

# 杂题选讲

uyom

July 31, 2023

# 目录

- 1 [CSP-S 2022] 星战
- 2 [NOIP2022] 喵了个喵
- 3 [NOIP2022] 比赛
- 4 [NOIP2021] 方差
- 5 [CSP-S 2021] 交通规划
- 6 [NOIP2020] 移球游戏
- 7 [NOIP2020] 微信步数
- 8 [CSP-S2019] 树上的数
- 9 [NOIP2017 提高组] 宝藏

- 有一个  $n$  个点  $m$  条边的无向图，每条边都有激活和失活两种状态，初始时均为激活状态。
- 接下来会有  $q$  次操作，操作的种类有：
  - ① 失活某条边（保证这条边是激活的）；
  - ② 失活以某个点为终点的所有边；
  - ③ 激活某条边（保证这条边是失活的）；
  - ④ 激活以某个点为终点的所有边。
- 你需要在每次操作后，判断是否每个点都恰好有一条出边是激活的。
- $n, m, q \leq 5 \times 10^5$

- 考虑给每个点随机一个权值  $w_i$ 。
- 那么发现所有点的出度均为一就是统计每条边的起点所在点的点权和所有点的权值和一致。
- 于是我们对每个点维护一下以它为终点的失活的边的起点点权和以及激活的边的起点点权和就行。
- 时间复杂度  $O(n + m + q)$ 。

- 有  $n$  个栈和  $m$  张牌，牌的种类一共有  $k$  种，每种牌都有偶数张。
- 开始时所有的栈都是空的。这个游戏有两种操作：
  - 选择一个栈，将牌堆顶上的卡牌放入栈的顶部。如果这么操作后，这个栈最上方的两张牌种类相同，则会自动将这两张牌消去。
  - 选择两个不同的栈，如果这两个栈**栈底**的卡牌种类相同，则可以将这两张牌消去，原来在栈底上方的卡牌会成为新的栈底。如果不同，则什么也不会做。
- 你需要构造一个能消去所有卡牌的方案。
- $n \leq 300$ ,  $m \leq 2 \times 10^6$ ,  $2n - 2 \leq k \leq 2n - 1$

- 我们先来考虑  $k = 2n - 2$  怎么做。
- 这个部分是简单的，我们直接每个栈放两个数，留一个空栈。
- 进来一个已经出现过的数，就根据之前出现的时候在栈顶还是栈底然后操作就行。
- $k = 2n - 1$  的时候，我们依然考虑延续这个思路：尽量保持  $n - 1$  个栈里放 2 个元素，留一个空栈的状态。
- 考虑什么时候我们不能维持这个状态了，也就是  $n - 1$  个栈都已经放了 2 个元素了，然后再进来了一个没出现的元素，我们设这个元素为  $x$ 。
- 关键在于这个时候我们怎么操作来保持我们想要保持的状态。

- 我们考虑一下放  $x$  之后到下一次  $x$  出现的位置这段区间。
- 如果这段区间里的数全是现在栈顶的数，那就直接把  $x$  放空栈就行，因为后面只会出/入栈顶，用不到空栈处理底的操作，等到第二个  $x$  来的时候这个栈就又能空出来了。
- 否则，就是说这个区间至少存在一个数是现在某个栈的栈底，我们记最早出现的那个栈底是  $u$ ， $u$  所在的栈的栈顶是  $v$ 。
- 考虑从当前到  $u$  最早出现时， $v$  出现的次数：
  - 如果  $v$  出现偶数次，那就把  $x$  放这个栈栈顶，后面来的  $v$  都丢在空栈里，最后空栈会空出来，然后放  $u$  消掉。
  - 如果  $v$  出现奇数次，那就把  $x$  放空栈，后面来的  $v$  丢在栈顶，这样  $u$  来的时候  $v$  已经被奇数个  $v$  干掉了，可以直接丢进栈里面。
- 模拟上述过程即可，时间复杂度是  $O(m)$  的。

