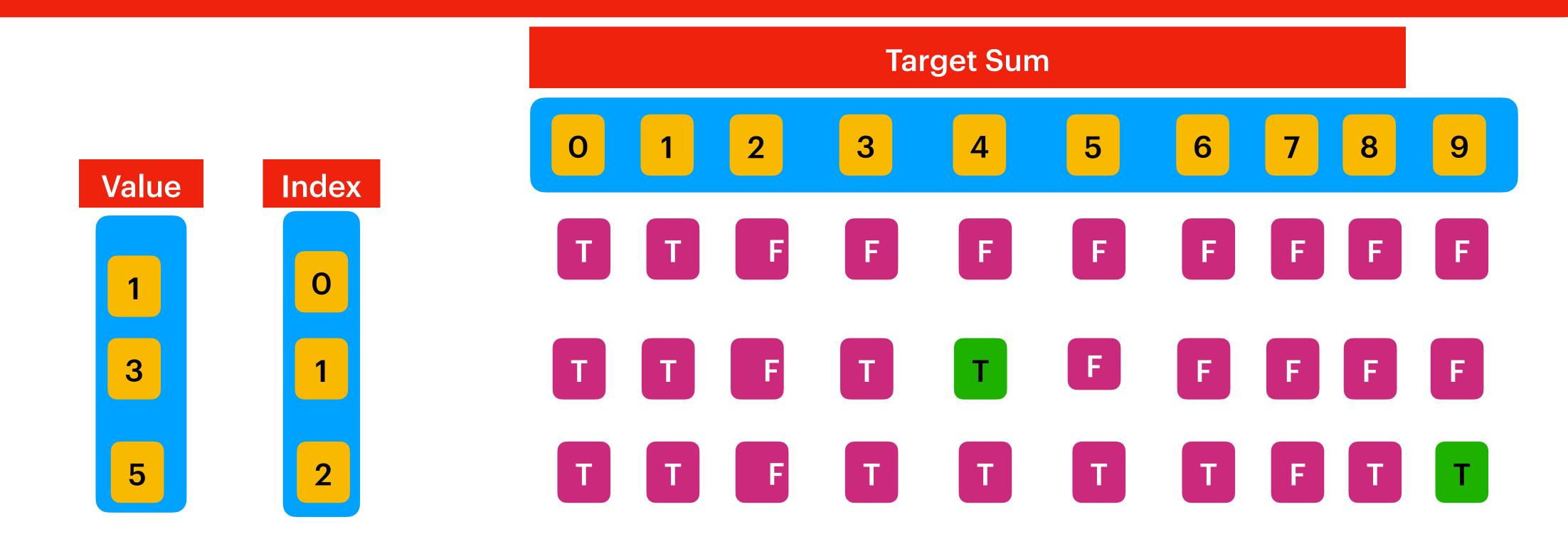
Problem Statement: Given a set of positive numbers, assume that always subset exit with given sum 'S', so print at least one possible subset.

Ex: {1,3,5}, S = 8 Expected Output: {3,5}

Its a Back Tracking problem



## Algorithm:

It's a Back Tracking problem. Move to the top of the record. In our example i=2, sum = 8 . dp[2][8] = True

1. In our use case when dp[i][sum] is true, we would need to check does it copied from previous problem or not.

we have two possibilities here

- 1) when dp[i-1][sum] is false and dp[i][sum] is true, it means we includes the current "i".
- 2) when dp[i-1][sum] is true and dp[i-1][sum-arr[i] is true, it means we includes the current "i"

Once we included current element then reduce the sum. All the iterations move on to previous index

## **Equal Subset Sum Partition:**

Given a set of positive numbers, find if we can partition it into two subsets such that the sum of elements in both the subsets is equal.

