杂题选讲2

仓鼠

写在前面

- 讲课人水平不大行，也没做过啥题
- 所以看似杂题选讲，实则原题选讲/简单题选讲
- 一些题可能需要一点点前置技能
- 所以略微归纳了一下，讲部分题的时候会先讲一下前置技能

对称性原理-翻折法

- 对称性原理在OI中最多的应用就是翻折法
- 来看这么一个例子：要找两条从$(1,1)$走到(N, M)的路径，满足这两条路径除了起点和终点不相交，计数方案数
- 先拆成两个起点到终点的路径
- 假设它们两独立，那么方案数就是直接相乘
- 对于相交的情况，在第一次相交的位置翻折，就相当于交换了起点（或者交换了终点），方案数还是直接相乘
- 两个结果相减就是答案

Jiry Matchings

给一棵边权树，$f(i)$ 表示 i 条边的最大权匹配，求出 $f(1), f(2), \ldots, f(n-1)$.
$n \leq 2 \cdot 10^5$，时限 6s.

Jiry Matchings

- 考虑如何做两个凸函数的$\max +$卷积，可以考虑差分后归并
- 再考虑一个经典的树上分治FFT做法，即对轻儿子做分治FFT，对所有重链做分治FFT。把上面那个做法套上去就可以了

小星星

- Source： UOJ#185 ZJOI2016Day1T3

小Y是一个心灵手巧的女孩子，她喜欢手工制作一些小饰品。她有 n 颗小星星，用 m 条彩色的细线串了起来，每条细线连着两颗小星星。有一天她发现，她的饰品被破坏了，很多细线都被拆掉了。这个饰品只剩下了 $n-1$ 条细线，但通过这些细线，这颗小星星还是被串在一起，也就是这些小星星通过这些细线形成了树。

小Y找到了这个饰品的设计图纸，她想知道现在饰品中的小星星对应着原来图纸上的哪些小星星。如果现在饰品中两颗小星星有细线相连，那么要求对应的小星星原来的图纸上也有细线相连。

小Y想知道有多少种可能的对应方式。只有你告诉了她正确的答案，她才会把小饰品做为礼物送给你呢。

- $N \le 17$

小星星

- 从另一个角度看排列：长度为N的每个元素都在$[1, N]$之间的整数序列，要求每个数出现了至少一次
- 对于每个数，出现了至少一次可以看成：没有出现的时候就违反了限制
- 用容斥的做法，枚举违反了的限制集合。具体计数的时候直接树形dp

Ribbons on Tree

- Source：ARC101E
- 给定一颗有偶数个点的树，分成$\frac{N}{2}$对，将每对之间的路径覆盖
- 求有多少种匹配方案满足，树上的任意一条边都被覆盖
- $N \leq 5000$

Ribbons on Tree

- 考虑一个暴力的容斥过程。先钦定一些边一定没有被覆盖，然后计数
- 那么就是在去掉这些边后形成的每个联通块中两两匹配
- 把这个过程写成一个树形dp的过程就可以了
- 设$F[u][x]$表示u的子树中和u相连联通块大小为x的方案数，这里是带上了容斥系数的方案数
- 在把儿子的信息合并上来的时候，是在做一个卷积
- 每次需要决定每条边是否被钦定，如果被钦定有一个-1的系数

Ribbons on Tree

```
DFS(u)
{
        Size[u] = 1
        for (v | v is son of u)
        {
                DFS(v)
                for (x = 0 ~ min(Size[u], K))
                        for (y = 0 ~ min(Size[v], K))
                                //do something
                Size[u] += Size[v]
        }
}
```

普及一下，这个代码的时间复杂度是$O(NK)$，当然$K \le N$

On the Bench

- Source：codeforces840C 经典问题
- 有N种球，第i种球有A_i个，求排列方式，使得相同种类的球不相邻
- 每种球内部是否区分只是答案后面是否有个系数的差别，不用纠结
- $\sum_i A_i \leq 3000$

