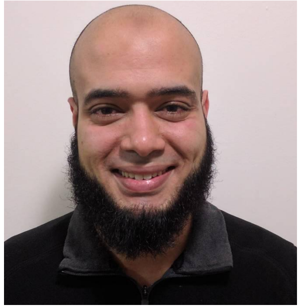
# Algorithms

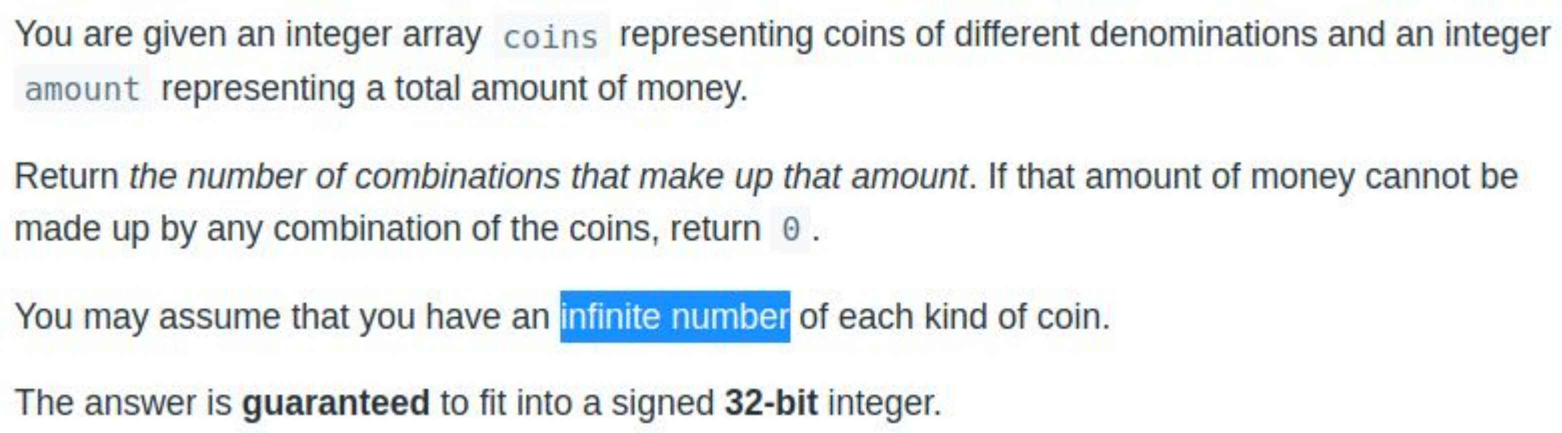
# DP Homework 1

**Mostafa S. Ibrahim**
*Teaching, Training and Coaching for more than a decade!*

*Artificial Intelligence & Computer Vision Researcher*
*PhD* from Simon Fraser University - Canada
*Bachelor / Msc* from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



