#题目

http://codeforces.com/contest/914/problem/C

#分析

##（一）reduce规则

###例1：13-->1，需要三步：

|  |  |  |
| --- | --- | --- |
| 十进制 | 二进制 | 1的个数 |
| 13 | 1101 | 3 |
| 3 | 11 | 2 |
| 2 | 10 | 1 |

###例2：1023-->1，需要3步

|  |  |  |
| --- | --- | --- |
| 十进制 | 二进制 | 1的个数 |
| 1023 | 1111111111 | 10 |
| 10 | 1010 | 2 |
| 2 | 10 | 1 |

###例3：255-->1，需要4步

|  |  |  |
| --- | --- | --- |
| 十进制 | 二进制 | 1的个数 |
| 255 | 1111111 | 7 |
| 7 | 111 | 3 |
| 3 | 11 | 2 |
| 2 | 10 | 1 |

##（二）特别数字（Special Number）

以1~13为例，

1-->1需要0步，

2-->1需要1步：2=(10)B-->1

3-->1需要2步：3=(11)B-->2, 2=(10)B-->1

4-->1需要1步：4=(100)B-->1

5-->1需要2步：5=(101)B-->2, 2=(10)B-->1

6-->1需要2步：6=(110)B-->2, 2=(10)B-->1

7-->1需要3步：7=(111)B-->3, 3=(11)B-->2, 2=(10)B-->1

8-->1需要1步：8=(1000)B-->1

9-->1需要2步：9=(1001)B-->2, 2=(10)B-->1

10-->1需要2步：10=(1010)B-->2, 2=(10)B-->1

11-->1需要3步：11=(1011)B-->3, 3=(11)B-->2, 2=(10)B-->1

12-->1需要2步：12=(1100)B-->2, 2=(10)B-->1

13-->1需要3步：13=(1101)B-->3, 3=(11)B-->2, 2=(10)B-->1

14-->1需要3步：14=(1110)B-->3, 3=(11)B-->2, 2=(10)B-->1

15-->1需要2步：15=(1111)B-->4, 4=(100)B-->1

16-->1需要1步：16=(10000)B-->1

这里的步数，即为题目中的k。

当k = 1时，表示经过1步可以转化为1。在1~16中，有4个数符合要求：2，4，8，16。即特别数字有3个。

当k = 2时，表示经过2步可以转化为1。在1~16中，有7个数符合要求：3，5，6，9，10，12，15。即特别数字有7个。

当k = 3时，表示经过4步可以转化为1。在1~16中，有3个数符合要求：7，11，13，14。即特别数字有4个。

##（三）求n中1的个数。

n = 1，1的个数为1

n = 2 = (10)B，1的个数为1

n = 3 = (11)B，1的个数为2

n = 4 = (100)B，1的个数为1

n = 5 = (101)B，1的个数为2

n = 6 = (110)B，1的个数为2

n = 7 = (111)B，1的个数为3

n = 8 = (1000)B，1的个数为1

n = 9 = (1001)B，1的个数为2

n = 10 = (1010)B，1的个数为2

n = 11 = (1011)B，1的个数为3

n = 12 = (1100)B，1的个数为2

n = 13 = (1101)B，1的个数为3

n = 14 = (1110)B，1的个数为3

n = 15 = (1111)B，1的个数为4

n = 16 = (10000)B，1的个数为1

……

##（四）动态规划

根据（二）中的分析，可以利用动态规划求步数。用dp[x]来表示某数经过x步后转化为1。

dp[1] = 0

dp[2] = dp[2中1的个数] + 1 = dp[1] + 1 = 1

dp[3] = dp[3中1的个数] + 1 = dp[2] + 1 = 2

dp[4] = dp[4中1的个数] + 1 = dp[1] + 1 = 1

dp[5] = dp[5中1的个数] + 1 = dp[2] + 1 = 2

dp[6] = dp[6中1的个数] + 1 = dp[2] + 1 = 2

dp[7] = dp[7中1的个数] + 1 = dp[3] + 1 = 3

dp[8] = dp[8中1的个数] + 1 = dp[1] + 1 = 1

dp[9] = dp[9中1的个数] + 1 = dp[2] + 1 = 2

dp[10] = dp[10中1的个数] + 1 = dp[2] + 1 = 2

dp[11] = dp[11中1的个数] + 1 = dp[3] + 1 = 3

dp[12] = dp[12中1的个数] + 1 = dp[2] + 1 = 2

dp[13] = dp[13中1的个数] + 1 = dp[3] + 1 = 3

dp[14] = dp[14中1的个数] + 1 = dp[3] + 1 = 3

dp[15] = dp[15中1的个数] + 1 = dp[4] + 1 = 2

dp[16] = dp[16中1的个数] + 1 = dp[1] + 1 = 1

代码为：

```

int ones(int n)

{

int cnt = 0;

while(n)

{

if(n%2 == 1)

{

cnt++;

}

n /= 2;

}

return cnt;

}

