

## A

将  $s$  可能的取值范围看作若干个区间  $[l_i, r_i]$ , 则每个时刻区间会扩张至  $[l_i - 1, r_i + 1]$ , 答案就是区间的并集包含的数字数目。

题目的限制条件就是在  $t$  时刻的区间中除去  $[L, R]$  相交的部分。

用 set 维护每一个区间, 在区间重合的时刻进行合并。记录区间两两之间重合的时间, 将区间重合、限制以及询问的时间离线排序后按时间顺序维护即可。

## B

从大到小枚举 gcd, 假设枚举到了  $d$  这个数, 提取出编号为  $d, 2d, \dots$  的节点, 对于它们两两之间的路径上的边, 它们的答案可以是  $d$ 。

一个想法是每次暴力枚举两个节点, 通过跳到 lca 找到路径上的所有边, 然后每条边维护答案的最大值。

考虑一条边如果答案已经可以是  $d$ , 那么在枚举  $d - 1, d - 2, \dots$  时由于答案变小, 可以不用提取出这条边了。

于是可以将这条边缩掉, 假设这条边连接了  $(v, fa_v)$ , 那么下次暴力跳 lca 的时候可以直接从  $v$  跳到  $fa_{fa_v}$ 。

用并查集维护每个点下次跳的时候应该跳到哪个点, 每次暴力跳一条边就缩一条边, 时间复杂度均摊  $O(n)$ 。

## C

比特跳跃跳两次不如直接一步到位, 因为对于任意  $a, b$  都有  $x|a + b|z \geq x|z$ 。

同时, 如果需要通过比特跳跃跳到  $k$ , 则最好从  $k$  的二进制表示的子集跳, 如果不能的话由于  $k|1 \leq k + 1$ , 直接从 1 跳过去肯定不劣。

所以 1 到  $i$  的最短路一共四种情况:

- 不用比特跳跃
- 1 比特跳跃到某个点再通过原来的边走过去
- 1 走到  $k$  的二进制的一个子集再比特跳跃过去
- 1 直接比特跳跃过去

所以先从 1 向每个点连接边权为比特跳跃代价的边跑一遍 dijkstra 求出最短路, 这样可以覆盖情况 1, 2, 4。

对于情况 3, 由于 1 走到  $k$  的二进制的一个子集这一部分的最短路已经被计算过, 可以枚举  $k$  的子集中的最小值更新从 1 到  $k$  的最短路, 可以通过 DP 求解。

## D

将二叉树当成一棵由 01 串构成的 Trie, 则问题变为寻找  $S$  中子树形成的 Trie 是否包含  $T$  中的全部字符串  $t_1, t_2, \dots, t_k$ 。

对  $S$  进行 dfs, 搜索到节点  $u$  时前面的节点组成了一个字符序列, 检查  $t_1, t_2, \dots, t_k$  是否分别是该字符序列的一个后缀, 如果  $t_i$  是后缀, 则将  $u$  的  $|t_i|$  级祖先的 cnt+1。回溯的过程如果发现某个节点  $v$  的 cnt=k, 则该点符合题意。

匹配过程可以使用哈希或AC自动机实现。

## E

---

如果  $t_i \leq \frac{m}{3}$ , 则直接构造  $(3t_i, 2t_i)$ , 这样的数对在算法执行过程中只会向  $t$  数组中加入  $t_i$  一个数。

反之, 因为  $b_i > t_i, 2b_i + t_i > m$ , 故  $a_i = b_i + t_i$ , 下一次递归时有  $b_i = t_i + x_1$ , 因为若  $b_i = 2t_i + x$  则  $a_i = 3t_i + x > m$ , 不符合题意。同时  $x_1$  也应该小于  $\frac{m}{3}$ 。

考虑一直递归, 直到最后以  $b = x_p$  结束, 则  $x_p | t_i$ , 且  $x_1, x_2, \dots, x_p$  都是小于  $\frac{m}{3}$  的数。

多选不如直接选择  $x_1 = x_p$ , 剩下的数可以用去和另一个  $> \frac{m}{3}$  的数一起构造或子集单独构造。需要满足  $x_1 | t_i$ 。

所以每一个  $> \frac{m}{3}$  的数需要和另一个  $\leq \frac{m}{3}$  的数进行匹配, 不能匹配完则无解, 用二分图匹配求解即可。

## F

---

如果一条边  $(a, b, w)$  被一个三元组  $(u, v, l)$  定义为有用的 (此处假设起点为  $u$ , 起点为  $v$  同理), 则必须满足

$$d(u, a) + w + d(b, v) \leq l$$

暴力枚举边和三元组是  $O(n^4)$  的, 考虑优化, 对上式移项可得:

$$w + d(b, v) \leq l - d(u, a)$$

枚举边  $(a, b, w)$  和  $v$ , 对于每个  $(v, a)$  只需要通过枚举  $(u, v, l)$  预处理出最大的  $l - d(u, a)$  即可。

时间复杂度  $O(n^3)$ 。