浅谈数位类统计问题

山东省青岛第二中学 刘聪

【摘要】

在信息学竞赛中，有一类与数位有关的区间统计问题。这类问题往往具有比较浓厚的数学味道，无法暴力求解，需要在数位上进行递推等操作。本文通过几个例子，简要介绍了解决此类问题的基本思想和方法。

【关键字】

数位 区间 统计 递推 树 二进制

【正文】

在信息学竞赛中，有这样一类问题：求给定区间中，满足给定条件的某个 D 进制数或此类数的数量。所求的限定条件往往与数位有关，例如数位之和、指定数码个数、数的大小顺序分组等等。题目给定的区间往往很大，无法采用朴素的方法求解。此时，我们就需要利用数位的性质，设计 $\log(n)$级别复杂度的算法。解决这类问题最基本的思想就是“逐位确定”的方法。下面就让我们通过几道例题来具体了解一下这类问题及其思考方法。

【例题 1】Amount of degrees (ural 1057)

题目大意：

求给定区间[X,Y]中满足下列条件的整数个数：这个数恰好等于 K 个互不相等的 B 的整数次幂之和。例如，设 X=15，Y=20，K=2，B=2，则有且仅有下列三个数满足题意：

$$17 = 2^4+2^0,$$
$$18 = 2^4+2^1,$$
$$20 = 2^4+2^2。$$

输入：第一行包含两个整数 X 和 Y。接下来两行包含整数 K 和 B。

输出：只包含一个整数，表示满足条件的数的个数。

数据规模：$1 \le X \le Y \le 2^{31}-1$，$1 \le K \le 20$， $2 \le B \le 10$。

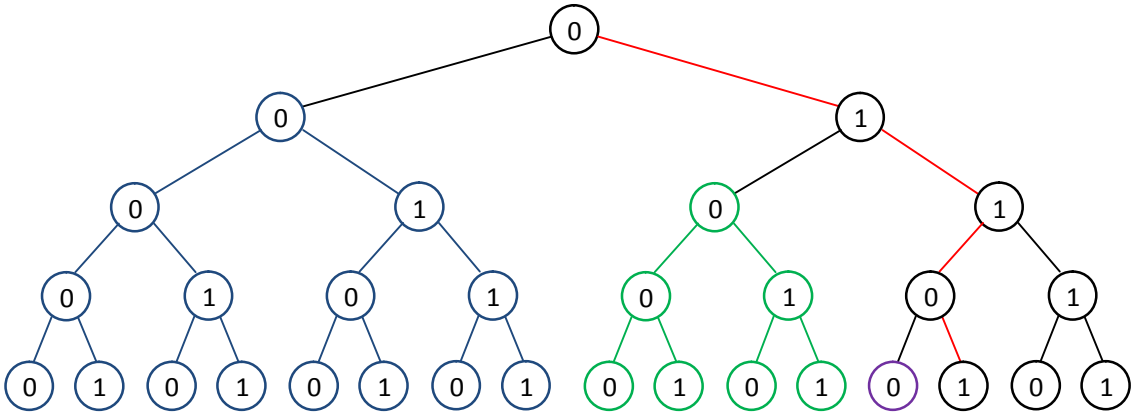
分析：

所求的数为互不相等的幂之和，亦即其 B 进制表示的各位数字都只能是 0 和 1。因此，我们只需讨论二进制的情况，其他进制都可以转化为二进制求解。

很显然，数据范围较大，不可能采用枚举法，算法复杂度必须是 $\log(n)$级别，因此我们要从数位上下手。

本题区间满足区间减法，因此可以进一步简化问题：令 *count*[*i*..*j*]表示[*i*..*j*]区间内合法数的个数，则 *count*[*i*..*j*]=*count*[0..*j*]-*count*[0..*i*-1]。换句话说，给定 *n*，我们只需求出从 0 到 *n* 有多少个符合条件的数。

假设 *n*=13，其二进制表示为 1101，*K*=3。我们的目标是求出 0 到 13 中二进制表示含 3 个 1 的数的个数。为了方便思考，让我们画出一棵高度为 4 的完全二叉树：



