数位统计问题选讲

Claris

Hangzhou Dianzi University

2016 年 2 月 8 日

Rownanie

对于一个正整数 n，定义 $f(n)$ 为它十进制下每一位数字的平方的和。

Rownanie

对于一个正整数 n，定义 $f(n)$ 为它十进制下每一位数字的平方的和。

现在给定三个正整数 k, a, b，请求出满足 $a \le n \le b$ 且 $k \times f(n) = n$ 的 n 的个数。

Rownanie

对于一个正整数 n，定义 $f(n)$ 为它十进制下每一位数字的平方的和。

现在给定三个正整数 k, a, b，请求出满足 $a \leq n \leq b$ 且 $k \times f(n) = n$ 的 n 的个数。

- $1 \leq k, a, b \leq 10^{18}, a \leq b$。

Rownanie

对于一个正整数 n，定义 $f(n)$ 为它十进制下每一位数字的平方的和。

现在给定三个正整数 k, a, b，请求出满足 $a \le n \le b$ 且 $k \times f(n) = n$ 的 n 的个数。

- $1 \le k, a, b \le 10^{18}, a \le b$。
- Source：PA 2015

Rownanie

- 当 n 取 $10^{18}-1$ 时，$f(n)$ 最大，此时 $f(n)=9^2\times 18=1458$。

Rownanie

- 当 n 取 $10^{18}-1$ 时，$f(n)$ 最大，此时 $f(n)=9^2\times 18=1458$。
- 枚举所有可能的 $f(n)$，检验 $f(k\times f(n))$ 是否等于 $f(n)$ 即可。

Rownanie

- 当 n 取 $10^{18}-1$ 时，$f(n)$ 最大，此时 $f(n)=9^2\times 18=1458$。
- 枚举所有可能的 $f(n)$，检验 $f(k\times f(n))$ 是否等于 $f(n)$ 即可。
- 时间复杂度 $O(9^2\log^2 b)$。

Gray code

给定一个长度为 n 的仅包含'0'、'1'、'?' 的字符串，你需要给所有的问号决定填 0 还是 1。

Gray code

给定一个长度为 n 的仅包含'0'、'1'、'?' 的字符串，你需要给所有的问号决定填 0 还是 1。

填完之后，将这个串看成二进制数，转化成格雷码，如果格雷码中第 i 个字符是 1，那么你将获得 a_i 点分数。求可以得到的分数的最大值。

Gray code

给定一个长度为 n 的仅包含'0'、'1'、'?' 的字符串，你需要给所有的问号决定填 0 还是 1。

填完之后，将这个串看成二进制数，转化成格雷码，如果格雷码中第 i 个字符是 1，那么你将获得 a_i 点分数。求可以得到的分数的最大值。

- $1 \le n \le 200000, 1 \le a_i \le 1000$。

Gray code

给定一个长度为 n 的仅包含'0'、'1'、'?' 的字符串，你需要给所有的问号决定填 0 还是 1。

填完之后，将这个串看成二进制数，转化成格雷码，如果格雷码中第 i 个字符是 1，那么你将获得 a_i 点分数。求可以得到的分数的最大值。

- $1 \le n \le 200000, 1 \le a_i \le 1000$。
- Source：2015 Multi-University Training Contest 7

Gray code

- 二进制码转换为格雷码的方法为将二进制码右移一位然后与原来的二进制码按位异或。

Gray code

- 二进制码转换为格雷码的方法为将二进制码右移一位然后与原来的二进制码按位异或。
- 设 $f(i,j)$ 表示考虑了前 i 位，二进制码中第 i 位填了 j 的最大分数，则：

Gray code

- 二进制码转换为格雷码的方法为将二进制码右移一位然后与原来的二进制码按位异或。
- 设 $f(i,j)$ 表示考虑了前 i 位，二进制码中第 i 位填了 j 的最大分数，则：
- $f(i,0) = \max(f(i-1,0), f(i-1,1)+a_i)$
  $f(i,1) = \max(f(i-1,1), f(i-1,0)+a_i)$

Gray code

- 二进制码转换为格雷码的方法为将二进制码右移一位然后与原来的二进制码按位异或。
- 设 $f(i,j)$ 表示考虑了前 i 位，二进制码中第 i 位填了 j 的最大分数，则：
- $f(i,0) = \max(f(i-1,0), f(i-1,1) + a_i)$
  $f(i,1) = \max(f(i-1,1), f(i-1,0) + a_i)$
- 时间复杂度 $O(n)$。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

给定 A, B，请统计出 $[A, B]$ 内所有平衡的数的个数。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

给定 A, B，请统计出 $[A, B]$ 内所有平衡的数的个数。

- $1 \le A \le B \le 10^{18}$。

Balanced Numbers

一个数被称为是平衡的数当且仅当对于所有出现过的数位，偶数出现奇数次，奇数出现偶数次。

给定 A, B，请统计出 $[A, B]$ 内所有平衡的数的个数。

- $1 \leq A \leq B \leq 10^{18}$。
- Source：SPOJ 10606

Balanced Numbers

- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么 $ans = cal(B) - cal(A - 1)$。

