2698. Find the Punishment Number of an Integer

Given a positive integer `n`, return *the **punishment number*** of `n`.

The **punishment number** of `n` is defined as the sum of the squares of all integers `i` such that:

- `1 <= i <= n`
- The decimal representation of `i * i` can be partitioned into contiguous substrings such that the sum of the integer values of these substrings equals `i`.

Example 1:

```
Input: n = 10
Output: 182
Explanation: There are exactly 3 integers i in the range [1, 10] that satisfy the
conditions in the statement:
- 1 since 1 * 1 = 1
- 9 since 9 * 9 = 81 and 81 can be partitioned into 8 and 1 with a sum equal to 8 +
1 == 9.
- 10 since 10 * 10 = 100 and 100 can be partitioned into 10 and 0 with a sum equal
to 10 + 0 == 10.
Hence, the punishment number of 10 is 1 + 81 + 100 = 182
```

Example 2:

```
Input: n = 37
Output: 1478
Explanation: There are exactly 4 integers i in the range [1, 37] that satisfy the
conditions in the statement:
- 1 since 1 * 1 = 1.
- 9 since 9 * 9 = 81 and 81 can be partitioned into 8 + 1.
- 10 since 10 * 10 = 100 and 100 can be partitioned into 10 + 0.
- 36 since 36 * 36 = 1296 and 1296 can be partitioned into 1 + 29 + 6.
Hence, the punishment number of 37 is 1 + 81 + 100 + 1296 = 1478
```

Constraints:

- `1 <= n <= 1000`

2698. Find the Punishment Number of an Integer

Overview

We are given a positive integer `n`, and our task is to return its **punishment number**.

The **punishment number** is the sum of the squares of all integers `i` that satisfy two conditions:

1. **Range**: `i` must be within the range `1 <= i <= n`.
2. **Partition**: The decimal representation of `i * i` can be partitioned into contiguous substrings such that the sum of these substrings equals `i`.

In other words, for each integer in the range `[1, n]`, we check whether the digits of its squared value can be split so that the resulting sum matches the original number.

Let's look at examples where the squared integer's digits can be partitioned as described:

Number	Squared	Partition	Summation
1	1 * 1 = 1	1	1 = **1**
9	9 * 9 = 81	81	8 + 1 = **9**
10	10 * 10 = 100	100	10 + 0 = **10**
36	36 * 36 = 1296	1296	1 + 29 + 6 = **36**

As we can see, multiple ways exist to split the digits of a squared integer, leading to different summations. Our goal is to find at least one valid partition for each integer in the given range and sum up the squares of all numbers that satisfy the condition.

2698. Find the Punishment Number of an Integer

```
/*
    Recursion: find all combinations of non-overlapping substrings
    // Time complexity: O(n 8^{log_10 n})
    // Space complexity: O(log_10 n)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
class Solution{
    public:
        int punishmentNumber(int n){
            // Recursive function to get all possible partitions
            // Time complexity: O(8^{log_10 n})
            // Space complexity: O(4 log_10 n+2)=O(log_10 n)
            auto solve=[&](std::string& s,int index,int target,auto& self)->bool{
                if(index==s.size()) return target==0;
                int sum=0;
                for(int split=index;split<s.size();++split){
                    sum=sum*10+(s[split]-'0');
                    if(sum>target) return false;
                    if(self(s,split+1,target-sum,self)) return true;
                }
                return false;
            };

            int ans=0;
            for(int i=1;i<=n;++i){
                int sq=i*i;
                std::string sq_str=std::to_string(sq);
                // If a partition sum is equal to i, then and i*i to the answer
                if(solve(sq_str,0,i,solve)) ans+=sq;
            }
            return ans;
        }
};
```