250710奇怪的数（题解）

> 题目链接：
> https://www.matiji.net/exam/brushquestion/30/4497/C2CBD34082148550EF198C50D10DBDC7?from=1

小码哥很喜欢研究数字，这天小码妹问了他一个问题：“请问一个数字的各个位数上的数字之和为 10 的数有什么性质吗”，小码哥没找到关于这种数字的资料，于是自行将这种数字命名为奇怪的数，并且他还引申出了一个新问题：从 1 到 n 的整数中有多少个数字是奇怪的的数。

比如19、28、37、127都是符合条件的数。

输入格式：
一行一个整数nn
输出格式：
一行一个整数，代表符合条件数的个数

输入：
100
输出：
9

解法一：

```cpp
#include <bits/stdc++.h>
using namespace std;

bool digitSum(int x) {
    int sum = 0;
    while (x > 0) {
        sum += x % 10;
        x /= 10;
    }
    return sum==10;
}

int main() {
    int n;
    cin >> n;

    int count = 0;
    for (int i = 1; i <= n; ++i) {
        if (digitSum(i)) {
```

```
            count++;
        }
    }

    cout << count << endl;
    return 0;
}
```

解法二：数位DP

核心思想：

考虑 **所有不超过 n 的数中，数位和恰好等于 10 的个数**，可以用「数位 DP」求解：

- `dfs(pos, sum, limit)` 表示从低位到高位第 `pos` 位，当前数位和为 `sum`，是否仍受最高位约束 `limit` 的合法数的个数。

状态变量：

- `pos`：当前位数
- `sum`：当前数位和
- `limit`：当前是否受上界（原数n）限制

转移逻辑：

枚举当前位数能填的每个数 `d`（0~9），进行递归：

```
res += dfs(pos - 1, sum + d, limit && (d == up));
```

- 若 `limit==true`，当前位不能超过 `digit[pos]`（对应原数n的当前位）
- 若 `limit==false`，则可以放心填0~9

边界条件：

当 `pos == -1`（即已经没有更多位了）：

- 若 `sum == 10`，说明这是一个“奇怪的数”，返回 `1`
- 否则返回 `0`

```
#include <bits/stdc++.h>
using namespace std;

const int N = 20;
long long dp[N][150]; // dp[pos][sum]：从第pos位开始，数位和为sum的方案数
```

```cpp
int digit[N];

// dfs函数：
// pos：当前正在处理的数位
// sum：当前已经累计的数位和
// limit：是否受原数字n的限制
long long dfs(int pos, int sum, bool limit) {
    if (pos == -1)  // 所有位都处理完了
        return sum == 10;

    if (!limit && dp[pos][sum] != -1)
        return dp[pos][sum];

    int up = limit ? digit[pos] : 9;  // 当前位最大可填值
    long long res = 0;

    for (int d = 0; d <= up; ++d) {
        res += dfs(pos - 1, sum + d, limit && (d == up));
    }

    if (!limit) dp[pos][sum] = res;
    return res;
}

// 主函数，预处理digit数组，然后调用dfs
long long solve(long long n) {
    int len = 0;
    while (n) {
        digit[len++] = n % 10;
        n /= 10;
    }

    memset(dp, -1, sizeof dp);
    return dfs(len - 1, 0, true);  // 从最高位开始
}

int main() {
    long long n;
    cin >> n;
    cout << solve(n) << endl;
    return 0;
}
```