为了方便起见，树的根用 0 表示。这样，这棵高度为 4 的完全二叉树就可以表示所有 4 位二进制数（0..2^4-1），每一个叶子节点代表一个数。其中，红色路径表示 *n*。所有小于 *n* 的数组成了三棵子树，分别用蓝色、绿色、紫色表示。因此，统计小于 13 的数，就只需统计这三棵完整的完全二叉树：统计蓝子树内含 3 个 1 的数的个数、统计绿子树内含 2 个 1 的数的个数（因为从根到此处的路径上已经有 1 个 1），以及统计紫子树内含 1 个 1 的数的个数。注意到，只要是高度相同的子树统计结果一定相同。而需要统计的子树都是“右转”时遇到的。当然，我们不能忘记统计 *n* 本身。实际上，在算法最初时将 *n* 自加 1，可以避免讨论 *n* 本身，但是需要注意防止上溢。

剩下的问题就是，如何统计一棵高度为 *i* 的完全二叉树内二进制表示中恰好含有 *j* 个 1 的数的个数。这很容易用递推求出：设 *f*[*i*,*j*]表示所求，则分别统计左右子树内符合条件数的个数，有 *f*[*i*,*j*]=*f*[*i*-1,*j*]+*f*[*i*-1,*j*-1]。

这样，我们就得出了询问的算法：首先预处理 *f*，然后对于输入 *n*，我们在假想的完全二叉树中，从根走到 *n* 所在的叶子，每次向右转时统计左子树内数的个数。下面是 C++代码：

```
void init()//预处理f
{
    f[0][0]=1;
    for (int i=1;i<=31;++i) {
        f[i][0]=f[i-1][0];
        for (int j=1;j<=i;++j) f[i][j]=f[i-1][j]+f[i-1][j-1];
    }
}
int calc(int x,int k)//统计[0..x]内二进制表示含k个1的数的个数
{
    int tot=0,ans=0;//tot记录当前路径上已有的1的数量，ans表示答案
    for (int i=31;i>0;--i)
    {
        if (x&(1<<i)) {
            ++tot;
            if (tot>k) break;
```

```
            x=x^(1<<i);
        }
        if ((1<<(i-1))<=x) {
            ans+=f[i-1][k-tot];
        }
    }
    if (tot+x==k) ++ans;
    return ans;
}
```

最后的问题就是如何处理非二进制。对于询问 n，我们需要求出不超过 n 的最大 B 进制表示只含 0、1 的数：找到 n 的左起第一位非 0、1 的数位，将它变为 1，并将右面所有数位设为 1。将得到的 B 进制表示视为二进制进行询问即可。

预处理递推 f 的时间复杂度为 $O(\log^2(n))$，共有 $O(\log(n))$次查询，因此总时间复杂度为 $O(\log^2(n))$。

实际上，最终的代码并不涉及树的操作，我们只是利用图形的方式来方便思考。因此也可以只从数位的角度考虑：对于询问 n，我们找到一个等于 1 的数位，将它赋为 0，则它右面的数位可以任意取，我们需要统计其中恰好含有 $K-tot$ 个 1 的数的个数（其中 tot 表示这一位左边的 1 的个数），则可以利用组合数公式求解。逐位枚举所有"1"进行统计即可。

我们发现，之前推出的 f 正是组合数。同样是采用"逐位确定"的方法，两种方法异曲同工。当你觉得单纯从数位的角度较难思考时，不妨画出图形以方便思考。

【例题 2】Sorted bit sequence (spoj 1182)

题目大意：

将区间[m,n]内的所有整数按照其二进制表示中 1 的数量从小到大排序。如果 1 的数量相同，则按照数的大小排序。求这个序列中的第 k 个数。其中，负数使用补码来表示：一个负数的二进制表示与其相反数的二进制之和恰好等于 2^{32}。

