Calculus 题解

注意到题中所给的所有函数均为发散。所以只需要检查是否所有的构成函数的系数均为 0 即可。

Kanade Loves Maze Designing 题解

注意到允许一个 $\mathcal{O}(n^2)$ 的算法，考虑用 +-1 法统计每种颜色即可。即，记 c_i 为颜色 i 的出现次数，r 为从某点处开始 DFS，到点 i 得到的答案，进入该点时，给 c_i 加一，如果这种颜色第一次出现，则给 r 加一，记录答案，退出该点时，如果给 c_i 减一后这种颜色不会出现，则 r 减一。

时间复杂度：$\mathcal{O}(n^2)$，空间复杂度：$\mathcal{O}(n)$。

关键词：DFS

Cycle Binary 题解

对于一个字符串 S，如果我们将其表示为如题目所述的 $kP+P'$ 的形式，则称 P 为 S 的一个循环节，P 的长度为 S 的一个周期。

题目要求我们对所有长度为 n 的 01 串，按其最小周期分类计算贡献。设 $f(x)$ 表示长度为 n，**最小**周期为 x 的 01 串的个数，则我们要求的是（设为 $g(n)$）：

$$g(n)=\sum_{x=1}^{n}\left\lfloor\frac{n}{x}\right\rfloor f(x) \tag{C.1}$$

首先考虑如何求 $f(x)$。长度为 x 的 01 串一共有 2^x 个，其中，存在周期 y 满足 $y<x$ 且 $y|x$ 的串显然不能计算在 $f(x)$ 内，因为其有更小的周期 y。我们很自然地作出以下猜想：

- $f(x)$ 等于长度为 x 的 01 串中，所有不存在周期 y 满足 $y<x$ 且 $y|x$ 的串的个数。

为了验证上述猜想，我们需要用到 **周期引理（Periodicity Lemma）**：

- 设字符串 S 有周期 x 和 y，满足 $x+y\le|S|+\gcd(x,y)$，那么 $\gcd(x,y)$ 也是 S 的一个周期。

设有长度为 x 的 01 串 T 和长度为 n 的 01 串 $S=kP+P'$，T 不存在周期 y 满足 $y<x$ 且 $y|x$。假设 S 存在一个比 x 更小的周期 y，由周期引理：

1. 如果 $x+y\le n+\gcd(x,y)$，则 $\gcd(x,y)$ 也是 S 的周期。同时 $\gcd(x,y)|x$，$\gcd(x,y)$ 也是 T 的一个周期，这与 T 不存在周期 y 满足 $y<x$ 且 $y|x$ 相矛盾。故假设不成立，x 就是 S 的最小周期，我们的猜想正确，并得到 $f(x)$ 的计算公式：

$$f(x)=2^x-\sum_{y<x,y|x}f(y) \tag{C.2}$$

2. 如果 $x+y>n+\gcd(x,y)$，我们无法通过周期引理来证明之前的猜想。

在情况 1 中，x 满足的条件等价于 $x\le\lfloor n/2\rfloor+1$。对于情况 2，观察式 $(\mathrm{C}.1)$ 可以发现，$x>\lfloor n/2\rfloor$ 的 $f(x)$ 系数都是 1，于是我们不需要分别计算这些 $f(x)$，用总串数 2^n 减去 $x\le\lfloor n/2\rfloor$ 的 $f(x)$ 之和即可。式 $(\mathrm{C}.1)$ 可化为：

$$
\begin{aligned}
g(n) &= \sum_{x=1}^{\lfloor n/2 \rfloor} \left\lfloor \frac{n}{x} \right\rfloor f(x) + 2^n - \sum_{x=1}^{\lfloor n/2 \rfloor} f(x) \\
&= 2^n + \sum_{x=1}^{\lfloor n/2 \rfloor} \left(\left\lfloor \frac{n}{x} \right\rfloor - 1 \right) f(x)
\end{aligned} \tag{C.3}
$$

