{% note info %} **摘要** Title: 1960. 闪烁 Tag: 状态压缩，位运算，找环 Memory Limit: 64 MB Time Limit: 1000 ms {% endnote %}

Powered by:**NEFU AB-IN**

Link

@TOC

1960. 闪烁

- 题意

> 农夫约翰对牛棚里昏暗的灯光感到不满，刚刚安装了一个新吊灯。 新吊灯由 N 个灯泡组成，这 N 个灯泡围成一圈，编号为 0~N−1。奶牛对这个新吊灯非常着迷，并且喜欢玩以下游戏： 对于第 i 个灯泡，如果在 T−1 时刻，它左侧的灯泡（当 i>0 时，为第 i−1 个灯泡；当 i=0 时，为第 N−1 个灯泡）是开着，那么在 T 时刻，就切换这个灯泡的状态。 这个游戏将持续 B 单位时间。 给定灯泡的初始状态，请确定在 B 单位时间后，它们的最终状态。

- 思路

可以看到一共最多16个灯泡，也就是最多`1 << 16`个状态，而B有10^{15}，肯定会有循环节，所以就是要找环 其次就是如何更新二进制串的问题，可以采用**状态压缩**的方法

  - 将二进制转换为十进制进行比较和存储
  - 利用位运算进行计算
    - `n >> k & 1`：**代表n第k位的数值是多少**
    - `n |= (1 << k)`：**代表将n的第k位转化为1**
    - `n & -n`：**返回n的最后一位1**
  - 最后找出循环节，只需要走余数步即可

- 代码

如果要函数中也使用某个变量，则声明为global

```
'''
Author: NEFU AB-IN
Date: 2022-01-14 15:37:40
FilePath: \ACM\Acwing\1960.py
LastEditTime: 2022-01-14 21:21:21
'''

p = [-1] * (1 << 16)
n = 0


def update(state):
    res = 0
```

```
    for i in range(n):  #枚举每一位
        j = (i - 1 + n) % n  #代表前一位的位置
        x = state >> i & 1  #state i 位置上的位置
        y = state >> j & 1  #state j 位置上的位置
        res |= (x ^ y) << i  #stste i 位置上的数变成x^y
    return res


def printS(state):
    for i in range(n):
        print(state >> i & 1)


def solve():
    state = 0
    global n
    n, b = map(int, input().split())
    for i in range(n):
        x = int(input())
        state |= x << i  #状态压缩，将二进制串储存成十进制
    i = 1
    while True:
        state = update(state)  #更新状态
        if i == b:
            printS(state)
            return
        elif p[state] == -1: #说明没有出现过
            p[state] = i
        else:  #说明进入下一个循环了
            len = i - p[state]  #循环节
            r = (b - i) % len  #余数
            while r > 0:
                state = update(state)
                r -= 1
            printS(state)
            return
        i += 1


if __name__ == "__main__":
    solve()
```