T1 技能

为方便，后文将账号位置称为起点，关键点称为终点。

sol1

$m \le 10$

预处理每一个起点到每一个终点的距离，暴力枚举哪一个起点到哪一个终点。

时间复杂度 $O(nm^2 + n!m)$ 。

能拿 40 分。

正解

显然，从贪心角度考虑，对于每一棵子树，肯定是子树内的起点和终点优先匹配，匹配不了的再到子树外去匹配。

可以证明，子树内匹配肯定比子树内外匹配更优。

那么我们就可以直接得出，对于每一条边，它被账号便利的次数肯定是子树内起点与终点数量之差。

直接对于每一条边，统计答案就可以。

因为 w 最大到 10^{100} ，所以我们需要写一个高精。

只需要高精乘低精和高精度加法，直接数组模拟就可以了。

时间复杂度是 $O(100n)$ 。

T2 奶茶代金券

考虑到两个大于 $\frac{m}{2}$ 的奶茶组合起来和单独购买浪费的钱数是一样的（虽然本题中无法单独购买奶茶），想要省钱，那么应该考虑组合两个小于 $\frac{m}{2}$ 的奶茶，或者组合一个大于 $\frac{m}{2}$ 的（下文简称“大的”），一个小于 $\frac{m}{2}$ 的（下文简称“小的”）。

贪心思路应该是优先将大的和小的匹配进行购买，如果匹配结束后还有多余的“小的”，那么将“小的”之间两两匹配；如果匹配结束之后还有多余的“大的”，那么无法继续匹配。

考虑大的和小的进行匹配的过程，恰好小于 $\frac{m}{2}$ 的奶茶只能和恰好大于 $\frac{m}{2}$ 的奶茶匹配。例如 $m = 10$ 的时候，4 只能和 6 匹配，如果和 7、8、9 匹配，则浪费的钱数和单独购买相同，导致无意义贪心。而价格为 1 的奶茶可以和 6、7、8、9 匹配，我们可以感觉到，所有小于 $\frac{m}{2}$ 的奶茶中，越接近 $\frac{m}{2}$ 的奶茶匹配的选择越少，所以应该被优先匹配。

最终思路：将所有“小的”从大到小排序，然后开始匹配，匹配的时候将所有“大的”从小到大进行匹配，类似双指针维护匹配过程。

最终再判断是否还有多余的“小的”，将这些“小的”两两匹配。（随便怎么匹配都是一样的）

T3 帮助

sol1

暴力枚举两个人，看第一个人会不会帮助第二个人。

如果会帮的话直接帮。

时间复杂度 $O(n^2)$ ，可以拿 30 分。

sol2

所有 t 都相等。

显然对于每一个人，要么他不愿意帮助任何一个人，要么他愿意帮助所有人。

要么他愿意接受所有人的帮助，要么不愿意接受任何一个人的帮助。

所以对于每一个，直接看他会不会帮助其他人，愿不愿意接受其他人的帮助。

直接统计答案就可以。

时间复杂度 $O(n)$ ，可以拿 10 分。

注意，从这里以后的算法，都要特判一下，存不存在某个人帮助了自己的情况。

sol3

$t_i \leq 10$

首先我们可以对 a, b, c, d 做一些处理。

显然，如果 $a > 10$ ，这个人不会接受任何人的帮助。

如果 $b > 10$ ，我们可以令 $b \leftarrow 10$。

c 和 d 同理。

然后我们可以预处理一个数组 $g_{i,j}$ 表示所有成绩为 i 的且愿意帮助成绩为 j 的人，所做题目数量之和。

显然，我们只需要预处理 $0 \leq i, j \leq 10$ 的 g 值。

然后对于每一个人 i ，求出 $g_{c_i \sim d_i, t_i}$ 即可。

时间复杂度 $O(10n)$ ，能拿 10 分。

sol4

$a_i = b_i = c_i = d_i$

和上面差不多。

预处理一个 $g_{i,j}$ ，表示所有成绩为 i 的且愿意帮助成绩为 j 的人，所做题目数量之和。

有值的 g 最多只有 n 个，用map存储即可。

第 i 个人的答案就是 g_{c_i, t_i}。

时间复杂度 $O(n \log n)$ ，能拿 10 分。

sol5

$a_i = 0, b_i = 10^9$

没有 a, b 的限制了，每一个人都愿意接受其他人的帮助。

我们首先将所有人按照成绩，升序排序。

那么对于每一个人，他愿意帮助的人（在排序之后）一定是一个区间。

并且我们可以直接使用二分，求出他愿意帮助的人是哪一个区间。

然后维护答案的差分序列，最后前缀和即可。

时间复杂度 $O(n\log n)$，能拿 10 分。

所有暴力就好啦，一共能拿 70 分。

sol6

还是 $a_i=0,b_i=10^9$ 。

这是另一个做法，和正解更靠近一些。

还是先将所有人按照成绩，升序排序。

还是用二分求出每一个人，愿意帮助哪一个区间。

我们再维护一个 g_i ，表示所有愿意帮助成绩为 i 的人，所做题目数量之和。

容易发现 g_i 是可以递推的。

$$g_i=g_{i-1}+\sum_{c_p=i}f_p-\sum_{d_p=i-1}f_p$$

你还可以发现，在递推的过程中， g 的值只变化了不到 $2n$ 次。

我们这样就可以一边递推 g ，一边算每一个人的答案。

时间复杂度一样，是 $O(n\log n)$ 。

正解

和前面一样的排序，二分。

差不多的套路，维护 $g_{i,j}$ 表示所有成绩为 j 且愿意帮助成绩为 i 的人，所做题目数量之和。

$g_{i,j}$ 也可以递推，式子长这样：

$$g_{i,j}=g_{i-1,j}+\sum_{c_p=i\text{且}t_p=j}f_p-\sum_{d_p=i-1\text{且}t_p=j}f_p$$