On the Bench

- 限制是相邻两个球不能相同。当违反限制的时候，可以看成相邻两个球粘在了一起
- 对于第i种球，如果违反了$j(0 \le j < A_i)$个限制，就相当于有j对相邻的球被粘在了一起，方案数是$\binom{A_i-1}{j}$，带上容斥系数就是$(-1)^j$，这时候可以看成有$A_i - j$个球拿出去任意排列
- 用背包dp把对每种球的容斥过程合并到一起即可。时间复杂度是$O((\sum_i A_i)^2)$的
- 也可以用多项式优化做到更好的复杂度

青春猪头少年不会梦到兔女郎学姐

- Source：IOI2019集训队作业by马耀华
- 有N种球，第i种球有A_i个
- 对于一个序列，把它看成首尾相连的。一个序列的权值定义为每个极大相同颜色连续段长度的乘积
- 求所有序列的权值和
- 对998244353取模
- $\sum_i A_i \leq 2 \times 10^5$

青春猪头少年不会梦到兔女郎学姐

- 想象这么一种暴力。假如枚举每种颜色最后分段是什么样的，那么可以直接用前面说的容斥做法统计方案数，乘上这种分段带来的价值加到答案里面去
- 实际上可以把每种暴力的结果合并到一起，丢到后面的容斥的过程里面去
- 先优化暴力的过程，数量为A_i的物品分成n段的贡献和用之前说的插板法计算
- 然后用FFT计算出，每个分成n段的情况在容斥的时候又被分成了m段的方案数

青春猪头少年不会梦到兔女郎学姐

- 直接用分治FFT把容斥的情况合并到一起
- 注意需要处理一个首尾成环的情况，可以考虑加一维表示目前有没有确定开头的颜色
- 对于开头的颜色，需要特殊处理贡献

mythological I

- Source：TCO2013 Round 3A Hard 增强版
- 给定如下不等式组
- $\forall 1 \le i \le n, x_i \le t$
- $\sum_{i=1}^{m} x_i \le S$
- 给定S, t, n, m，求解数
- $S \le 10^{18}$
- $n \le m \le 10^9$
- $t \le 10^9 \quad n \cdot t \le S$
- $m - n \le 10^3$

mythological I

- 假如暴力枚举前n个变量的取值，令它们的和为X，那么后面的变量方案数可以用组合数算出答案就是$\binom{S-X}{m-n}$
- 把$\binom{S-X}{m-n}$展开成一个关于x的$m-n$次多项式$F(x)$
- 那么只要对于每个$0 \le k \le m-n$的k，均计算出$\sum_{\forall 1\le i\le n, x_i\le t}(x_1+x_2+\cdots+x_n)^k$，然后代入到$F(x)$里面即可
- 记一个长度为$m-n+1$的向量G_l，其中$G_{l,k}$表示的就是$\sum_{\forall 1\le i\le l, x_i\le t}(x_1+x_2+\cdots+x_l)^k$，不难发现由$G_a$和$G_b$可以直接$O((m-n)^2)$求出$G_{a+b}$。直接倍增算出$G_n$即可

GYM100958 I

- 求出有多少个字符串二元组(S, T)，满足如下条件：
- S的长度为N
- T的长度为M
- T是S的一个子串
- S的字符集大小为A，你可以理解为，S中每个元素都是1到A的一个正整数
- 请你求出答案，对10^9+7取模
- $1 \le M \le 50 \quad 1 \le N \le 200 \quad M \le N \quad 1 \le A \le 10^9$

GYM100958 I

- 考虑这么一个事情。假如确定了T，如何计算有多少个满足条件的S
- 这个问题可以直接dp，设$F[i]$表示，只考虑S的前i个元素，同时T出现在了$[i-M+1, i]$的部分上，同时这是其第一次在S中出现的方案数
- 转移十分显然，可以考虑容斥，减掉 T 并非第一次出现的方案数。$F[i] = A^{i-M} - \sum_{j=M}^{i-1} F[j] \times P(i, j)$，其中$P(i, j)$表示$T$同时出现了在以$i$和$j$结尾的位置上的方案数

GYM100958 I

- 考虑P(i, j)如何计算
- 当两个对应的串不重叠时，其中间的部分可以任意确定
- 当两个对应的串重叠时，只要满足一个border的限制即可

GYM100958 I

- 可以注意到，两个串的dp是完全相同的，当且仅当这两个串拥有的border集合相同
- 搜出所有可行的border集合，实践证明只有几千种，计算每种border集合对应的串的数量
- 下面展示一下搜border的代码

