

# 生日蜡烛

## 【题意简述】

有  $n$  根蜡烛和  $n$  个开关，每根蜡烛至多由两个开关控制。第  $i$  根蜡烛由第  $i$  个开关控制，此外，还可能由另一个开关  $j$  控制，并且满足当  $i > 1$  时  $j < i$ ，当  $i = 1$  时  $j$  为任意数。有  $m$  个操作，动态修改某根蜡烛的状态和控制它的开关，求每次操作后把所有蜡烛都熄灭最少要按几次开关，无解输出 -1。

## 【数据范围】

10%的数据满足： $n、m \leq 10$

20%的数据满足： $n、m \leq 1000$

另有 20%的数据满足：控制蜡烛的开关不会更改，且控制蜡烛  $i$  的开关为  $i$  和  $(i+n-2)\%n+1$

另有 30%的数据满足：控制蜡烛的开关不会更改

100%的数据满足： $n、m \leq 100000$

## 【算法分析】

### 算法 1:

每个开关按奇数次和按一次效果一样，按偶数次和不按效果一样。因此可以先  $2^n$  枚举每个开关按或不按，再检查所有蜡烛最终是否熄灭。

时间复杂度:  $O(mn2^n)$

空间复杂度:  $O(n)$

期望得分: 10 分

### 算法 2:

先枚举第一个开关按或不按，第  $i(i > 1)$  个开关便可以由之前的开关和第  $i$  根蜡烛的状态推导出来。最后再检查第 1 根蜡烛即可。

时间复杂度:  $O(mn)$

空间复杂度:  $O(n)$

期望得分: 20 分

### 算法 3:

将问题抽象一下，把开关看做点，蜡烛看做连接两点的边。若蜡烛是亮的，则控制它的两个开关中应按一个，否则两个开关应同时按或同时不按。

为了方便处理，我们可以增设 0 号开关，并规定它不能按，这样当蜡烛只由一个开关控制时便不需特殊考虑。

由于第  $i$  根蜡烛一定由第  $i$  个开关控制，若是把控制  $i$  号蜡烛的另一个开关看做  $i$  的前驱，那么每个点有且仅有一个前驱，得到的图是环套树森林。

类似**算法 2**，我们先不考虑第 1 根蜡烛，直接将 1 号点连到 0 号点，计算答案后再枚举 1 号开关按或不按，环套树森林就简化成了树。

20%的数据中，树的初始形态为链，且结构不发生改变，那么，当蜡烛  $i$  状态改变时，第  $i$  个开关到第  $n$  个开关的状态全部需要改变。

这是可以用线段树来维护的，修改操作即将一段区间取反。

询问最少要按多少次开关，别忘了先枚举 1 号开关，再查询控制 1 号蜡烛的开关的状态，检查 1 号蜡烛最终是否熄灭。若是最终熄灭，序列中 1(我们用 1 或 0 代表按或不按)的个数就是要按开关的次数，否则当前解不合法。

**时间复杂度:** $O((n+m)\log n)$

**空间复杂度:** $O(n)$

**期望得分:** 20 分

结合**算法 2**，可以得到 40 分

**算法 4:**

50%的数据中，树的形态不发生改变，那么，当蜡烛  $i$  状态改变时，以  $i$  为根的子树中开关的状态全部需要改变。

子树改变使我们联想到了 dfs 括号序列。

我们对一棵树进行深度优先遍历。

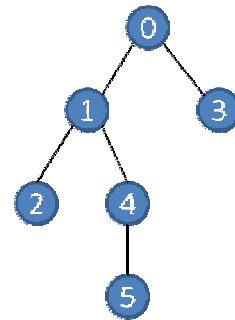
如右图，其 dfs 括号序列为：

0 1 2 2 4 5 5 4 1 3 3 0

可以发现，以  $i$  为根的子树即两个  $i$  之间的连续一段。

利用 dfs 括号序列，我们可以设计如下算法：求出树的 dfs 括号序列，将开关  $i$  的信息(0 或 1)存入前一个  $i$ ，将后一个  $i$  看做 2。蜡烛  $i$  状态改变时，我们需要将子树  $i$  对应的 dfs 括号序列中的 0 变为 1, 1 变为 0。

这同样可以用线段树来维护。



**时间复杂度:** $O((n+m)\log n)$

**空间复杂度:** $O(n)$

**期望得分:** 50 分

结合**算法 2**，可以得到 70 分

**算法 5:**

树的形态改变，对应着 **dfs** 括号序列的变化。更改某个节点的父亲，即将一段 **dfs** 序列移至另一处。

这是可以用平衡树来维护的。

修改  $i$  号蜡烛，就是将开关  $i$  对应的区间插入至新位置，并根据蜡烛  $i$  与控制它的开关的状态判断子树是否需要取反。

**时间复杂度:**  $O((n+m)\log n)$

**空间复杂度:**  $O(n)$

**期望得分:** 100 分