例如，当 m=0，n=5 时，排序后的序列如下：

编号	十进制数	二进制表示
1	0	0000 0000 0000 0000 0000 0000 0000 0000
2	1	0000 0000 0000 0000 0000 0000 0000 0001
3	2	0000 0000 0000 0000 0000 0000 0000 0010
4	4	0000 0000 0000 0000 0000 0000 0000 0100
5	3	0000 0000 0000 0000 0000 0000 0000 0011
6	5	0000 0000 0000 0000 0000 0000 0000 0101

当 m=-5，n=-2 时，排序后的序列如下：

编号	十进制数	二进制表示
1	-4	1111 1111 1111 1111 1111 1111 1111 1100
2	-5	1111 1111 1111 1111 1111 1111 1111 1011
3	-3	1111 1111 1111 1111 1111 1111 1111 1101
4	-2	1111 1111 1111 1111 1111 1111 1111 1110

输入：包含多组测试数据。第一行是一个不超过 1000 的正整数，表示测试数据数量。每组数据包含 m,n,k 三个整数。

输出：对于每组数据，输出排序后的序列中第 k 个数。

数据规模：$m \times n \geq 0$， $-2^{31} \leq m \leq n \leq 2^{31}-1$ ，$1 \leq k \leq \min\{n-m+1, 2\ 147\ 473\ 547\}$。

分析：

我们首先考虑 *m*、*n* 同正的情况。

由于排序的第一关键字是 1 的数量，第二关键字是数的大小，因此我们很容易确定答案中 1 的个数：依次统计区间[*m*,*n*]内二进制表示中含 1 的数量为 0,1,2,…的数，直到累加的答案超过 *k*，则当前值就是答案含 1 的个数，假设是 *s*。利用例一的算法可以解决这个问题。同时，我们也求出了答案是第几个[*m*,*n*]中含 *s* 个 1 的数。因此，只需二分答案，求出[*m*,*ans*]中含 *s* 个 1 的数的个数进行判断即可。

由于每次询问的复杂度为 O(log(*n*))，故二分的复杂度为 $O(\log^2(n))$，这同时也是预处理的复杂度，因此此算法较为理想。

m<0 的情况，也不难处理，我们只要忽略所有数的最高位，求出答案后再将最高位赋回 1 即可。或者也可以直接将负数视为 32 位无符号数，采用同正数一样的处理方法。两种方法都需要特别处理 *n*=0 的情况。

【例题 3】Sequence (spoj 2319)

题目大意：

给定所有 K 位二进制数：0,1,…,2^K-1。你需要将它们分成恰好 M 组，每组都是原序列中连续的一些数。设 $S_i(1 \leq i \leq M)$表示第 i 组中所有数的二进制表示中 1 的个数，S 等于所有 S_i 中的最大值。你的任务是令 S 最小。

输入：两个整数 K 和 M。

输出：一个整数表示 S 的最小值。答案不保证可用 64 位整数储存。

数据规模：$1 \leq K \leq 100, 1 \leq M \leq 100, M \leq 2^K$。

分析：

直接做不容易做，我们可以通过二分答案来处理。

显然，假如我们知道了 *S* 的值，可以贪心求出序列最少被分为几组。当 *S* 变大时，分组数量是非增的，根据这个单调性，我们二分 *S*，然后贪心求出[0,2^K-1]的区间最少被划分的组数进行判断

剩下的问题就是如何求出组数。实际上这非常简单，由于 *M* 较小，因此我们每次从当前起点 *a* 开始找到最大的 *b* 满足 *count*(*b*)-*count*(*a*-1)≤*S*（其中 *count*(*i*)表示 0..*i* 的所有整数的二进制中 1 的数量），再以 *b*+1 为起点继续寻找，直到某次找到的 *b* 超过 2^K-1，或者已有区间数量超过 *M*。

利用逐位确定的方法，我们很容易求出 count(*i*)：类似例一，在高度为 *K* 的完全二叉树中，从根走到叶子 *i*，右转的时候累加左子树的权值。也就是找到 *i* 的所有二进制为 1 的数位，计算其权值。其中要注意累加之前已有的 1 的数量。

