The 9th CCPC Harbin

题目讲解

2023-11-05

A. Go go Baron Bunny!

讲题视频

B. Memory

- 首先有 $Mood(i) = \frac{Mood(i-1)}{2} + a_i$，但如果直接进行浮点数运算则会因为精度误差而无法得到精确的结果，所以本题的关键在于避开浮点数的运算。
- 因为a_i都是整数，所以我们可以用一个二元组(f_i, b_i)来表示$Mood(i)$，其中f_i表示$Mood(i)$是否有小数部分，b_i表示$Mood(i)$的整数部分（向下取整）：
- $Mood(i) < 0$ 等价于 $b_i < 0$
- $Mood(i) = 0$ 等价于 $b_i = 0$ 且 $f_i = false$
- $Mood(i) > 0$ 等价于 $b_i > 0$，或者 $b_i = 0$ 且 $f_i = true$
- 初始有$f_0 = false, b_0 = 0$，那么$(f_i, b_i)\ (i = 1, 2, \cdots, n)$ 二元组的维护方式如下：
- $f_i = f_{i-1}$ or $(b_{i-1} \mod 2) \neq 0$
- $b_i = a_i + \lfloor \frac{b_{i-1}}{2} \rfloor$

C. Karshilov's Matching Problem II

- 预处理出 T 串每个后缀与 S 串的 LCP，这部分可以用拓展KMP（Z函数）$O(n)$ 处理。
- 对于每个询问 $[L,R]$ ，可以在 $[L,R]$ 中找到最靠左的 mid 满足 $mid-1+LCP(T[mid,n],S)>R$ 。这一步存在多种 $O(\log n)$方法查找。
- 对于 $[mid,R]$ 的这部分，和 S 的某个前缀一致，预处理直接算出即可。
- 对于 $[L,mid-1]$ 的这部分，因为每个位置和 S 的LCP都不会超出 R，所以每个位置答案都是 LCP 长度的前缀代价和，也可以预处理后 $O(1)$ 算出。
- 综上，复杂度 $O(\log n)$。
- 设置题目时放过了 $O(n\sqrt{n})$ 的莫队做法，如果是 $O(n\sqrt{n\log n})$ 的做法可能会有一点卡常。

D. A Simple MST Problem

首先考虑 $[L, R]$ 范围中存在质数的情况。基于Kruskal，此时对于任何一个数 x ，只需要考虑价值为 $\omega(x)$ 和 $\omega(x)+1$ 的边即可（要么找一个质因子集合包含 x 的数相连，要么和那个质数相连）。

我们可以预处理出每个整数 x 在 10^6 范围内的两个值：

- l_x：小于 x 但其质因子集合被 "x 的质因子集合" 包含的最大的数。
- r_x：大于 x 但其质因子集合被 "x 的质因子集合" 包含的最小的数。

这两个值都可以 $O(n \log n)$ 处理。（用欧拉筛求出最小质因子，即可 $O(\log n)$ 分解质因子。搜索以遍历 x 质因子集合的数）

D. A Simple MST Problem

质因子集合相同的整数之间，可以先互相连接形成连通块，视为一个点。每个点贪心向任何一个自己包含的点连边即可，最后会剩下一些没有出度的点。这些点当中必有一个单质数点，贪心连向这个点即可。基于预处理，该贪心过程可以 $O(n)$ 实现。

对于 $[L, R]$ 范围中不存在质数的情况，说明这个区间很小（可以筛一遍看看区间的最大值），直接prim最小生成树即可。

E. Revenge on My Boss

首先考虑在给定序列的情况下，Bob会如何决策。

定义 $B = \sum_{i=1}^{n} b_i$。Bob会找到 m 以最大化

$$(\sum_{i=1}^{m} a_i + \sum_{i=m}^{n} b_i) \cdot c_m = (B + \sum_{i=1}^{m} a_i - b_i + b_m) \cdot c_m$$

我们不妨设 $d_i = a_i - b_i$ ，$\{sumd_i\}$ 表示 $\{d_i\}$ 的前缀和数组。

接下来考虑二分答案 P ，检验是否存在一种顺序，去放置这些城市，使得 $\forall m, 1 \leq m \leq n$ ，均满足 $(B + sumd_m + b_m) \leq P/c_m$ 也即

$$sumd_{m-1} \leq P/c_m - B - a_m$$

该限制的右侧是和放置顺序无关的量，记为 e_m，而左侧是在选第 m 个城市之前的城市的 d 值前缀和。

检验该问题，是一个经典的贪心问题。

E. Revenge on My Boss

首先找到 $d_i <= 0$ 的城市，他们按照 e_i 的值从大到小排列去选取，如果出现了不可选取的城市自然而然就失败。这一步贪心的降低了后面的 $sumd$ 。对于 $d_i > 0$ 的城市，他们按照 $e_i + d_i$ 从小到大排序，如果中间出现了不可选取的城市也就失败。如果最后每个城市都放上去了，就说明 P 是合法的。（证明略）