# Problem #1: LeetCode 518 - Coin Change 2



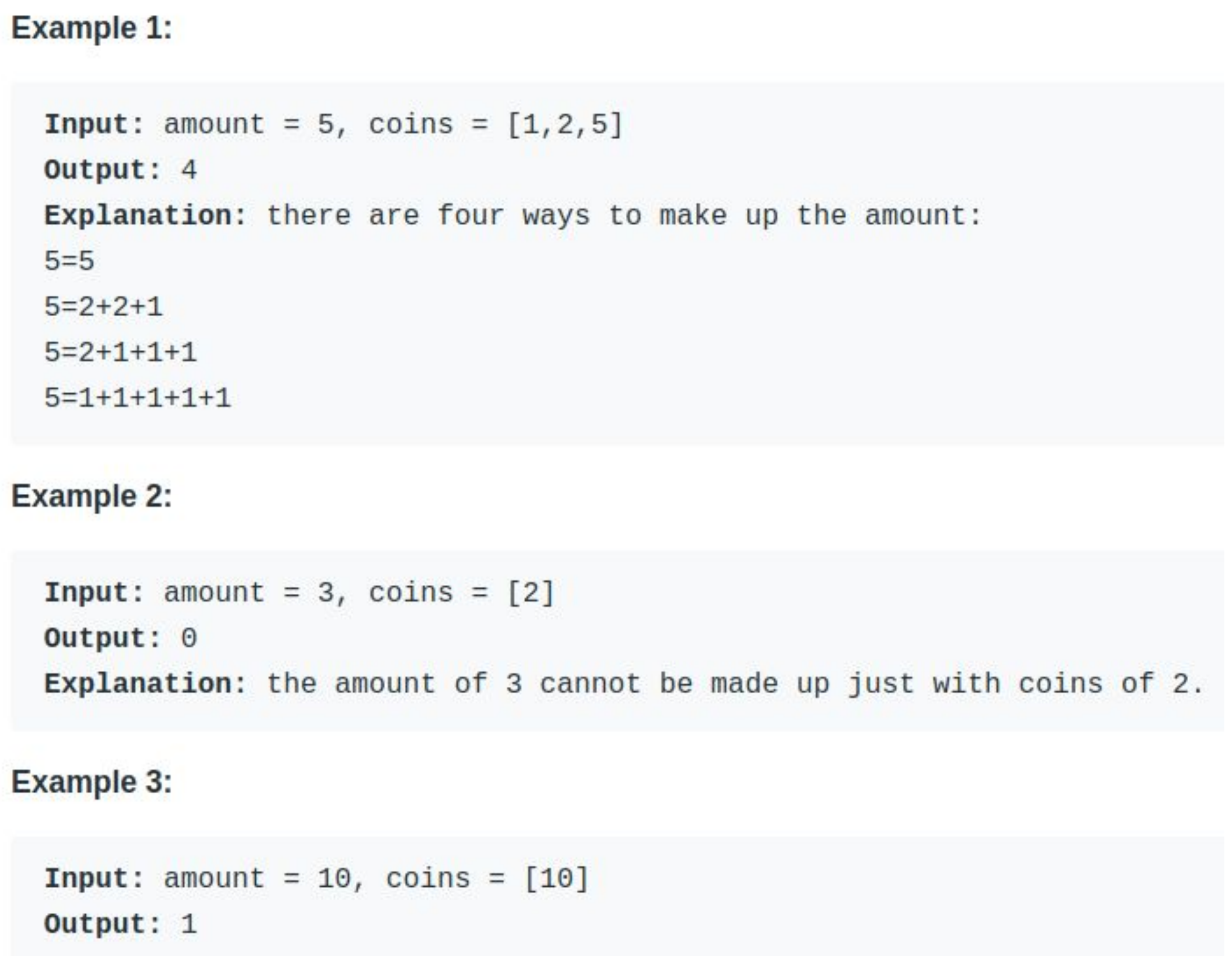
You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return *the number of combinations that make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `0`.

You may assume that you have an infinite number of each kind of coin.

The answer is **guaranteed** to fit into a signed **32-bit** integer.

- All the values of coins are unique. 300 max coins with values [1, 5000]
- Note, this problem is similar to subset sum with following changes
  - We can take the same items 0 or many times
  - We want to count possible cases NOT just if possible to make a sum
    - Coin Change 1 is about boolean possibility



### Example 1:

```
Input: amount = 5, coins = [1,2,5]
Output: 4
Explanation: there are four ways to make up the amount:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1
```

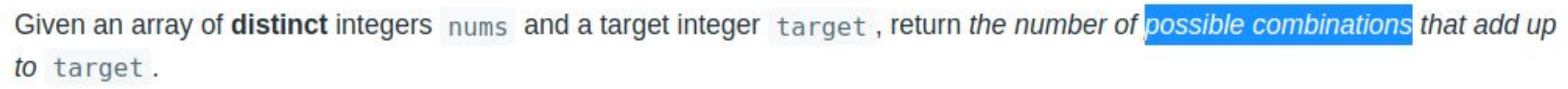
### Example 2:

```
Input: amount = 3, coins = [2]
Output: 0
Explanation: the amount of 3 cannot be made up just with coins of 2.
```

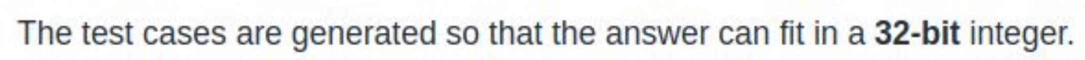
### Example 3:

```
Input: amount = 10, coins = [10]
Output: 1
```

# Problem #2: LeetCode 377 - Combination Sum IV

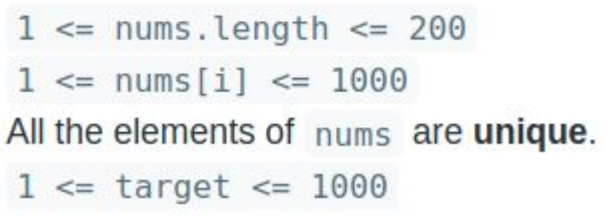


Given an array of **distinct** integers `nums` and a target integer `target`, return *the number of possible combinations that add up to* `target`.