寻找 *b* 的方法也不难：设 *find*(*i*)表示最小的 *k* 满足 0..*k* 的数的二进制表示中 1 的数量不超过 *i* 的最大 *k*，则 *b*=*find*(*count*(*a*-1)+*s*)。*find*(*i*)也使用逐位确定的方法求解，在树中从根向叶子行走，若已有权值加上当前节点左子树的权值不超过 *i* 则向右走，否则向左走。

count(*i*)和 *find*(*i*)的复杂度都是 O(*K*)，因此算法总复杂度为 O(*MK*log(*S*))加上高精度的复杂度。

【例题 4】Tickets (sgu 390)

题目大意：

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，直到已卖出的票的编号的数位之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，售票员持有的票的编号是从 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。

输入：三个整数 L,R 和 k。

输出：输出一个整数，表示问题的答案。

数据规模：$1 \le L \le R \le 10^{18}$，$1 \le k \le 1000$。

分析：

本题与例 3 有些类似，也是对区间内的整数进行连续的分组。与例 3 不同的是，本题中 k 的值较小，也就表示分组数量非常巨大，不可能求出每一个分组。

让我们同之前一样，从特殊情况开始考虑。假设给定的区间是$[1..10^h-1]$（也就是一棵完整的高度为 h 的完全十叉树），考虑如何由子问题组合出原问题。这里我们遇到的问题是，两棵子树间的“合并”较难处理：前一棵子树的最后一个区间未必是满的。亦即，一棵子树的头几个元素会并入前一棵子树最后的分组当中。为了解决这个问题，我们引入一维，记录每棵子树之前有多少空间。因此就得到了下面的递推算法：

设 $f[h,sum,rem]$表示对于高度为 h 的完全十叉树，在这个区间内的每个数需要累加的数字和为 sum，在这个区间之前有 rem 的空间，所能得出的分组数量。f 的返回值需要记录两个数：$f[h,sum,rem].a$ 记录分组数量，$f[h,sum,rem].b$ 记录其最后一个小组剩余的空间，以便于与下一个子树合并。f 可由其十个子树推出：

```
for i:=0 to 9 do f[h,sum,rem]:=merge(f[h,sum,rem],f[h-1,sum+i,f[h,sum,rem].b]);
```

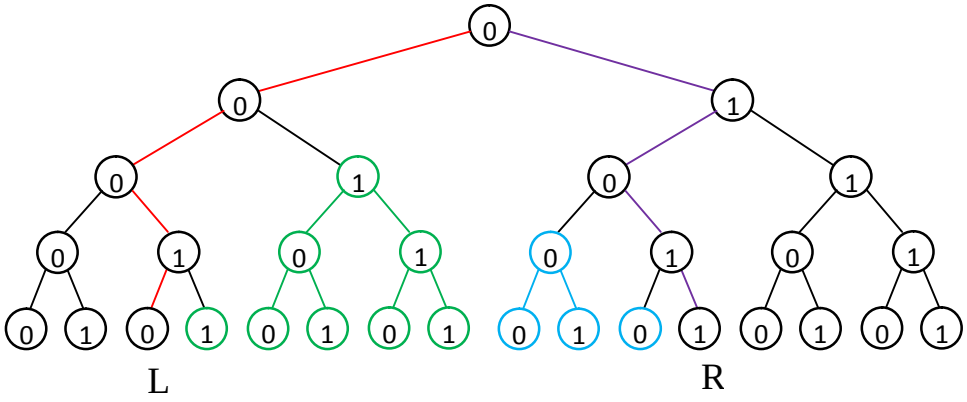
其中，merge 函数表示合并两个记录，它的伪代码是：

```
function merge(x,y);
  x.a:=x.a+y.a;
  x.b:=y.b;
  return x;
```

当 h=0 时，若 rem>0，则 $f[h,sum,rem].a$=0，否则 $f[h,sum,rem]$=1；b 需要根据 sum 和 k 的关系求出。

这样，我们就求出了完整子树的分组数量。接下来的工作就比较容易了：在完全十叉树中，我们从叶子 L 走到叶子 R，计算途中遇到的完整子树并加以合并即可。遍历的时候，从 L 所在的叶子一直向上走直到 L 和 R 的最近公共祖先的下一层，途中计算路径右侧的兄弟子树。然后在 LCA 的下一层，从 L 的祖先走到 R 的祖先，途中计算经过的子树。最后，向下走到 R，途中计算左侧的兄弟子树。当然，不能忘记统计 L 和 R 所在的叶子。

