2025 全国信息学奥林匹克联赛 NOIP
第六场模拟赛**题解**

2025/7/12

（请选手务必仔细阅读本页内容）

题目名称	网格选点	数学作业	维护数组	洗牌
题目类型	传统型	传统型	传统型	传统型
英文题目名称	kgrid	homework	database	shuffle
输入文件名	kgrid.in	homework.in	database.in	shuffle.in
输出文件名	kgrid.out	homework.out	database.out	shuffle.out
输出文件名	2s	1s	1s	2s
内存上限	512M	512M	512M	512M
测试点数目	/	/	10	/
每个测试点分值	/	/	10	/
附加样例文件	有	有	有	有
结果比较方式	全文比较	全文比较	全文比较	全文比较

1 网格选点

(kgrid.cpp)

【题解】

算法一: 暴力枚举 K 个点，判断是否在一条直线上

算法二：枚举两个点 (本质枚举一条直线)，计算该直线上的点的个数，然后组合计数一下。但是这样会重复计算。那么我们考虑枚举两个点（本质枚举一条线段），计算该线段上的点的个数为 P, 为了避免重复，强制选择线段的两个端点，该线段对答案的贡献为 $\binom{P-2}{K-2}$

时间复杂度 $O((NM)^2 \times gcd$的复杂度), 其中 gcd的复杂度 用于计算线段上的点个数。

算法三：对上述枚举进行优化，大量线段是可以平移的，强制其中一个端点为 (0, 0), 因此我们只需要枚举另外一个端点，然后考虑所有平移线段和一些翻转的情况即可。

时间复杂度 $O(NM \times gcd$的复杂度)

2 数学作业

(homework.cpp)

【题解】

我们可以用一棵二叉树表示一个有效的表达式，该树有 n 片叶子和 $n-1$ 个内部节点。叶子对应于问号，每个内部节点对应于函数 max 或 min。

问题现在可以重新表述为在叶子上写出 1 到 n 的一个排列，然后将这些值向根节点传播。每个子任务的解决方案都涉及从输入中解析字符串，并将其转换为这样的一棵二叉树。

第一个子任务可以通过尝试 $\{1, 2, \ldots, n\}$ 的所有排列，替换到表达式中并评估来解决，时间复杂度为 $O(n! \cdot n)$。

第二个子任务可以使用状态压缩的动态规划解决。状态为 $dp[node][mask]$，表示在该节点中，如果允许的值来自于该 mask，所能获得的所有可能值的集合。对于转移，我们尝试将 mask 划分为两个子 mask（每个子 mask 对应一个子节点），以所有可能的方式进行划分。时间复杂度为 $O(3^n \cdot n^2)$，但实际上运行速度要快得多，因为可以丢弃所有 mask 中 1 的数量与子树中的节点数量不匹配的状态。

还有一种随机化解法可以解决第二个子任务。我们猜测可获得的数集形成一个区间（后来证明这个猜测是正确的）。然后我们尝试评估尽可能多的不同排列，并存储我们遇到的最小值和最大值。最后，我们声明输出为该区间的长度。给读者的挑战：证明这种方法成功的概率足够高。

第三个子任务是通过从根节点开始，查找同一类型节点（全为 min 或全为 max）的最长序列来解决。假设有 k 个这样的节点，解答即为 $n-k$。证明留作练习。

现在让我们看看一个节点及其子树。设 S 为一个大小等于子树中叶子数量的不同数字集合。完整的解决方案需要观察到在这个节点中使用 S 中的数作为叶子的值，可以获得的数集形成一个 S 中的区间。此外，我们必须想出一种方法，将左右子节点的区间结合起来，以获得该

节点的区间。证明是递归的，同时也描述了获取所述区间的方法，从而允许我们通过树状动态规划解决问题。

假设某个节点有一个左子节点，包含 L 片叶子；右子节点包含 R 片叶子。同时，左子节点可能值的区间为 $[a,b] \subseteq [1,L]$，右子节点的区间为 $[c,d] \subseteq [1,R]$。