和上面的一样，所有的 g 变化次数加在一起不超过 $2n$ 次。

还是一样，可以在计算答案过程中计算出 g 的值。

最后统计答案的时候，求的就是 $\sum_{j=a_i}^{b_i}g_{i,j}$ 。

所以我们先离散化，然后使用树状数组递推 g 的值。

时间复杂度还是 $O(n\log n)$ 。

T4 神奇的变换

一句话题意：给你一个序列和若干个询问，每次询问给出一个区间，求出这个区间所有数字乘起来之后，莫比乌斯函数值，约数个数，约数和。

约定：为了方便计算时间复杂度，后文将 $1\sim10^3$ 内质数个数（共 168 个）记作 $O(\sqrt{n})$，将 $1\sim10^6$ 内质数个数（共约 78000 个）记作 $O(n)$ 。

sol1

$n,q\le 1$

你可以按照题面进行模拟。

时间复杂度 $O(a)$ ，能拿 20 分。

sol2

$n,q\le 10^3$

首先我们知道肯定不能先求出 $\prod a_i$ 的值再算答案（废话

考虑有什么东西，可以代替这个序列，计算出答案。

我们发现，假如可以求出，每一个质数，在这个区间内的出现次数，同样可以把答案算出来。

所以我们先筛出 $1\sim 10^3$ 的质数，然后对序列中的每一个数字，分解质因数。

最后，对于每一个询问，暴力便利区间，并统计出现的质数，计算答案。

时间复杂度是 $O(nq\log a\log mod)$ 。

为啥是 2 个 $\log$ 呢。

第一个是因为，每一个数字，最多可能会有 $\log$ 个质因数。

第二个是因为，每一次计算贡献的时候，肯定是先把原来（统计的这个质数）的贡献除掉，然后再乘上新的贡献，这里我们需要预处理逆元。

感觉有点卡。

所以我们可以提前预处理逆元。

因为我们是知道所有质数出现次数的，假如一个质数出现了 i 次，我们显然只需要预处理这个质数出现 $0\sim i$ 次的时候，贡献（以及逆元）分别是多少。

时间复杂度是 $O(n\log n\log a+nq\log n)$ 。

可以拿 40 分（包括前面的 5 个点）

sol3

$a_i\le 10^3$

感觉和sol2的思路差不多。

因为所有 $a\le 10^3$ ，所以出现的质因数肯定也 $\le 10^3$ 。

预处理 $pre_{i,j}$ 表示，前 i 个数中，第 j 个质数出现的次数。

询问的时候暴力枚举 $1\sim 1000$ 中的质因数，根据他的出现次数统计答案。

时间复杂度 $O(n\log n\log a+q\sqrt{n})$ 。

能拿 20 分。

sol4

$$\prod_{j=l_i}^{r_i} a_j\le 10^7$$

不难发现，我们只需要线性筛，就可以知道 $1 \sim 10^7$ 中所有数的答案了，所以我们只需要快速求出一个区间内所有数的乘积。

随便选一个大一点的质数当作模数，然后对这个序列做前缀积，直接逆元就可以了。

时间复杂度 $O(10^7 + q \log mod)$ 。

能拿 40 分（包括前 5 个点）。

暴力分一共 80 分。

正解

个人感觉和sol3关系比较大。

首先我们会发现一个很不错的性质：序列中的每一个数字，最多只“包含”一个大于 1000 的质数。

而正好，小于等于 1000 的质数对答案造成的贡献，我们可以用sol3的方法来处理。

也就是说，我们要解决的，就只是大于 1000 的质数所造成的贡献。

考虑对序列进行分块，块长为 $\sqrt{n}$。

我们再预处理另一个数组 $pre'_{i,j}$ 表示，前 i 个**块**中，第 j 个质数出现了几次。

我们还可以预处理一个数字 $val_{l,r}$ 表示，第 l 个块到第 r 个块中所有数，答案是多少。

这个数组的话，直接枚举起点，然后一个一个加数字。

然后再来看统计答案。

首先对于一个询问的区间 $[l, r]$ ，它肯定是由中间的整块和两边的散块构成的。

对于中间的所有整块，我们已经知道了答案（val 数组）以及所有质数出现次数（pre' 数组）。

然后对于两边的散块，我们直接一个一个数字加进去并且更新答案就可以了。

时间复杂度是 $O((n+q)\sqrt{n} + n \log n \log a)$