以下图为例：（为了方便起见这里使用二叉树，其思想与十叉树是一致的）



我们首先求出 L 和 R 的最近公共祖先（在这里就是根节点），然后从 L 所在的叶子向上走，每走一步就计算所有右侧的兄弟子树（图中两个绿色子树）。走到 LCA 的下一层后，再向右走到 R 的祖先。然后向下走到 R 所在叶子，计算途径的左侧兄弟子树（两个蓝色子树）。

需要特殊处理的是 L=R 的情况，以及 $R=10^{18}$ 的情况（如果只计算到 18 层）。另外，如果最后一个分组未满则不计入总组数，需要特别注意。

递推求 f 的复杂度为 $O(k\log^2(R))$，需要询问的子树数量是 $O(\log(R))$，因此总复杂度是递推的 $O(k\log^2(R))$。递推建议采用记忆化搜索实现。

【例题 5】Graduated Lexicographical Ordering (zju 2599)

题目大意：

考虑所有从 1 到 n 的整数。我们设一个整数 x 的十进制各位数字之和为 w(x)。

让我们重新将整数排序：对于整数 a 和整数 b，如果 w(a)<w(b)，则 a 在 b 之前。如果 w(a)=w(b)，那么 a 在 b 之前当且仅当 a 的十进制表示在字典序中小于 b 的十进制表示。例如：

$120 <_{grlex} 4$ 因为 w(120) = 1 + 2 + 0 = 3 < 4 = w(4)。

$555 <_{grlex} 78$ 因为 w(555) = 15 = w(78) 并且 "555" 的字典序小于 "78"。

$20 <_{grlex} 200$ 因为 w(20) = 2 = w(200) 并且 "20" 的字典序小于 "200"。

给定整数 n 和 k，你需要求出 1-n 中的数按照上述方法排序后，k 是第几个数，以及第 k 个数是多少。

输入：包含多组数据，每组数据包含两个整数 n 和 k。输入的最后一行是 n=k=0。

输出：对于每组数据，输出一行两个整数，分别表示 k 在 1-n 中的编号，以及第 k 个数是多少。

数据规模：$1 \le k \le n \le 10^{18}$。

分析：

首先考虑第一问：给定一个数 k，求其顺序。数字之和比 k 小的数一定在 k 之前，我们累加这些数。为此，我们需要一个函数 $count(n,i)$，统计[1,n]中数字和为 i 的数有多少，实现方法类似之前的例题，这里就不再赘述了。然后，我们需要求出在[1,n]中所有数字和为 $sumof(k)$的数满足字典序小于 k 的数量。

两个位数相同的数的字典序就是大小顺序，但由于数字不可以有前导 0，因此长度不同的数字比较起来会很麻烦。为了避免这种情况，我们只比较位数相同的数：分别求出 k，$k\times10$，$k\times100$，……以及 $k/10$，$k/100$，……亦即，从 1 到 n 枚举 k 的位数并添 0 或删位。对于每个长度的 k'，求出与之位数相同而小于 k 的数的个数，也就等于 $count(k',sumof(k))-f[lengthof(k')-1,sumof(k)]$。这里，记号 $sumof(k)$表示 k 的数位之和，$lengthof(k')$表示 k'的十进制长度，$f[i,j]$是求 $count(n,i)$时需要预处理的数据，表示所有长度为[0..10^i-1]的数中各个数位之和为 j 的数的个数。这样我们就求出了第一问。