Balanced Numbers

- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么 $ans = cal(B) - cal(A - 1)$。
- 考虑如何计算 $cal(n)$，对于每个数字，定义状态 0 为没出现过，1 为出现了奇数次，2 为出现了偶数次，那么可以用一个 10 位 3 进制数来表示所有数字的状态。

Balanced Numbers

- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么 $ans = cal(B) - cal(A - 1)$。
- 考虑如何计算 $cal(n)$，对于每个数字，定义状态 0 为没出现过，1 为出现了奇数次，2 为出现了偶数次，那么可以用一个 10 位 3 进制数来表示所有数字的状态。
- 设 $f(i, S, 0)$ 表示从高到低填了前 i 位，所有数字的状态为 S，且当前数字已经小于 n 的数字个数。$f(i, S, 1)$ 表示从高到低填了前 i 位，所有数字的状态为 S，且当前数字等于 n 的数字个数，然后 DP 即可。

Balanced Numbers

- 设 $cal(n)$ 为 $[1, n]$ 中平衡的数的个数，那么 $ans = cal(B) - cal(A-1)$。
- 考虑如何计算 $cal(n)$，对于每个数字，定义状态 0 为没出现过，1 为出现了奇数次，2 为出现了偶数次，那么可以用一个 10 位 3 进制数来表示所有数字的状态。
- 设 $f(i, S, 0)$ 表示从高到低填了前 i 位，所有数字的状态为 S，且当前数字已经小于 n 的数字个数。$f(i, S, 1)$ 表示从高到低填了前 i 位，所有数字的状态为 S，且当前数字等于 n 的数字个数，然后 DP 即可。
- 时间复杂度 $O(3^{10} \log B)$。

一个不含前导 0 的数 x 各个数位上的数字之积记为 $f(x)$。

数字之积

一个不含前导 0 的数 x 各个数位上的数字之积记为 $f(x)$。

给定 n, L, R，求 $[L, R)$ 中满足 $0 < f(x) \leq n$ 的数的个数。

一个不含前导 0 的数 x 各个数位上的数字之积记为 $f(x)$。给定 n, L, R，求 $[L, R)$ 中满足 $0 < f(x) \le n$ 的数的个数。

- $0 < L < R < 10^{18}, n \le 10^{9}$。

数字之积

一个不含前导 0 的数 x 各个数位上的数字之积记为 $f(x)$。给定 n, L, R，求 $[L, R)$ 中满足 $0 < f(x) \leq n$ 的数的个数。

- $0 < L < R < 10^{18}, n \leq 10^9$。
- Source：BZOJ 3679

数字之积

- 与上题一样，还是考虑计算 $[1, R)$ 内满足条件的数的个数。

数字之积

- 与上题一样，还是考虑计算 $[1, R)$ 内满足条件的数的个数。
- 数字之积非常大，但是这些数字的质因子只可能是 2、3、5、7。

数字之积

- 与上题一样，还是考虑计算 $[1, R)$ 内满足条件的数的个数。
- 数字之积非常大，但是这些数字的质因子只可能是 2、3、5、7。
- 所以设 $f(i, cnt_2, cnt_3, cnt_5, cnt_7, j)$ 为从高到低填了前 i 位，2、3、5、7 的个数分别为 cnt_2、cnt_3、cnt_5、cnt_9，是否小于 R 的状态为 j 的数字个数，然后 DP 即可。

数字之积

- 与上题一样，还是考虑计算 $[1, R)$ 内满足条件的数的个数。
- 数字之积非常大，但是这些数字的质因子只可能是 2、3、5、7。
- 所以设 $f(i, cnt_2, cnt_3, cnt_5, cnt_7, j)$ 为从高到低填了前 i 位，2、3、5、7 的个数分别为 cnt_2、cnt_3、cnt_5、cnt_9，是否小于 R 的状态为 j 的数字个数，然后 DP 即可。
- 时间复杂度 $O(\log_2 n \log_3 n \log_5 n \log_7 n \log R)$。

Beautiful numbers

给定 L, R，求 $[L, R]$ 中所有可以被它所有非零数位整除的数的个数。

Beautiful numbers

给定 L, R，求 $[L, R]$ 中所有可以被它所有非零数位整除的数的个数。

- $1 \le L \le R \le 9 \times 10^{18}$。

给定 L, R，求 $[L, R]$ 中所有可以被它所有非零数位整除的数的个数。

- $1 \le L \le R \le 9 \times 10^{18}$。
- Source：Codeforces 55 D

Beautiful numbers

- 与上题一样，还是考虑计算 $[1, R]$ 内满足条件的数的个数。

Beautiful numbers

- 与上题一样，还是考虑计算 $[1, R]$ 内满足条件的数的个数。
- 注意到 $\operatorname{lcm}(1,2,3,4,5,6,7,8,9) = 2520$，且 1 到 9 中任意一个集合的 lcm 只有 49 种。所以可以考虑设 $f(i,j,k,l)$ 为从高到低填了前 i 位，之前拼成的数字 $\bmod 2520 = j$，已选用数位的 lcm 为 k，是否小于 R 的状态为 l 的数字个数，然后 DP 即可。

