**K. J. Somaiya College of Engineering, Mumbai-77**
(A Constituent College of Somaiya Vidyavihar University)
**Department of Computer Engineering**

**Batch:-** B-2 **Roll No:-** 16010122151

**Experiment No:-**08

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

## Title: Implementation of sum of subset Algorithm

**Objective:** To learn the Backtracking strategy of problem solving for Sum of subset

**CO to be achieved:**

CO 2 Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies.

**Books/ Journals/ Websites referred:**

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**

**Pre Lab/ Prior Concepts:**

Data structures, Concepts of algorithm analysis

**Historical Profile:**

Subset sum problem is to find subset of elements that are selected from a given set whose sum adds up to a given number K. We are considering the set contains non-negative values. It is assumed that the input set is unique (no duplicates are presented).

One way to find subsets that sum to K is to consider all possible subsets. A power set contains all those subsets generated from a given set. The size of such a power set is 2N.

***Input:***

A vector X={x1,x2… xn} for all n elements in the set where Xi=0 (element not added) or xi=1 (element added in the solution tuple).

***Output:***

Summation of the chosen numbers must be equal to given number M and one number can be used only once.

BACKTRACKING CONDITION

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

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

---

**New Concepts to be learned:**
Application of algorithmic design strategy to any problem, Backtracking method of problem solving Vs other methods of problem solving problem sum of subset and its applications.

---

**Algorithm:**

Algorithm sumOfSub(s, k, r)

{//It is assumed w[1]<=m and Sigma(i=1 to m)w[i]>=m

//generate the left child. Note: s+w(k)<=M since Bk-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]); //moving to next sub-problem.

Similarly, assume the array is presorted and we found one subset. We can generate next node excluding the present node only when inclusion of next node satisfies the constraints.

if ((S+ r- W[k]>=m)and (S+ W[k+1]<=m)) then//generate right {
//child and those satisfying 2 bounding functions

X{k]=0;

sumOfSub (S, k+1, r- W[k]);
}}

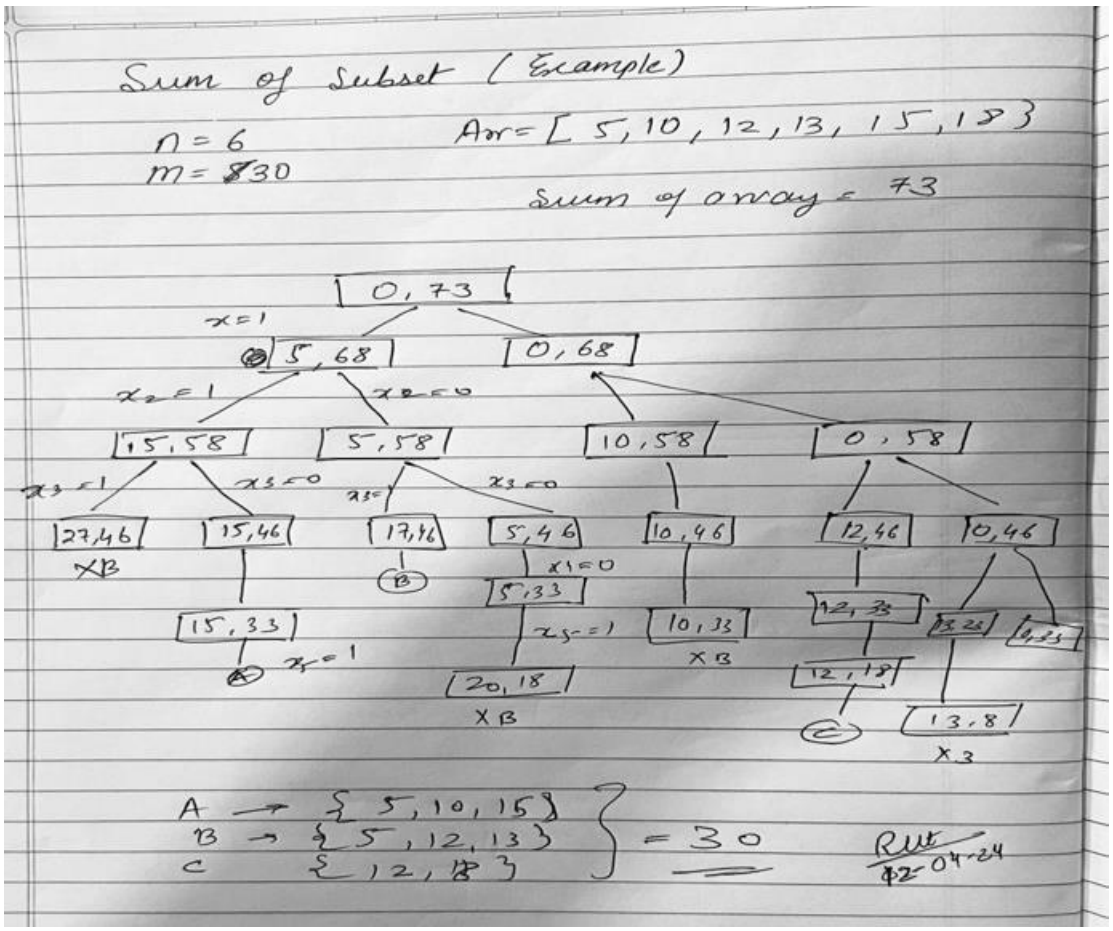
**Implementation(Code):**

```
def sum_of_subset(X, M):
    n = len(X)
    subset = []
    current_sum = 0
    remaining_sum = sum(X)

    def sum_of_sub(s, k, r):
        nonlocal current_sum, remaining_sum, subset
        # Base case: Subset found
        if s == M:
            print(subset)
            return

        # Explore left child: include the current element if possible
        if s + X[k] <= M:
            subset.append(X[k])
            sum_of_sub(s + X[k], k + 1, r - X[k])
            subset.pop()  # backtrack

        # Explore right child: exclude the current element
        if s + r - X[k] >= M and s + X[k + 1] <= M:
            sum_of_sub(s, k + 1, r - X[k])

    # Sort X (assuming it's not presorted)
    X.sort()

    # Call the recursive function starting with the first element
    sum_of_sub(current_sum, 0, remaining_sum)

# Main program
if __name__ == "__main__":
    # Input X as a list of integers separated by spaces
    X = list(map(int, input("Enter the list of integers:- ").split()))
    # Input M as the target sum
    M = int(input("Enter the target sum (M): "))

    print("The sum of subset solutions for the above list of integers and
target sum is as follows ")


    # Call the sum of subset function with the input X and M
    sum_of_subset(X, M)
```