然后考虑第二问。依次将 k 减去 $count(n,1)$,$count(n,2)$,…直到 k 即将小于 0，则可确定答案的数字之和。接下来只需求出[1,n]中数字之和为 s、字典序为 k 的数是几。遗憾的是，因为字典序不同于大小顺序，不可以利用第一问进行二分。我们可以使用一种较为暴力的逐位确定方法，一位位确定答案的数位。由于答案位数未知，因此我们需要首先确定答案的首位（枚举首位数字，用第一问的算法判断数量是否超过 k）。然后，每次在已有的答案右边添加一位，若添加之后合法数的个数恰为 k 则返回当前答案，否则继续添加直到出解。由于答案一定存在，添加过程不会超过 $O(\log(n))$次。

【总结】

从以上几道例题可以看出，数位类问题涉及的基本操作类似，模型性明显，在这些基本操作的基础上便可以演变出各种类型题目的变换。

解决问题的核心思想就是“逐位确定”思想。较常规的思想是在数位的基础上进行思考。与之对应的是树的思想，将数字看做完全二叉树进行处理。前者较为抽象，后者更加具体。因此在题目稍嫌繁琐时，使用后者往往可以化抽象为具体，使我们的思路更加清晰。

由于基本操作的复杂度是 O(log(*n*))级别的，因此在处理一些较繁琐问题时，可以适当牺牲时间复杂度，对一些子问题采用二分、穷举等方法以降低思考和编程复杂度。

在这里，我选择了几个比较有代表性的例子来介绍数位类的统计问题，希望能对读者有所启发。而更多的感知与技巧，还需要大家在做题的过程中慢慢摸索。

【感谢】

感谢青岛二中臧方青老师对我的指导。
感谢唐文斌教练提供例题 4。
感谢绍兴一中的董华星同学提供例题 2。
感谢所有关心、支持过我的老师、同学。

【附录】

例 1 原题：

Amount of degrees

Create a code to determine the amount of integers, lying in the set [X;Y] and being a sum of exactly K different integer degrees of B.

Example. Let X=15, Y=20, K=2, B=2. By this example 3 numbers are the sum of exactly two integer degrees of number 2:

17 = 24+20,
18 = 24+21,
20 = 24+22.

Input

The first line of input contains integers X and Y, separated with a space ($1 \le X \le Y \le 2^{31}-1$). The next two lines contain integers K and B ($1 \le K \le 20$; $2 \le B \le 10$).

Output

Output should contain a single integer — the amount of integers, lying between X and Y, being a sum of exactly K different integer degrees of B.

Sample

Input:

15 20

2

2

Output:

3

例 2 原题：

Sorted bit squence

Let's consider the 32 bit representation of all integers i from m up to n inclusive ($m \le i \le n$; $m \times n \ge 0$, $-2^{31} \le m \le n \le 2^{31}-1$). Note that a negative number is represented in 32 bit Additional Code. That is the 32 bit sequence, the binary sum of which and the 32 bit representation of the corresponding positive number is 2^{32} (1 0000 0000 0000 0000 0000 0000 0000 0000 in binary).

For example, the 32 bit representation of 6 is 0000 0000 0000 0000 0000 0000 0000 0110

and the 32 bit representation of -6 is 1111 1111 1111 1111 1111 1111 1111 1010

because

0000 0000 0000 0000 0000 0000 0000 0110 (6)
+
1111 1111 1111 1111 1111 1111 1111 1010 (-6)

= 1 0000 0000 0000 0000 0000 0000 0000 0000 (2^{32})

Let's sort the 32 bit representations of these numbers in increasing order of the number of bit 1. If two 32 bit representations that have the same number of bit 1, they are sorted in lexicographical order.

For example, with m = 0 and n = 5, the result of the sorting will be

No.	Decimal number	Binary 32 bit representation
1	0	0000 0000 0000 0000 0000 0000 0000 0000
2	1	0000 0000 0000 0000 0000 0000 0000 0001
3	2	0000 0000 0000 0000 0000 0000 0000 0010
4	4	0000 0000 0000 0000 0000 0000 0000 0100
5	3	0000 0000 0000 0000 0000 0000 0000 0011
6	5	0000 0000 0000 0000 0000 0000 0000 0101

with m = -5 and n = -2, the result of the sorting will be

No.	Decimal number	Binary 32 bit representation
1	-4	1111 1111 1111 1111 1111 1111 1111 1100
2	-5	1111 1111 1111 1111 1111 1111 1111 1011
3	-3	1111 1111 1111 1111 1111 1111 1111 1101
4	-2	1111 1111 1111 1111 1111 1111 1111 1110

Given m, n and k ($1 \le k \le \min\{n - m + 1, 2\,147\,473\,547\}$), your task is to write a program to find a number corresponding to k-th representation in the sorted sequence.