- 给定两个长度为  $n$  的序列  $a, b$ , 有  $q$  次询问, 每次询问会给出两个整数  $l, r$ , 求:

$$\sum_{x=l}^r \sum_{y=x}^r (\max_{i=x}^y a_i) (\max_{i=x}^y b_i)$$

- $1 \leq n, q \leq 2.5 \times 10^5$



- 我们离线所有询问，对右端点  $r$  进行扫描线。
- 在扫描过程中，我们设  $X_l$  和  $Y_l$  分别表示  $a_{l\dots r}$  和  $b_{l\dots r}$  的最大值。
- 在扫描右端点的同时，我们维护  $a, b$  的单调栈，这样如果扫描到  $i$ ， $i$  弹栈弹掉的区间的  $X$  就会修改为  $a_i$ ， $Y$  会修改为  $b_i$ 。
- 如果我们要查询的  $l, r$  的答案，那么相当于在扫描到  $r$  的时候，查询区间  $[l, r]$  的历史  $X \times Y$  和。
- 对每个位置  $i$  再维护历史  $X_i \times Y_i$  和  $sum_i$ ，现在需要支持的操作就是：
  - 对  $X$  区间加。
  - 对  $Y$  区间加。
  - 对所有  $i$ ，更新  $sum_i \leftarrow sum_i + X_i \times Y_i$ 。
  - 对  $sum$  询问区间和。

- 对序列建立线段树，每个线段树结点维护  $s_x, s_y, s_{xy}, s_{sum}$  分别表示区间内  $x, y, x \times y, sum$  的和。
- 还需要维护若干标记  $addx, addy$  表示区间加  $x$  和区间加  $y$  的标记，此外还有一些关于历史和的标记  $upd, hx, hy, hxy$  分别表示还有几次  $sum_i \leftarrow sum_i + x_i \times y_i$  的历史累加操作没下放，这几次没下放的操作的  $addx$  之和， $addy$  之和， $addx \times addy$  之和。
- 整理好要维护哪些标记后，逐个耐心推下怎么下传就行。
- 时间复杂度  $O((n + q) \log n)$ 。

- 还有一种更好想的方式，是把状态写成矩阵形式，矩阵维护长度，序列  $x$  的和，序列  $y$  的和， $x_i \times y_i$  的和和历史和。
- 每个节点的矩阵都是两个儿子的矩阵的和，每次更新就是乘一个矩阵，然后矩阵乘法有结合律和分配率，那么每次更新也是可以打 tag 的。这样朴素实现就能做到  $O(5^3 q \log n)$ 。
- 可以把矩阵写出来，注意到需要维护的矩阵本质不同的数只有 7 个，直接维护那 7 个数然后手动矩乘即可。

- 给定长度为  $n$  的非严格递增正整数数列  $a$ 。每次可以进行的操作是：任意选择一个正整数  $1 < i < n$ ，将  $a_i$  变为  $a_{i-1} + a_{i+1} - a_i$ 。
- 求在若干次操作之后，该数列的方差最小值是多少。
- $1 \leq n \leq 400$ ,  $a_i \leq 600$  或者  $1 \leq n \leq 10^4$ ,  $a_i \leq 50$

- 先将答案化为  $n \sum_i a_i^2 - (\sum_i a_i)^2$ 。
- 容易证明，操作对数列的影响就是交换差分数组的相邻两项。
- 一个重要的结论是，最优情况下，这个新的差分序列必然是一个单谷序列。证明在之后会给出。
- 考虑将所有差分从小到大排序后依次插入，令  $f_{i,j}$  表示已经考虑了前  $i$  个差分值，现有的  $a_i$  之和为  $j$  时的  $\sum_i a_i^2$  的最小值。令  $d_i, s_i$  分别表示差分值和差分值的前缀和。
- 转移就考虑放最左边还是最右边：
  - 最左边：  $f_{i-1,j} + d_i^2 \times i + 2 \times j \times d_i \rightarrow f_{i,j+i \times d_i}$
  - 最右边：  $f_{i-1,j} + s_i^2 \rightarrow f_{i,j+s_i}$
- 答案就是  $\min_i (n \times f_{n,i} - i^2)$ 。
- 这样直接做复杂度是  $O(n^2 v)$  的，但是  $d_i = 0$  的时候是不用转移的，所以可以做到  $O(n \min(n, v) v)$ 。

