Newton School of Technology

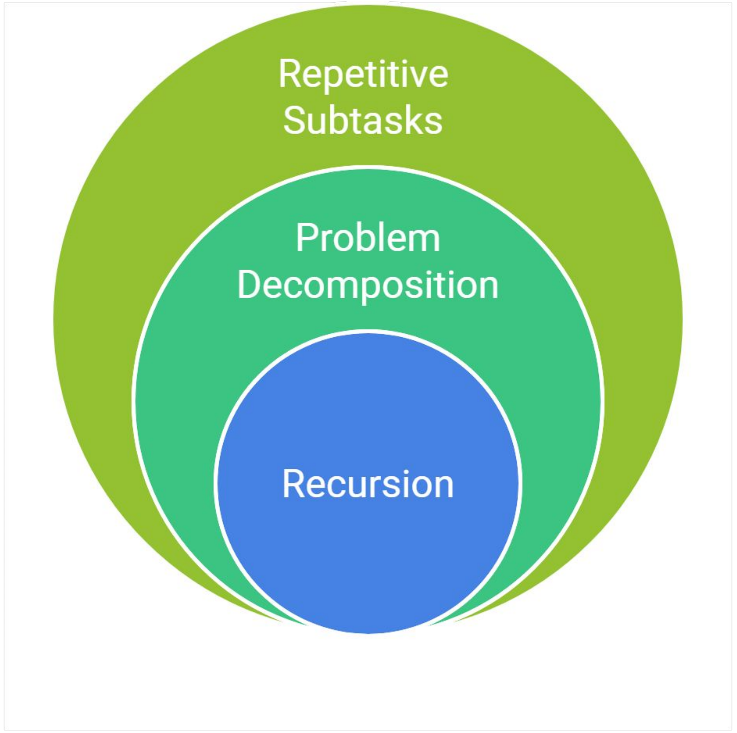
9

# Recursion – Part IV

by Gladden Rumao

## Lecture Agenda :

1. Count of subsets with sum k
2. Find all subsets with sum k



Newton School of Technology

## Count of Subsets with sum K

## Description

You are given **an array** nums of size n containing positive integers, and a **target sum k**.

Your task is to find the number of ways to select a subset of elements from the array such that the sum of the chosen elements is equal to the target sum k

4

## Example

Input : K = 5 ,

| | | | |
|---|---|---|---|
| 1 | 4 | 4 | 5 |

Output : 3

Explanation :

| | |
|---|---|
| 1 | 4 |

| | |
|---|---|
| 1 | 4 |

5

| |
|---|
| 5 |

Newton School of Technology

## How many different paths/possibilities we have?

To find the subset we have 2 options :

- Include current element
- Other option ?

## How many different paths/possibilities we have?

To find the subset we have 2 options :

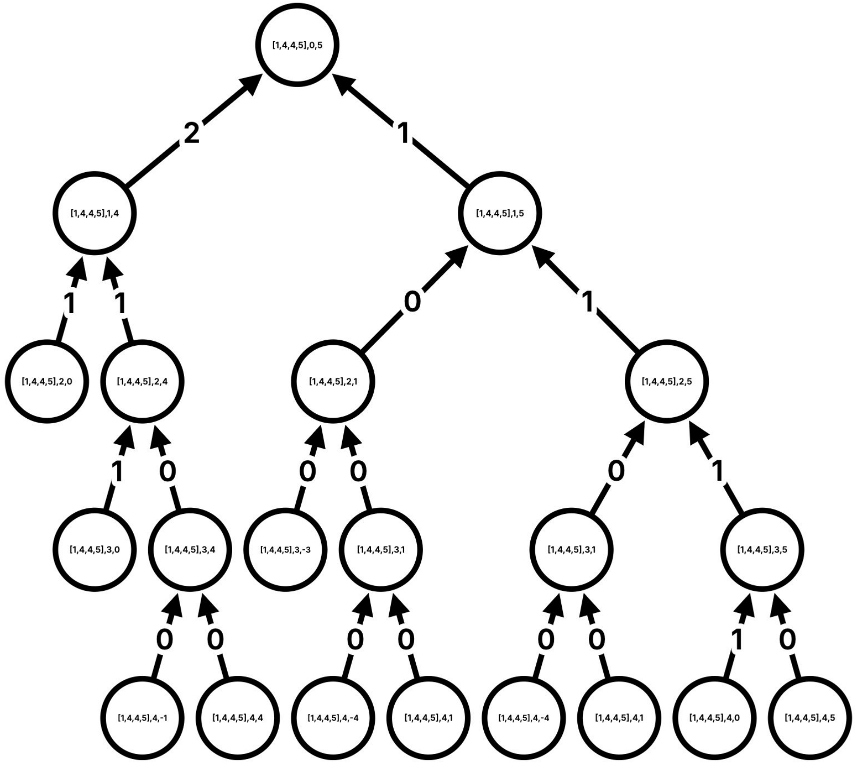
- Include current element
- Exclude current element

Newton School of Technology

## When should i stop or what will be my base case?

1. When subset sum value is equals to target value
2. When we reached the ending index

## Recursive Tree:



# Approach

**Base Case**

1. If the target sum k becomes 0, it means we have successfully found a valid subset. Return 1 to count this subset.
2. If the current index goes out of bounds (i.e., index==len(nums)) or the target sum becomes negative, return 0 as no valid subset is possible.

