T1

分类讨论题，手玩可以发现答案不会超过 2。

40 分

暴力枚举所有可能的情况，时间复杂度为 $\mathcal{O}(n!)$。

60 分

当序列 A 中 0 的个数不超过 $\lfloor\frac{n+1}{2}\rfloor$ 时，可以发现答案一定是 0，因为可以 $0,x,0,y,0,\ldots$ 这样交替着放，保证没有两个 0 是挨着的。

正解

答案为 0 就是上面这种情况。

答案为 1 有以下两种情况：

- 所有数都是 0。
- 存在一个数大于 1，因为不能让 0 和 1 挨着，所以用这个大于 1 的数把 0 和 1 隔开就行。

否则答案一定为 2。

T2

30 分

暴力搜索划分方案即可。

60 分

写一个简单 dp 就行，设 $dp_{i,j}$ 表示划分到了 i，当前 $\gcd=j$ 是否可行，转移是简单的，时间复杂度 $\mathcal{O}(n^3V)$，其中 V 是值域。

80 分

手玩可以发现答案一定是 $\frac{n}{2}$。

正解

注意到一个关键性质：$\gcd(a+b,c)\geq\gcd(a,b,c)$。

证明是容易的，设 $d=\gcd(a,b,c)$，那么 $d|(a+b)$，于是 $\gcd(a+b,c)\geq d$ 肯定成立。

这个性质说明最优划分一定只会划分为两段，否则将相邻的两段合并起来一定更优。

于是滚一个前缀和，暴力枚举分割点即可，时间复杂度 $\mathcal{O}(n\log V)$。

T3

20 分

暴力 dfs 即可。

$m = 1$

考虑在二进制上解决该问题。

因为最初只有一块蛋糕，那么这个问题就变得非常简单。

从高位向低位考虑，如果当前 n 这一位为 1，那么就选择最小的蛋糕一直切，直到切出了当前二进制位对应大小的蛋糕。

正解

首先无解是好判断的，所有蛋糕的和小于 n 就无解，否则可以把蛋糕都切成大小为 1。

考虑到 n 很大，而且 a_i 有一个 2 的幂次的限制，不如在二进制下考虑问题。

因为把高位的 1 往下一位拆会有 1 的代价，而低位的 1 向高位的合成是免费的，于是从低位向高位贪心。

假设现在贪心到了第 k 位（n 的二进制表示下第 k 位为 1），如果当前存在一块蛋糕大小为 2^k，那么直接用了，剩下的蛋糕向高位合成（两块大小为 2^k 的蛋糕可以当成一块大小为 2^{k+1} 的蛋糕），保证蛋糕的充分利用。

如果不存在，怎么办呢？考虑将大的蛋糕往下面拆，找到最小的一块比 2^k 大的蛋糕，往下一直切，知道切出一块大小为 2^k 的蛋糕即可。

这样贪心的正确性显然，时间复杂度 $\mathcal{O}(m + \log^2 n)$。

T4

20 分

暴力枚举每一行翻不翻转，然后计算答案即可，时间复杂度 $\mathcal{O}(m^2 2^n)$。

40 分

注意到如果钦定 (i, j) 这个位置为 1，那么所有行的翻转情况就固定了！

这告诉我们什么呢？可以直接枚举钦定为 1 的位置，然后确定所有行的翻转情况并计算答案。

枚举的位置数是 $\mathcal{O}(nm)$ 的，每次计算答案是 $\mathcal{O}(nm)$ 的，于是复杂度变成了 $\mathcal{O}(n^2 m^2)$。

60 分

考虑优化上面的做法，可以发现每次重新计算答案将整个矩阵扫一遍是非常愚蠢的，将翻转情况与上次不同的行重新拿出来更新一下就行。时间复杂度 $\mathcal{O}(nm \max(n, m))$。

80 分

计算哪些列只有一个 1 的过程可以使用 bitset 优化，时间复杂度 $\mathcal{O}(\frac{nm \max(n,m)}{w})$。

正解

真的有必要枚举位置后每次都去算一遍答案吗？

考虑如果 $(i, j), (k, l)$ $(j \neq l)$ 两个位置能够同时为 1，那么两个位置所对应的行的翻转情况一定是一样的！

于是考虑对每个位置所对应的行的翻转情况进行 hash，一种行的翻转情况对应 hash 值计算方式如下：$\sum_{i=1}^{n} C_i p^i$，其中 C_i 表示第 i 行是否翻转，p 为取的一个大质数。将所有的 hash 值丢进一个 map 里，相同值个数最多的就是答案。

时间复杂度 $\mathcal{O}(nm)$。