智算之道复赛题解

A 数字

$b = 1$ 直接输出 900 。

$b = 2$ 时，判断 aa 最后一位是否为 2 的倍数，是输出 900 ，否则输出 0 （奇数还是偶数只由最后一位决定）。

对于 100% 的数据，枚举前三位，判断枚举的数接上 aa 是否是 b 的倍数，是则给答案计数器 +1 。

时间复杂度 $O(1)$ 。

B 网格

如果在魔法格点 (a, b) 花费 $w2$ 走到 $(a+1, b+1)$ 不如花费两次 $w1$ 到 $(a+1, b+1)$ 划算，即 $2 \times w1 \leq w2$ ，直接输出 $2 \times n \times w1$ 。

对于 $2 \times w1 > w2$ 的情况，经过的魔法格点越多，代价就越小。

$n \leq 3$ 时，暴力搜索路径。

魔法格点都在对角线上的情况，显然能够同时经过所有魔法格点。

对于 100% 的数据，令 f_i 表示以第 i 个点作为路径中最后一个魔法格点的情况下，最多能经过多少魔法格点。

一条路径上经过的魔法格点行列坐标必须都是递增的，故转移方程为

$$f_i = \max\{f_j\} + 1, x_j < x_i, y_j < y_i$$

需要先将某一维排序以保证更新顺序，或使用记忆化搜索。

答案就是 $2 \times (n - \max\{f_i\}) \times w1 + \max\{f_i\} \times w2$ 。

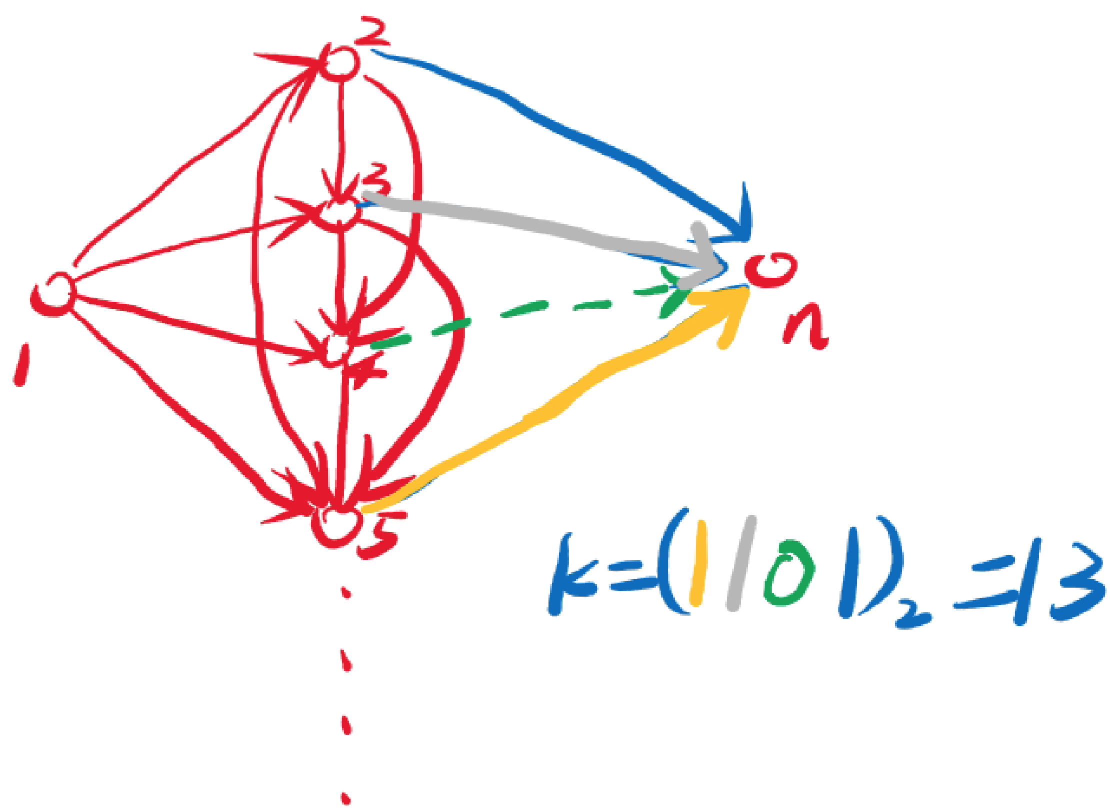
时间复杂度 $O(k^2)$ 。

C 有向无环图

k 给定的点，手动构造出符合条件的图。

第三个测试点的点数要求很宽松，直接 1 向 $2\cdots k+1$ 连边， $2\cdots k+1$ 向 n 连边。

对于 100% 的数据，给出一种通用的构造方法，可以结合下图理解



$1, n$ 位于两侧，中间有 `64-__builtin_clzll(k)` 个点（也就是二进制下 k 最高位 1 的位置 $+1$ ，如 $k=8$ 就有 4 个点， $k=7$ 有三个点）， 1 向中间所有点连边，中间的每个点向 n 以外所有编号比它大的点连边。