**Recursive Call**

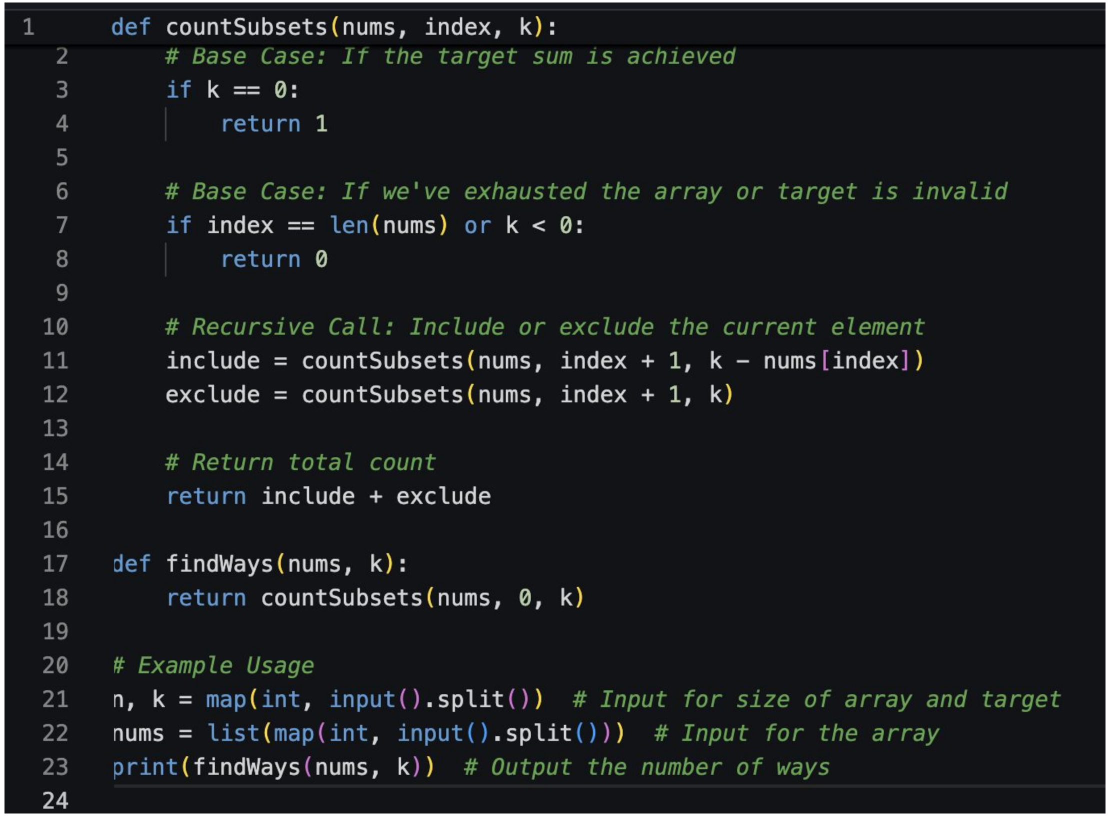
At each index, consider two possibilities:

1. **Include the current element** in the subset:
   - Subtract the value of the current element from k, and move to the next index.
2. **Exclude the current element** from the subset:
   - Keep k unchanged and move to the next index.

**Return Type**

The result for the current function call is the sum of these two recursive calls.The function returns an **integer**, representing the total number of valid subsets that achieve the target sum k.

# Code



```python
def countSubsets(nums, index, k):
    # Base Case: If the target sum is achieved
    if k == 0:
        return 1

    # Base Case: If we've exhausted the array or target is invalid
    if index == len(nums) or k < 0:
        return 0

    # Recursive Call: Include or exclude the current element
    include = countSubsets(nums, index + 1, k - nums[index])
    exclude = countSubsets(nums, index + 1, k)

    # Return total count
    return include + exclude

def findWays(nums, k):
    return countSubsets(nums, 0, k)

# Example Usage
n, k = map(int, input().split())  # Input for size of array and target
nums = list(map(int, input().split()))  # Input for the array
print(findWays(nums, k))  # Output the number of ways

```

## Code Analysis

**Time Complexity :** O(2^N)

**Space Complexity :** O(N)

Newton School of Technology

## Find all subsets with target sum equals K

## Description

You are given an integer array arr of size n and a target integer target.

