Sky 不会算数

考虑每一个二进制位的答案：

- **and**

每个数这个二进制位是 1 贡献为 1 否则为 0。

如果 l 这一位为 0 则贡献为 0，否则就看是否有进位，如果没有进位则贡献为 1，反之为 0。

- **or**

每个数这个二进制位是 0 贡献为 0 否则为 1。

如果 l 这一位为 1 则贡献为 1，否则就看是否有进位，如果没有进位则贡献为 0，反之为 1。

- **xor**

考虑每一二进制位的进位情况，如：

000 1111000011110000 **11**

不难发现中间的异或均为 0，关键就是看两边，如果 1 的个数为奇数，则贡献为 1。

注意特殊考虑第一位的情况： $01010101...$

Sky 不会图论

算法一

Floyd / Dijkstra / BFS 求出每个标记点到其他点的最短路。

时间复杂度： $O(n^3)$ $O(k(n+m))$

期望得分： $10-20$

算法二

对于链的情况，由于边权非负，距离点 u 最近的其他标记点只可能是 u 左 / 右最靠近 u 的标记点。

正序 / 倒序遍历两次，用一个变量记录上个标记点到当前访问到的点的距离，更新答案即可。

时间复杂度： $O(n)$

结合前述，期望得分 $10-30$

算法三

所有点的父亲都是 1 号点时，按照 1 号点是否为标记点进行讨论。

- 是， 1 号点的答案是最小的边权，其他点的答案是该点到 1 的边权。
- 不是，找出离 1 号点最近和次近的标记点 u,v ， u 的答案是 $\mathrm{dis}(1,u)+\mathrm{dis}(1,v)$ ，其他点 w 的答案则是 $\mathrm{dis}(1,w)+\mathrm{dis}(1,u)$ 。

时间复杂度： $O(n)$

结合前述，期望得分 $10-40$

算法四

树形 DP

首先向祖先方向更新信息。对于每个点求出子树内离它最近的标记点是哪个，记录距离（简称为子树内的答案，子树外同）；再利用第一次求出的信息反向更新，求出子树外的答案，进行合并。

但考虑如下情况：

v 是 u 子树内离 u 最近的点， v 会成为 u 子树内的答案。

若 v 子树外的答案本应是点 w 但 $2\times\mathrm{dis}(u,v)<\mathrm{dis}(w,v)$ ， v 就会用 $u\to v\to u$ 这段路径代替 $u\to w$ ， u 子树外的答案就成了本身。

所以要同时维护次小值。~~不排除有优秀实现不需要这样做~~

时间复杂度： $O(n)$

期望得分： 100

算法五

新建一个点向所有标记点连一条权值为 0 的边，并作为源点跑 Dijkstra，可以求出离点 u 最近的标记点 $f(u)$ 和它们之间的距离 $dis(u)$。若一条边 $< u, v, w >$ 满足 $f(u) \neq f(v)$，那么 $f(u)$ 经过这条边到 $f(v)$ 的最短距离一定是 $dis(u) + w + dis(v)$，找出所有满足条件的边，取最小值。

时间复杂度： $O(n \log n)$

期望得分： 70

PS：算法五也适用于图，但并不知道能否在树上做到 $O(n)$。~~有算法五的线性做法请告诉我~~

Sky 不会走路

对于 $n, m \leq 50, m - n \leq 10$ 的情况，暴搜即可。

对于 $n, m \leq 10^5$ 的情况，先求一次最短路长度 l，然后二分答案 a，每次只保留权值不小于 a 的边和可以经过的点求最短路，根据求出的长度是否等于 l 来判断是否可行。

对于一般情况，注意到所有最短路构成一个 DAG，可以求出之后在上面 DP。令 f_u 表示最短路 DAG 上从 1 到 u 的路径上最小边的最大值，若 u 可以经过，有
$f_u = max\{min(f_v, w)\}$，否则 $f_u = 0$。

通过 Dijkstra 算法求最短路 DAG 可以做到 O(m log n)。 注意 SPFA 的复杂度是 O(nm)，一定要用的话请自行承担
风险。

对于 $n, m \leq 1000$ 的情况，是给用复杂度较差的方法求最短路 DAG 的同学留的。

对于 $k = 0$，可以少一个特判。

Sky 不会计数

DAG

容易发现我们只需按照拓扑序跑一个背包即可，因为每个节点只能用来加或减或不操作，所以需要跑分组背包，复杂度是 $O((n+m)T)$，可能有点卡常，于是可以考虑用 bitset 来优化背包的过程，这时复杂度是 $O(\frac{(n+m)T}{64})$，可以轻松通过这个部分分。

基环树

基环树上的问题一般都是把环拿出来特殊处理，所以我们考虑怎么处理这棵树里的环。
我们可以反复的在环上走，而操作 3 则让我们不必顾虑这么做的后果，因此环上任何一个点都可以给我们带来任意次的贡献。

尽管你可以把这个环缩成一个点后跑完全背包，但是这么做很容易被卡掉，所以我们需要研究一下怎么优雅的处理掉这个环。

容易发现如果环上有两个点的点权 a, b，根据裴蜀定理，我们可以用 a, b 来将计数器进行幅度为 $\gcd(a, b)$ 的操作，于是这两个点可以缩成一个权值为 $\gcd(a, b)$ 的点。不断进行这个过程可以将环缩成一个权值为 $\gcd\{w_i\}$ 其中 i 是在还上的点的点。

然后对这个点特殊标记一下，按照 DAG 的做法跑，遇到这个点的时候跑完全背包就好了。

至于说会不会因为计数器只能在 $[0, T]$ 之间而不能将 a, b 的操作简化为任意次 $\gcd(a, b)$ 的操作，这是不可能的，因为 $1 \leq w_i \leq \frac{T}{2}$，这就保证了无论当前计数器的值是什么，总有办法进行幅度为 $\gcd(a, b)$的操作。

复杂度是 $O(\frac{(n+m)T \log \max\{w_i\}}{64})$ 。

一般图

tarjan 后按照基环树的办法处理环就行了。

复杂度和基环树部分相同。