10. Subset Sum I | Recursion

Given a list(Arr) of N integers, print sums of all subsets in it. Output should be printed in increasing order of sums.

Example 1:

```
Input:
N = 2
Arr = [2, 3]
Output:
0 2 3 5
Explanation:
When no elements is taken then Sum = 0.
When only 2 is taken then Sum = 2.
When only 3 is taken then Sum = 3.
When element 2 and 3 are taken then
Sum = 2+3 = 5.
```

N = 3

[3, 1, 2]

{ } → 0 {3 2} → 5

{3} → 3 {1 2} → 3

{1} → 1 {3 1 2} → 6

{2} → 2

{3 1} → 4

In Order ⇒ 0 1 2 3 3 4 5 6 [8 subset]

2^3

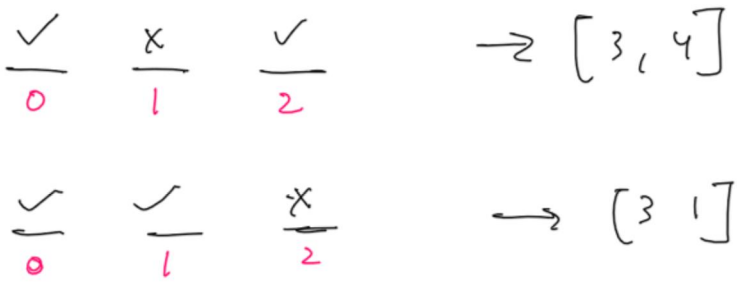
① Brute force :