Beautiful numbers

- 与上题一样，还是考虑计算 $[1, R]$ 内满足条件的数的个数。
- 注意到 $\text{lcm}(1, 2, 3, 4, 5, 6, 7, 8, 9) = 2520$，且 1 到 9 中任意一个集合的 lcm 只有 49 种。所以可以考虑设 $f(i, j, k, l)$ 为从高到低填了前 i 位，之前拼成的数字 $\bmod 2520 = j$，已选用数位的 lcm 为 k，是否小于 R 的状态为 l 的数字个数，然后 DP 即可。
- 对于 k，可以在一开始预处理出所有 49 种情况，对其进行重标号，以剔除无效状态。

Beautiful numbers

- 与上题一样，还是考虑计算 $[1, R]$ 内满足条件的数的个数。
- 注意到 $\operatorname{lcm}(1,2,3,4,5,6,7,8,9) = 2520$，且 1 到 9 中任意一个集合的 lcm 只有 49 种。所以可以考虑设 $f(i,j,k,l)$ 为从高到低填了前 i 位，之前拼成的数字 $\bmod 2520 = j$，已选用数位的 lcm 为 k，是否小于 R 的状态为 l 的数字个数，然后 DP 即可。
- 对于 k，可以在一开始预处理出所有 49 种情况，对其进行重标号，以剔除无效状态。
- 时间复杂度 $O(2520 \times 49 \log R)$。

Xorequ

给定方程 $x \text{ xor } 3x = 2x$，给定正整数 n，任务如下：

Xorequ

给定方程 $x \; xor \; 3x = 2x$，给定正整数 n，任务如下：

(1) 求出所有小于等于 n 的正整数中，有多少个数满足该方程。

给定方程 $x \text{ xor } 3x = 2x$，给定正整数 n，任务如下：

(1) 求出所有小于等于 n 的正整数中，有多少个数满足该方程。

(2) 求出所有小于等于 2^n 的正整数中，有多少个数满足该方程。

Xorequ

给定方程 $x\ xor\ 3x = 2x$，给定正整数 n，任务如下：

(1) 求出所有小于等于 n 的正整数中，有多少个数满足该方程。

(2) 求出所有小于等于 2^n 的正整数中，有多少个数满足该方程。

- $1 \le n \le 10^{18}$。

Xorequ

给定方程 $x \; xor \; 3x = 2x$，给定正整数 n，任务如下：

(1) 求出所有小于等于 n 的正整数中，有多少个数满足该方程。

(2) 求出所有小于等于 2^n 的正整数中，有多少个数满足该方程。

- $1 \leq n \leq 10^{18}$。
- Source：BZOJ 3329

Xorequ

- 将方程移项，得 $x\ xor\ 2x = 3x$，又因为 $x + 2x = 3x$，所以 $x\ and\ 2x = 0$，否则异或结果一定小于 $3x$。

Xorequ

- 将方程移项，得 $x \ \mathit{xor}\ 2x = 3x$，又因为 $x + 2x = 3x$，所以 $x \ \mathit{and}\ 2x = 0$，否则异或结果一定小于 $3x$。
- $x \ \mathit{and}\ 2x = 0$ 等价于 x 的二进制表示中没有相邻的 1。

Xorequ

- 将方程移项，得 $x\ xor\ 2x = 3x$，又因为 $x + 2x = 3x$，所以 $x\ and\ 2x = 0$，否则异或结果一定小于 $3x$。
- $x\ and\ 2x = 0$ 等价于 x 的二进制表示中没有相邻的 1。
- 考虑数位 DP，设 $f(i, j, k)$ 为从高到低填了前 i 位，最后一个 1 离 i 的距离为 j，是否小于 n 的状态为 k 的数字个数，然后 DP 即可求出第一问。

Xorequ

- 将方程移项，得 $x \text{ xor } 2x = 3x$，又因为 $x + 2x = 3x$，所以 $x \text{ and } 2x = 0$，否则异或结果一定小于 $3x$。
- $x \text{ and } 2x = 0$ 等价于 x 的二进制表示中没有相邻的 1。
- 考虑数位 DP，设 $f(i,j,k)$ 为从高到低填了前 i 位，最后一个 1 离 i 的距离为 j，是否小于 n 的状态为 k 的数字个数，然后 DP 即可求出第一问。
- 对于第二问，设 $g(i,j)$ 为 i 位二进制数中最高位为 j 且满足条件的数字个数，则：
  $g(i,0) = g(i-1,0) + g(i-1,1), g(i,1) = g(i-1,0)$
  即 $g(i,0) = g(i-1,0) + g(i-2,0)$，所以答案即为斐波那契数列第 $n+2$ 项，矩阵快速幂即可。