# 结论的证明

- 考虑化一下答案的式子：

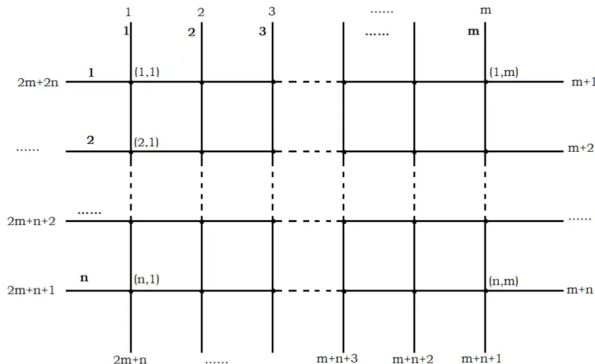
$$\begin{aligned} n \sum_i a_i^2 - \left( \sum_i a_i \right)^2 &= n \sum_i \left( \sum_{j \leq i} d_j \right)^2 - \left( \sum_i \sum_{j \leq i} d_j \right)^2 \\ &= n \sum_j \sum_k d_j d_k (n - \max(j, k)) - \\ &\quad \sum_j \sum_k d_j d_k (n - j)(n - k) \\ &= \sum_j \sum_k d_j d_k (-n \times \max(j, k) + n(j + k) - jk) \\ &= \sum_j \sum_k d_j d_k (n \times \min(j, k) - jk) \\ &= \sum_i d_i^2 i(n - i) + 2 \sum_j \sum_{j < k} d_j d_k (n - k)j \end{aligned}$$

# 结论的证明

- 注意到对于  $i > n/2$ , 如果  $d_i > d_{i+1}$ , 交换两者会让两个  $\sum$  都变小。
- 同理对于  $i \leq n/2$ , 如果  $d_i < d_{i+1}$ , 交换两者会让两个  $\sum$  都变小。
- 因此最优的差分序列是一个单峰序列。

# [CSP-S 2021] 交通规划

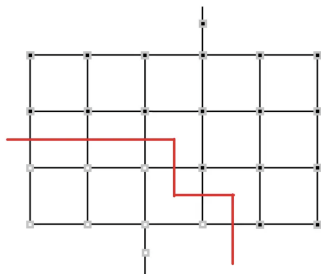
- 有一个  $n \times m$  的网格，里面有  $2mn - m - n$  条线段，外面还有  $2m + 2n$  条射线，他们都带有一个权值。



- $T$  次给定  $k$  个在某几条射线上的点，它们已经被染成黑色或白色，要求出将整个网格染色后，端点颜色不同的边的最小权值和。
- $1 \leq n, m \leq 500, T \leq \sum k \leq 50$



- 先考虑  $k = 2$ , 若此时两个附加点颜色相同, 显然答案为 0, 否则整张图一定会被染成一白一黑的两个连通块。
- 比如下图:

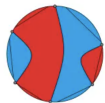


- 相当于就是找到一条从左边的平面到右边平面的一条路把两个点割开。

- 考虑多个询问点时，那么这个网格图上被钦定的颜色必定两色交替，且个数为偶数。



- 效仿  $k = 2$ ，我们相当于是要把这几段隔开来。也就是红-蓝间隔和蓝-红间隔两两匹配，使得匹配的路径和最小，两两之间的最短切割路径可以通过建平面图跑最短路得到。



- 至于最小权匹配，可以直接 KM 或者费用流，当然由于这个题特殊性质也可以区间 DP（因为分割线交叉了可以看作不交叉，看作在接头的地方碰头再分开）。
- 时间复杂度是  $O(knm \log(nm) + k^3)$  的。

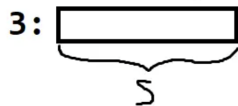
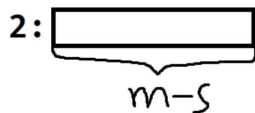
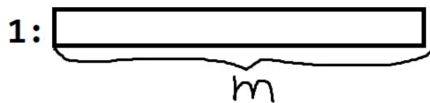
- 给定  $n + 1$  个栈，栈的高度限制为  $m$ 。
- 初始时前  $n$  个上每个有  $m$  个球，最后一个为空。球分为  $n$  种颜色，每种恰好  $m$  个。
- 一次操作可以把一个栈顶的元素弹出放到一个另一个栈顶，但是不可以使栈溢出或下溢。现要把同种颜色的球移动到同一个栈上。
- 你需要构造一个在 820000 次操作内的方案。
- $2 \leq n \leq 50, 2 \leq m \leq 400$

