The 4th Simons Cup Solution

Author: Shunpower, 2025/3/14

Supported by nzhtl1477, Qingbaiqwq

A. 盛放 (zime)

Source：BZOJ4722 由乃

> 显然，这是 lxl 出的题。~~原题题面包含敏感内容。~~

放在 T1，没有加强！既不卡常，也不难写！

首先考虑要做的那个判定到底是什么东西。考虑先把集合大小匀到每个数上，那么相当于我们的值域在 $[1,v]$ 之间。然后你可以发现不交就是来搞笑的，我们把交抠掉就可以了。

于是我们可以任意选择区间的子集。只需判断是否存在两个子集的和相等即可。这会产生 2^{len} 种选法，但是和的值域只有 $[1,len\times v]$！抽屉原理告诉我们此时答案一定是 `YES`。

显然剩下的情况 len 一定特别小了，事实上至多只有 13。所以我们可以先不进行 k 次方的修改，扫到的时候再进行修改。具体来说，修改的时候我们只需记录从最初到现在进行了多少次 k 次方，这显然可以用一个树状数组简单完成。

考虑因为每个数据 k 都是固定的，我们显然可以进行倍增完成。

暴力出每个数是多少之后本质是一个 13 个元素的 01 背包，当背包中有一项被重复赋值为 1 的时候就说明找到解了。这显然可以使用 `bitset`！

不妨令 $c=13$，于是复杂度 $\mathcal{O}(q(c\log q+\frac{c^2v}{w}))$。我不知道有没有卡掉每次 2^c 枚举的人，这玩意太邪恶了。

闲话：我没想清楚 k 不固定的时候怎么做，如果 v 不是质数的话直接用扩展欧拉定理好像 case 会很多。这是 T1 也没必要这么加强。

B. 千言万雨 (violence)

Source：LEQ and NEQ, AtCoder Regular Contest 115 E

对于 n 比较小的情况可以直接线段树维护整体 dp，因为带 $\log$ 所以是不可能过的。

这种暴力做法是一定不能放过的！我们考虑正规组合做法。接下来会有一些比较炫酷的魔法。

容易想到容斥。考虑钦定有多少个位置 $x_i=x_{i+1}$，此时容易有一个 dp $f_{i,j}$ 表示前 i 个位置里有 j 个这样相等的位置，然而这个状态的转移显然需要维护前一项的取值，所以并不好做。直接做应该是 $\mathcal{O}(n^3)$ 到 $\mathcal{O}(n^4)$ 的。

我们发现，这样位置的数量和极长等值连续段的数量有关。至少 i 个这样的位置意味着至多 $n-i$ 个连续段，并且这两个东西的钦定显然是可以一一对应的。所以我们只要把原序列钦定划分成 $n-i$ 个连续段就行了。所以我们直接考虑 dp：$f_{i,j}$ 表示前 i 个位置里钦定了 j 个连续段的方案数。我们发现这个状态是好做的，我们只要每次合并一个段当成连续段就行，因为它和前面的关系无所谓。于是这个东西变成简单的序列划分 dp，转移直接考虑合并连续段产生的贡献显然是这一段的 $\min$，所以：

$$f_{i,j}\leftarrow\sum_{k=0}^{i-1}f_{k,j-1}\times\min_{l=k+1}^{i}a_l$$

从而答案就是 $ans_n = \sum_{i=0}^{n-1} (-1)^i f_{n,n-i}$

直接做可以做到 $\mathcal{O}(n^4)$。使用 ST 表优化做到 $\mathcal{O}(n^3)$。这些都是 trivial 的优化。

可以发现状态是 $\mathcal{O}(n^2)$ 的，我们试图优化状态。注意到我们最后只关心状态的奇偶性，并且恰巧转移都是 $j \leftarrow j-1$ 且系数与 j 无关，所以我们可以直接把 j 变成 $0/1$ 描述奇偶性。转移变成：

$$f_{i,j} \leftarrow \sum_{k=0}^{i-1} f_{k,1-j} \times \min_{l=k+1}^{i} a_l$$

直接做可以做到 $\mathcal{O}(n^2)$。

熟悉历史和的朋友们都知道，我们这个东西长得像扫描线一样，后面那个按 $\min$ 贡献显然是一个单调栈的形式。数据结构学傻了就可以上线段树，做到 $\mathcal{O}(n \log n)$，可能还跑不过别人整体 dp。但是直接前缀和就可以小常数 $\mathcal{O}(n)$。