并将 $f(x)$ 的计算公式 $(\mathrm{C}.2)$ 移项得：

$$
2^x = \sum_{y|x} f(y) = f * 1 \tag{C.4}
$$

对式 $(\mathrm{C}.4)$ 使用杜教筛，对式 $(\mathrm{C}.3)$ 使用数论分块（注意杜教筛求出的约 $2\sqrt{n}$ 个值就是数论分块所需的端点处的前缀和），即可快速计算出 $g(n)$。其中杜教筛需要预处理 $x \le n^{2/3}$ 的 $f(x)$，我们根据 $f(x)$ 的计算公式 $(\mathrm{C}.2)$，使用类似埃氏筛的方法，枚举倍数进行贡献即可。

时间复杂度 $O(n^{2/3} \ln n^{2/3} + \sum_T n^{2/3})$，空间复杂度 $O(n^{2/3})$。

Display Substring 题解

二分答案，然后使用任意一种后缀数据结构 check 即可。

如何 check?

- 后缀数组：所有后缀的所有前缀即所有子串。我们遍历后缀数组，对于每个后缀，其越长的前缀能耗越大，于是可以二分找到能耗小于等于要 check 的值的前缀的个数，再减去重复部分即可。而每个后缀被重复统计的部分就是 `height` 数组对应的值。
- 后缀自动机/后缀树：这两种后缀数据结构都是将本质不同的子串记录在其结点上。每个结点表示的子串都是形如 $\mathrm{Suffix}(T, i) + S$ 的串（即取 T 的一个后缀和 S 连接构成的串，在后缀树上则是再翻转一次的串）。显然我们取的 T 的后缀越长，其表示的子串能耗越大，于是可以二分这个长度来找到满足条件的子串的个数，每次 check 对每个结点做一次二分即可。

时间复杂度 $O(n \log n \log k)$。

Didn't I Say to Make My Abilities Average in the Next Life?! 题解

考虑对 $[1, n]$ 范围内的询问，实际上所求为

$$
\left[\sum_{1 \le l \le r \le n} \frac{\max a_{l \ldots r} + \min a_{l \ldots r}}{2} \Big/ \frac{n(n+1)}{2} \right] \pmod{10^9 + 7}
$$

这是一个经典的笛卡尔树或单调栈问题。分别考虑最大值与最小值的贡献，可以两次使用对应方法以 $\mathcal{O}(n)$ 的时间复杂度求出。

具体说来，我们可以尝试对每个 i 统计最大的区间 $[p, q]$ $(p \le i \le q)$，满足 $\max a_{p \ldots q} = a_i$。那么 a_i 作为最大值的贡献就是 $a_i(i - p + 1)(q - i + 1)$。

加入多组询问后，可以考虑将询问离线，利用单调栈解决。首先按询问右端点排序，从小到大枚举右端点处理每个询问。对于每个左端点维护当右端点是枚举点时询问的答案。

在单调栈的加入过程中，可能会有从单调栈中删除一些点的操作。我们会发现这种操作实际上是对于单调栈中这个点到单调栈下一个点之间的答案依次增加一个等差数列。然后再把最大值设置成新的值。

这个过程可以通过维护线段树的区间加和区间推平实现。这样我们就处理了所有右端点小于当前枚举点的区间贡献。

而删除完之后，对于每个左端点来说，其对询问的贡献取决于其右值最大的点。这也可以用线段树维护。

请注意程序实现问题。

时间复杂度：$\mathcal{O}((n+m)\log n)$，空间复杂度：$\mathcal{O}(n+m)$。

同时存在一种算法，在计算最大值或最小值的贡献时，按从左到右和从右到左的顺序分别计算两遍最值最大范围，根据贡献为 $a_i(i-p+1)(q-i+1)$ 进行统计即可。时间复杂度为 $\mathcal{O}(n\log n+m)$。

关键词：离线算法，线段树

Directed Minimum Spanning Tree 题解

假设输入图是强连通的，Edmond's Algorithm 的收缩阶段如下。

1. 对于每个点，选择一个指向它的权重最小的弧。
2. 设C是由选定的弧形成的任意一个环。对于属于C的每条弧(u,v)，从指向v的所有弧的权重中减去$w(u,v)$。
3. 将C收缩到一个顶点c，并删除自环。当该图仅包含一个顶点，则终止。

