Stack

我们把询问 $[l,r]$ 过程中“成功”的对叫关键点。

观察询问 $[l,r]$ 相对于 $[l+1,r]$ 关键点集合的变化：

1.离 l 最近的点若与之 a 值不同，且 b 值比 b_l 小，那么它不再是关键点。去掉关键点后重复这一过程。

2.将 l 加入关键点集合。

对于一个 r ，我们记 p_i 表示最小的 l ，使得询问 $[l,r]$ 时 i 还是关键点。模拟上述过程可以 $O(n)$ 求出每个 p_i。发现对于不同的 r, p_i 是不变的。故可以令 $r=n$， $O(n)$ 求出每个 p_i。

此时我们发现答案就是 $[l,r]$ 中 $p_i \leq l$ 的 i 个数，差分后二维数点即可。

复杂度 $O((n+q)\log n)$。

Discuss

$10-30$ 分做法

对任意两个集合判断，可以使用 `bitset` ，时间复杂度 $O(\frac{n^3}{w})$ 。

100 分做法

首先将所有集合按集合大小从小到大排序。

考虑枚举位置 x，对于所有包含 x 的考虑：

如果相邻两个没有包含关系，那么它们一定满足条件。

这样做用 `bitset` 加速是 $O(\frac{n^3}{w})$ 的。考虑维护它们的包含关系。

对于每个 x ，包含关系都是一条链，需要维护的包含关系一定是树形结构，否则已经找出了一对满足条件的集合。

考虑把这条链加到森林上，如果能够成功加入那么考虑下一个位置 x 。

否则从下至上，第一个不和森林匹配的点和它的前驱一定满足条件。

只会判断 $O(n)$ 次包含关系，用 `bitset` 判断时间复杂度 $O(m+\frac{n^2}{w})$ 。

可以通过 70 分。

发现对于这个森林，只会判断父亲是否包含儿子，所以把儿子包含的所有题目向父亲的集合中查询，用哈希表维护总复杂度是 $O(m)$ 。

因此每组数据的复杂度为 $O(n+m)$ 。

Sort

为了方便描述，记 $g(i,j,k)$ 表示 $a_{k,i}+a_{k,j}$。

$k=2$ 和 $n\le 3000$ 的 20 分很容易拿到，不再赘述。

算法一

考虑 $k=3$ 时怎么做。

对于每个排列 p_1,p_2,p_3, 统计满足 $g(i,j,p_1)<g(i,j,p_2)<g(i,j,p_3)$ 的对 (i,j) 的 $a_{p_2,i}+a_{p_2,j}$ 之和即可（如果不等式不会取到等号的话）。

发现 $g(i,j,p_1)<g(i,j,p_2)$ 可以写成 $a_{p_1,i}-a_{p_2,i}<a_{p_2,j}-a_{p_1,j}$ 。从而可以用 $3!$ 次二维偏序解决原问题。

但对于不等式取到等号的对，不管怎么算都会漏或者重。一种简单的处理方式是把每行乘 3, 然后第 i 行加 $i-1$ 来避免取等。

结合前 20 分对应的算法可得 50 分。

算法二

对于 $k=4$ 的情况，记 $h(i,j,k)$ 表示三个满足 $s\ne k$ 的 $g(i,j,s)$ 的中位数。

可以注意到答案等于

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m}\left(g(i,j,k)-\frac{1}{2}h(i,j,k)\right)$$

g 的和是容易计算的， h 的和套用 $k=3$ 的做法即可。

这是一个大常数 $O(n\log n)$ 做法（瓶颈是 24 次二维偏序）。

直接写可能常数稍大有点卡，不过把排列 p_1,p_2,p_3 和 p_3,p_2,p_1 在一个归并排序里做就可以卡掉近一半常数，可以保证通过。

Bonus

测试点 $6-9$ 留给常数较大的正解。

也可以用 $min-max$ 容斥将 $k=4$ 的问题转化为 $k=3$ 的问题，细节处理可能与本题解不同。