The test cases are generated so that the answer can fit in a **32-bit** integer.

- The difference between this problem and previous one is the permutations matter
    - Now [1+1+2], [2+1+1], [1+2+1] are 3 possible ways not just one



`1 <= nums.length <= 200`

`1 <= nums[i] <= 1000`

All the elements of `nums` are **unique**.

`1 <= target <= 1000`

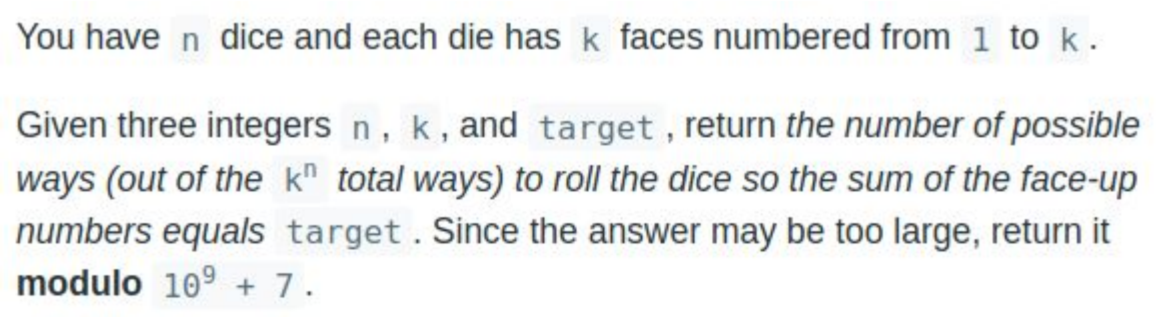
### Example 1:

```
Input: nums = [1,2,3], target = 4
Output: 7
Explanation:
The possible combination ways are:
(1, 1, 1, 1)
(1, 1, 2)
(1, 2, 1)
(1, 3)
(2, 1, 1)
(2, 2)
(3, 1)
Note that different sequences are counted as different combinations.
```

### Example 2:

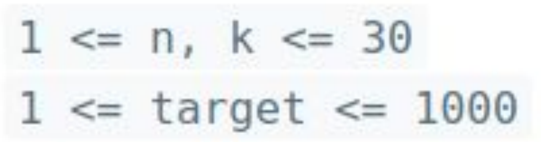
```
Input: nums = [9], target = 3
Output: 0
```

