2787. Ways to Express an Integer as Sum of Powers

Medium | Topics | Companies | Hint

Given two **positive** integers n and x.

Return *the number of ways* n *can be expressed as the sum of the* x^{th} *power of* ***unique*** *positive integers, in other words, the number of sets of unique integers* $[n_1, n_2, ..., n_k]$ *where* $n = n_1^x + n_2^x + ... + n_k^x$.

Since the result can be very large, return it modulo $10^9 + 7$.

For example, if $n = 160$ and $x = 3$, one way to express n is $n = 2^3 + 3^3 + 5^3$.

Example 1:

> **Input:** n = 10, x = 2
> **Output:** 1
> **Explanation:** We can express n as the following: $n = 3^2 + 1^2 = 10$.
> It can be shown that it is the only way to express 10 as the sum of the 2^{nd} power of unique integers.

DP

n_i range = [1: n * (1/k)]

```
for i in range(n_i):
        set.add(i **k)
```

dp sum

```python
class Solution:
    def numberOfWays(self, n: int, x: int) -> int:
        MOD = 10 ** 9 + 7
        max_possible = math.ceil(n ** (1/x))
        powers = [m ** x for m in range(1, max_possible + 1)]
        #print(max_possible, powers)

        dp = [0] * (n+1)
        dp[0] = 1

        for p in powers:
            for s in range(n, p-1, -1):
                dp[s] = (dp[s] + dp[s-p]) % MOD

        #print(dp)

        return dp[n]
```

max_possi == ceil() not int()
    if == int(), got floating point precision problem
    eg(n=64, x=3, if int(), max_possi = 3.9999999..)

sol1: ceil(), with extra powers, take more memory but safe

sol2: int(round(...))

sol3: avoid float root entirely:

```
max_p = 1
while (max_p + 1) ** x <= n:
        max_p += 1
```