BITACM 俱乐部 2023 年暑假集训 Rating#1 Solution

2023 年 7 月 22 日

A 图

$f_{i,j}$ 表示第 i 个点到达 j 的方案数，答案为 $f_{k,1}$。

e_j 为给出的 k 条边形成的图中，j 相邻的点集，可以列出方程:

$$f_{i,j} = \sum_{k=1}^{n} f_{i-1,k} - \sum_{v \in e_j} f_{i-1,v} - f_{i-1,j}$$

式子中第一项可以前缀和优化，时空复杂度为 $O(n^2)$。

B 区间

f_i 表示处理了所有 $r \leq i$ 的区间，可以得到的最大的可划分集合的对数。g_i 表示处理了所有 $r \leq i$ 的区间，在留下一个未配对集合的情况下，可以得到的最大的可划分集合对数。

按区间右端点从小到大处理，$f_i = \max_{j=l}^{i} g_l + 1, g_k = \max(g_k, \max_{j=1}^{l-1} f_j)$，$l$ 为当前区间左端点.

g 中第二项可以前缀和优化，对 g 的更新是区间取最大，可以线段树优化，时间复杂度为 $O(n \log n)$。

C 道路

两个集合之间的两两最短路的最小值。

裸的最短路。

建立一个 S 和一个 T，S 向 A 集合中的所有点连权值为 0 的边，B 集合中的所有点向 T 集合中连权值为 0 的边。以 S 为源点跑最短路，dis_T 就是答案。

没卡 SPFA。

D 鱼鱼学算术

双指针板子。

由于给定数列有序，对于一个固定的 A_i，满足 $A_j - A_i = B$ 的下标 j 显然一定是连续的，且 j 的区间随着 A_i 的值变大一定会向右移动，于是考虑双指针。一个指针用来从头开

始枚举 i，一个用来记录 j，对于所有 A_i 值相同的 i，我们只需要记录个数并在一处计算即可。显然此时 i 和 j 的移动一定是从左到右的。

实际上直接拿个桶存放就行，值域较小。

E 奇异的空间

搜索。

1. 把所有的能量域标出来。

2. 从 1 号能量域开始 dfs，每次搜到一个能量域时，从下向上看有没有能量管线连接到这个能量域，如果有，那么就沿着这条管线走，看它连接到了哪个能量域，再搜索连接到的这个能量域。

3. 回溯的时候输出编号。

F Brotato 的奇妙冒险

令 $b_i = Qa_i$，问题转化为在数组 b 中找到与 Pa_i 距离最近的数。枚举 i，在 b 数组上二分即可。

时间复杂度 $O(n \log n)$

G 鱼鱼之怒

单调栈板子。

在每次往后面加入人后，考虑可能挡住鱼鱼的人的变化：不妨设加入之前可能挡住鱼鱼的人的编号集合为 Q，则先取后加入的来进行比较，可以看出是一个后进先出的结构，用栈。如果对应的身高比当前的高，则仍然是可能挡住鱼鱼的人。若身高比当前的低，则其不再是可能挡住鱼鱼的人，移出栈。最后再将当前的人加入栈，因为他作为最后一个一定是可能挡住鱼鱼的人。在每次加入和移出的时候维护一下异或和即可。

时间复杂度 $O(n)$.

H 原神启动

由于原神 id 最多只有 18 位 (10^{18} 对应的星铁 id 是 2，不考虑)，能表示出的最大星铁 id 是 10^{18}，当且仅当原神 id 为 18 个 9，其他情况不可能比 18 个 10 相乘还要大。

因此，只需要筛选出 10^{18} 以内的所有星铁 id，就能保证不会有遗漏。

下面考虑如何预处理出所有星铁 id。

注意到星铁 id 是由 $1 \sim 10$ 内的数字相乘得来的，而 $1 \sim 10$ 内的质数只有 $2, 3, 5, 7$，因此所有星铁 id 都能质因数分解为 $2, 3, 5, 7$ 的乘积；反之，如果一个数含有不是 $2, 3, 5, 7$ 的质因子，那么就一定不是星铁 id。

一个朴素的想法是用 set，每次取出一个数字分别乘以 $2, 3, 5, 7$ 插入 set 中，利用 set 自带的排序功能就可以得出第 k 个星铁 id。本题数据比较宽松，这种 $O(n \log n)$ 的做法是可以过的。

还有一个 $O(n)$ 的做法，对 $2, 3, 5, 7$ 分别建立四个指针，$p2$ 表示上一个乘以 2 得来的数字的位置，以此类推。如果当前的 a_i 是某个数乘 2 得来的，那一定是 $a_{p2} \times 2$。

对于这个结论可以简单证明一下，a_{p2} 之前的数乘 2 得到的 2 的幂次不会大于 a_{p2} 的，可以假设把 a_i 的其他幂次不变，2 的幂次减一 (等于 a_{p2} 中 2 的幂次)，这样得到的一个数肯定会在 a_{p2} 到 a_{i-1} 之间出现过。a_{p2} 之后的数乘 2 得到的数又肯定大于 $a_{p2} \times 2$，所以只能是 $a_{p2} \times 2$ 其他三个指针同理。

在插入 a_i 时，只需要比较 $a_{p2} \times 2, a_{p3} \times 3, a_{p5} \times 5, a_{p7} \times 7$.