根据上述过程建立一棵树T。假设我们正在收缩一个环C，如果弧(u,v)属于C，则将一个从c到v的弧以$w(u,v)$的权重连接在T中。我们可以证明，任何根在r的DMST的权重等于所有环收缩的权重之和减去从T的根到叶子r的路径上的所有弧的权重之和。对于一个 case，总的时间复杂度将是$O(m\log n)$。

Increasing Subsequence 题解

定义 `dp[i]` 为 `a[1...i]` 中以 `a[i]` 为结尾的极长上升子序列个数。则 `dp[i]` 对 `dp[j]` 有贡献当且仅当 `i` 到 `j` 之间没有 `a[i]` 到 `a[j]` 之间的元素。`dp[i]` 初值为 `1` 当且仅当 `a[i]` 前面没有比 `a[i]` 小的。`dp[i]` 对答案有贡献当且仅当 `a[i]` 后面没有比 `a[i]` 大的。这样就有了一个朴素的$O(n^2)$做法。

通过分治并对两侧分别维护单调栈，我们可以把时间复杂度优化到$O(n\log^2 n)$。

当处理区间 `[l,r]` 时，令 `m` 为 `[l,r]` 的中点，对 `a[l,r]` 中所有元素按值依次从小到大处理。对 `[l,m]` 和 `[m+1,r]` 分别建单调栈。左边单调栈中的元素值从小到大，位置从大到小。右边的单调栈中元素值从小到大，位置从小到大。处理左边的元素时直接往单调栈里丢，处理右边的元素 `a[i]` 时通过单调栈找到 `[m+1,i-1]` 中比他小的最靠右侧的元素 `a[j]`。然后在左侧的单调栈中找到比 `a[j]` 大的第一个元素和比 `a[i]` 小的最后一个元素。区间 `[l,m]` 对 `dp[i]` 的贡献就来自单调栈里这两个元素之间的元素的 `dp` 值之和。

Lawn of the Dead 题解

考虑所有点的个数减去不能到达的点的个数，即为可以到达的点的个数。

根据题意，有地雷的地方是不可以到达的。由于僵尸只会向右和向下走，当某个点的左边和上方都不可达时，该点不可达，并会对自己右边的点和下方的点造成影响。

由于空间很大但地雷数有限，可以从上往下逐行对每一行的地雷排序后进行处理。对每个地雷，找到从自己的右上角点 $(x-1,y+1)$ 开始的从左往右的连续不可达区域的范围，那么 x 这行的这个范围也不可达。可以用线段树来实现区间查询和区间覆盖。每一行处理完后查询该行不可达的点数，累加后用总点数减即得到答案。

License Plate Recognition 题解

将车牌垂直投影到一维空间，比如将有 `#` 的列记为 1，没有的为 0，然后寻找每个字符对应的左右边界即可。需要注意的是，根据图像特征，所有的数字和英文字母的投影结果一定是一段连续的区间，但是汉字不一定，如"川"、"鄂"。解决方法为从右向左先找数字和英文字母的对应区间，剩下的即为汉字。

Pony Running 题解

有环概率 dp 典型题，使用经典求法高斯消元。

首先对于每一个点都可以写成一个线性方程，
$dp(i,j)=p_{ij0}dp(i-1,j)+p_{ij1}dp(i+1,j)+p_{ij2}dp(i,j-1)+p_{ij3}dp(i,j+1)+1$，对于格子外的点 $dp(i,j)=0$。

这样可以列出 nm 个线性方程，设 dp 值为矩阵 x，其系数构成矩阵 A，常数构成矩阵 B，即求 $Ax=B$。

首先可以用高斯消元求出第一个 A^{-1}，时间复杂度 $O((nm)^3)$

查询操作要求 x 矩阵之和，即求 $x=A^{-1}B$，修改操作维护 A^{-1} 即可。

1. 修改操作考察伍德伯里矩阵恒等式
   $(A+UCV)^{-1}=A^{-1}-A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}$ 维护 A^{-1}，时间复杂度 $O((nm)^2)$