Xorequ

- 将方程移项，得 $x \text{ } xor \text{ } 2x = 3x$，又因为 $x + 2x = 3x$，所以 $x \text{ } and \text{ } 2x = 0$，否则异或结果一定小于 $3x$。
- $x \text{ } and \text{ } 2x = 0$ 等价于 x 的二进制表示中没有相邻的 1。
- 考虑数位 DP，设 $f(i, j, k)$ 为从高到低填了前 i 位，最后一个 1 离 i 的距离为 j，是否小于 n 的状态为 k 的数字个数，然后 DP 即可求出第一问。
- 对于第二问，设 $g(i, j)$ 为 i 位二进制数中最高位为 j 且满足条件的数字个数，则：
  $g(i, 0) = g(i-1, 0) + g(i-1, 1), g(i, 1) = g(i-1, 0)$
  即 $g(i, 0) = g(i-1, 0) + g(i-2, 0)$，所以答案即为斐波那契数列第 $n+2$ 项，矩阵快速幂即可。
- 时间复杂度 $O(\log n)$。

A Math Problem

有一个函数，满足 $f(1) = 1$，且对于任意正整数 n，有：

A Math Problem

有一个函数，满足 $f(1)=1$，且对于任意正整数 n，有：
$3\times f(n)\times f(2n+1)=f(2n)\times(1+3f(n)), f(2n)<6\times f(n)$。

有一个函数，满足 $f(1)=1$，且对于任意正整数 n，有：$3\times f(n)\times f(2n+1)=f(2n)\times(1+3f(n)), f(2n)<6\times f(n)$。

现给定两个正整数 n 和 m，对 1 到 n 里每个整数 i，计算 $f(i) \bmod m$，请对于 0 到 $m-1$ 的每个整数 i，统计有多少数字计算结果为 i。

A Math Problem

有一个函数，满足 $f(1) = 1$，且对于任意正整数 n，有：

$3 \times f(n) \times f(2n+1) = f(2n) \times (1 + 3f(n)), f(2n) < 6 \times f(n)$。

现给定两个正整数 n 和 m，对 1 到 n 里每个整数 i，计算 $f(i) \bmod m$，请对于 0 到 $m-1$ 的每个整数 i，统计有多少数字计算结果为 i。

- $1 \le n \le 10^{18}, 1 \le m \le 65537$。

A Math Problem

有一个函数，满足 $f(1)=1$，且对于任意正整数 n，有：

$3\times f(n)\times f(2n+1)=f(2n)\times(1+3f(n)), f(2n)<6\times f(n)$。

现给定两个正整数 n 和 m，对 1 到 n 里每个整数 i，计算 $f(i) \bmod m$，请对于 0 到 $m-1$ 的每个整数 i，统计有多少数字计算结果为 i。

- $1\le n\le 10^{18}, 1\le m\le 65537$。
- Source：2015 ACM/ICPC 亚洲区北京站

A Math Problem

- 不难发现 $f(2n) = 3f(n), f(2n+1) = 3f(n) + 1$。

A Math Problem

- 不难发现 $f(2n) = 3f(n), f(2n+1) = 3f(n) + 1$。
- 将 n 转为二进制表示，考虑数位 DP，设 $f(i, j, k)$ 为从高到低填了前 i 位，已经填了数的部分的 f 值对 m 的余数为 j，是否小于 n 的状态为 k 的数字个数，然后 DP 即可。

A Math Problem

- 不难发现 $f(2n) = 3f(n), f(2n+1) = 3f(n) + 1$。
- 将 n 转为二进制表示，考虑数位 DP，设 $f(i, j, k)$ 为从高到低填了前 i 位，已经填了数的部分的 f 值对 m 的余数为 j，是否小于 n 的状态为 k 的数字个数，然后 DP 即可。
- 时间复杂度 $O(m \log n)$。

XHXJ's LIS

给定 L, R, K，对于 $[L, R]$ 内的每个整数，将其看成字符串，求出最长上升序列。请统计所有 LIS 长度为 K 的数字个数。

XHXJ's LIS

给定 L, R, K，对于 $[L, R]$ 内的每个整数，将其看成字符串，求出最长上升序列。请统计所有 LIS 长度为 K 的数字个数。

- $0 < L \le R < 2^{63} - 1, 1 \le K \le 10$。

XHXJ's LIS

给定 L, R, K，对于 $[L, R]$ 内的每个整数，将其看成字符串，求出最长上升序列。请统计所有 LIS 长度为 K 的数字个数。

- $0 < L \leq R < 2^{63} - 1, 1 \leq K \leq 10$。
- Source：HDU 4352

XHXJ's LIS

- 回忆 $O(n \log n)$ 求 LIS 的过程，维护一个上升序列，每次新加一个数的时候，用它去替换里面最小的大于它的数，最后序列长度就是答案。

XHXJ's LIS

- 回忆 $O(n \log n)$ 求 LIS 的过程，维护一个上升序列，每次新加一个数的时候，用它去替换里面最小的大于它的数，最后序列长度就是答案。
- 对于本题，因为数位只可能是 0 到 9，所以可以考虑用一个 10 位二进制数来唯一确定这个需要维护的序列。