GYM100958 I

```cpp
vector<int> dfs(__int128 S, int ways)
{
        Ans = Add(Ans, Mult(ways, dp(S)));
        int last = *(border.rbegin());
        vector<int> Ret(N);//Ret[n]表示以当前border集合为前缀的长度为n的串有多少个
        Ret[last] = Add(Ret[last], ways);
        for (int x = last + 1, temp;x < N;++x)
        {
                border.push_back(x), temp = ways;
                for (int i = 0, v;i < border.size() - 1;++i)
                {
                        v = border[i];
                        if (v + v > x && !(S & (static_cast<__int128>(1) << static_cast<__int128>(v + v - x))))
                                goto loop;
                }
                if (last + last <= x)
                        temp = Mult(temp, power[x - (last + last)]);
                temp = Sub(temp, Ret[x]);
                if (temp)
                {
                        vector<int> y = dfs(S | (static_cast<__int128>(1) << static_cast<__int128>(x)), temp);
                        for (int i = 0;i < N;++i)
                                Ret[i] = Add(Ret[i], y[i]);
                }
                loop :
                border.pop_back();
        }
        return Ret;
}
```

Unicyclic Graph Counting

- Source： CODE FESTIVAL 2017 Elimination Tournament Round 3
- 计数有多少个有标号环套树，第i个点的度数为D_i，对大质数取模
- $N \le 300$

Unicyclic Graph Counting

- 假定我们已经得到了一个环，大小为K。下面我们考虑一种新的Prüfer编码方式，不删除环上的点，只删除树上的点。那么树上点u在这个编码中出现次数一定为$D_u - 1$，环上点v在这个编码中出现次数一定为 $D_v - 2$ ，并且序列的最后一个点必须是环上的点。可以发现，当环的形状确定后，这样的编码方式就和换套树一一对应了。编码总长度为$N - K$
- 直接做DP就可以了，记录当前选了多少个点在环上以及序列的最后一个点是否确定，时间复杂度为$O(N^2)$

斯特林反演

- $[m=n]=\sum_{k=m}^{n}(-1)^{n-k}\begin{bmatrix}n\\k\end{bmatrix}\begin{Bmatrix}k\\m\end{Bmatrix}$
- $[m=n]=\sum_{k=m}^{n}(-1)^{n-k}\begin{Bmatrix}n\\k\end{Bmatrix}\begin{bmatrix}k\\m\end{bmatrix}$
- 上面两个东西叫做反转公式，由它们可以直接得到斯特林反演
- $f(n)=\sum_{k=0}^{n}\begin{Bmatrix}n\\k\end{Bmatrix}g(k)\Leftrightarrow g(n)=\sum_{k=0}^{n}(-1)^{n-k}\begin{bmatrix}n\\k\end{bmatrix}f(k)$
- $f(n)=\sum_{k=n}^{+\infty}\begin{Bmatrix}k\\n\end{Bmatrix}g(k)\Leftrightarrow g(n)=\sum_{k=n}^{+\infty}(-1)^{k-n}\begin{bmatrix}k\\n\end{bmatrix}f(k)$

方阵

- Source：2018雅礼集训
- 对于一个$\mathrm{N} \times \mathrm{M}$方阵，每个格子填上$[1, \mathrm{C}]$中的数，求任意两行、两列均不同的方案数
- $\mathrm{N}, \mathrm{M} \leq 5000$

方阵

- 设f(n)表示n行M列方阵，满足任意两列互不相同的方案数
- $f(n) = P_{c^M}^{n}$
- 设g(n)表示n行M列方阵，满足任意两行两列互不相同的方案数
- 考虑组合意义显然有， $f(n) = \sum_{k=0}^{n} \left\{ {n \atop k} \right\} g(k)$。那么用斯特林反演就可以求出答案g(N)了

异或图

- Source： BZOJ4671
- 定义两个点集相同的无向图异或的结果，为边集写成二进制数后异或再转成新边集得到的无向图
- 给定S个点集大小为N的无向图，问有多少个图的集合异或结果为一个联通图
- $N \leq 10$
- $S \leq 60$