# Problem #3: LeetCode 1155 - Number of Dice Rolls With Target Sum

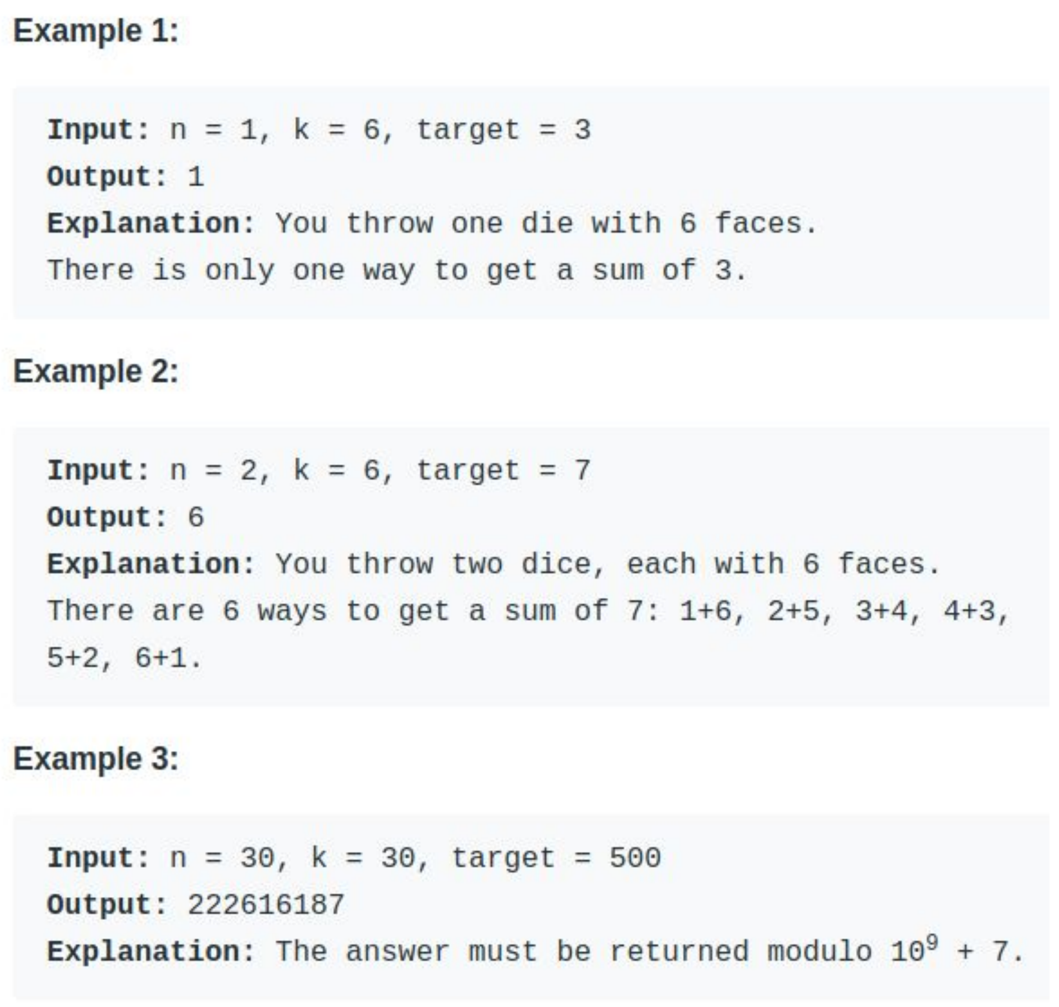


You have `n` dice and each die has `k` faces numbered from `1` to `k`.

Given three integers `n`, `k`, and `target`, return *the number of possible ways (out of the* $k^n$ *total ways) to roll the dice so the sum of the face-up numbers equals* `target`. Since the answer may be too large, return it **modulo** $10^9 + 7$.



```
1 <= n, k <= 30
1 <= target <= 1000
```



### Example 1:

**Input:** n = 1, k = 6, target = 3
**Output:** 1
**Explanation:** You throw one die with 6 faces.
There is only one way to get a sum of 3.

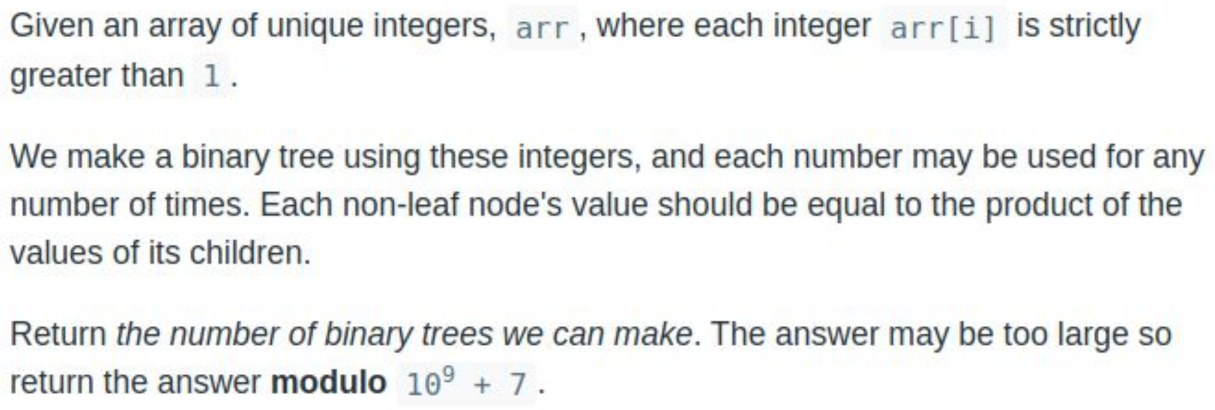
### Example 2:

**Input:** n = 2, k = 6, target = 7
**Output:** 6
**Explanation:** You throw two dice, each with 6 faces.
There are 6 ways to get a sum of 7: 1+6, 2+5, 3+4, 4+3, 5+2, 6+1.

### Example 3:

**Input:** n = 30, k = 30, target = 500
**Output:** 222616187
**Explanation:** The answer must be returned modulo $10^9 + 7$.