XHXJ's LIS

- 回忆 $O(n \log n)$ 求 LIS 的过程，维护一个上升序列，每次新加一个数的时候，用它去替换里面最小的大于它的数，最后序列长度就是答案。
- 对于本题，因为数位只可能是 0 到 9，所以可以考虑用一个 10 位二进制数来唯一确定这个需要维护的序列。
- 设 $f(i, S, j)$ 为从高到低填了前 i 位，之前部分的序列情况为 S，是否小于 R 的状态为 j 的数字个数，然后 DP 即可。

XHXJ's LIS

- 回忆 $O(n \log n)$ 求 LIS 的过程，维护一个上升序列，每次新加一个数的时候，用它去替换里面最小的大于它的数，最后序列长度就是答案。
- 对于本题，因为数位只可能是 0 到 9，所以可以考虑用一个 10 位二进制数来唯一确定这个需要维护的序列。
- 设 $f(i, S, j)$ 为从高到低填了前 i 位，之前部分的序列情况为 S，是否小于 R 的状态为 j 的数字个数，然后 DP 即可。
- 时间复杂度 $O(2^{10} \log R)$。

数字统计

将 $[L, R]$ 中的所有整数用 M 位二进制数表示 (允许前导 0)。现在将这些数中的每一个作如下变换:

数字统计

将 $[L, R]$ 中的所有整数用 M 位二进制数表示 (允许前导 0)。现在将这些数中的每一个作如下变换:

从这个数的最低两位开始，如果这两位都是 0，那么 $X=1$，否则 $X=0$。将这两位删去，然后将 X 放在原来最低位的位置上。重复这个变换直到这个数只剩下一位为止。

数字统计

将 $[L, R]$ 中的所有整数用 M 位二进制数表示 (允许前导 0)。现在将这些数中的每一个作如下变换：

从这个数的最低两位开始，如果这两位都是 0，那么 $X=1$，否则 $X=0$。将这两位删去，然后将 X 放在原来最低位的位置上。重复这个变换直到这个数只剩下一位为止。

例如 01001 的变换过程如下：

01001 -> 0100 -> 011 -> 00 -> 1

数字统计

将 $[L, R]$ 中的所有整数用 M 位二进制数表示 (允许前导 0)。现在将这些数中的每一个作如下变换：

从这个数的最低两位开始，如果这两位都是 0，那么 X =1，否则 X =0。将这两位删去，然后将 X 放在原来最低位的位置上。重复这个变换直到这个数只剩下一位为止。

例如 01001 的变换过程如下：

01001 -> 0100 -> 011 -> 00 -> 1

现在的问题是变换后的所有数中，值为 Y(Y 为 0 或 1) 的有多少个？

数字统计

将 $[L, R]$ 中的所有整数用 M 位二进制数表示 (允许前导 0)。现在将这些数中的每一个作如下变换：

从这个数的最低两位开始，如果这两位都是 0，那么 $X=1$，否则 $X=0$。将这两位删去，然后将 X 放在原来最低位的位置上。重复这个变换直到这个数只剩下一位为止。

例如 01001 的变换过程如下：

01001 -> 0100 -> 011 -> 00 -> 1

现在的问题是变换后的所有数中，值为 Y(Y 为 0 或 1) 的有多少个?

- $1 \leq M \leq 200, 0 \leq L \leq R \leq 2^M$。

数字统计

将 $[L,R]$ 中的所有整数用 M 位二进制数表示 (允许前导 0)。现在将这些数中的每一个作如下变换:

从这个数的最低两位开始，如果这两位都是 0，那么 $X=1$，否则 $X=0$。将这两位删去，然后将 X 放在原来最低位的位置上。重复这个变换直到这个数只剩下一位为止。

例如 01001 的变换过程如下:

01001 -> 0100 -> 011 -> 00 -> 1

现在的问题是变换后的所有数中，值为 Y(Y 为 0 或 1) 的有多少个?

- $1 \le M \le 200, 0 \le L \le R \le 2^M$。
- Source：BZOJ 3780

数字统计

- 与之前的问题不同，因为运算的性质，本题没有办法从高位向低位填数，只能从低位向高位填数。

数字统计

- 与之前的问题不同，因为运算的性质，本题没有办法从高位向低位填数，只能从低位向高位填数。
- 重新设计状态，设 $f(i,j,k)$ 表示已经填了后 i 位，转化后的数字最终为 j，后 i 位与 R 后 i 位的大小关系为 k 的方案数。这里的 k 要么是小于，要么是大于等于。