Input: {1, 2, 3, 4}

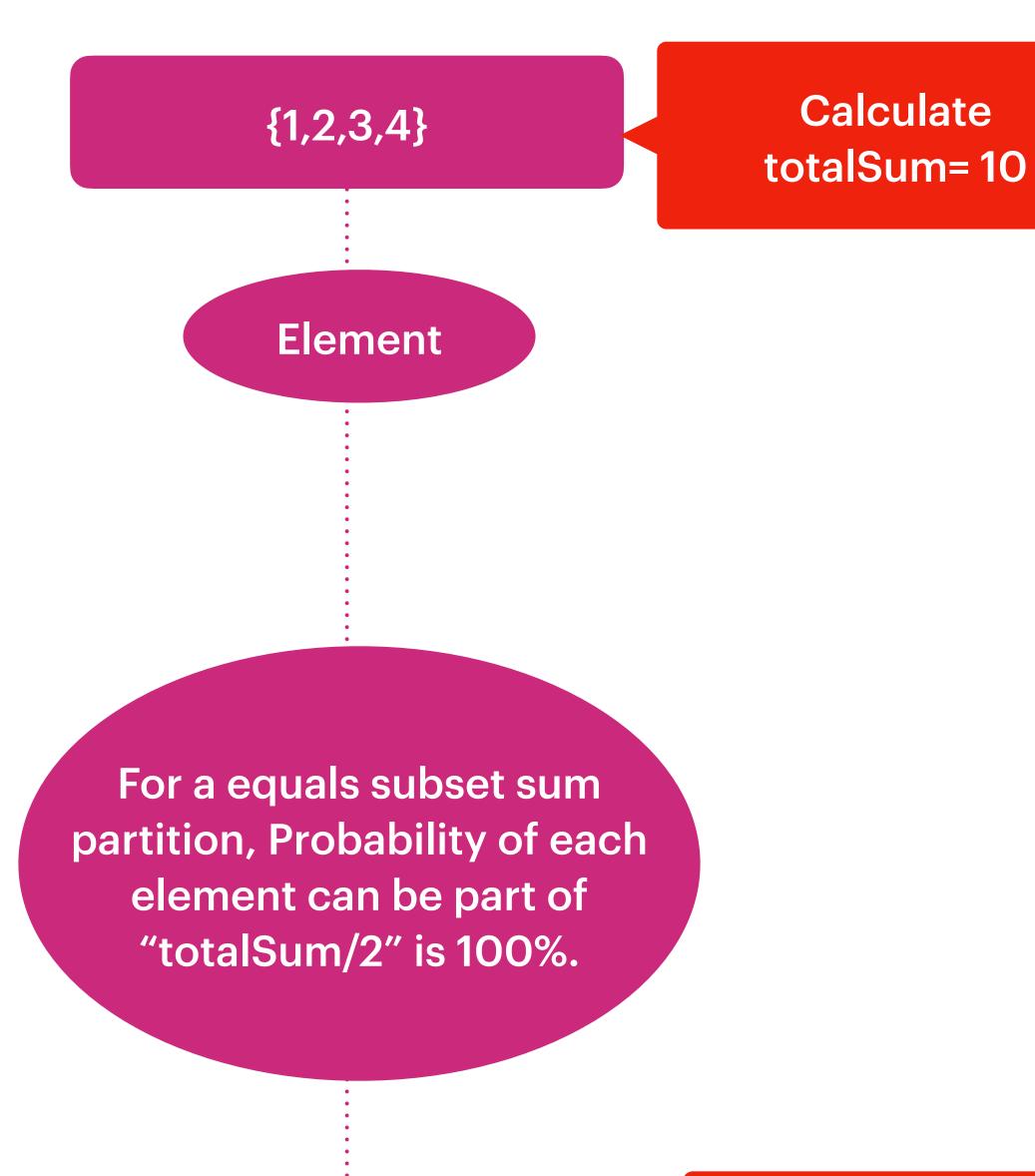
**Output: True** 

Explanation: The given set can be partitioned into two subsets with equal sum: {1, 4} & {2, 3}

Input: {2, 3, 4, 6}

**Output: False** 

Explanation: The given set cannot be partitioned into two subsets with equal sum.



targetSum = sum/2

Same as previous subset sum example. Here the targetSum = totalSum/2.