A suitable message is to be displayed if the given problem instance doesn't have a solution.

Problem:

- Given positive numbers w_i, $1 \le i \le n$, and m, this problem requires finding all subsets of w_i whose sums are „m".
- All solutions are k-tuples, $1 \le k \le n$. Explicit constraints:
  $x_i \in \{j \mid j \text{ is an integer and } 1 \le j \le n\}$.

Implicit constraints:

No two xi can be the same.

The sum of the corresponding wi" be m.

$$B_k(x_1, \ldots, x_k) = true \text{ iff } \sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i \ge m$$

$$\text{and } \sum_{i=1}^{k} w_i x_i + w_{k+1} \le m$$

$x_i < x_i+1$, $1 \le i < k$ (total order in indices) to avoid generating multiple instances of the same subset (for example, (1, 2, 4) and (1, 4, 2) represent the same subset).

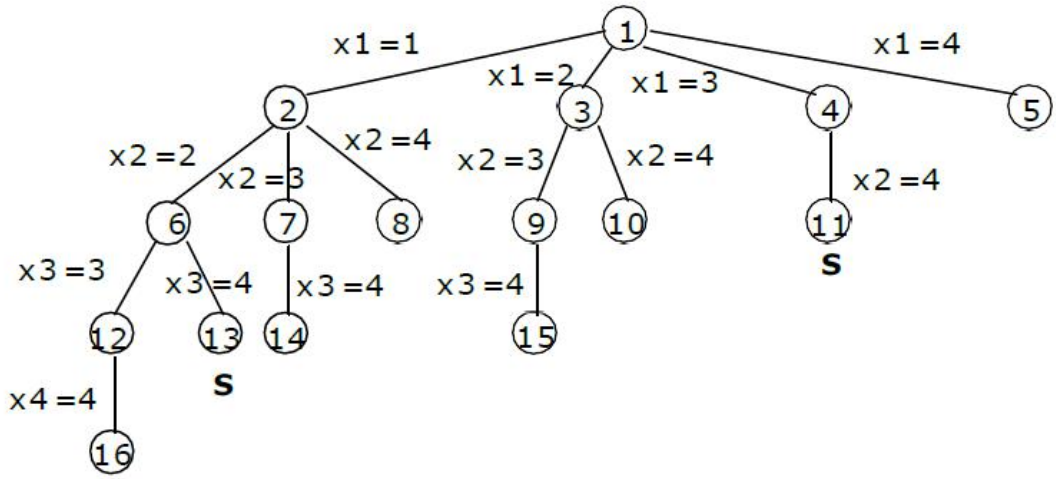
A better formulation of the problem is where the solution subset is represented by an n-tuple (x_1, ... , x_n) such that $x_i \in \{0, 1\}$.

Algorithm:

```
Algorithm SumOfSub(s, k, r)
// Find all subsets of w[1 : n] that sum to m. The values of x[j],
// 1 ≤ j < k, have already been determined. s = Σ_{j=1}^{k-1} w[j] * x[j]
// and r = Σ_{j=k}^{n} w[j]. The w[j]'s are in nondecreasing order.
// It is assumed that w[1] ≤ m and Σ_{i=1}^{n} w[i] ≥ m.
{
     // Generate left child. Note: s + w[k] ≤ 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 ≤ j ≤ n.
     else if (s + w[k] + w[k + 1] ≤ m)
          then SumOfSub(s + w[k], k + 1, r − w[k]);
     // Generate right child and evaluate B_k.
     if ((s + r − w[k] ≥ m) and (s + w[k + 1] ≤ m)) then
     {
          x[k] := 0;
          SumOfSub(s, k + 1, r − w[k]);
     }
}
```

Procedure:
n = 4, w = (11, 13, 24, 7) and m = 31, the desired subsets are (11,13, 7) and (24, 7).

- ✓ Solution to the problem constructed by considering the each element in the set.
- ✓ Decision is made either to select or reject the element.
- ✓ If the element is selected it is added to the partial sum.
- ✓ If it is rejected then it is not added to the partial solution.
- ✓ Constraint is verified after selecting an element if constraint violated algorithm backtrack to consider the next alternative.
- ✓ If constraint satisfied algorithm proceeds to make decision on next element.



The tree corresponds to the variable tuple size formulation.
The edges are labeled such that an edge from a level i node to a level i+1 node represents a value for x_i.
At each node, the solution space is partitioned into sub - solution spaces.
All paths from the root node to any node in the tree define the solution space, since any such path corresponds to a subset satisfying the explicit constraints.
The possible paths are (1), (1, 2), (1, 2, 3), (1, 2, 3, 4), (1, 2, 4), (1, 3, 4), (2), (2,3), and so on.
Thus, the left mot sub-tree defines all subsets containing w_1, the next sub-tree defines all subsets containing w_2 but not w_1, and so on.

Sample Implementation:

```
#include<stdio.h>
#include<conio.h>
int s[10], d, n, set[10], count = 0;
void display(int);
int flag = 0;
void main ()
{
   int subset(int,int);
   int i;

   printf ("Enter the number of elements in set\n");
   scanf ("%d", &n);
   printf ("Enter the set values\n");
   for (i = 0; i < n; ++i)
      scanf ("%d", &s[i]);
   printf ("Enter the sum\n");
   scanf ("%d", &d);
   printf ("The progrm output is\n");
   subset (0, 0);
   if (flag == 0)
      printf ("there is no solution");
   getch ();
}

int subset (int sum, int i)
{
   if (sum == d)
      {
         flag = 1;
         display (count);
```

```
            return 0;
        }
    if (sum > d || i >= n)
        return 0;
    else
        {
            set[count] = s[i];


            count++;
            subset (sum + s[i], i + 1);
            count--;
            subset (sum, i + 1);
        }
}

void
display (int count)
{
    int i;
    printf ("{");
    for (i = 0; i < count; i++)
        printf ("%d ", set[i]);
    printf ("}");

}
```

Output:

```
Enter the number of elements in set
5
Enter the set values
1
3
4
5
6
Enter the sum
9
The progrm output is
{1 3 5 }{3 6 }{4 5 }
```