这个做法技巧性比较强。题解区中还有其他的一些线性做法可以通过这道题：建立小根笛卡尔树再树上 dp、拆暴力 dp 式子等，这些做法有了起点之后都比较简单，这里不再赘述了。事实上我本来是想强制在线把这些做法卡死的，但是这是 ~~ARC 的 F~~ T2，不是搬题人炫技的地方。

C. 想念特效药 (rhythm)

Source: QOJ8553. Exchanging Kubic, CCPC2023 Final

> 场上为什么只过了 $\mathcal{O}(1)$ 个队呢？~~因为证明太困难了。~~

我们主要介绍一下杜老师给出的漂亮的做法，在本题部分分的引导下应该是可以感性地想出的。原题解是一个不太好玩的四边形不等式。

首先我们使用 n 次询问可以问出每个数的正负，并且简单地确定所有的正数。我们把 0 也算成负数。

我们先考虑性质 C，它意味着整个序列是正负相间的。

考虑我们查询相邻的 $+-+$ 三项，显然会有两种情况：

- 答案不是两个正数，这说明我们使用了负数，那么我们可以直接把负数求出来。因为这一段的选法要么是两边的正数段或者全选，于是我们可以把这个段合并起来成一个新的正数，然后迭代下一轮。
- 答案是两个正数之一，这说明我们没有使用负数。进一步地，这意味着这个负数的绝对值不小于两个正数的 $\min$。

既然有 $\min$ 相关我们不妨放到最小值处考虑来消除影响（B 性质）。考虑最小的正数两边的 $+-+-+$（中间那个正数就是最小的正数）。

考虑查询这两个，如果有一边合并了我们就合并迭代下去。如果两头都没有合并，这说明我们在最大子段和中不可能单独选中间这个东西，也不可能截断在负数处，所以这一段的选法只有可能是两边的正数段或者全选。于是我们可以把中间的 $-+-$ 合并起来成一个新的负数，然后迭代下一轮。

合并时需要将其中一个负数*设置为中间那个正数的相反数。这样做其实是贪心地为了确保另一个负数有数能用的最优策略。感性理解一下，如果我们使得赋值的负数和那个正数一起选小于 0，那么在构造出一个区间同时包含这两个数和后面的一串数时，可能会因为过小导致剩下的负数无法填入了。

在 C 性质的情况下，由于每个负数段合并起来之后也只会有一个未确定的负数，每个正数段始终是确定性的，正确性证明是较为容易的（考虑每一轮归纳一下，我们只需证明合并过程不影响答案就可以了）。

考虑没有性质 C 怎么办。容易想到我们应该可以初始就进行一次极长正数段和极长非正数段的合并，然后再进行 C 性质做法。此时正数段因为是确定性的所以没有影响；负数段里面虽然可能有多个未确定的值，但由于最大子段和不可能截断在负数内部，所以负数段可以看成是要么全选要么全不选，也没有影响。此外可以发现对其赋值这种问题根本不是问题，因为其中包括的所有负数段都是一个一个的数，那么我们可以直接给一个数赋值，剩下的全写成 0，同样是因为最大子段和不可能在这个负数段里面截断。我们应该可以较为严格地把没有 C 性质的问题的正确性证明规约于有 C 性质的问题。

