

20260912模拟赛题解

A 购买

闲话

原题：CF1801B，绿

为什么题目英文名叫 `manyhacks`？因为出题人、验题人和出题人认识的几个高手在这题上都 FST 了。

一些比较典型易错的 hack 都放在 Sub 1 了。

你可能需要注意的点包括，是否使用了 `multiset`，是否开了 `long long`，是否处理正确了初值（即两人都至少拥有一项礼物）

第一问

只用求最大值的话。

如果可以在同一个商店买两样礼物，那么答案就是 $\max(\max a - \min b, \max b - \min a)$ 。

取不到这个值当且仅当 $\max a$ 和 $\min b$ 来源于同一个商店，或 $\max b$ 和 $\min a$ 来源于同一个商店。这个时候退而求其次到次大值即可。

第二问

要求最小值。考虑按照 a 对所有 (a_i, b_i) 排序。我们假设给蜗牛（下文称为 A）买的东西最贵是 a_i 。

显然第 $1, 2, 3, \dots, (i-1)$ 样商品，可以给 A 买，也可以给 B 买。

第 $(i+1), (i+2), \dots, n$ 样商品，只能给 B 买。令 $su f_i = \max_j^n b_j$ ，那么给 B 买的东西价格最大值一定至少是 $su f_{i+1}$ 。

如果 $su f_{i+1}$ 已经比 a_i 大了，那么在给 A 买的东西最贵是 a_i 的情况下，答案最小就是 $su f_{i+1} - a_i$ 。

反之答案就是 $b_1, b_2, \dots, b_{i-1}$ 中距离 a_i 最接近的那件商品与 a_i 的距离，应使用 `multiset` 维护。

时间复杂度 $O(n \log n)$ 。实现优雅的话常数应该不怎么大。

B 整理

先观察，后推式子。

Subtask 1

我是爆搜大神！ 直接枚举排列，然后把伪代码翻译成 C++，数一下几个排列执行了 a, b 的比较。

可以获得 7 分。

Subtask 2

我是状压 dp 大神！出题人不是状压 dp 大神。可能存在若干 $O(2^n \text{poly}(n))$ 做法。

可以获得 26 分。

Subtask 3

我是观察大神！

Observation 1. 任何两个数 a, b 至多只会发生一次比较。

发生完比较后，两个数会被放在同一个数组中。接下来不可能发生比较。

不难发现 a 和 $a + 1$ 一定会比较，输出 $n!$ 即可。

结合上述算法可以获得 30 分。

Subtask 4

Subtask 5

我是组合计数大神！

Observation 2. 两个数组 A 和 B 合并时，想让一个数 $a \in A$ 和另一个数 $b \in B$ 发生比较，充要条件是 B 中不存在任意一个满足 $a < t < b$ 的数 t 。

可以自己手玩一下。如果存在这样的数，那么 a 会和 t 发生比较后归并掉了，就碰不到 b 了。

你可以把整个线段树状物的归并排序合并树画出来，然后枚举 a, b 在哪次合并相遇的。

如果相遇时， $|A| = k_1, |B| = k_2$ ，根据组合意义可知，合并 A, B 时比较 a, b ，可能的 (A, B) 方案数为：

$$k_1!k_2! \binom{n - (b - a + 1)}{k_1 - 1} \binom{n - k_1 - 1}{k_2 - 1}$$

$|A|, |B|$ 可以互换，这个也要算。然后对于每个 n, a, b 你都重复这个过程时间复杂度是 $O(\sum n \log n)$ 的，只能获得 75 分。

Subtask 6

我是线段树的性质大神！

线段树同层长度不同区间最多只有 2 个，所以整棵线段树不同区间个数是 $O(n)$ 的。

发现对于相同的 $(|A|, |B|)$ ，答案的计算是一样的。这样的二元组只有 $O(\log n)$ 个，记忆化一下即可。时间复杂度 $O(T \log n)$ 。

