# 题目

给你一个整数 n，请你帮忙计算并返回该整数「各位数字之积」与「各位数字之和」的差。

示例 1：

输入：n = 234

输出：15

解释：

各位数之积 = 2 \* 3 \* 4 = 24

各位数之和 = 2 + 3 + 4 = 9

结果 = 24 - 9 = 15

示例 2：

输入：n = 4421

输出：21

解释：

各位数之积 = 4 \* 4 \* 2 \* 1 = 32

各位数之和 = 4 + 4 + 2 + 1 = 11

结果 = 32 - 11 = 21

提示：

1 <= n <= 10^5

# 分析

要解决“计算整数各位数字之积与和的差”的问题，核心是逐位提取数字并分别计算乘积与总和，最终返回两者差值。

解题思路：

1、逐位提取数字：通过循环对整数n进行“取余”和“整除”操作——n % 10获取当前最后一位数字，n = n / 10移除最后一位，直到n变为0（所有数字处理完毕）。

2、计算乘积与总和：

- 初始化product为1（乘法单位元，确保初始值不影响累乘结果），每次提取数字后将其乘入product。

- 初始化sum为0（加法单位元，确保初始值不影响累加结果），每次提取数字后将其加入sum。

3、返回结果：最终结果为product - sum。

代码实现：

class Solution {

public:

int subtractProductAndSum(int n) {

int product = 1; // 各位数字之积，初始为1（乘法不影响结果）

int sum = 0; // 各位数字之和，初始为0（加法不影响结果）

while (n > 0) {

int digit = n % 10; // 提取当前最后一位数字

product \*= digit; // 累乘到乘积

sum += digit; // 累加至总和

n = n / 10; // 移除最后一位，继续处理剩余数字

}

return product - sum; // 返回积与和的差

}

};

代码解释

- 变量初始化：product初始为1（避免初始值0导致乘积始终为0），sum初始为0（符合加法逻辑）。

- 循环逻辑：每次循环提取一位数字，更新乘积和总和后缩小n，直到所有数字处理完毕。

- 结果计算：直接返回“乘积 - 总和”，无需额外复杂处理。

复杂度分析

- 时间复杂度：O(k)，k为整数n的位数（如n=10^5时k=6），循环次数等于位数，每次操作均为O(1)。

- 空间复杂度：O(1)，仅使用product、sum、digit 3个常数变量，无额外空间开销。