数字统计

- 与之前的问题不同，因为运算的性质，本题没有办法从高位向低位填数，只能从低位向高位填数。
- 重新设计状态，设 $f(i,j,k)$ 表示已经填了后 i 位，转化后的数字最终为 j，后 i 位与 R 后 i 位的大小关系为 k 的方案数。这里的 k 要么是小于，要么是大于等于。
- 对于两个相同位数的数，我们首先比较它们的首位，如果相同，我们再比较它们去掉首位后的部分，这恰好就是 k，按照这种方法进行状态转移即可。

数字统计

- 与之前的问题不同，因为运算的性质，本题没有办法从高位向低位填数，只能从低位向高位填数。
- 重新设计状态，设 $f(i, j, k)$ 表示已经填了后 i 位，转化后的数字最终为 j，后 i 位与 R 后 i 位的大小关系为 k 的方案数。这里的 k 要么是小于，要么是大于等于。
- 对于两个相同位数的数，我们首先比较它们的首位，如果相同，我们再比较它们去掉首位后的部分，这恰好就是 k，按照这种方法进行状态转移即可。
- 时间复杂度 $O(M)$。

Tickets

有一位售票员给乘客售票，对于每位乘客，他会卖出多张连续的票，直到已卖出的编号的所有位置上的数的和不小于给定的正数 k。然后他会按照相同的规则给下一位乘客售票。

Tickets

有一位售票员给乘客售票，对于每位乘客，他会卖出多张连续的票，直到已卖出的编号的所有位置上的数的和不小于给定的正数 k。然后他会按照相同的规则给下一位乘客售票。

初始时，售票员持有的编号是从 L 到 R 的连续整数。请你求出，售票员可以售票给多少位乘客。

Tickets

有一位售票员给乘客售票，对于每位乘客，他会卖出多张连续的票，直到已卖出的编号的所有位置上的数的和不小于给定的正数 k。然后他会按照相同的规则给下一位乘客售票。

初始时，售票员持有的编号是从 L 到 R 的连续整数。请你求出，售票员可以售票给多少位乘客。

- $1 \le L \le R \le 10^{18}, 1 \le k \le 1000$。

Tickets

有一位售票员给乘客售票，对于每位乘客，他会卖出多张连续的票，直到已卖出的编号的所有位置上的数的和不小于给定的正数 k。然后他会按照相同的规则给下一位乘客售票。

初始时，售票员持有的编号是从 L 到 R 的连续整数。请你求出，售票员可以售票给多少位乘客。

- $1 \le L \le R \le 10^{18}, 1 \le k \le 1000$。
- Source：SGU 390

Tickets

- 首先弱化问题，假设给定的区间为 $[1, 10^h - 1]$，我们可以用一棵完全十叉树来描述所有的数，且我们可选的是 L 到 R 中间的这部分。准确的说，设 L 和 R 的 LCA 为 x，那么就是 L 到 x 路上右边的部分以及 R 到 x 路上左边的部分。

Tickets

- 首先弱化问题，假设给定的区间为 $[1, 10^h - 1]$，我们可以用一棵完全十叉树来描述所有的数，且我们可选的是 L 到 R 中间的这部分。准确的说，设 L 和 R 的 LCA 为 x，那么就是 L 到 x 路上右边的部分以及 R 到 x 路上左边的部分。
- 设 $f(i, j, k)$ 表示最后 i 位没有限制，已经确定部分的数位和为 j，一开始剩余容量为 k 时，分配完毕后最终的状态，$f(i, j, k).a$ 表示票数，$f(i, j, k).b$ 表示最终剩余容量，则有如下转移：

Tickets

- 首先弱化问题，假设给定的区间为 $[1, 10^h - 1]$，我们可以用一棵完全十叉树来描述所有的数，且我们可选的是 L 到 R 中间的这部分。准确的说，设 L 和 R 的 LCA 为 x，那么就是 L 到 x 路上右边的部分以及 R 到 x 路上左边的部分。
- 设 $f(i,j,k)$ 表示最后 i 位没有限制，已经确定部分的数位和为 j，一开始剩余容量为 k 时，分配完毕后最终的状态，$f(i,j,k).a$ 表示票数，$f(i,j,k).b$ 表示最终剩余容量，则有如下转移：
- for(t=0;t<=9;t++){

```
    f(i,j,k).a=f(i,j,k).a+f(i-1,j+t,f(i,j,k).b).a;
    f(i,j,k).b=f(i-1,j+t,f(i,j,k).b).b;
}
```

Tickets

- 设 $W(A, B, C, L, R)$ 表示当前考虑最后 A 位，之前考虑部分的数位和为 B，一开始剩余容量为 C，是否需要考虑上下界时的结果，也可以通过类似方法进行转移。

Tickets

- 设 $W(A, B, C, L, R)$ 表示当前考虑最后 A 位，之前考虑部分的数位和为 B，一开始剩余容量为 C，是否需要考虑上下界时的结果，也可以通过类似方法进行转移。
- 对于 W 的求解，考虑使用记忆化搜索来完成，一方面实现难度低，另一方面可以剔除大量无效状态。