因此得到 $O(n \log n \log V)$ 的算法，其中 V 是答案值域。

F. Palindrome Path

本题要求最后的操作序列是个回文串，所以我们考虑在正向操作的同时考虑反过来添加相同的操作，题解做法的核心在于构造一个"走一步"的连套操作，使得当前起点“走一步”之后，回文部分的影响不改变终点。

如果能够实现这个连套操作，那么就从(sr, sc)开始给迷宫建一棵dfs树，然后沿着树边遍历每个格子，最后再走到(er, ec)即可。不妨假设我们想往右走一步，即从(sr, sc)走到$(sr, sc + 1)$，具体的连套操作构造方法如下：

F. Palindrome Path

考虑基于(sr, sc)不断进行L操作，为了保持操作序列的回文性，我们也要从后往前添加L操作，然后我们需要逆推经过这一系列L操作之后能恰好停在(er, ec)的格子，所以在(sr, sc)进行L操作的同时也要基于(er, ec)进行它的逆操作，即R操作，假设分别操作成功了k次之后有某一边无法再成功操作，此时两边的格子分别到了$(sr, sc-k), (er, ec+k)$这两个位置，那么有两种情况：

1. $(sr, sc-k)$ 无法再成功进行L操作，但$(er, ec+k)$可以再成功进行一次R操作。此时再让两边分别进行一次L操作和R操作后，$(sr, sc-k)$不动，而$(er, ec+k)$移动到$(er, ec+k+1)$，之后再让$(sr, sc-k)$和$(er, ec+k+1)$进行$k+1$次R操作和L操作，那么两个格子最后会停在$(sr, sc+1)$和(er, ec)上，达成预期。

F. Palindrome Path

2. $(er, ec+k)$无法再成功进行R操作。首先让$(sr, sc-k)$进行$k+1$次R操作，显然它会移动到$(sr, sc+1)$，关键在于$(er, ec+k)$这个格子，虽然按照道理进行$k+1$次逆操作，即L操作后它会移动到$(er, ec-1)$，但是在这个逆操作过程中我们只是想要逆推出进行$k+1$次R操作和k次L操作之后能恰好停在(er, ec)的格子，现在我们考虑直接把(er, ec)带入这个过程，可见其进行$k+1$次R操作之后会停在$(er, ec+k)$，这是假设的前提，然后进行k次L操作之后又会回到(er, ec)，正好就是符合预期的。

F. Palindrome Path

- 综上，我们可以把构造(sr, sc)到(er, ec)的回文操作序列的问题规约成$(sr, sc+1)$到(er, ec)的问题，不妨设$(sr, sc+1)$到(er, ec)的一个回文操作序列是P，那么按照上述分析，(sr, sc)到(er, ec)的回文操作序列就可以是 LL$\cdots$LRR$\cdots$R P R$\cdots$RRL$\cdots$LL ，其他方向的规约方法类似。
- 可见规约一次所需要添加的长度最多是$4\max(n, m)$，需要规约的次数，即在dfs树上需要移动的步数最多是$3nm$，所以这个做法所构造的操作序列的一个上界是$12nm\max(n, m)$，是在可接受范围之内的。

G. The Only Way to the Destination

如果我们可以把所有空格子的上下左右联通图建出来，问题等价于判断原图有没有环路。

但是数据过大，做不到。此时利用放置墙的性质，一列一列考虑迷宫。每一列中，迷宫会被墙切割成若干段空格子的区间。我们把每列中的一个空格区间，视作一个结点，这样最多有 $O(m+k)$ 个结点。

其中，全为空（没有放墙的列）的列会有 $O(m)$ 个。根据题目要求，只要 $n \geq 2$，如果有两个相邻的列都是空的，说明一定不符合条件；而 $n=1$ 的时候又较为容易特判。因此只需要考虑 m 与 $O(k)$ 同级的情况了。这样，结点数量变为 $O(k)$ 。

G. The Only Way to the Destination

接下来考虑结点之间的连边：我们处理每两个相邻的列：对于列1中的空格段，如果与列2中的空格段恰好在一行有交集，则把这两个空格段的表示结点进行连边（如果有两行有交集，意味着出现了田字型的环）。这样的连边可以用双指针实现，只会连 $O(k)$ 条边。