异或图

- 设g(n)表示恰好有n个联通块的方案数
- f(n)的直观组合定义不好描述，但是却很好求，直接枚举集合划分，然后要求在不同集合中的点必不联通即可，等价于在不同集合中的点必无连边。直接列出方程然后求解数即可
- 那么用斯特林反演就可以求出答案g(1)了

小C的岛屿

- 来源： IOI2018集训队作业by任轩笛
- 有N座岛屿，初始时没有边。每座岛屿都有一个概率值p_i和一个友好列表A_i。小c站在1号岛屿，依次执行以下操作：
- （1）设现在在岛屿x，有p_x的概率产生一条图中尚未存在的随机无向边，不会产生自环。
- （2）如果此时所有岛屿仍未联通，她会在当前点的友好列表中，等概率随机选择一个，走到那座岛屿上。并把不满意度增加1，然后重复（1）。否则就结束这个过程。
- 求她的期望不满意度，对一个大质数取模
- $n \le 50$

小C的岛屿

- 设$\mathrm{F}(\mathrm{x},\mathrm{m})$表示，当前在x号岛屿，已经加入了m条边，图还未联通，不满意度的期望。答案就是$\mathrm{F}(1,0)$
- 设$\mathrm{G}(\mathrm{m})$表示当前有m条边且原图未联通，加入了一条原图中未存在的边，图联通的概率。那么就是说，有$1-\mathrm{G}(\mathrm{m})$的概率，在当前有m条边且未联通的情况下，再加入一条边原图仍然未联通
- 如果按照m分层，那么$\mathrm{F}(\mathrm{x},\mathrm{m})$能转移的状态都是当前层和后面的层，故可以按照层做分层高斯消元。层数有$\mathrm{O}(\mathrm{N}^2)$层，每一层有$\mathrm{O}(\mathrm{N})$个变量，故消元复杂度为$\mathrm{O}(\mathrm{N}^5)$

小C的岛屿

- 设$A(n,m)$表示n个点m条边无向图的数目，$B(n,m)$表示n个点m条边联通无向图的数目，$A(n,m)=\binom{\binom{n}{2}}{m}$
- $G(m)=\frac{\sum_{x=1}^{n-1}\binom{n-1}{x-1}\sum_{e=0}^{m}B(x,e)B(n-x,m-e)}{\left(A(N,m)-B(N,m)\right)\times(\binom{N}{2}-m)}$
- 分母就是N个点M条边不联通无向图选择一条边加进去的总方案数
- 分子就是N个点M条边不联通无向图选择一条边加进去后变联通的方案数。就是在做一个枚举桥边两边联通块点数和边数的事情
- 问题变为如何快速计算$B(n,m)$

小C的岛屿

- 根据斯特林反演，$\sum_{s=1}^{n}\left\{ {n \atop s} \right\}(-1)^{s-1}(s-1)! = [n=1]$
- 把n看成联通块数，考虑等式左边的组合意义进行计数。将所有点划分成若干个集合，不在同一个集合中的点不连边，在同一个集合中的任意连边，计数方案数。如果集合个数为s，那么容斥系数就是$(-1)^{s-1} \times (s-1)!$
- 方案数的计算可以考虑集合大小为$\{a_1, a_2, \cdots, a_s\}$，方案数为$\binom{\binom{a_1}{2}+\binom{a_2}{2}+\cdots+\binom{a_s}{2}}{m}$。$(s-1)!$可以看成除了1所在以外的$s-1$个联通块任意排列的方案数。所以可以直接dp，设$H[n][m]$表示n个点，可选边的总数为m，可以排列联通块并带上容斥系数的方案数
- 最后枚举1所在联通块大小，时间复杂度$O(N^4)$

AGC028F2

给定一个 N 行 N 列的网格，这个网格有 $N \times N$ 个方格。从上往下数第 i 行且从左往右数第 j 列的方格，可以用一个二元组 (i, j) 来描述它。每个方格要么是空的，要么被一个障碍物所占据。同时，每个空的方格上都写了一个数字。如果 $A_{i,j}$ 是 1 到 9 中某一个自然数，那么 (i, j) 是一个写了 $A_{i,j}$ 的空方格；否则 $A_{i,j} =' \#'$，(i, j) 是一个被障碍物所占据的方格。