Using Power Set

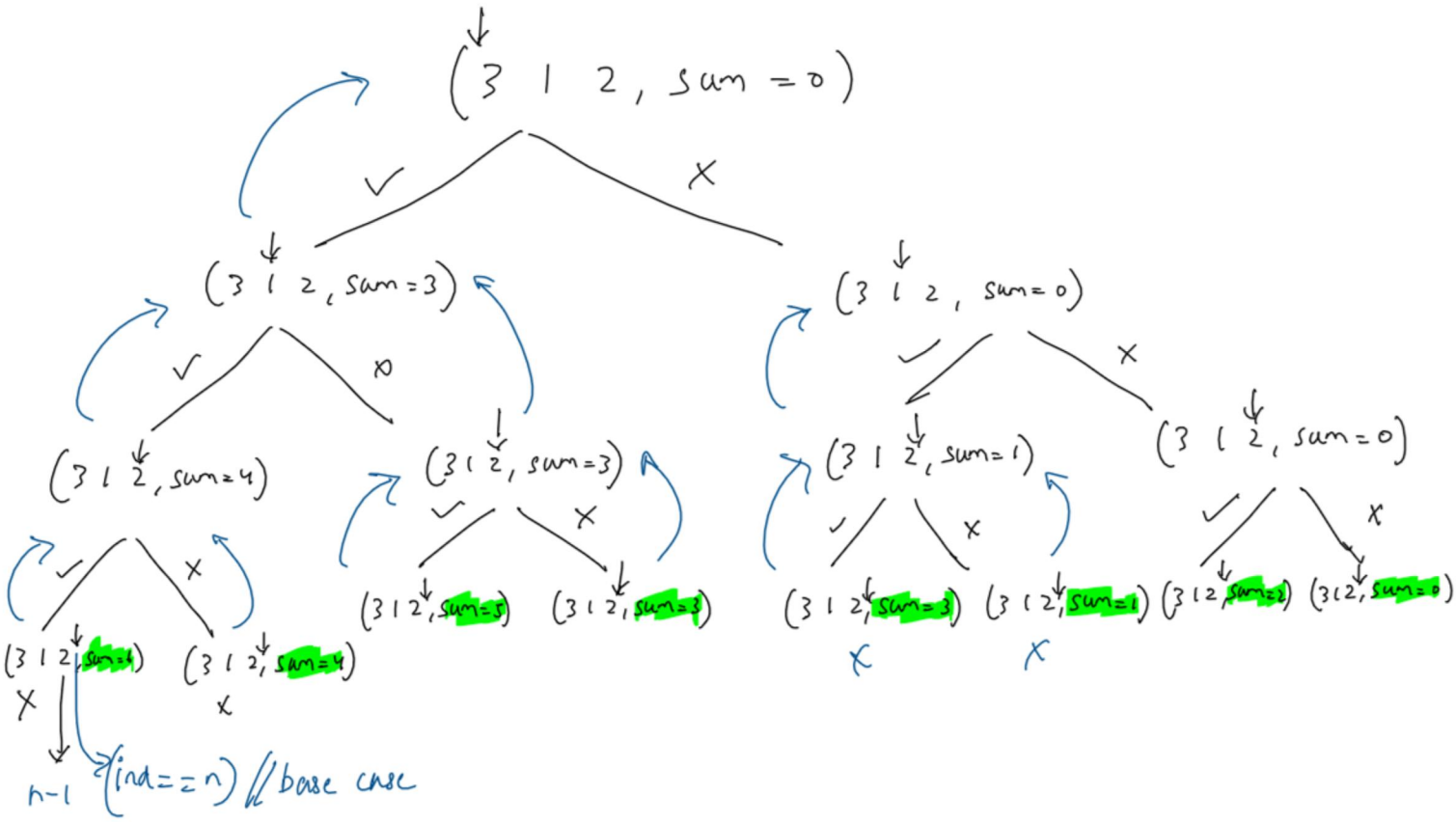
T.C ⇒ $O(2^N \times N)$

② Optimal Solution :

[3 1 4]



Intution: The main idea is that on every index you have two option either to select the element to add it to your subset (pick) or not select the element at the index and move to the next index (not_pick).

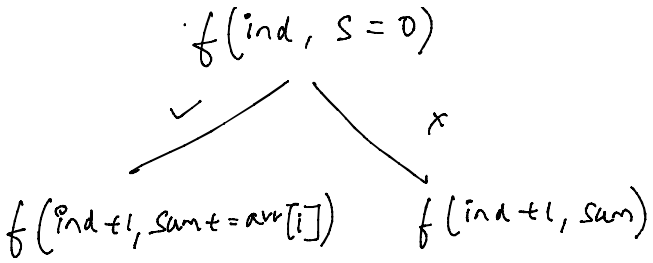


o/p ⇒ [6, 4, 5, 3, 3, 1, 2, 0]

↳ sort it ⇒ o/p

Pseudo Code:

```
f(ind, s = 0)
```



Base case: (ind == n)

T.C ⇒ $O(2^N + 2^N(\log(2^N)))$ → Y? $2^N \log 2^N$ NOT $O(N \log N)$ because of size is 2^N.

Sorting

S.C ⇒ $O(2^N)$

```java
import java.util.ArrayList;
import java.util.Collections;

class Solution{

    void func(int ind, int sum, ArrayList<Integer> arr, int N, ArrayList<Integer> sumSubset){
        // base case
        if(ind == N){
            sumSubset.add(sum);
            return;
        }

        //pick the element
        func(ind + 1, sum + arr.get(ind), arr, N, sumSubset);
        // not pick the element
        func(ind, sum, arr, N, sumSubset);
    }

    ArrayList<Integer> subsetSums(ArrayList<Integer> arr, int N){
        ArrayList<Integer> sumSubset = new ArrayList<>();
        func(ind: 0, sum: 0, arr, N, sumSubset);
        Collections.sort(sumSubset);
        return sumSubset;
    }
}
```

```cpp
#include<bits/stdc++.h>
using namespace std;

class Solution{
    public:
    void func(int ind, int sum, vector<int> &arr, int N, vector<int> &sumSubset){
        if(ind == N){
            sumSubset.push_back(sum);
            return;
        }
        // pick the element
        func(ind + 1, sum + arr[ind], arr, N, sumSubset);
        // not pick the element
        func(ind + 1, sum, arr, N, sumSubset);
    }


    public:
        vector<int> subsetSums(vector<int>arr, int N){
            vector<int> sumSubset;
            func(0, 0, arr, N, sumSubset);
            return sumSubset;
        }
};
```