这四个数字，如果取的是 $a_{p2} \times 2$，那就把 $p2$ 加一，其他同理。

在预处理出星铁 id 的过程中，如果发现当前数大于 10^{18}，就可以结束预处理。我们设预处理出的星铁 id 个数为 n。

通过预处理，就可以在 $O(1)$ 的时间内获得第 k 个星铁 id。如果 $k > n$，说明不存在。对于 $k \leq n$，可以用贪心策略获取对应的最小原神 id。

可以很容易想到一个贪心策略，先看星铁 id 能拆出多少个 10，再看能拆出多少个 9，以此类推。至于为什么这样是最优的，假设当前能拆成 4×5，那么在原来的原神 id 中表示就是 34，如果把 4×5 改成 2×10，原来的表示就是 19，比 34 更小。

另外还需要判断，如果最小原神 id 大于 18 位，说明当前星铁 id 也是不存在的。因为预处理只是保证了不会有遗漏，可能会有多余解。

I 如何获得肉

题意就是选择一个长度为 m 的区间，使得区间中最大数和次大数的和最大。

维护一个单调队列，最大数一定是队首，次大数可能是队列中的第二个元素，也可能是出队元素中最大的。用一个变量存储出队元素的最大值即可。

J 游戏人生 I

暴力搜索或者二进制枚举某个游戏选或者不选。

复杂度 $O(2^N)$.

K 游戏人生 II

如果暴力搜索或者二进制枚举某个游戏选或者不选的话，复杂度会寄掉。

使用 meet in the middle，枚举一半的 $\sum t - \sum v$ 的值，再枚举另一半，记录和为零的个数。

复杂度是 $O(n2^{\frac{n}{2}})$。

不过得用个 hash 表存一下，貌似 map 平均在 2000ms 左右, unordered 平均 800ms(验题人说的)，pbds 里的 gp_hash_table 能到 500ms 左右。

L 字符串

$f_{i,j}$ 表示原串长为 i，简写后为 j 的字符串的数量。每次在原串尾端添加一段连续字符。

$$f_{i,j} = \sum_{k=1}^{i} 25 \times f_{i-k,j-len(k)-1}$$

$len(k)$ 为数字 k 的长度。对于 $1-9, 10-99, 100-999, 1000-3000$，这些 $len(k)$ 相同的长度，前缀和优化。时间复杂度 $O(len(n)n^2)$。

M 我是签到题？

由于相邻的牌朝上的面写有的数字必须不同，因此我们很容易得到 dp 方程：

$f_{i,0} += f_{i-1,0}$，$a_i \neq a_{i-1}$.

$f_{i,0} += f_{i-1,1}$，$a_i \neq b_{i-1}$.

$f_{i,1} += f_{i-1,0}$，$b_i \neq a_{i-1}$.

$f_{i,1} += f_{i-1,1}$，$b_i \neq b_{i-1}$.

其中 $f_{i,0}, f_{i,1}$ 分别表示考虑前 i 张牌，第 i 张牌选正面 or 反面的合法方案数。

这样我们就得到了不需要修改的做法。

容易想到把转移方程写成矩阵形式，

$$\begin{bmatrix} f_{i,0} \\ f_{i,1} \end{bmatrix} = \begin{bmatrix} [a_i = a_{i-1}] & [a_i = b_{i-1}] \\ [b_i = a_{i-1}] & [b_i = b_{i-1}] \end{bmatrix} \begin{bmatrix} f_{i-1,0} \\ f_{i-1,1} \end{bmatrix}$$

显然 $f_{n,0}, f_{n,1}$ 就是 i 从 2 到 n 的转移矩阵的乘积，再乘上初始的矩阵 $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ 即可。

因为有修改，容易想到维护区间乘积，因此我们用线段树维护区间内转移矩阵的乘积。

而区间查询就是初始矩阵乘以区间内矩阵的积。

这样操作 $1, 2, 4$ 都可以容易地解决了，单点修改的时候需要修改两个位置的转移矩阵。

操作 3 是出题人一拍脑子想出来的，但是却把出题人脑子成功弄过载。

注意到，我们无论怎么翻转，都对区间查询的答案没有任何影响，由对称性可以显然发现。那么翻转的影响到底在哪呢？——在于翻转后的单点更新！我对一个翻转过的牌进行单点更新，那么我必须要获取它的真实状态（到底正面和反面的值是多少），否则就会导致 bug。

因此，我们考虑对于区间翻转，对区间打一个 tag 表示区间是否被翻转。然后在单点更新和区间翻转时下传 tag，当 tag 被传到我们要更新的叶节点时，若 tag $= 1$，那么将此时的 a_i, b_i 互换，然后再按照需求做单点更新。单点更新结束后，重新生成和这个点相关的转移矩阵，也就是单点更新矩阵即可。

时间复杂度 $O(n \log n)$.