由于至多查询两次可以减少掉一个负数段，两头是负数段的可以直接删掉，于是负数段的总数一定是严格少于 $\frac{n}{2}$ 的。于是总次数为 $2n$ 以内。

此处给出一个严谨（？）正确性证明的参考链接，网上也能找到很多乱七八糟的做题记录。有人如果更明白了可以教教我 qwq。

至于四边形不等式做法，根本没人写。唯一的资料是原题解。

总而言之，找最小值合并等操作最后时间复杂度是可以轻易做到 $\mathcal{O}(n^2)$ 的。因为交互库复杂度就这么高所以没必要再进一步优化了。

实现上看似很难写，实则也容易写得很难写。注意到我们确定负数时是在抵消贡献，可以看成是直接删除了两个连续段，这样就不用维护负数段里面的情况，好写很多。~~当然这并不影响 std 写得丑。~~

*：这里我有一个没搞清楚的问题：是不是每次都必然要选择同侧的（边界上只有一侧的除外，显然不影响）？证明中疑似需要使用这条性质。欢迎搞清楚的选手和我交流。

闲话：这题一开始 `grader.cpp` 在读取 `long long a[]` 的时候使用了 `%d` 标识符，导致所有 a 序列中的数都被读成 `unsigned` 了，于是几乎所有数据的询问次数都在 $n+\mathcal{O}(1)$。考试前一天才发现。

D. 不止两票 (fearless)

Source：UVA12267 Telephone Network

> 我故意交换了 C 和 D。

其实是很简单的刻画题。~~不是阅读题~~

题目结合图片应该还是比较好懂的。看完题容易想到我们应该先观察刻画一波合法调用路径集合的形态，不然根本做不了。

感受一下，直接刻画一个合法的调用路径集合的形态是比较困难的，我们不妨考虑什么样的调用路径集合会不合法。由于每一层专家系统是递归定义的，我们不妨考虑第 n 层到第 $n-1$ 层的连线的情况：

- i 号节点上的调用和 $i+2^{n-1}$ 号节点上的调用（$i\in[0,2^{n-1})$）不能往同一个 $S(n-1)$ 走，不然两条调用路径就会同时占用一个节点。并且可以注意到，如果在这一层上分开了，之后它们就**独立**了。
- 对于同一条调用，输入输出两边必须往同一个 $S(n-1)$ 走，不然之后就连不起来了。
- 每个节点上的调用只能要么往上要么往下。

由于非常漂亮的独立性，可以发现在每一层的每一个专家系统都满足以上三条限制的调用路径集合将必然满足条件。

这种限制启发我们使用图论的办法解决这个题。“反色连边，同色限制”，我们把不允许同时满足的条件之间连边，跑二分图染色取一种颜色出来就能得到一个方案。

考虑每一层开 2^{n+2} 个点，分别表示输入侧向哪个开关，输出侧向哪个开关。连边比较显然。连完边之后跑二分图染色取一种颜色的当这一层的路径方案。然后可以发现向两个 $S(n-1)$ 走的调用路径形成了两个子问题，可以分治递归下去继续做。

做到 $S(0)$ 的时候就可以退出输出方案了。

无解当且仅当不是二分图或者调用在输入输出节点就会重叠。然而事实上，chenxia25 在原题题解证明了图中一定没有奇环，从而必然是二分图。

因为有 n 层，每层总体来说都是 2^n 个点，边数和点数同阶，所以总时间复杂度是 $\mathcal{O}(Tn2^n)$。

需要注意一下常数。

闲话：这题的 UVA 数据远远没有开满……我本地测 100 组满数据，我跑了 9 秒，两篇题解分别跑了 23 秒和 38 秒。所以最后本来是削成了 30 组数据时限 7.5 秒。但是这个速度是按照我的笔记本电脑开的，实际上在我校电脑上最慢只需要跑 2.1 秒，因此考虑到评测机性能强大，时限最终确定为 5 秒。

后记

zime will be **fearless** with **rhythm** after suffering from **violence**.

愿节奏，旋律和原力与你我同在。