250701小码哥幸运日（题解）

> 题目链接：
> https://www.matiji.net/exam/brushquestion/19/4497/C2CBD34082148550EF198C50D10DBDC7?from=1

#以下题目可能有错别字等，完整题目请访问上述链接
数字 2 是小码哥的幸运数字，小码哥认为他从出生开始算的天数中凡是带 2 的一定是他的幸运日，现在给出两个数字 L 和 R ，即从他出生开始算的第 L 天到 第 R 天，请求出区间 [L,R] 的所有天数幸运值之和。

对于幸运值的定义：2 出现的次数之和:
比如给定区间 [2, 22]，数字2在数2中出现了1次，在数12中出现1次，在数20中出现1次，在数21中出现1次，在数22中出现2次，所以数字2在该区间内一共出现了6次，也就是说该区间的幸运值之和为6。

输入格式:
一行两个整数L和R
输出格式:
一行一个整数，表示小码哥的幸运值之和

输入:
2 22
输出:
6

其中：0≤L≤R≤1e5

为什么简单题，用复杂的解法?

1 题目简单，但思维模型很重要

- 这题表面“暴力遍历”就能做，`R ≤ 1e5`，暴力不超时。
- 但背后蕴含了**经典算法思维**：
  - **按位统计（数学推导）**
  - **数位DP**
- 题目本身不难，但目的是借助这个“壳”讲解更通用的方法。

2 扩展性强，基础题引申高级解法

比如：

- 题目范围变大：如果 `R ≤ 1e9`，暴力必超时，只能靠按位统计或数位DP
- 题目改成统计任意数字出现次数，按位统计模板直接用
- 题目增加复杂限制（如某些位不能为0），必须用数位DP

3 简单题，复杂想

- 竞赛不会一上来直接扔极难题，大多从简单壳子引申拓展。
- 你掌握了“暴力、按位统计、数位DP”三个层次，同样套路能解决：
  - 统计1出现次数
  - 统计某区间中数字d出现次数
  - 统计不含某数字的数
  - 统计符合奇偶、大小组合条件的数

解法一：一次遍历

过于简单，贴个代码

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

int count(int x) {
    int cnt = 0;
    while (x) {
        if (x % 10 == 2) cnt++;
        x /= 10;
    }
    return cnt;
}

int main() {
    int L, R;
    cin >> L >> R;

    int ans = 0;
    for (int i = L; i <= R; ++i) {
        ans += count(i);
    }

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

解法二：按位统计（数学公式）

本篇题解核心方法：
由于本题 0≤L≤R≤1e5，所以暴力没有超时；如果 0≤L≤R≤1e9，那么暴力大概率超时，所以建议学一下以下两种方法

(1) 如何统计区间内 2 出现的次数？
例如区间 [L,R] ，分别统计 0~R 中 2 出现的次数 count1 和 0~L-1 中 2 出现的次数 count2，区间 [L,R]中 2 出现的次数即为 count1-count2，接下来的问题就是如何高效统计出 0~R 和 0~L-1 中 2 出现的次数，此时我们想到了按位统计

(2) 什么是按位统计？
简单来说，就是逐“位”分析每一位上特定数字或特定状态的出现次数；例如对本体来说，即分别统计每一位上 2 出现的次数，最后相加汇总

(3) 如何实现按位统计？
(3.1)假设有以下问题
例如 n=3256
逐位分析，每次考虑第 k 位（个位是第0位，十位是第1位，百位是第2位，依此类推）

定义：
high：第 k 位左边的高位部分
cur：第 k 位的数字
low：第 k 位右边的低位部分
factor = 10^k：当前位的权重（个位是1，十位是10，百位是100，依此类推）

目标是计算第 k 位上，数字 2 出现的次数。

(3.2)贡献公式推导
针对第k位，假设n = high × factor × 10 + cur × factor + low
例子：若n = 3256,k = 2(百位)

factor = 100
high = 3(百位左边的部分)
cur = 2(百位)
low = 56(百位右边的部分)

(3.3)三种情况分析
情况一：cur<2
此时第k位上无法出现2。
推理：
高位high决定了前缀组合总数
每个组合下，低位部分从0到factor-1
但因为第k位cur<2,所以这些组合不会出现2

贡献 = high × factor

情况二：cur==2
此时第k位恰好是2。
推理：
高位high的前缀组合中，每个组合可以有factor个完整组合
但第飞位等于2时，还要额外考虑低位具体数值

贡献= high × factor + (low + 1)
说明:
high×factor:完全组合下2出现次数
low+1:当前组合下，低位0到low,满足第k位是2

情况三：cur>2
此时第k位比2大，意味着前缀组合可以多出一轮完整组合。
推理：
高位high前缀组合产生high×factor次出现
因为当前位cur>2,还多出一轮完整组合

贡献=(high + 1) × factor

(3.4)
根据(3.2)和(3.3)中分析可知百位（k=2）上cur==2，所以
贡献 = high×factor+(low+1) = 3*100+(56+1) = 357

同理：
个位（k=0）（cur=6>2）的贡献=(high + 1) * factor=326
十位（k=1）（cur=5>2）的贡献=(high + 1) * factor=330
千位（k=3）（cur=3>2）的贡献=(high + 1) * factor=1000

所以0~3256中共出现 357 + 326 + 330 +1 000 = 2013 个 2

了解按位统计之后，接下来结合实际代码来更深入理解

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

ll count(ll n) {
    ll res = 0;
    ll i = 0;
    ll factor = 1;

    while (n / factor != 0) {
        ll low = n % factor;
        ll cur = (n / factor) % 10;
        ll high = n / (factor * 10);

        if (cur < 2) {
            res += high * factor;
        } else if (cur == 2) {
```

```
            res += high * factor + (low + 1);
        } else { // cur > 2
            res += (high + 1) * factor;
        }

        factor *= 10;
    }
    return res;
}

int main() {
    ll L, R;
    cin >> L >> R;

    cout << count(R) - count(L - 1) << endl;
    return 0;
}
```

该模板稍作修改，可拓展：

- 统计任意数字 `d` 的出现次数（只需改判断条件）
- 统计区间其他特定特性（如：统计所有偶数、统计含某些特定位的数（如只让统计个位上出现2的次数））

解法三：数位DP

本题无复杂限制，按位统计数学推导即可，DP虽能做但属大材小用。
由于篇幅问题，解法三这里不过多解释，此处贴个代码，有需要的可以单独了解

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

const int N = 20;
ll dp[N][N];
int digit[N];

ll dfs(int pos, int cnt, bool limit) {
    if (pos == -1) return cnt;

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

    int up = limit ? digit[pos] : 9;
    ll res = 0;
    for (int d = 0; d <= up; ++d) {
        res += dfs(pos - 1, cnt + (d == 2), limit && (d == up));
    }
```

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

ll count(ll n) {
    int len = 0;
    while (n) {
        digit[len++] = n % 10;
        n /= 10;
    }
    memset(dp, -1, sizeof(dp));
    return dfs(len - 1, 0, true);
}

int main() {
    ll L, R;
    cin >> L >> R;
    cout << count(R) - count(L - 1) << endl;
    return 0;
}
```

所有题目、代码、思路等如有不理解的地方：

- 可以在群里提问
- 也可以私聊当天的分享人

遇到难以解决的问题及时讨论，不要自己闷头纠结！