Experiment No. 12

Title: Write a program to find solution to Sum of Subsets Problem using backtracking

Theory:

In the search for fundamental principles of algorithm design, backtracking Represents one of the most general technique. Many problems which deal with searching for a set of solutions or which ask for an optimal solution satisfying some constraints can be solved using the backtracking formulation. Given n distinct positive numbers (usually called weights) and we desire to find all combination of these numbers whose sum are m. This is called the sum of subset problem. For a node at level i the left child corresponds to $x_i = 1$ and the right to $x_i = 0$.

A simple choice for the bounding functions is $B_k(x_1, \ldots, x_k) =$ true iff

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

Clearly $x_1, \ldots, x_k$ cannot lead to an answer node if this condition is not satisfied. The bounding functions can be strengthened if we assume the w_i's are initially in nondecreasing order. In this case $x_1, \ldots, x_k$ cannot lead to an answer node if

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

The bounding functions we use are therefore

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

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

Since our algorithm will not make use of B_n, we need not be concerned by the appearance of w_{n+1} in this function. Although we have now specified all that is needed to directly use either of the backtracking schemas, a simpler algorithm results if we tailor either of these schemas to the problem at hand. This simplification results from the realization that if $x_k = 1$, then

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

Algorithm SumOfSub(s, k, r)
// Find all subsets of $w[1:n]$ that sum to m. The values of $x[j]$,
// $1 \le j < k$, 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 $w[1] \le m$ and $\sum_{i=1}^{n} w[i] \ge m$.
{
    // Generate left child. Note: $s + w[k] \le m$ 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 $w[j] > 0$, $1 \le j \le n$.
    else if $(s + w[k] + w[k+1] \le m)$
        then SumOfSub$(s + w[k], k+1, r - w[k])$;
    // Generate right child and evaluate B_k.
    if $((s + r - w[k] \ge m)$ **and** $(s + w[k+1] \le m))$ **then**
    {
        $x[k] := 0$;
        SumOfSub$(s, k+1, r - w[k])$;
    }
}

Complexity:

As there are 2^n possible subsets generated from n elements to be checked for summation. Hence the time complexity is given as $O(2^n)$.