Your task is to find and Return all the subsets of arr such that the sum of the elements in each subset is equal to target.

## Example

Input : K = 6,

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

Output :

| | | |
|---|---|---|
| 1 | 2 | 3 |

| | |
|---|---|
| 2 | 4 |

15

## How can we store the subset for different options?

For including can I add current element into list ?

For excluding can I remove the current element that we just added ?

Newton School of Technology

## When should i stop or what will be my base case?

1. When subset sum value is equals to target value
2. When we reached the ending index

## Approach

**Base Case**

1. **Target Sum Reached**: If the current sum of the elements in the current_subset equals the target, it means we've found a valid subset. We should add this subset to the result list.
   - Condition: if current_sum == target:
   - Action: Add current_subset to result.
2. **End of Array**: If we've reached the end of the array (i.e., the index exceeds the length of the array), the recursion should terminate without adding any further subsets.
   - Condition: if index == len(arr):
   - Action: Return (end of recursion).

**Recursive Call**

At each index, we have two recursive choices:

1. **Include the current element** in the subset:
   - Add the current element (arr[index]) to the current_subset, and call the recursive function for the next index, increasing the sum by arr[index].
   - Recursive Call: recursive(index + 1, current_subset, current_sum + arr[index])

## Approach

2) **Exclude the current element** from the subset:

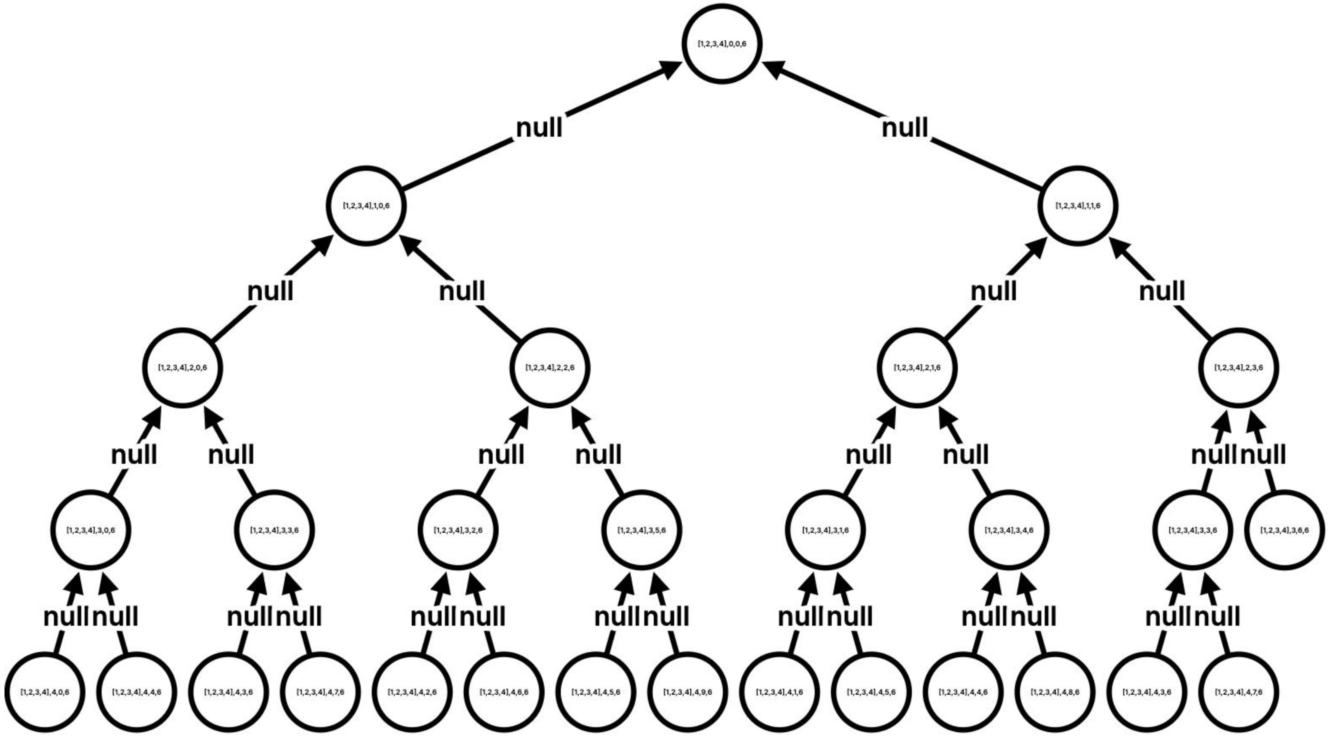
- Remove the last element from the current_subset (backtrack), and call the recursive function for the next index, keeping the sum unchanged.
- We are not including the current element into current_sum. So, needs to remove it from the current_subset also to find the new subset.
- Recursive Call: recursive(index + 1, current_subset, current_sum)