# Problem #4: LeetCode 823 - Binary Trees With Factors

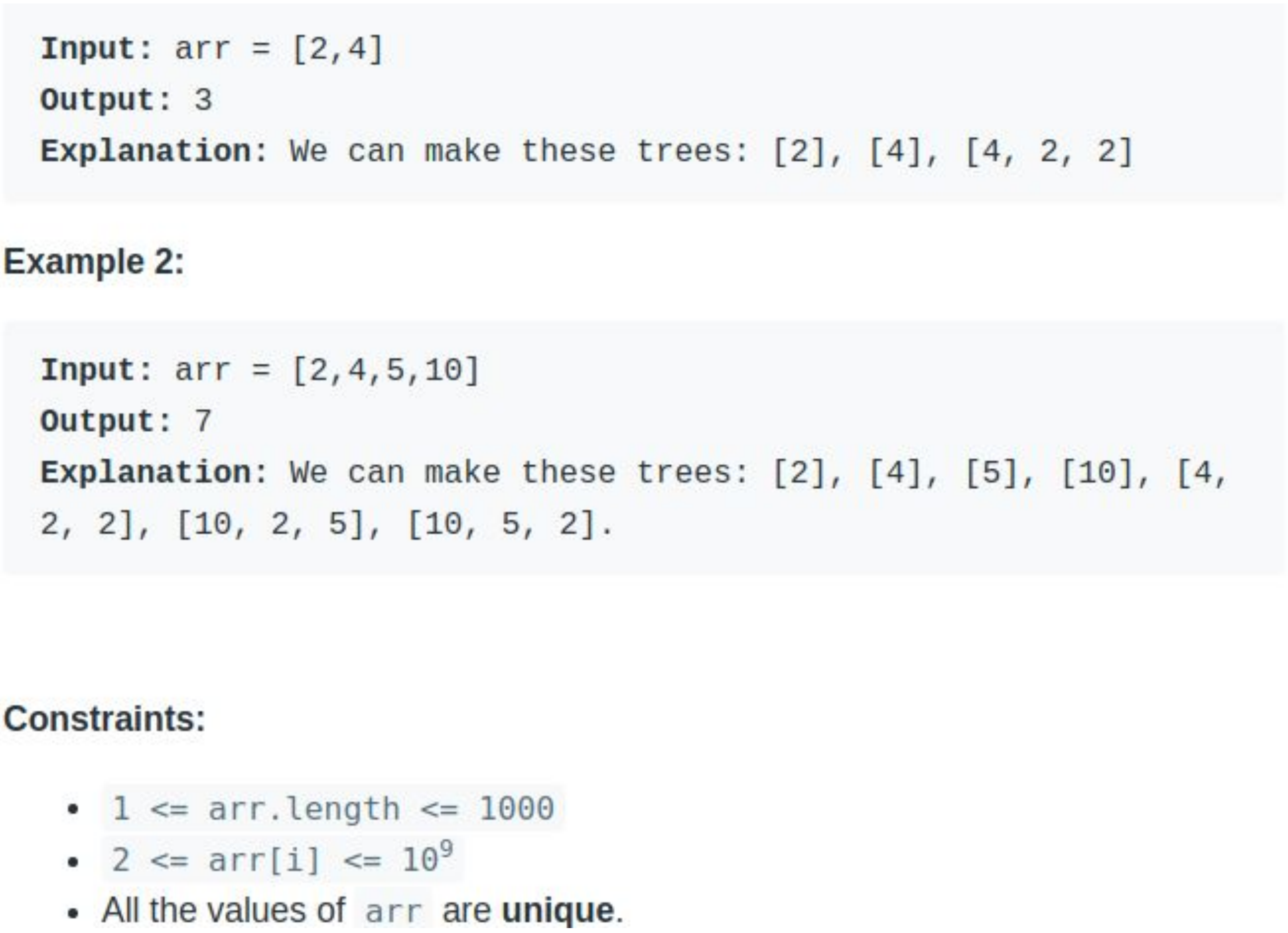


Given an array of unique integers, `arr`, where each integer `arr[i]` is strictly greater than `1`.

We make a binary tree using these integers, and each number may be used for any number of times. Each non-leaf node's value should be equal to the product of the values of its children.

Return *the number of binary trees we can make*. The answer may be too large so return the answer **modulo** $10^9 + 7$.

- Hint: define cnt_trees(x): Return how many trees that has x as a **root**
- Carefully handle overflows (for languages like C++, Java)



```
Input: arr = [2,4]
Output: 3
Explanation: We can make these trees: [2], [4], [4, 2, 2]
```

**Example 2:**

```
Input: arr = [2,4,5,10]
Output: 7
Explanation: We can make these trees: [2], [4], [5], [10], [4,
2, 2], [10, 2, 5], [10, 5, 2].
```

**Constraints:**

- `1 <= arr.length <= 1000`
- $2 <= arr[i] <= 10^9$
- All the values of `arr` are **unique**.

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."