我们称一个方格 Y 是一个方格 X 可达的 (或者说 X 可到达 Y)，当且仅当以下三个条件被满足：

(1) X 和 Y 不是同一个方格；

(2) X 和 Y 都是空的；

(3) 存在一条从 X 到 Y 的路径，满足在一个方格处时，接下来只往下或者往右走到一个相邻的空方格。

考虑所有的二元组 (X, Y) 满足 X 可到达 Y，定义一个这样的二元组价值为 X 和 Y 上数字的乘积，请求出所有满足条件二元组价值的和。

$1 \leq N \leq 1500$

$A_{i,j}$ 是一个 1 到 9 中某一个自然数或者是 #。

AGC028F2

对于这样网格图上路径统计相关的问题，很自然可以想到用网格图分治的做法。具体而言，每次考虑一个子矩形内部对答案的贡献，就把这个子矩形分成两部分，分别计算两部分 (也分别是两个子矩形) 对答案的贡献求和，再加上跨越两个子矩形的贡献，就是整个子矩形对答案的贡献了。在接下来的一些描述中，我们会直接忽视那些有障碍物的点。

假设当前考虑的子矩形大小为 $H \times W$，即它有 H 行 W 列。我们假设 $W \leq H$。下面考虑把这个子矩形尽量均分成上下两个部分，设上面的部分为 U、下面的部分为 D，均分意味着 U 的行数是 $H_U = \lceil \frac{H}{2} \rceil$，$D$ 的行数为 $H_D = \lfloor \frac{H}{2} \rfloor$。根据前面说的分治做法，现在考虑计算 $X \in U$、$Y \in D$ 的 (X, Y) 对答案产生的贡献和。下面做出一些定义。

AGC028F2

定义 3.7.1.

$Left(i,j)$ 表示最小的 x，满足 $D(1,x)$ 可以到达 $D(i,j)$。

$Right(i,j)$ 表示最大的 x，满足 $D(1,x)$ 可以到达 $D(i,j)$。

$Top(j)$ 表示 U 中可以到达 $D(1,j)$ 的点中，行标号的最小值，这里的标号指的是在 U 中的标号。

$Bot(j)$ 表示 $D(1,j)$ 可以到达的点中，行标号的最大值。

$Mpoint(a,b)$ 表示 $D(1,a)$ 和 $D(1,b)$ 能同时到达的点中，行标号的最小值。

$Brh(a,b,l)$ 表示 $D(1,a)$ 和 $D(1,b)$ 在 D 的前 l 行中可以同时达到点的点权和。

$Reachable(a)$ 表示 $D(1,a)$ 可以到达点的点权和。

这里的定义中有一些特殊情况。如果不存在这个函数要找的点，当这个函数是求“最大”时定义其值为 $-\infty$，是求“最小”时定义其值为 $+\infty$。

AGC028F2

$\mathrm{Left}(i,j), \mathrm{Right}(i,j), \mathrm{Top}(j)$ 和 $\mathrm{Bot}(j)$ 这四个函数可以直接求。具体而言，可以把网格图按照能够直接到达的关系连有向边，看成一个有向无环图 DAG。在这个 DAG 上做一遍 dp 即可求出这四个函数在每个位置处的值。这个 dp 的时间复杂度是 $O(HW)$ 的。

考虑如何对于 $1 \le a \le b \le W$ 求出 $\mathrm{Mpoint}(a,b)$。首先，所有的 $\mathrm{Mpoint}(a,a) = 1$。对于剩下的情况，考虑所有的 $D(i,j)$ 满足 $\mathrm{Left}(i,j) \le a < b \le \mathrm{Right}(i,j)$，若不存在这样的点则 $\mathrm{Mpoint}(a,b) = +\infty$，否则找到令行标号即 i 最小的点，设其为 $D(p,q)$。下面分情况讨论：

(1)$p > \min\{\mathrm{Bot}(a), \mathrm{Bot}(b)\}$ 时，$\mathrm{Mpoint}(a,b) = +\infty$。

这一点是显然的，因为在这样的条件下，找不到符合条件的点。

AGC028F2

(2)$p \le \min\{\mathrm{Bot}(a), \mathrm{Bot}(b)\}$ 时，$\mathrm{Mpoint}(a, b) = p$。