如果该节点是 max 类型，节点区间的下限是 $a+c$。实际上，我们把 $[1,L+R]$ 中最小的 $a+c-1$ 个数称为 “小数”，其余的称为 “大数”。假设左子树中有 x 个小数，右子树中有 y 个小数。由于 $x+y=a+c-1$，至少有一个满足 $x<a$ 或 $y<b$。假设 $x<a$，则左子树中第 a 小的数是大数，所以左右子树的最大值也将是大数。这表明节点值至少为 $a+c$。这个值也是可获得的：将 $[1+c,L+c]$ 放入左子树，其余的放入右子树。我们可以使得左子树的值为 $a+c$，右子树的值为 c。

如果节点是 min 类型，则下限为 $min(a,c)$。证明类似于上述情况。此时，小数是小于 $min(a,c)$ 的数，其余的为大数。相同的推理表明节点值至少为 $min(a,c)$。如果 $a<c$，可以通过将 $[1,L]$ 放入左子树，将 $[L+1,L+R]$ 放入右子树获得该值。$a \geq c$ 的情况与此类似。

对于上限，我们可以证明类似的结果。如果节点是 max 类型，上限为 $max(b+R,d+L)$；如果是 min 类型，上限为 $b+d-1$。为了证明中间的值是可获得的，我们只需要稍微改变左右子树中数字的分配方式，类似于达到边界值的构造。总时间复杂度为 $O(n)$。

3 维护数组

(database.cpp)

【题解】

10 分，直接暴力

20 分，使用 Trie 求 xor 最大。

30 分，基于以上，多了一个异或操作，我们可以维护一个全局的异或和，在查询的时候同时异或一下这个即可。

40 分，又多了一个插入操作，我们希望后面查询的时候查询这个值，但是直接插入会被全局变量给打乱。所以我们希望插入的值异或上全局变量是真实值，那么我们直接插入真实值异或上全局变量的值即可。

以上 40 分基本考察 01 $Trie$ 的基本功。

对于 70 分，与和或操作类似，只介绍与操作。与操作相当于，把 trie 树中某一位全变成 0，相当于合并两颗子树。

我们不妨建立一个时间轴，记录下每个时间的操作（全变 0/1，查询）。然后我们给树上每个节点记录一个修改时间，当查询道某个结点的时候我们判断一下这个节点的修改时间是否在这一位全局的上次修改之前，如果是的话我们就暴力合并一下两棵子树，之后更新一下时间。这样做复杂度是对的，接下来我们证明这个复杂度。我们每合并一次节点，会使字典树的节点个数少一。而只有插入操作会产生节点，也就是最多产生 $31(n+m)$ 个节点，因此我们的合并操作次数少于 $31(n+m)$ 。

对于 100 分, 对比 70 分的做法, 多了个异或操作，我们可以用当前时间，某一位的修改时间，某个节点的修改时间这三个数来维护我们的更新。

如果遍历到一个点时, 如果节点修改时间早于这一位的修改时间, 我们暴力的合并一下, 然后更新节点修改时间为这一位的修改时间。之后，如果节点修改时间早于当前时间，我们可以用前缀和的方法算出这两个时间点之间异或了多少个 1，奇数个的话，我们交换一下左右儿子即可，之后再次更新节点修改时间为当前时间。复杂度仍为 $O((n+m)log(a))$ 。

4 洗牌

(shuffle.cpp)

【题解】

我们称抽到一张 鬼牌 并重排牌序之前的抽卡流程为一轮。

$$\begin{aligned} E(\text{游戏结束时期望抽牌数}) &= \sum_{i=1}^{\infty} P(\text{游戏在第}i-1\text{ 轮没有结束}) \times E(\text{第}i\text{ 轮抽的牌数}) \\ &= \sum_{i=1}^{\infty} P(\text{游戏在第}i-1\text{ 轮没有结束}) \times E(\text{一轮抽的牌数}) \\ &= E(\text{轮数}) \times E(\text{一轮抽的牌数}) \end{aligned}$$