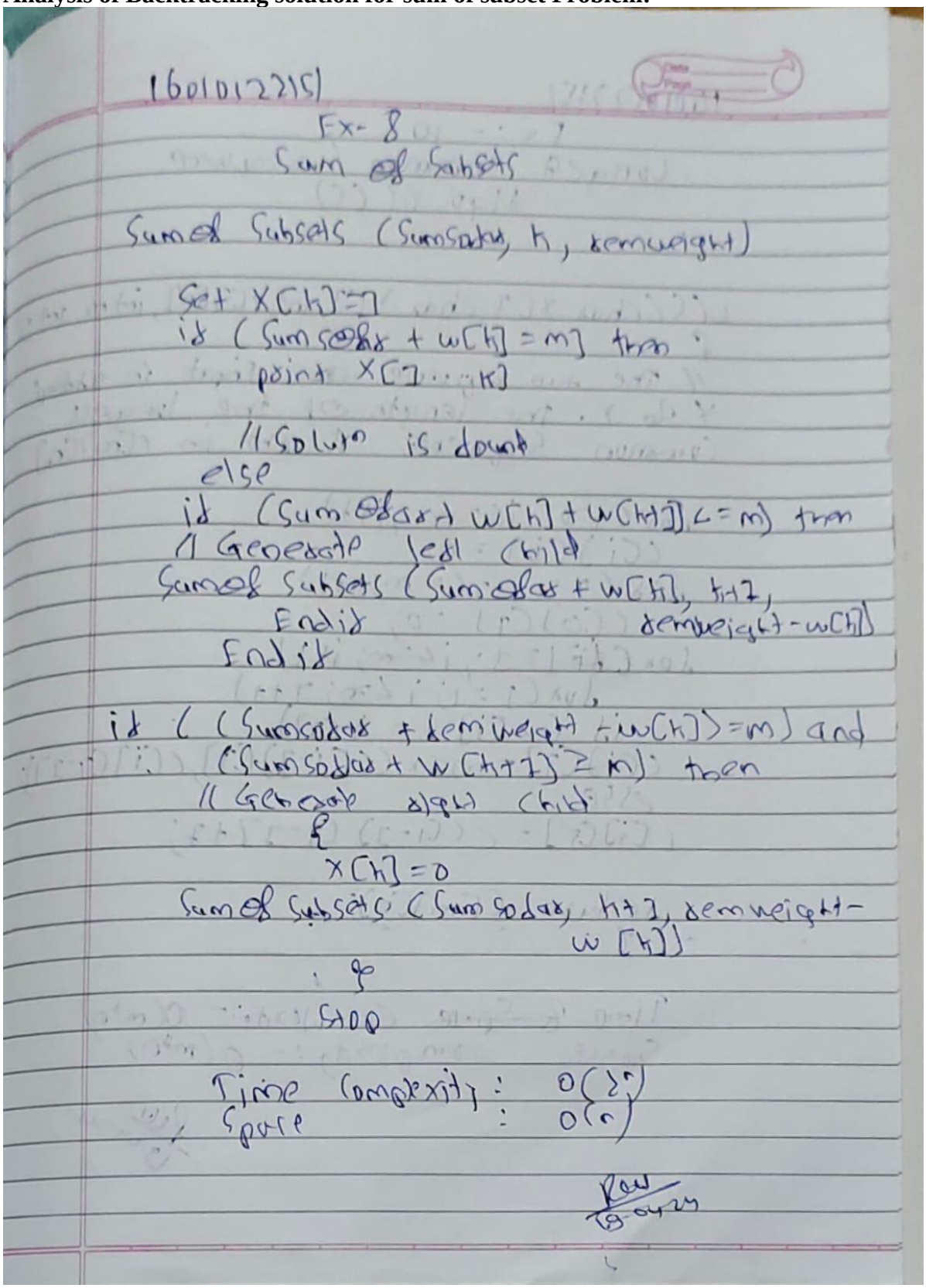
**Output:**

```
Enter the list of integers:- 5 10 12 13 15 18
Enter the target sum (M): 30
THE SUM OF SUBSET SOLUTIONS FOR THE ABOVE LIST OF INTEGERS AND TARGET
    SUM IS AS FOLLOWS
[5, 10, 15]
[5, 12, 13]
[12, 18]

=== Code Execution Successful ===
```

**Example sum of subset Problem along with state space tree:**



**Analysis of Backtracking solution for sum of subset Problem:-**



**Conclusion:-**

We have successfully used backtracking in both code and paper to solve the sum of subset algorithm, and we have produced accurate and comparable results.