最后我们得到一张具有 $O(k)$ 个结点，$O(k)$ 条边的无向图，判环即可。

H. Energy Distribution

题意是求 $F(x_1, x_2, \ldots, x_n) s.t. x_i \geq 0, \sum x_i = 1$ 的极大值。

若极值点 $x_i > 0$，$x_j > 0$，则必须满足 $\nabla_{x_i} F = \nabla_{x_j} F$（否则微调一下 F 就能变大）

可以枚举取值为 0 的变量集合。剩余变量在最优分配下一定满足梯度相等的条件，高斯消元求得条件极值点即可。注意舍弃负解。

复杂度 $O(2^n n^3)$。

I. Rolling for Days

做法

- 由数据范围大致可以猜到这是一道状压dp题。
- 称抽到后需要留下来的卡为有效卡，抽到后放回卡池的卡为无效卡。卡池中的无效卡数量只有在抽出一张有效卡后这种卡需求被满足时才会增加，此时这种卡在卡池中多余的那些卡从有效卡转变为了无效卡，无效卡数量共变化 m 次。这启发我们设计一个集合作为动态规划的状态，表示每种卡是否抽够。
- 抽卡次数可能无限，不能考虑每一次抽卡。但假如我们知道了每次抽出有效卡的概率，由于无效卡张数的变动次数很少，我们可以较为方便地对抽有效卡的相关信息进行计算。
- 对于每一个有效卡序列，将产生这个有效卡序列的概率乘上这个有效卡序列期望抽无效卡的张数，就能得到这个有效卡序列对最终答案中期望抽无效卡张数的贡献，再把有效卡序列的长度 $\sum b_i$ 加上就得到了最终答案。

I. Rolling for Days

- 如果抽到一张有效卡后，这种卡的需求被满足，称这张卡为终止有效卡。有效卡序列中一定会有m张终止有效卡。
- 设 $f(i,S)$ 表示确定了有效卡序列的前i位，第i位是一张终止有效卡，满足了集合S中的卡牌种类的需求的所有有效卡序列对最终答案抽无效卡张数的贡献；$g(i,S)$ 表示生成上述有效卡序列的概率。
- 枚举 i 位放的终止有效卡的种类 x ，再枚举上一个终止有效卡的位置 j ，可以列出大致的转移方程如下：

I. Rolling for Days

$$f(i,S)=\sum_{x\in S}\sum_{j=0}^{i-1}(f(j,S-x)+g(j,S-x)\times\sum_{k=j+1}^{i}\frac{\sum_{y\in S-x}a_y-b_y}{n-k+1-\sum_{y\in S-x}a_y-b_y})\times P(A)$$

- **上述式子中，A 表示的是剩下 b_x-1 张有效的第 x 种卡在 i 位置之前这一事件**
- **$\frac{\sum_{y\in S-x}a_y-b_y}{n-k+1-\sum_{y\in S-x}a_y-b_y}$ 是用来表示在有效卡序列的第 k 位抽出无效卡的期望次数。这是由于抽有效卡序列的第k位时卡池中已经被抽走的 $k-1$ 张卡，而此时卡池中无效卡的张数是 $\sum_{y\in S-x}a_y-b_y$ 。**

I. Rolling for Days

- 尝试写转移方程的时候，我们突然发现了一个巨大的问题，那就是让第 x 种卡在 i 位置之前抽够的概率是没法算的。前 i 位填了 $S-x$ 集合中的种类后还剩下 $i-\sum_{y\in S-x} b_y$ 个空位，强制把一个 x 放在 i 位置之后，把 b_x-1 个 x 放在前 $i-1$ 位的空位中有 $\binom{i-1-\sum_{y\in S-x} b_y}{b_x-1}$ 种放法。
- 问题是，有效卡序列的每个位置卡池中有效卡的数量和第 x 种卡的数量都不一样. 也就是说在抽出有效卡的条件下，有效卡中抽出我们想要的第x种卡的概率是不同的，这个概率跟之前已经放下的那些有效卡的位置有关。这样的话，只用 (i,S) 状态是完全不够的。

I. Rolling for Days

- 不妨继续往下写。假设有效卡序列的第 i 个位置及以前第 x 种卡的个数为 $c_{i,x}$ ，则在有效卡序列的第 i 个位置抽出第 x 种卡的概率为 $\frac{a_x-c_{i-1,x}}{n-i+1-\text{位置}i\text{处无效卡数量}}$ 。
- 由于最后这些概率要相乘，我们不妨把分子和分母分别乘起来考虑。不管第 x 种卡放在了哪些位置，分子都一定是 $a_x, a_x-1, \ldots, a_x-b_x+1$ ；而分母的大小只跟具体的位置有关，跟 x 无关。当每种卡牌的需求都被满足时，整个有效卡序列一定被填满了，因此每个位置都会恰好被某种卡选中，都会当一次分母。我们可以在DP的过程中把分母的贡献提前计算。也就是说，枚举 i 位放的终止有效卡的种类 x ，再枚举上一个终止有效卡的位置 j 之后，分子计算第 x 种卡的贡献；分母不计算第 x 种卡的贡献，转而计算区间 $[j+1,i]$ 中位置的贡献，
- 因此我们可以写出完善后的转移方程：