可以获得 100 分！

C 包装

Subtask 1

我会爆搜！

爆搜即可。时间复杂度可以是 $O(n^n)$ 或 $O(n!)$ 。都能过。

Subtask 2

我会 map！

发现答案就是，本质相同的箱子的最大数量。直接用 map 状物维护即可。

可能对正解有启发作用，因为从数据结构的角度来说，这就是把区间加改成了单点加。

设计中，希望大家能通过该档部分“猜”出正解。不知道是否能起到这个目的。

Subtask 3

我会二分 / dp / 贪心！

对每个前缀分别做。

考虑分成 k 条链的充要条件是，能形成 $n - k$ 条包含关系。能形成包含关系 (i, j) 的条件是 $b_j \leq a_i$ 。

然后你发现你可以二分一下最大的 p 使得能找出 p 个 (i, j) 使得 $b_j \leq a_i$ 。

这个东西你只要选最小的 p 个 b 和最大的 p 个 a 即可。

时间复杂度 $O(n^2 \log n)$ 。我认为结合 Sub 2 的一些思考不难得出正解了。

Subtask 4

我会正解！了吗？

说实话不太懂怎么能卡在这里。

姑且认为给到不会写线段树的做法。时间复杂度 $O(n \max s_i)$ 。

可能有一些别的做法能过掉这个 sub。

正解

我会 Dilworth 定理！

据说 CN 选手普遍学 dilworth 定理太早导致容易忘，我来给大家复习一下！

最小链覆盖转成最长反链。

问题相当于选出一些盒子，两两无法嵌套。

我们只需要关注最大的 b_i 。设 $\max b_i = v$ ，那么只有 $a_i > v$ 的 i 才可以被选上。

因此答案为 $\max_v \sum_i [a_i > v \wedge b_i \leq v]$ ，中括号是艾弗森括号。

可以直接用线段树维护，时间复杂度 $O(n \log n)$ 。

验题人做法

我会自己发明 Dilworth 定理!

@[Joey](#) /bx

没能想到 Dilworth 定理，感性思考了一个做法，讲的不好的地方还请多多包含。

把所有箱子先按照存放容量 c_i 排序，那么箱子 i 可以放入的箱子 j 就是一个后缀，令这个后缀为 $[d_i, n]$ 。

从后向前考虑，问题转化为：

- 有若干个物品，第 i 个物品在 d_i 之前没买就消失。
- 每天只能购买一个物品。
- 问最少要抛弃多少个物品。

这是一个经典的贪心问题，解决方式是优先购买临期商品。

不妨换个思路，先存着购买额度，等到了某个商品的 ddl 再时间穿梭回去买它，这和上面的贪心方式是等价的。

于是考虑维护 c 上每个时刻的购买额度 t_i ，共有两种事件：

- i 时刻增加 1 的额度：由于要存额度，并且额度不会过期。 $\forall i \in [1, i], t_i \leftarrow t_i + 1$ 。
- i 时刻某个物品到达 ddl：
 - 该物品买到了：额度用掉一个，并且永久有效。 $\forall i \in [1, i], t_i \leftarrow t_i - 1$ 。
 - 该物品未买到：此时额度 ≤ 0 ，仍强制消耗一个额度，并定义整个过程抛弃的物品数为 $\max(-\min_{i=1}^n t_i, 0)$ 。因为下次发生抛弃事件必然先消耗了后面增加的所有额度，故可以保证正确性。 $\forall i \in [1, i], t_i \leftarrow t_i - 1$ 。

可以合并为 $\forall i \in [1, i], t_i \leftarrow t_i - 1$ 。

于是需要维护的操作就是前缀 ± 1 ，查全局最小值。存放 $[1, i]$ 的箱子所需的成本则可以通过按下标从小到大加入实现，时间复杂度 $O(n \log n)$ 。