首先 $\mathrm{Mpoint}(a, b) \ge p$ 是显然的，因为不满足所给条件的 $D(i, j)$ 一定无法被 $D(1, a)$ 和 $D(1, b)$ 同时到达。

下面证明 $\mathrm{Mpoint}(a, b) \le p$。

证明. 考虑从 $D(1, \mathrm{Left}(p, q))$ 走到 $D(p, q)$ 的任一路径 $Path_1$，和 $D(1, \mathrm{Right}(p, q))$ 走到 $D(p, q)$ 的任一路径 $Path_2$，此时有 $\mathrm{Left}(p, q) \le a < b \le \mathrm{Right}(p, q) \le q$。这时候从 $D(1, a)$ 走到第 $\mathrm{Bot}(a)(p \le \mathrm{Bot}(a))$ 行的路径 $Path$，必定会与 $Path_1$ 或 $Path_2$ 相交。具体而言，假设 $Path$ 经过了 $D(p, q)$，那么 $Path$ 与两条路径都有交点 $D(p, q)$；假设 $Path$ 经过了 $D(p, q_1)$ 满足 $q_1 < q$，那么 $Path$ 一定与 $Path_1$ 有交点；假设 $Path$ 经过了 $D(p, q_2)$ 满足 $q_2 > q$，那么 $Path$ 一定与 $Path_2$ 有交点。对于 $Path$ 从相交点开始，变换成与其相交路径后半部分走到 $D(p, q)$ 的部分，即就得到了 $D(1, a)$ 走到 $D(p, q)$ 的路径。同理，可以得到 $D(1, b)$ 走到 $D(p, q)$ 的路径。证毕。

□

综上所述，在这种情况下，$\mathrm{Mpoint}(a, b) = p$ 成立。上述的讨论过程，尤其是上面的证明部分中考虑相交路径的方法，在后文中多次用到，请读者引起注意。

AGC028F2

直接做二维前缀最小值求出每个点对应的 (p, q)，就可以保证求所有 $\text{Mpoint}(a, b)$ 的时间复杂度是 $O(HW + W^2)$ 即 $O(HW)$ 的了。

考虑如何 $\forall 1 \le a \le b \le W, 1 \le l \le H_D$ 快速查询 $\text{Brh}(a, b, l)$。

显然，$\forall l > m = \min\{\text{Bot}(a), \text{Bot}(b)\}$，一定满足 $\text{Brh}(a, b, l) = \text{Brh}(a, b, m)$。所以下面我们只考虑 $l \le m$ 的情况。

$\text{Mpoint}(a, b) > l$ 时，$\text{Brh}(a, b, l) = 0$, 根据这两个函数的定义，这是显然的。

$\text{Mpoint}(a, b) \le l \le m$ 时，就是在求满足 $x \le l$ 且 $\text{Left}(x, y) \le a \le b \le \text{Right}(x, y)$ 的 $D(x, y)$ 点权和，原因和求 $\text{Mpoint}(a, b)$ 的讨论一样，在此不再赘述。这个东西的计算，考虑用容斥的思想，分成四个部分：

(1) 加上 $x \le l$ 且 $\text{Left}(x, y) \le \text{Right}(x, y)$ 的 $D(x, y)$ 点权和；

(2) 减去 $x \le l$ 且 $a < \text{Left}(x, y) \le \text{Right}(x, y)$ 的 $D(x, y)$ 点权和；

(3) 减去 $x \le l$ 且 $\text{Left}(x, y) \le \text{Right}(x, y) < b$ 的 $D(x, y)$ 点权和；

(4) 加上 $x \le l$ 且 $a < \text{Left}(x, y) \le \text{Right}(x, y) < b$ 的 $D(x, y)$ 点权和。

AGC028F2

在从小到大枚举 1 的过程中，可以直接维护出每个 l 对应的 (1) 的结果；对 (2) 和 (3) 的查询用二维前缀和就可以做到 $O(1)$。对于 (4)，注意到满足 $a < \mathrm{Left}(x,y) \le \mathrm{Right}(x,y) < b$ 的 $D(x,y)$ 一定有 $x < \mathrm{Mpoint}(a,b) \le l$，这个性质像前面证明的时候一样直接考虑路径的相交就可以发现，所以就只是算 $a < \mathrm{Left}(x,y) \le \mathrm{Right}(x,y) < b$ 的 $D(x,y)$ 点权和了，条件和 l 无关，那么一开始也用二维前缀和预处理，就可以每次 $O(1)$ 查询了。所以在这些适当的预处理下，可以 $O(1)$ 查询 $\mathrm{Brh}(a,b,l)$。