Tickets

- 设 $W(A, B, C, L, R)$ 表示当前考虑最后 A 位，之前考虑部分的数位和为 B，一开始剩余容量为 C，是否需要考虑上下界时的结果，也可以通过类似方法进行转移。
- 对于 W 的求解，考虑使用记忆化搜索来完成，一方面实现难度低，另一方面可以剔除大量无效状态。
- 时间复杂度 $O(k \log^2 R)$。

Graduated Lexicographical Ordering

对于 $[1, n]$ 内的所有正整数，将它们按照各数位之和为第一关键字，字典序为第二关键字从小到大排序。

Graduated Lexicographical Ordering

对于 $[1, n]$ 内的所有正整数，将它们按照各数位之和为第一关键字，字典序为第二关键字从小到大排序。

给定 k，求出 k 的排名，并找到第 k 小的数。

Graduated Lexicographical Ordering

对于 $[1, n]$ 内的所有正整数，将它们按照各数位之和为第一关键字，字典序为第二关键字从小到大排序。

给定 k，求出 k 的排名，并找到第 k 小的数。

- $1 \leq k \leq n \leq 10^{18}$。

Graduated Lexicographical Ordering

对于 $[1, n]$ 内的所有正整数，将它们按照各数位之和为第一关键字，字典序为第二关键字从小到大排序。

给定 k，求出 k 的排名，并找到第 k 小的数。

- $1 \leq k \leq n \leq 10^{18}$。
- Source：ZOJ 2599

Graduated Lexicographical Ordering

- 对于第一问，设 k 的数位之和为 sum，枚举 1 到 $sum-1$ 的数，计算数位和为它们的数的个数，再计算数位和为 sum 且字典序比 k 小的数字个数，可以通过数位 DP 和枚举位数来完成。

Graduated Lexicographical Ordering

- 对于第一问，设 k 的数位之和为 sum，枚举 1 到 $sum-1$ 的数，计算数位和为它们的数的个数，再计算数位和为 sum 且字典序比 k 小的数字个数，可以通过数位 DP 和枚举位数来完成。
- 对于第二问，从小到大枚举数位之和，确定第 k 小的数的数位之和，然后从第一位开始借助第一问逐位确定每一位。

Graduated Lexicographical Ordering

- 对于第一问，设 k 的数位之和为 sum，枚举 1 到 $sum-1$ 的数，计算数位和为它们的数的个数，再计算数位和为 sum 且字典序比 k 小的数字个数，可以通过数位 DP 和枚举位数来完成。
- 对于第二问，从小到大枚举数位之和，确定第 k 小的数的数位之和，然后从第一位开始借助第一问逐位确定每一位。
- 时间复杂度 $O(\log^4 n)$。

Fibonacci

众所周知，斐波那契数列 F 满足：

Fibonacci

众所周知，斐波那契数列 F 满足：

$F_0 = 0, F_1 = 1, F_m = F_{m-1} + F_{m-2}(m \geq 2)$

众所周知，斐波那契数列 F 满足：

$F_0 = 0, F_1 = 1, F_m = F_{m-1} + F_{m-2} (m \geq 2)$

现在给出一个数字串 S，请找到任意一个 k，使得 F_k 以 S 为结尾，或判断无解。

Fibonacci

众所周知，斐波那契数列 F 满足：

$F_0 = 0, F_1 = 1, F_m = F_{m-1} + F_{m-2} (m \geq 2)$

现在给出一个数字串 S，请找到任意一个 k，使得 F_k 以 S 为结尾，或判断无解。

- 设 $|S|$ 为 S 的长度，$1 \leq |S| \leq 18$。

Fibonacci

众所周知，斐波那契数列 F 满足：

$F_0 = 0, F_1 = 1, F_m = F_{m-1} + F_{m-2} (m \geq 2)$

现在给出一个数字串 S，请找到任意一个 k，使得 F_k 以 S 为结尾，或判断无解。

- 设 $|S|$ 为 S 的长度，$1 \leq |S| \leq 18$。
- Source：PA 2015

Fibonacci

- 斐波那契数列的特性：斐波那契数列模 10^m 的循环节为 6×10^m。

Fibonacci

- 斐波那契数列的特性：斐波那契数列模 10^m 的循环节为 6×10^m。
- 从最低位开始，枚举这一位是什么，如果匹配最后一位，那么继续枚举最后第二位，形成一个 dfs 的过程。

Fibonacci

- 斐波那契数列的特性：斐波那契数列模 10^m 的循环节为 6×10^m。
- 从最低位开始，枚举这一位是什么，如果匹配最后一位，那么继续枚举最后第二位，形成一个 dfs 的过程。
- 如此搜索，状态数不会很多。

Fibonacci

- 斐波那契数列的特性：斐波那契数列模 10^m 的循环节为 6×10^m。
- 从最低位开始，枚举这一位是什么，如果匹配最后一位，那么继续枚举最后第二位，形成一个 dfs 的过程。
- 如此搜索，状态数不会很多。
- 为了处理前导 0 的问题，可以考虑找到答案后直接给答案强行加上 6×10^{18}。