- 先考虑构造  $n = 2$  时的方案。
- 我们设原来第 1 根柱子上有  $s$  个 1，然后我们就进行以下操作：



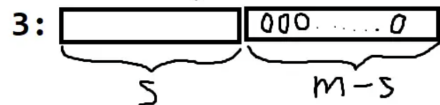
3:

# step 1

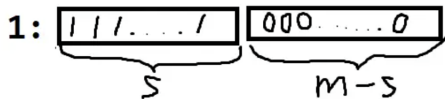


## step 2

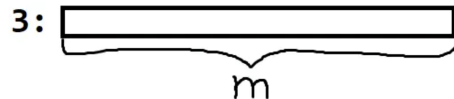
1:



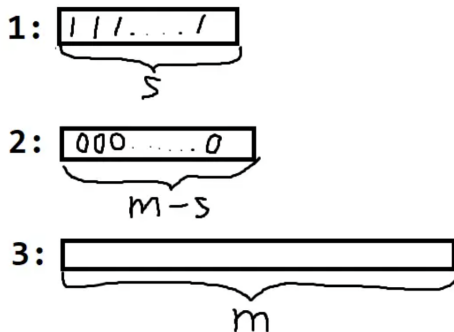
## step 3



2:




## step 4





## step 5

1:   
m

2:   
m

3:

- 于是我们可以在  $O(m)$  次操作内把两个栈里的  $m$  个扔到第一个栈中，再把另外  $m$  个扔到第二个栈中。
- 对于多种颜色，我们考虑把其转化到两种颜色求解。
- 考虑分治，先将颜色在  $[l, mid]$  中的球都扔到编号在  $[l, mid]$  的栈里， $[mid + 1, r]$  同理，然后递归求解。
- 具体怎么分的，就是拿两个指针一开始分别指向  $l$  和  $mid + 1$ ，每次看看这两个指针指向的栈里的数是  $[l, mid]$  更多还是  $[mid + 1, r]$  更多，如果是前者，就移动  $m$  个  $[l, mid]$  到  $i$ ，然后给  $i$  加一；否则移动  $m$  个  $[mid + 1, r]$  到  $j$ ，然后给  $j$  加一。
- 这样操作次数就是  $O(nm \log n)$  的。

- 有一个  $k$  维场地，第  $i$  维宽为  $w_i$ ，即第  $i$  维的合法坐标为  $1, 2, \dots, w_i$ 。
- 小 C 有一个长为  $n$  的行动序列，第  $i$  元素为二元组  $(c_i, d_i)$ ，表示这次行动小 C 的坐标由  $(x_1, x_2, \dots, x_{c_i}, \dots, x_k)$  变为  $(x_1, x_2, \dots, x_{c_i} + d_i, \dots, x_k)$ 。小 C 会将行动序列重复无限次，直到走出这个场地。
- 接下来，小 C 会以场地中的每个整点为起点，按照行动序列走直到走出场地。小 C 想知道他一共会走几步。
- 答案对  $10^9 + 7$  取模。
- $1 \leq n \leq 5 \times 10^5, 1 \leq k \leq 10, 1 \leq w_i \leq 10^9, d_i \in \{1, -1\}$

- 首先将问题转化一下，转为我们对每个  $i$  统计有几个起点走  $i$  步还没走出去。
- 不妨直接考虑每个  $\text{mod } n$  相同的  $i$ 。
- 令  $s_d, \text{premx}_{i,d}, \text{premn}_{i,d}$  分别表示走了一个周期  $d$  维的变化量，走了前  $i$  步时向正方向走的最大位移和向负方向走的最大位移。
- 对于走了  $k$  个完整周期又  $i$  步后的第  $d$  维来说，需要满足：
  - ① 如果  $s_d > 0$  则需要满足：
    - $x_d + \text{premn}_{n,d} \geq 1$
    - $x_d + s_d k + \text{premx}_{i,d} \leq w_i$
    - $x_d + s_d(k-1) + \text{premx}_{n,d} \leq w_i$
    - 整了一下就是
 