根据定义可以得到，$\forall 1 \le a \le W$,$\mathrm{Reachable}(a) = \mathrm{Brh}(a,a,H_D)$。

在进行了大量预处理后，就可以进行对问题的求解了。

AGC028F2

考虑用 $O(HW)$ 的 dp 对于每个 $U(i,j)$ 求出 $\mathrm{Min}(i,j)$ 表示最小的 y，满足其可以到达 $D(1,y)$；$\mathrm{Max}(i,j)$ 表示最大的 y，满足其可以到达 $D(1,y)$。忽视掉那些 $\mathrm{Min}(i,j)>\mathrm{Max}(i,j)$ 的点，即 $\mathrm{Max}(i,j)=-\infty$ 且 $\mathrm{Min}(i,j)=+\infty$、$U(i,j)$ 无法到达 D 中点的情况，那么将会有如下两个性质：

(1) 当固定一个 i_0 的时候，随着 j 的从小到大增加，$\mathrm{Min}(i_0,j)$ 和 $\mathrm{Max}(i_0,j)$ 均单调不降。即 $\mathrm{Min}(i_0)$ 和 $\mathrm{Max}(i_0)$ 有非严格单调性。

这里只提供证明的思路不再叙述具体的证明过程：考虑反证，假设一对违反条件的情况，以 $\mathrm{Min}(i_0)$ 的一对逆序对为例，即 $j_1<j_2$ 且 $\mathrm{Min}(i_0,j_1)>\mathrm{Min}(i_0,j_2)$，这时候考虑两条到对应点的路径必定相交，通过交换调整可以得到一条从 $U(i_0,j_1)$ 到 $D(1,\mathrm{Min}(i_0,j_2))$ 的路径，显然与假设和 Min 函数的定义矛盾。按照这个思路同样可以证明 $\mathrm{Max}(i_0)$ 的非严格单调性。

(2)$U(i,j)$ 可以到达 $D(1,y)$，当且仅当 $\mathrm{Min}(i,j)\le y\le \mathrm{Max}(i,j)$ 且 $\mathrm{Top}(y)\le i$。原因和求 $\mathrm{Mpoint}(a,b)$ 的讨论一样，在此不再赘述。

综上可以得到一个显然的做法，枚举 U 的每一行，然后从左到右扫描每一列，根据当前考虑点 $U(i,j)$，加入和删除一些 $D(1,y)$，维护当前考虑点可以到达的所有在 D 中点的点权和。

AGC028F2

现在需要支持的事情是，维护一个类似队列的东西，每次在队列 Q 的后端加入一个 $D(1,y)$ 或是在前面删除一个 $D(1,y)$，同时求 Q 中可以到达点的点权和。定义 $\mathrm{Only}(y)$ 表示：对于一个在 Q 中的 $D(1,y)$，其可以到达但是 Q 中所有列坐标大于 y 的点无法到达点的点权和。那么我们要查询的值就是 Q 中所有 $\mathrm{Only}(y)$ 的和，下面考虑如何维护 $\mathrm{Only}(y)$。前端删除是不会影响 $\mathrm{Only}(y)$ 的，只有后端插入会产生影响。对于后端新插入的 $D(1,y_0)$，有 $\mathrm{Only}(y_0)=\mathrm{Reachable}(y_0)$；同时还有 Q 中的一些位置会发生更改。