Input

The input consists of several data sets. The first line of the input file contains the number of data sets which is a positive integer and is not bigger than 1000. The following lines describe the data sets.
For each data set, the only line contains 3 integers m, n and k separated by space.

Output

For each data set, write in one line the k-th number of the sorted numbers.

Example

Sample input:

2

0 5 3

-5 -2 2

Sample output:

2

-5

例 3 原题：

Sequence

You are given the sequence of all K-digit binary numbers: 0, 1,..., 2^K-1. You need to fully partition the sequence into M chunks. Each chunk must be a consecutive subsequence of the original sequence. Let S_i ($1 \le i \le M$) be the total number of 1's in all numbers in the ith chunk when written in binary, and let S be the maximum of all S_i, i.e. the maximum number of 1's in any chunk. Your goal is to minimize S.

Input

In the first line of input, two numbers, K and M ($1 \le K \le 100$, $1 \le M \le 100$, $M \le 2^K$), are given, separated by a single space character.

Output

In one line of the output, write the minimum S that can be obtained by some split. Write it without leading zeros. The result is not guaranteed to fit in a 64-bit integer.

Example

Input:

3 4

Output:

4

例 4 原题：

Tickets

Conductor is quite a boring profession, as all you have to do is just to sell tickets to the passengers. So no wonder that once upon a time in a faraway galaxy one conductor decided to diversify this occupation. Now this conductor sells several tickets at a time to each passenger. More precisely, he successively gives tickets to the passenger until the sum of the digits on all the tickets given becomes not less than some integer number k. Then this process repeats for the next passenger. Initially conductor has a tape of tickets numbered successively from l to r, inclusive. This way of tickets distribution is quite good, because passengers are glad to get several tickets when they pay only for one. But there is one disadvantage. Since each passenger gets several tickets, it is possible

that conductor won't be able to serve all passengers. Your task is to help conductor in this difficult situation. You should calculate how many passengers is the conductor able to serve.

Input

Input file contains three integer numbers l, r and k ($1 \le l \le r \le 10^{18}$, $1 \le k \le 1000$) .

Output

Output should contain exactly one number — the answer to the problem.

Sample input

40 218 57

Sample output

29

例 5 原题：

Graduated Lexicographical Ordering

Consider integer numbers from 1 to n. Let us call the sum of digits of an integer number its weight. Denote the weight of the number x as w(x).

Now let us order the numbers using so called *graduated lexicographical ordering*, or shorter grlex ordering. Consider two integer numbers a and b. If w(a) < w(b) then a goes before b in grlex ordering. If w(a) = w(b) then a goes before b in grlex ordering if and only if the decimal representation of a is lexicographically smaller than the decimal representation of b.

Let us consider some examples.

- $120 <_{grlex} 4$ since w(120) = 1 + 2 + 0 = 3 < 4 = w(4).
- $555 <_{grlex} 78$ since w(555) = 15 = w(78) and "555" is lexicographicaly smaller than "78".
- $20 <_{grlex} 200$ since w(20) = 2 = w(200) and "20" is lexicographicaly smaller than "200".

Given n and some integer number k, find the position of the number k in grlex ordering of integer numbers from 1 to n, and the k-th number in this ordering.

Input

There are several lines in the input file, and each line stands two integers n and k (1 <= k <= n <= 10^{18}). A line with n = k = 0 ends up the input.

Output

For each line in the input, output one line in the output file. First print the position of the number k in grlex ordering of integer numbers from 1 to n, and then the integer that occupies the k-th position in this ordering.

Sample Input

20 10

0 0

Sample Output

2 14