   维护细节：更改一个点的四个概率转换为矩阵上即更改四个点的值。分解成四次修改矩阵 A 的操作。

   例如，给矩阵 A 的第 i 行，第 j 列加 Δ，若 A_{3*3}，$(i,j)=(2,3)$

   $$UCV=\begin{bmatrix}0&0&0\\0&0&\Delta\\0&0&0\end{bmatrix}$$

   则有，

   $$U_{n\times 1}=[0\quad 1\quad 0]^T\qquad U_i=1$$
   $$C_{1\times 1}=[\Delta]$$
   $$V_{1\times n}=[0\quad 0\quad 1]\qquad V_j=1$$

   而

   $$A^{-1}U=A^{-1}(:,i)$$
   $$VA^{-1}=A^{-1}(j,:)$$
   $$(C^{-1}+VA^{-1}U)^{-1}=(\frac{1}{\Delta}+A^{-1}(j,i))^{-1}=\frac{1}{1/\Delta+A^{-1}(j,i)}$$

   因此时间复杂度为 $O((nm)^2)$
2. 查询操作，求每个个点走出方格步数的期望之和，做一次矩阵乘法求 $x=A^{-1}B$ 即可，$O((nm)^2)$

总时间复杂度 $O((nm)^3+q(nm)^2)$

另外由于是网格图，可以使用直接消元法或主元法通过本题，详见《浅谈图模型上的随机游走问题》by 王修涵

Travel on Tree 题解

题目要求一个区间内的所有点对间的路径构成的“子树”的边权和，多次询问。

简单分析可以发现本题难以进行区间合并，于是考虑使用莫队算法。

问题转化为计算在一个的“子树”中加入（或删除）一个结点对总边权的贡献。设当前区间为 $[l,r]$，S 为结点 l 到 r 构成的集合，T' 为 S 内的所有点对间的路径构成的“子树”。设现在要加入的结点为 u。如果我们固定原树的根，并设 S 中所有结点的 LCA 是 v，则加入结点 u 的贡献可以分为两种情况：

1. $\mathrm{lca}(u,v)=v$，则我们需要找到从 u 到根，第一个在 T' 中出现的结点 w，此时贡献为 $\mathrm{dep}(u)-\mathrm{dep}(w)$。
2. $\mathrm{lca}(u,v)=w\neq v$，此时贡献为 $\mathrm{dep}(u)+\mathrm{dep}(v)-2\cdot\mathrm{dep}(w)$。

在情况 1 中，我们需要快速找到结点 w。考虑 DFS 序的性质，如果我们将 S 中的结点按 DFS 序排列，则只需找到结点 u 在 S 中的前驱（从大到小第一个 DFS 序小于 u 的）结点 x 和后继（从小到大第一个 DFS 序大于 u 的）结点 y。则要找的结点 w 为 $\mathrm{lca}(u,x)$ 和 $\mathrm{lca}(u,y)$ 中深度较大的一个。x,y 不存在时需特判。

但这样我们在维护 S 时不免要使用一棵树（线段树或平衡树），时间复杂为 $O(n\sqrt{n}\log n)$，仍不能接受。通过对删除操作的分析，可以发现如果使用链表维护 S 中的结点，则对于删除操作，计算贡献可以 $O(1)$ 完成，并且支持 $O(1)$ 撤回。于是可以考虑仅进行删除和撤回操作，使用回滚莫队解决本题。对于 LCA 的查询，我们需要使用查询复杂度为 $O(1)$ 的 RMQ-LCA 算法。

综上，本题时间复杂度为 $O(n\sqrt{n})$。std 的写法是手写链表和开栈记录要回滚的总边权，比赛环境下耗时 4500 ms 左右，非比赛环境下 12000 ms 左右。如果在代码中使用了 `std::list`，或在回滚时重新计算了一次贡献，则可能收获两倍甚至更大的常数，这在比赛中可以通过，但赛后可能无法通过。