$$1 - \text{premn}_{n,d} \leq x_d \leq w_i - s_d k - \max(\text{premx}_{i,d}, \text{premx}_{n,d} - s_d)$$
  - ② 如果  $s_d < 0$ ，同理我们可以得到：
 
$$1 - s_d k - \min(\text{premn}_{i,d}, \text{premn}_{n,d} - s_d) \leq x_d \leq w_x - \text{premx}_{n,d}$$
  - ③ 如果  $s_d = 0$ ，可以得到  $1 - \text{premn}_{n,d} \leq x_d \leq w_i - \text{premx}_{n,d}$

- 对于每一维，可以算出  $k$  最大能是多少，对于一个  $k$ ，对答案的贡献就是  $\prod_i (r_i - l_i + 1)$ 。
- 其中  $r_i, l_i$  分别表示  $i$  这维能取的起点的上界和下界。
- 根据前面我们的推导， $r_i - l_i + 1$  应该是一个关于  $k$  的一次函数，若干个关于  $k$  的一次函数乘起来，那就是关于  $k$  的一个多项式。
- 也就是说现在是有个多项式  $f(x)$ ，我们要求  $f(1) + f(2) + \cdots + f(\lim_k)$ ， $\lim_k$  表示  $k$  的上界，对多项式做前缀和依旧是多项式，只是最高次会增加一，于是直接插值即可。
- 时间复杂度  $O(nk)$ 。

# [CSP-S2019] 树上的数

- 给定一棵树和每个节点上的初始数字，初始数字构成了一个排列，删除一条边的效果是交换被这条边连接的两个节点上的数字交换。
- 设  $p_i$  表示删完  $n - 1$  条边后  $i$  在哪个节点，要求合理安排删除  $n - 1$  条边的删除顺序，使得排列  $p$  字典序最小。
- $1 \leq n \leq 2000$

- 考虑把一个数字从一个节点运送到另一个节点的过程。
- 相当于给路上经过的边安排了一个删除顺序的限制。
- 可以发现，限制总共有 3 种：
  - 对于起点，选择的边要求是这个点所有边中第一条删除的；
  - 对于中间的点，要求选择的两条边的删除顺序必须连续；
  - 对于终点，选择的边要求是这个点所有边中最后一条删除的。
- 那么考虑对于每个点分别都建立一张图，图上面的点代表这个点连出去的一条边。
- 在这张图上的一条有向边代表出点要在入点之后马上选择，同时记录这个点钦定的第一条边和最后一条边。每个点的图不相关。

- 那么考虑贪心，从小到大确定每个数字的最终位置，同时保证不矛盾。
- 矛盾的情况有三种：
  - 图的形式不是若干条链的形式；
  - 第一个点（边）有入边，最后一个点（边）有出边；
  - 第一个点（边）所在的链的链尾是最后一个点（边），但是还有其他的点不在链中。
- 从每个数字的起始点出发，保证从根到这个点的路径不会引起矛盾，更新答案即可。
- 具体图的情况可以用并查集维护，时间复杂度  $O(n^2\alpha(n))$ 。



- 给定一张  $n$  个点， $m$  条边的有重边的无向带权图。
- 你需要确定一个根，然后找到一棵生成树使得所有边，边权和到根的距离的乘积的和最小。
- $n \leq 12$

- 考虑一层一层铺这个生成树，令  $f_{i,S,S'}$  表示当前生成树已经铺了  $i$  层了，已经在生成树中的点集为  $S$ ，第  $i$  层的点集为  $S'$ 。
- 转移就考虑新加入一个子集作为新的一层，新加入的每个点显然是连向  $S'$  中和他有边的且边权最小的点，这个可以预处理。
- 这样的话复杂度是  $O(n4^n)$  的，实现精细一点可以过。
- 但其实， $S'$  这一维完全是可以省略的！
- 因为如果我们连向的点实际并不是在第  $i$  层的，这个方案必然不会是最优的（因为不如在连向的点的下一层就加入这个点，而不是现在加入），而我们这样 DP 也能覆盖到所有的情况，因此这样就是对的！
- 这样的话，复杂度就是  $O(n3^n)$  了。

Thank you for listening!