Print Subset of an Array

Problem Level: Hard

Problem Description:

Given an integer array (of length n), find and print all the subsets of the input array.

Note : The order of subsets is not important.

Sample Input 1:

```
3
15 20 12
```

Sample Output 1:

```
[] (this just represents an empty array, don't worry about the square brackets)
12
20
20 12
15
15 12
15 20
15 20 12
```

Approach to be followed:

All the subsets of an array can be found using recursion.

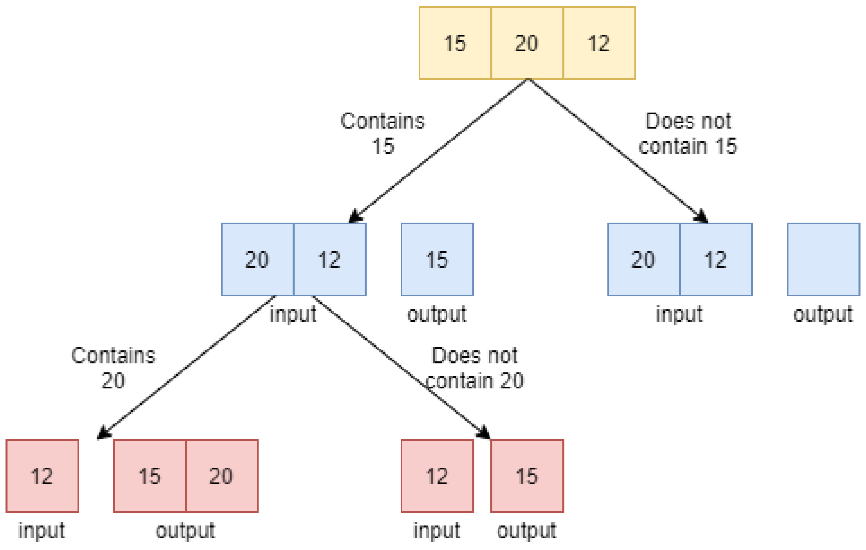
Suppose the input array is **"15 20 12".** For each element in the array, we have two options. First option is to find all the subsets, which do not include the element, and second is to find all the subsets which do include the element.

Since we want to print these subsets, we shall maintain an output array with us, which will store the output as we go along in the recursion. Using a variable **startIndex,** we will keep the account of which array elements are being fixed by us. When we reach a point, where this **startIndex** is equal to size of the array, we first print the output we have till now, and then end the recursion. For example, for the above mentioned array, the recursive calls made are-

1. Input array is [20, 12] and output array is [15]

2. Input array is [20, 12] and the output array is empty, [ ].

The first two levels of the recursive tree are shown below.



Steps:

1. Initialise an empty output array, and pass it in the recursive helper function, which will require **input**, **output**, and the **startIndex** (initially 0).
2. Inside it, initialise a **smallOutput** array to store output till now, with current size as 0.
3. Recursively call on outputs which do not contain the first element of the array, which means calling recursion from **startIndex** + 1.
4. Copy the contents of the **output** array into the **smallOutput.**
5. Append the element at the **startIndex** of the input array to the **smallOutput**.
6. Now, **smallOutput** stores the outputs which contain the element at the **startIndex**.
7. Recursively call on this new **smallOutput** to maintain those outputs which contain the fixed element from the **input** array.
8. For base case, if the **startIndex** equals the size of the input array, print the output till now, and return from the recursion.

Pseudo Code:

```
function helper(input[], startIndex, size,output[], oSize) {

     if startIndex equals size:

          loop from i = 0 till i = oSize:

               print output[i]

     return    // End Recursion
```

```
    declare smallOutput array
    smallOSize = 0
    helper(input, startIndex+1, size, output, oSize)

    int i = 0
    loop from i = 0 till i = oSize:
        smallOutput[i] = output[i]
    smallOutput[i] = input[startIndex]
    helper(input, startIndex+1, size, smallOutput, oSize+1)

function printSubsetsOfArray(input[], size)
    declare output array
    return helper(input, 0, size, output, 0)
```

Time Complexity: O(2^N), where N is the size of the input array. Since, for every index **i,** two recursive cases originate, hence the time complexity is **O(2^N)**.