AGC028F2

假设 $y' < y'' < y_0$，那么所有 $D(1, y')$ 和 $D(1, y_0)$ 可以同时到达的前 $\min\{\mathrm{Bot}(y'), \mathrm{Bot}(y'')\}$ 行中的点，$D(1, y'')$ 都可以到达，这个性质也是像前面一样考虑路径的相交就可以证明。所以存在 y'' 满足 $y' < y'' < y_0$，同时 $\mathrm{Bot}(y') \le \mathrm{Bot}(y'')$ 的 $\mathrm{Only}(y')$ 是不会发生改变的。所有可能发生改变的位置形成了一个序列 $J_1 < J_2 < \cdots < J_K$，根据前面的观察，它们一定满足 $\mathrm{Bot}(J_1) > \mathrm{Bot}(J_2) > \cdots > \mathrm{Bot}(J_K)$，这实际上就是一个类似单调栈的结构，维护所有成为了严格后缀最大值的位置。对于所有的 $\mathrm{Only}(J_i)$，有可能发生改变，从而变成新的 $\mathrm{Only}'(J_i)$。

对于 J_K，有：

$$\mathrm{Only}'(J_K) = \mathrm{Only}(J_K) - \mathrm{Brh}(J_K, y_0, \min\{\mathrm{Bot}(J_K), \mathrm{Bot}(y_0)\})$$

之后可以做一个向前递推的过程，当找到一个 $\mathrm{Bot}(J_p) > \mathrm{Bot}(y_0)$ 的 p 时，$\forall q < p, \mathrm{Only}'(J_q) = \mathrm{Only}(J_q)$，即都不会发生改变；

对于其它的 $p < K$，考虑简单的容斥可以得到，有：

$$\mathrm{Only}'(J_p) = \mathrm{Only}(J_p) - \mathrm{Brh}(J_p, y_0, \min\{\mathrm{Bot}(J_p), \mathrm{Bot}(y_0)\}) + \mathrm{Brh}(J_p, y_0, \min\{\mathrm{Bot}(J_{p+1}), \mathrm{Bot}(y_0)\})$$

AGC028F2

注意到，当把 $D(1, y_0)$ 加入到 Q 末端时，J 这个序列会有一个后缀被删除，而这些点和删除这些点后 J 的倒数第一个元素的 Only 才会发生改变，所以就像维护单调栈一样维护 J 即可，在维护的过程中顺便做出对 Only 的修改。注意对应到 Q 的前端删除，J 也会发生前端删除，所以具体实现的时候用双端队列来维护 J。

这样，每一行处理的时间复杂度都是 $O(W)$，总的时间复杂度是 $O(HW)$ 的。

综上所述，分治到每个大小为 $H \times W$ 的子矩形，进行处理的时间复杂度都是 $O(HW)$ 的，所以总的时间复杂度就是 $O(N^2 \log N)$ 的，可以通过本题。

CF1148H

给定一个最初为空的数组，需要支持以下操作：

- 给定 a, l, r, k，在数组末尾插入 a，然后查询有多少数对 (i, j) （$l \leq i \leq j \leq r$），满足 $\operatorname{mex}(\{a_i, a_{i+1}, a_{i+2}, \ldots, a_j\}) = k$。

强制在线。

$\operatorname{mex}(S)$ 表示集合 S 中最小的未出现的**自然数**。

$$1 \leq n \leq 2 \cdot 10^5$$

CF1148H

显然权值是O(N)的。考虑从小到大枚举右端点r，同时对于每个左端点l维护mex(l, r)。显然mex(l, r)随着l的减小单调不降，mex(l, r)被分成了连续若干段
考虑新加入一个数A[r + 1]，在以r的信息为基础上进行维护
首先mex(r + 1, r + 1) = mex{A[r + 1]}。对于之前的l，找到一个极大的段[l1, l2]，满足mex(l1, r) = mex(l1 + 1, r) = ... mex(l2, r) = A[r + 1]
(1)这样的段是空集。那么不会产生任何影响。更一般地，所有不在该段中的l均满足mex(l, r + 1) = mex(l, r)，可以不需要考虑它们的改变
(2)这样的段不是空集，那么这一段中的mex都会修改，同时满足单调不降的性质，并且任意A[r + 1] < mex(l1 <= l <= l2, r + 1) <= mex(l1 - 1, r)
如果把(2)中新产生的连续段给找出来的话，扫描完整个序列后，这一过程的总复杂度是O(N)的，因为相当于每次在数轴上删除一个点后加入若干点
用线段树维护每个权值的next（最近一次出现的位置），然后在线段树上DFS即可找出这些段
同时，为了维护答案，还需要对每个权值开一颗可持久化线段树来维护历史和