注意这里轮数和每轮抽的牌数并不独立，但由于 E(第i轮抽牌数) 都是一样的，所以我们可以将它们拆开。

计算 E(一轮抽的牌数)

我们可以把抽到 鬼牌 后结束一轮当成每轮都抽满了 $n+m$ 张牌，只不过第一张 鬼牌 之后的牌不算数。

我们考虑每张数字牌，它之前没有 鬼牌 的概率显然是 $\frac{1}{m+1}$，所以 E(一轮抽的牌数) 就是 $\frac{n}{m+1}+1$。(加上一张鬼牌)

对于特定的一个数字牌和所有的 m 张鬼牌，这张数字牌在这 $m+1$ 张牌中最先被抽到的概率是 $\frac{1}{m+1}$。考虑期望的线性性 n 张数字牌，能出现在第一张鬼牌前面的牌的张数就是 $\frac{n}{m+1}$

计算 E(轮数)

我们称一张数字牌为新的当且仅当之前没有抽到过这张牌。

令 t_i 表示抽到牌 i 的期望轮数，求的就是

$$\mathbb{E}\left(\max_i t_i\right)$$

min-max 容斥之后我们需要求的就是

$$\mathbb{E}\left(\min_{i\in S} t_i\right)$$

，这个式子显然只和 $|S|$ 有关的，所以只要计算所有可能的 $|S|$ 对应的值。对于一个 $|S|$，我们只关心 $|S|$ 张集合的牌和 m 张鬼牌，那么抽到 S 的牌的概率就是 $\frac{|S|}{|S|+m}$，期望就是其倒数。

所以结合 min - max 容斥答案就是

$$\sum_{i=1}^{n}\binom{n}{i}(-1)^{i+1}\frac{m+i}{i}$$

已知

$$\sum_{i=1}^{n}\binom{n}{i}(-1)^{i+1}\frac{1}{i}=\sum_{i=1}^{n}\frac{1}{i}$$

所以这部分的为 mH_n+1

其中 H_n 表示第 n 个调和数：$H_n=\sum_{k=1}^{n}\frac{1}{k}$。

其他思想思路:

$$\begin{aligned}E(\text{轮数})=&\sum_{i=1}^{n}E(\text{还剩}i\text{张牌没抽到过时抽到一张新牌所需轮数})\\&+E(\text{所有数字牌都抽到过后游戏结束所需期望轮数})\end{aligned}$$

在抽新牌时，已经抽到过的牌可以忽略不计（抽到了既不会是新牌，也不会让轮数 +1）。

所以

$E(\text{还剩}i\text{张牌没抽到过时抽到一张新牌所需轮数})$

$=E(\text{还剩}i\text{张牌没抽到过时抽到一张新牌所需抽 鬼牌 或新牌张数})-1$

-1 是因为最后一张新牌不会加轮数。每次抽到新牌的概率是$\frac{i}{m+i}$

$E(\text{还剩}i\text{张牌没抽到过时抽到一张新牌所需抽 鬼牌 或新牌张数})$ 就是$\frac{m+i}{i}$

所以$E(\text{还剩}i\text{张牌没抽到过时抽到一张新牌所需轮数})=\frac{m}{i}$

$E(\text{轮数})=mH_n+1$

最终结果

$$\begin{aligned}E(\text{游戏结束时期望抽牌数})&=E(\text{轮数})\times E(\text{一轮抽的牌数})\\&=(mH_n+1)\left(\frac{n}{m+1}+1\right)\end{aligned}$$

其中 H_n 表示第 n 个调和数：$H_n=\sum_{k=1}^{n}\frac{1}{k}$。