Kykneion asma

给定 a_0, a_1, a_2, a_3, a_4 以及 n，问有多少不含前导 0 的 5 进制 n 位数满足数字 i 的个数不超过 a_i。

- $1 \le n \le 20000, 0 \le a_i \le n$。

Kykneion asma

给定 a_0, a_1, a_2, a_3, a_4 以及 n，问有多少不含前导 0 的 5 进制 n 位数满足数字 i 的个数不超过 a_i。

- $1 \le n \le 20000, 0 \le a_i \le n$。
- Source：2015 ACM/ICPC 亚洲区沈阳站

Kykneion asma

- 设 $W(n, a_0, a_1, a_2, a_3, a_4)$ 代表可以有前导 0 的 n 位 5 进制数字，至多 a_i 个数字 i 的方案数，那么答案等于 $W(n, a_0, a_1, a_2, a_3, a_4) - W(n-1, a_0-1, a_1, a_2, a_3, a_4)$，相当于减去一定有前导 0 的方案数。

Kykneion asma

- 设 $W(n, a_0, a_1, a_2, a_3, a_4)$ 代表可以有前导 0 的 n 位 5 进制数字，至多 a_i 个数字 i 的方案数，那么答案等于 $W(n, a_0, a_1, a_2, a_3, a_4) - W(n-1, a_0-1, a_1, a_2, a_3, a_4)$，相当于减去一定有前导 0 的方案数。
- 考虑用容斥原理计算 $W(n, a_0, a_1, a_2, a_3, a_4)$:

Kykneion asma

- 设 $W(n, a_0, a_1, a_2, a_3, a_4)$ 代表可以有前导 0 的 n 位 5 进制数字，至多 a_i 个数字 i 的方案数，那么答案等于 $W(n, a_0, a_1, a_2, a_3, a_4) - W(n-1, a_0-1, a_1, a_2, a_3, a_4)$，相当于减去一定有前导 0 的方案数。
- 考虑用容斥原理计算 $W(n, a_0, a_1, a_2, a_3, a_4)$:
- 设 $f(i, S)$ 表示填了前 i 位，数字集合 S 是随便填且其它数字填的数量一定都要超过限制的方案数。

- 设 $W(n,a_0,a_1,a_2,a_3,a_4)$ 代表可以有前导 0 的 n 位 5 进制数字，至多 a_i 个数字 i 的方案数，那么答案等于 $W(n,a_0,a_1,a_2,a_3,a_4)-W(n-1,a_0-1,a_1,a_2,a_3,a_4)$，相当于减去一定有前导 0 的方案数。
- 考虑用容斥原理计算 $W(n,a_0,a_1,a_2,a_3,a_4)$:
- 设 $f(i,S)$ 表示填了前 i 位，数字集合 S 是随便填且其它数字填的数量一定都要超过限制的方案数。
- 根据容斥原理，如果 S 里有奇数个数，那么 $f(0,S)=1$，否则 $f(0,S)=-1$。

Kykneion asma

- 对于转移，设 $|S|$ 表示 S 中数字个数，那么首先有 $f(i,S) += f(i-1,S) \times |S|$，表示这些数可以随便填。

Kykneion asma

- 对于转移，设 $|S|$ 表示 S 中数字个数，那么首先有 $f(i,S)+=f(i-1,S)\times|S|$，表示这些数可以随便填。
- 然后枚举一个不在 S 集合中的数 j，一开始强行填上 a_j+1 个数后，它一定超过了限制，且接下来可以随便填了，所以有 $f(i,S\bigcup j)+=f(i-a_j-1,S)\times C(i-1,a_j)$。

Kykneion asma

- 对于转移，设 $|S|$ 表示 S 中数字个数，那么首先有 $f(i,S) += f(i-1,S)\times|S|$，表示这些数可以随便填。
- 然后枚举一个不在 S 集合中的数 j，一开始强行填上 a_j+1 个数后，它一定超过了限制，且接下来可以随便填了，所以有 $f(i,S\bigcup j) += f(i-a_j-1,S)\times C(i-1,a_j)$。
- 最后 $W(n,a_0,a_1,a_2,a_3,a_4) = f(n,\{0,1,2,3,4\})$。

Kykneion asma

- 对于转移，设 $|S|$ 表示 S 中数字个数，那么首先有 $f(i,S)+=f(i-1,S)\times|S|$，表示这些数可以随便填。
- 然后枚举一个不在 S 集合中的数 j，一开始强行填上 a_j+1 个数后，它一定超过了限制，且接下来可以随便填了，所以有 $f(i,S\bigcup j)+=f(i-a_j-1,S)\times C(i-1,a_j)$。
- 最后 $W(n,a_0,a_1,a_2,a_3,a_4)=f(n,\{0,1,2,3,4\})$。
- 时间复杂度 $O(5\times 2^5\times n)$。

Thank you!