3) **Return Type**

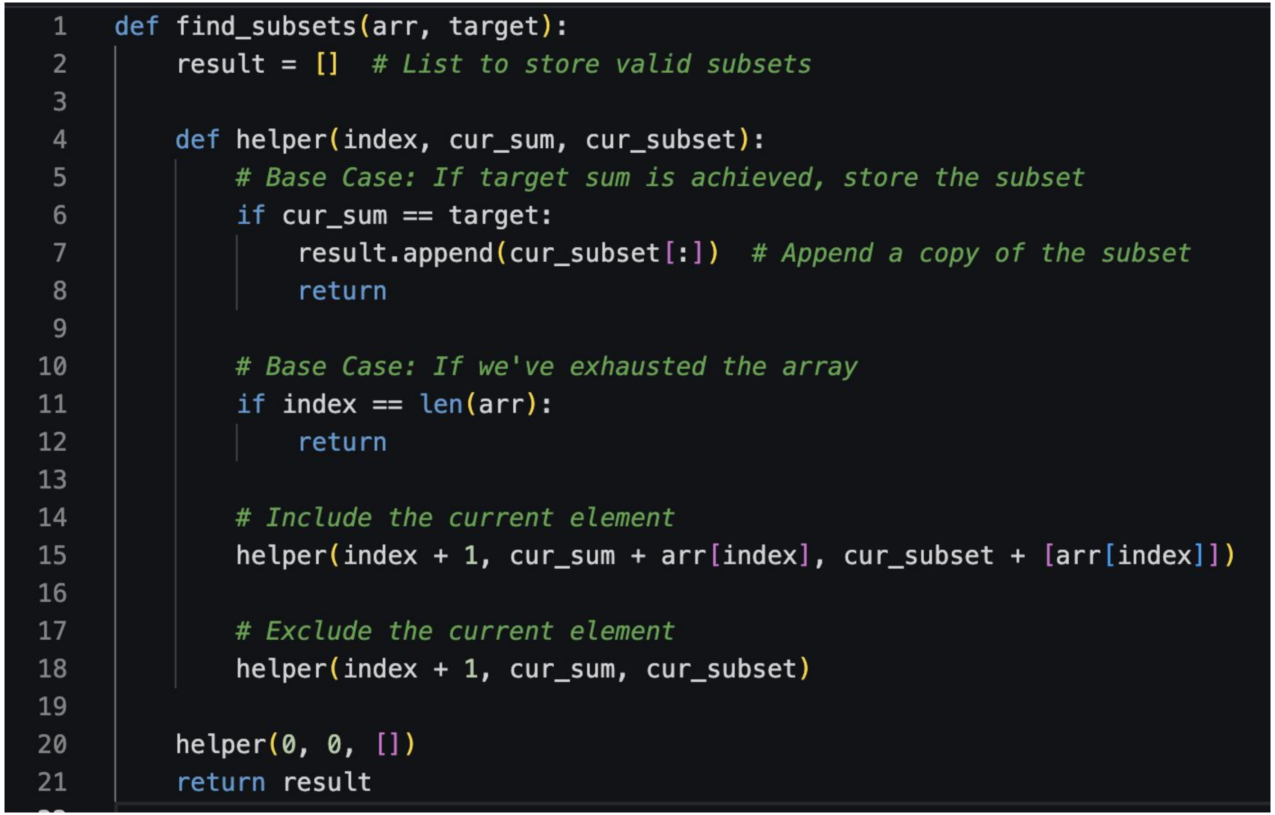
The function returns a list of lists (result), where each inner list represents a subset whose sum equals the target.

- **Type**: List[List[int]]
    - Each element of the list (result) is a list that represents a subset of elements whose sum equals the target.

## Recursive Tree:



# Code



```python
def find_subsets(arr, target):
    result = []  # List to store valid subsets

    def helper(index, cur_sum, cur_subset):
        # Base Case: If target sum is achieved, store the subset
        if cur_sum == target:
            result.append(cur_subset[:])  # Append a copy of the subset
            return

        # Base Case: If we've exhausted the array
        if index == len(arr):
            return

        # Include the current element
        helper(index + 1, cur_sum + arr[index], cur_subset + [arr[index]])

        # Exclude the current element
        helper(index + 1, cur_sum, cur_subset)

    helper(0, 0, [])
    return result
```

## Code Analysis

Newton School of Technology

**Time Complexity :** O(2^N * N)

**Space Complexity :** O(2^N * N)

## Summary

In this lecture we studied about problems in which we have made two recursive calls and two parameters are changing

In first problem , **count subsets with target sum K** , index and K are changing parameters

In second problem , **find all subsets with target sum K** , index and K are changing parameters

## Thanks for Attending!