```

##（五）求组合

```

void combination()

{

for(int i = 0; i <= 1000; i++)

{

com[i][0] = 1;

}

for(int i = 1; i <= 1000; i++)

{

for(int j = 1; j <= 1000; j++)

{

com[i][j] = (com[i-1][j-1] + com[i-1][j])%MOD;

}

}

}

```

##（六）利用组合求特别数字

###例1：n = (1101)B，k = 1

① 最高位的1替换成0，则四位数为0XXXX。符合条件的有0100, 0010, 0001。也就是C(3, 1)，注意到0001变成1需要0步，不符合k=1，要去掉。所以比n=1101小的特别数字有C(3, 1) - 1个。

② 次高位的1替换成0，则四位数为10XX，这里两个X都必须为0，才符合条件。所以比n=1101小的特别数字有C(2, 0) 个。

③ 第三位为0，不用计算。

④ 把第四位的1替换成0，则四位数为1100。不符合题意。

综上，答案为C(3, 1) - 1 + C(2, 0) = 3

###例2：n = (1101)B，k = 2

① 最高位的1替换成0，则四位数为0XXX。

所以比n=1101小的特别数字有C(3, 2)个，即0011, 0101, 0110。

② 次高位的1替换成0，则四位数为10XX。

所以比n=1101小的特别数字有C(2, 1)个，即1001, 1010。

③ 第三位本身即为0，不用计算。

④ 把第四位的1替换成0，则四位数为1100。符合题意。

1100所对应的组合可看成是C(0, 0)。

综上，答案为C(3, 2) + C(2, 1) + C(0, 0) = 3

###例3：n = (1101)B，k = 3

① 最高位的1替换成0，则四位数为0XXXX。

所以比n=1101小的特别数字有C(3, 3)个，即0111。

② 把次高位的1替换成0，则四位数为10XX。

所以比n=1101小的特别数字有C(2, 2)个，即1011。

③ 第三位本身即为0，不用计算。

④ 把第四位的1替换成0，则四位数为1100。不符合题意。

⑤ 最后要计算一下所有的1都没被0替换的情况，即n = 1101本身，这个数也符合要求。

综上，答案为C(3, 3) + C(2, 2) + 1 = 3

###例4：n = (10000)B, k = 2

① 把最高位的1替换成0，则变为XXXX。

比n=10000小的数有C(4, 2)个，即0011，0101，1001，0110，1010，1100

② 第2、3、4、5位数都是0，不用计算。

③ 最后要计算一下所有的1都没被0替换的情况，即n = 10000本身，这个数也符合要求。

综上，答案为C(4, 2) + 1 = 7

#完整代码

```

#include <bits/stdc++.h>

using namespace std;

#define MOD 1000000007

#define MAX 1000

int dp[MAX + 1];

long long com[MAX + 1][MAX + 1];

int ones(int n)

{

int cnt = 0;

while(n)

{

if(n%2 == 1)

{

cnt++;

}

n /= 2;

}

return cnt;

}

void combination()

{

for(int i = 0; i <= MAX; i++)

{

com[i][0] = 1;

}

for(int i = 1; i <= MAX; i++)

{

for(int j = 1; j <= MAX; j++)

{

com[i][j] = (com[i - 1][j - 1] + com[i - 1][j]) % MOD;

}

}

}

int main()

{

string n;

int k;

combination();

dp[1] = 0;

for(int i = 2; i <= MAX; i++)

{

dp[i] = dp[ones(i)] + 1;

}

cin >> n >> k;

if(k == 0)

{

cout << "1\n";

return 0;

}

long long oneCnt = 0, ans = 0;

for(int i = 0; i < n.size(); i++)

{

if(n[i] == '0')

{

continue;

}

// 把第j位上的1用0来代替

for(int j = max(oneCnt, 1LL); j <= n.size(); j++)

{

if(dp[j] == k - 1)

{

long long temp = com[n.size() - i - 1][j - oneCnt];

ans = (ans + temp) % MOD;

// 1-->0，需要0步，需要把这种情况排除掉

if(i == 0 && k == 1)

{

ans = (ans + MOD - 1) % MOD;

}

}

}

oneCnt++;

}

int cnt = 0;

for(int i = 0; i < n.size(); i++)

{

if(n[i] == '1')

{

cnt++;

}

}

// 最后要考虑n本身，能否构成一个special number

if(dp[cnt] == k - 1)

{

ans = (ans + 1) % MOD;

}

cout << ans << endl;

return 0;

}

```