可能和 dilworth 做法是等价的？

D 玩

Subtask 1,2

对于每次询问暴力求出 $\text{path}(u, v)$ 。注意到任意时刻的集合 S 都是由 $\text{path}(u, v)$ 中若干个极长子段得到的，因此状态数不超过 $O(2^l)$ ，暴力 dp 即可。

Subtask 3

考虑计算 SG 函数，这样每个子问题都是求一段区间的 SG 值。由于树的形态是一条链，可以对于每个区间预处理 SG 值，枚举该轮操作的字符并暴力拆分成子问题即可做到 $O(n^3)$ 。

Subtask 5

考虑加速算法 2 中的转移过程。记 $f_{i,j}$ 表示区间 $[i, j]$ 的子问题，辅助状态 $g_{i,j}$ 表示将区间 $[i, j]$ 按照字符 s_{i-1} 拆分成子问题后的 SG 值异或和。只需再预处理每个位置的下一个对应字符的位置即可 $O(|\Sigma|)$ 转移 f 以及 $O(1)$ 转移 g 。

Subtask 7

定义 $fl_{i,c}$ ：表示从位置 i 之前第一个字符 c 之后到 i 的 **SG 值**， $fr_{i,c}$ ：表示从位置 i 到 i 之后第一个字符 c 之前的 **SG 值**。

一个状态 $[i, j]$ 被称为**关键状态**，当且仅当满足以下两个条件：

1. $s_{i-1} = s_{j+1}$ （即左右边界外的字符相同）；
2. 区间 $[i, j]$ 中**不存在**字符 s_{i-1} 。

关键状态的数量为 $\mathcal{O}(n)$ 级别，其中 n 为字符串长度。

预处理步骤：

1. **暴力预处理所有关键状态的 SG 值。**
2. 对于每个关键状态 $[i, j]$ ，枚举字符 c ，并通过 $fr_{i,c}$ 和 $fl_{j,c}$ 进行状态转移。
3. **记忆化**： $fl_{i,c}$ 和 $fr_{i,c}$ 不用对每个 i 和 c 都计算，只需在处理关键状态时进行记忆化存储。

转移复杂度分析：

- 关键状态 $[i, j]$ 仅在其左端点前是 i 之前**第一次出现字符** c ，且右端点后是 j 之后**第一次出现字符** c 时，才会被转移。
- 预处理的时间复杂度为 $\mathcal{O}(n|\Sigma|^2)$ ，其中 $|\Sigma|$ 是字符集大小。

查询时操作包括：

1. 对每个字符的关键状态做**前缀和**计算。
2. 前后缀的非关键状态部分：
 - 要么是已预处理的 $fl_{i,c}$ 或 $fr_{i,c}$
 - 要么仍是查询串的前缀（且右端点后是 l 之后第一次出现的字符），或类似的后缀结构。

总时间复杂度为： $\mathcal{O}(n|\Sigma|^2 + q|\Sigma|)$ ，其中 q 是查询次数。

Subtask 9

考虑对于每个点 u ，不断往上跳到第一个颜色为 c 的点 $\text{up}[u][c]$ 。求出 $u \sim \text{up}_{u,c}$ 的树链 SG 值 $\text{val}_{u,c}$ ，这个也可以用倍增求出，倍增 2^k 个颜色为 c 的点去做。

然后考虑回答询问，先枚举分割的颜色 c ，然后利用刚刚的预处理的信息往上跳，直到 LCA 之前的最后一个颜色为 c 的点。需要特殊处理 LCA 的颜色就是 c 的情况。

然后你发现这就只需要递归地求包含 LCA 的这一段路径就行了。

发现这个过程，只会访问形如这样的路径：钦定一个颜色 c ，从 LCA 开始往 u 或 v 扫直到扫到一个颜色第一次出现（当然，也有可能直接扫到 x, y ），中间部分形成的路径。自然地，这样的路径只 $\mathcal{O}(|\Sigma|^2)$ 种。

直接记忆化即可。单次询问的时间复杂度就变成了 $\mathcal{O}(|\Sigma|^3 \log n)$ 。4s 比较多余。