接下来对 k 进行二进制分解，若第 i 位(0-indexed)为 1 则连一条 $i+2$ 到 n 的边。

构造的原理大概是 拓扑排序的过程中，对于中间的点，每有一个点出队，其他点的 dp 值会乘上 2 ; 而是否加上一个点到 n 的边决定了这个点出队时的 dp 值是否贡献会到答案中 (编号为 i 的点会贡献 2^{i-2}) 。

有一个地方需要注意：这个方法中边 $1-n$ 是独立的，所以要特判第一个测试点，直接按照上文构造点数会 >2 。

时间复杂度 $O(\log^2 k)$ 。

D 分数

$n \le 7$ 手玩出每次乘上的数。

$n \le 5000$ 模拟题目过程，记第 i 个数的分母是 f_i ，每次乘 q 时将所有 f_i 赋值为 $\gcd(f_i, q)$ 。

$n \le 50000$ 用 $n \le 5000$ 的程序将所有 q 打表。

100% 的数据需要打表观察/了解一些性质：

- 仅当 $\prod q = \operatorname{lcm}(1, 2, \cdots, i)$ 时前 i 个数的分母会变成 1 。
- 求 lcm 的本质是对所有质因子的次数取 $\max$ 。

设 $\operatorname{lcm}(1, 2, \cdots, i-1) = \prod p_j^{k_j} \quad (p_j \in P)$ ，仅当 $i = p^k$ 时会出现 $f_1 \cdots f_{i-1}$ 分母均为 1 而 f_i 分母不为 1 的情况（i 唯一质因子 p 的次数高于 $\operatorname{lcm}(1, 2, \cdots, i-1)$ 中 p 的次数），此时 i 会对答案贡献一个 p 。

用线性筛或者埃氏筛求出 $1 \cdots n$ 中所有 p^k 并标记 p 是多少即可。

时间复杂度 $O(n)$ 或 $O(n \log\log n)$ 。

~~实测埃氏筛比线性筛快~~

E 树数数

$n \le 100, m \le 100$

$O(1)$ 修改点权/颜色；询问时枚举所有点对，对于每个点对暴力爬树找最近黑色公共祖先，对权值求和。

$n \le 2000, m \le 2000$

记 $f(i) = \frac{(i+1)\times i}{2}; LBA(u)$ 为 u 的最近黑色祖先，不存在则为 0 ; $siz(u)$ 为 u 的子树大小 ; $g(u)$ 为 u 的子树集合($u \notin g(u)$) 。

考虑维护每个 w_i 会贡献的次数 cnt_i ，询问点 u 的答案就是 $\sum cnt_i \times w_i, i \in g(u)$ (u 是白色则要加上 $w_{LBA(u)} \times cnt_u$)。

$cnt_u = f(siz(u)) - \sum cnt_i, i \in g(u)$ 。

$O(1)$ 修改点权，询问时遍历 u 的子树对 $cnt_i \times w_i$ 求和，修改颜色时重新对树进行 dfs 求出新的 cnt_i 。

n 更大的情况，考虑将 cnt_i 、 $cnt_i \times w_i$ 放到数据结构上维护：

- 将对子树的询问转化到 dfs 序中对区间的询问。
- 黑点变白点，将 cnt_u 上传到 $cnt_{LBA(u)}$ ，白点变黑点再将这个值减掉，具体过程：
  1. 白变黑，数据结构中维护的 cnt_u 加上 $f(siz(u)) - \sum cnt_i, i \in g(u)$ ，若 $LBA(u) \neq 0$ 则数据结构中维护的 $cnt_{LBA(u)}$ 减去 cnt_u 刚刚增加的值。

  2. 黑变白，与白变黑相反，先给 $cnt_{LBA(u)}$ 加上 $f(siz(u)) - \sum cnt_i, i \in g(u)$ 再给 cnt_u 减去这个值。

- 修改权值，对维护的 $cnt_i \times w_i$ 进行单点修改。

树高较小时，可以暴力爬树找 LBA 。

对于 100% 的数据，在快速维护 cnt 、 $cnt \times w_i$ 的基础上将求 LBA 的过程用轻重链剖分+线段树优化到 $O(\log^2 n)$ 即可。

时间复杂度 $O(m \log^2 n)$ 。