A

结论：对于所有 $x \in [0, 2^k)$，所有数异或和为 0 的方案数和异或和为 x 的方案数是相等的，考虑把所有数都异或上 x 就能实现两边方案的一一对应。

所以，直接把异或和为 0 的限制去掉，答案只需要除以 2^k 就行了。

然后假装第二列没有最后一个数，计算方案数，最终答案乘以 $2^k - n$ 即可。

剩下的方案数，考虑对于每一列的数不同的容斥，答案如下：

$$\sum_{i=0}^{n} \binom{n}{i} (2^k)^{\underline{i}} \left((2^k - i)^{\underline{n-i}} \right)^2 = (2^k)^{\underline{n}} \sum_{i=0}^{n} \binom{n}{i} (2^k - i)^{\underline{n-i}}$$

其中 $n^{\underline{m}}$ 是 n 的 m 阶下降幂，即 $n \times (n-1) \times \cdots \times (n-m+1)$。

B

直接 dp 即可，设 $f_{i,j}$ 表示 $1 \sim i$ 基站中，i 建了基站，i 的上一个基站在 j 的最小费用，转移如下：

$$f_{i,j} = a_i + \min_{x=\max(1,i-k)}^{j-1} f_{j,x}$$

容易发现 $i - j$ 至多为 k，所以实际上的状态数是 $O(nk)$ 的，同时转移是 $O(k)$ 的，所以暴力是 $O(nk^2)$ 的。

优化只需要把转移用后缀 $\min$ 处理一下即可每次 $O(1)$ 转移，时间复杂度 $O(nk)$。

C

通过一丢丢观察力可以发现，答案是 $2^{n-1} - 1$，接下来说明为什么。

设 $f(n)$ 表示 n 的答案。首先需要发现一个性质：题中描述的操作是可逆的，而且它不关注最小步数，所以怎样操作都可以。

考虑整个过程，一定是先把除了最大的其他一整堆圆盘，分到剩余 $n-2$ 个柱子上，然后把 n 移动过去，再把原来分出去的圆盘依次移动回去就行了。

那么最大的那一堆，只能有一个圆盘，即 $f(1) + 1$；次大的那一堆，由于第一堆已经空出来了，所以这一堆可以有 $f(2) + 1$ 个圆盘；以此类推。

所以 $f(n) = \sum_{i=1}^{n-1} 1 + f(i)$，可得 $f(n) = 2^{n-1} - 1$。

D

线段树合并做法：对于一个子树，维护每个颜色的出现次数 x，则权值为 $\max(x, c-x)$，c 是该颜色的全局出现次数，然后线段树二分查询答案，合并时只需要使用线段树合并即可。但是需要注意空子树也是有权值的，实现上这是唯一的细节。时间复杂度 $O(n\log n)$，空间复杂度 $O(n\log n)$。

树上启发式合并做法：直接全局开一个线段树维护每个颜色的 $\max(x, c-x)$，然后直接树上 DSU 就做完了。时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n)$。

E

签到题，当 $d \le \lceil \frac{m}{n} \rceil$ 时输出 `Yes` 否则输出 `No`。

F

区间 dp，设 $f_{l,r,i,j}$ 表示区间 $[l,r]$ 的糖果，最后只剩下了一个类别为 i 等级为 j 的糖果的最大快乐值。$g_{l,r}$ 表示区间 $[l,r]$ 的糖果全部吃掉了的最大快乐值。转移如下：

$$
\begin{aligned}
f_{l,r,i,j} &\leftarrow g_{l,k} + f_{k+1,r,i,j} \\
f_{l,r,i,j} &\leftarrow f_{l,k,i,j} + g_{k+1,r} \\
f_{l,r,i,j} &\leftarrow f_{l,k,i,j-1} + f_{k+1,r,i,j-1} \\
g_{l,r} &\leftarrow f_{l,r,i,j} + V_i \times P^{j-1}
\end{aligned}
$$

G

直接树形 dp 即可，$f_{u,0/1/2}$，需要记录三种状态：0 表示 u 和 u 的所有儿子都没有建立过基站；1 表示 u 没有建立基站，但是 u 的某个儿子建立了基站；2 表示 u 建立了基站。

转移：$f_{u,0}$ 需要从所有 $f_{v,1}$ 转移来；$f_{u,1}$ 需要有至少一个儿子选择的是 $f_{v,2}$，剩下的选择 $f_{v,1}$；$f_{u,2}$ 直接从所有 $\min(f_{v,0}, f_{v,1}, f_{v,2})$ 转移来即可。

H

考察 Floyd 算法，用 Floyd 算法可以求出最小环的长度，同时算出路径的方案数，但是统计的时候，如果对于每个点都统计一下的话，一个包含 k 个点的最小环会被计算 k 次，不可行。所以需要对于 Floyd 有一定的了解，为什么要按照 k,i,j 的顺序枚举，实际上就是一个点一个点加入，维护两两点对之间的最短路，那么每次插入一个点时计算包含这个点的最小环和方案数即可做到不算重。时间复杂度 $O(n^3)$，空间复杂度 $O(n^2)$。

I

需要特判 $n=1$ 的情况，剩下的情况可以发现 1 号位置可以不用考虑，所以每次操作的 $[l,r]$，可以拆成 $[1,l-1]$ 和 $[1,r]$，这样就能解决奇偶性的问题，所有只需要考虑最多操作 k 次即可。

然后发现，只需要每次贪心把最前面的一段极长的 0 的连续段修改为全 1 的就是最优的，时空复杂度 $O(n)$。

J

考虑一开始所有物品都没有升级，算出极差最大值，假设 a 属性极差最大，那么容易发现，此时 a 属性的最小值对应的那个物品，一定需要升级，要不然极差一点无法变小。所以把它升级完之后，重复上述过程直到无法升级为止。时间复杂度 $O(n \log n)$，瓶颈在于离散化/排序，后续部分都是 $O(n)$ 的。

K

对于每个询问，枚举相遇的位置为 i，则 Alice 到达 i 的时间可以表示为 $\operatorname{dis}(S_a, i) + 2k \operatorname{dis}(S_a, T_a)$ 或 $\operatorname{dis}(T_a, i) + (2k+1) \operatorname{dis}(S_a, S_b)$，对于 Alice 和 Bob 分别讨论一下是那种情况，那么列出方程：

$$2k_a \operatorname{dis}(S_a, T_a) + r_a = 2k_b \operatorname{dis}(S_b, T_b) + r_b$$

用 exgcd 求解一下 k_a 的最小正整数解即可。但是这样的时间复杂度是 $O(n^2 + nm \log V)$ 的，无法通过。

但是我们发现，每次求解 exgcd 的参数是一样的，仅有 r_a, r_b 不同，只需先计算 $ax + by = \gcd(a, b)$ 的解 (x_0, y_0)，然后每次直接乘以一个倍数即可。时间复杂度 $O(n^2 + m \log V + nm)$。

但是实际上我们可以通过数论手段（类欧几里得算法/万能欧几里得算法）将这道题优化到 $O(n + m(\log n + \log V))$。

L

枚举 v_3, v_8，然后中间的四个点，只需要从所有 v_3, v_8 的公共邻居中任取即可，剩下两个点，从 v_3 的剩余邻居中取出来即可，然后 v_3, v_8 公共邻居个数可以使用 bitset 求出，时间复杂度 $O(\frac{n^3}{\omega})$。