I. Rolling for Days

$$f(i,S)=\sum_{x\in S}\sum_{j=0}^{i-1}(f(j,S-x)+g(j,S-x)\times\sum_{k=j+1}^{i}\frac{\sum_{y\in S-x}a_y-b_y}{n-k+1-\sum_{y\in S-x}a_y-b_y})$$

$$\times\frac{\prod_{k=1}^{b_x}(a_x-k+1)}{\prod_{k=j+1}^{i}(n-k+1-\sum_{y\in S-x}a_y-b_y)}\times\binom{i-1-\sum_{y\in S-x}b_y}{b_x-1}$$

$$g(i,S)=\sum_{x\in S}\sum_{j=0}^{i-1}g(j,S-x)\times\frac{\prod_{k=1}^{b_x}(a_x-k+1)}{\prod_{k=j+1}^{i}(n-k+1-\sum_{y\in S-x}a_y-b_y)}\times\binom{i-1-\sum_{y\in S-x}b_y}{b_x-1}$$

- **完善后的式子中，$f(i,S)$与$g(i,S)$的意义都发生了变化，表示的是提前计算了特定位置抽特定种类卡的概率的分母的贡献后的结果。**
- **预处理调和级数、前缀和优化后可以实现$\Theta(nm2^m)$的总时间复杂度。**

J. Game on a Forest

考虑单棵树的 SG 函数。

结论：对于有奇数个结点的树，其 SG 函数值为 1 。对于有偶数个结点的数，其 SG 函数值为 2 。

因此只需要枚举第一步删除的点或边，维护删除后森林中偶数个结点的树的数量以及奇数个结点的树的数量即可，容易用 $O(n)$ 复杂度的算法实现。

证明可以用到数学归纳法:

- 树的结点个数为 n 。$n = 1, 2$ 时显然成立。在 $n < x$ 成立时，证明 $n = x$ 也成立。

J. Game on a Forest

- 如果 x 是奇数，那么删掉一个结点，得到若干棵子树，他们中的奇子树一定有偶数个，偶子树可以为奇数个或者偶数个，因此可以到达后继状态的 SG 值可以是 $0, 2$，如果删掉的是叶子，那后继状态 SG 为 2 而因为一定至少一个结点有偶数度数，所以一定也存在 SG 为 0 的后继状态。如果删掉一条边，那么两侧子树异奇偶，后继状态 SG 为 3。因此 SG 为 $mex\{0, 2, 3\} = 1$。
- 如果 x 是偶数，那么删掉一个结点，得到若干棵子树，他们中的奇子树一定有奇数个，偶子树可以为奇数个或者偶数个，因此可以到达后继状态的 SG 值可以是 $1, 3$，如果删掉的是叶子，那后继状态 SG 为 1。如果删掉一条边，那么两侧子树同奇偶，后继状态 SG 为 0。因此 SG 为 $mex\{0, 1\} = 2$，或 $mex\{0, 1, 3\} = 2$。

K. Game on a Forest

讲题视频

L. Palm Island

将排列映射之后，可以等价为排序问题。

考虑冒泡排序，该算法执行 n 轮，每轮进行冒泡。我们可以看做，有一个 head 指针来处理冒泡：当 $a[head] < a[head+1]$ 时，指针直接后移；而当 $a[head] > a[head+1]$ 时，先交换两者，指针再后移。

我们直接可以把 a_1 看做指针指着的元素，操作1就可以看做head的后移，而操作2就可以看做交换然后后移。于是就可以实现 $n-1$ 轮的冒泡排序，为了把head重新移回起点，每轮都需要执行 n 步。容易知道 $n-1$ 轮冒泡之后已经有序，因此可以在 n^2-n 次操作内完成，考虑到设置为前期题，放松一点调整为 n^2 次。

M. Painter

- 本题可能有很多种做法，题解是用时光倒流的做法：对于每个最后渲染的像素，枚举这个渲染操作之前的所有画图操作，看影响到了这个像素的最靠后的画图操作是哪个，那么这个像素的颜色就是最后影响到它的画图操作的颜色。这样时间复杂度是$O(nA)$的，其中A是渲染面积总和。
- 本题可能有其他复杂度更优的方法，留给大家有兴趣研究。