Master algorithms together on Binary Search! Create a room, invite your friends, and race to finish the problems.

Daily Coding Problem #272

Problem

This problem was asked by Spotify.

Write a function, `throw_dice(N, faces, total)`, that determines how many ways it is possible to throw `N` dice with some number of faces each to get a specific total.

For example, `throw_dice(3, 6, 7)` should equal `15`.

Solution

Note that with one die, there are only two cases to consider. Namely, if the total is less than or equal to the number of faces, there is one way to throw that total; otherwise, it is impossible.

For two dice, we must examine all the possible outcomes from the first roll, and see if there is a corresponding second roll that sums up to the total. Generalizing this to arbitrary numbers of dice, we can use the following recursive relationship:

```
f(N, total) = Σ f(N - 1, total - m) for m in M
```

In other words, for any new die, we can sum up all the ways that die could form the total by adding one of its faces to the accumulated total of the previous dice. Here is how we could code this up:

could code this up:

```
def throw_dice(n, faces, total):
    if n == 1:
        return 1 if total <= faces else 0

    ways = 0
    for face in range(1, min(total, faces + 1)):
        ways += throw_dice(n - 1, faces, total - face)

    return ways
```

Since for each face, we must recursively call our function `N` times, this will take $O(M^N)$ time.

To improve the efficiency of our solution, we can use dynamic programming. We will start by creating a matrix that stores the number of ways of using `N` dice to make a given total.

Initially, as in the recursive solution, we only know the solution for one die: this will be `1` if the total is less than or equal to the number of faces, else `0`.

Then, looping through each dice, each potential total, and each face value, we accumulate the ways to get from the old total to the new by adding a single face value. Eventually, the bottom right corner of our matrix will store the result for `N` dice and our required total.

```
def throw_dice(n, faces, total):
    ways = [[0 for _ in range(total + 1)] for _ in range(n)]

    for t in range(1, total + 1):
        ways[0][t] = 1 if t <= faces else 0

    for dice in range(1, n):
        for t in range(1, total + 1):
            for face in range(1, min(t, faces + 1)):
                ways[dice][t] += ways[dice - 1][t - face]

    return ways[-1][-1]
```

Since we must loop through each die, each total, and each face value, the time and space complexity will be `O(N * M * T)`.

© Daily Coding Problem 2019

Privacy Policy

Terms of Service

Press