IOI2016 中国国家集训队第一次作业

中山纪念中学 杨家齐

2016 年 1 月 22 日

目录

第一章　试题泛做 (1)

1.1 Annual Parade

1.1.1 问题描述

给定一个 n 个点, m 条边的有向带权图, 你需要用一些路径来覆盖其中的点: 对于每一条路径, 你需要支付的代价为路径上的边的权值和; 对于每一条不是环的路径, 你需要额外支付 c 元; 对于每一个没有被覆盖的点, 你也需要额外支付 c 元.

有 k 组询问, 每一组询问会给定当前的 c 值, 然后你需要回答最小的代价.

1.1.2 数据规模和约定

- $2 \leq n \leq 250$
- $1 \leq m \leq 30000$
- $1 \leq k \leq 10000$
- 图中没有自环
- 边权与 c 值都小于等于 10000

时间限制: 4s.

1.1.3 分析

我们尝试用网络流来解决这一问题: 先把每个点 i 拆成 i 和 $i\prime$, 然后对于每一对点 i 和 j, 我们在 i 和 $j\prime$ 之间连一条以最短路为边权的边.

然后我们可以发现, 每一次增广, 我们都会使得总费用减少 c 并增加本次增广的权值. 由于在 Successive SAP 算法中, 增广路的权值是单调递增的, 因此, 我们可以只做一次费用流, 然后把每次增广所得到的权值都存下来. 然后对于每一个 c, 我们二分出最佳的不增广的点, 然后就可以得到答案了.

时间复杂度: $O(\mathrm{MaxFlow}(n, n^2) + k \log n)$.

1.2 Two Companies

1.2.1 问题描述

给定一棵 n 个结点的树, 其中有 k 条路径, 第 i 条路径是从 $x[i]$ 到 $y[i]$ 的, 并且选择第 i 条路径可以获得一个收益 $z[i]$. 所有的路径被分成了两组, 你选择的路径中, 不在同一个组的路径是不能相交的. 求可能的最大收益.

1.2.2 数据规模和约定

- $1 \leq n \leq 10^5$
- 每组的路径数 m_1, m_2 不超过 700

时间限制: 2s.

1.2.3 分析

首先, 如果我们把路径看作点, 然后在不能够同时选的路径间连一条边的话, 那么我们可以发现, 这个新图是个二分图, 并且它的最大点权独立集就是答案.

那么我们考虑怎么构这个二分图: 我们可以直接 m^2 枚举, 然后我们主要的问题就是如何快速判断两条路径是否相交. 不妨设 $c_i = \mathrm{LCA}(a_i, b_i)$, 我们有一个简便的方法: 如果路径 (a_1, b_1) 和 (a_2, b_2) 相交, 那么 $c_1 \in (a_2, b_2) \vee c_2 \in (a_1, b_1)$. 这个结论比较显然.

时间复杂度: $O(m^2 \log n + \mathrm{MaxFlow}(m, m^2))$.

1.3 Sereja and Arcs

1.3.1 问题描述

数轴上等距分布着 n 个点: $(1, 0)$ 到 $(n, 0)$, 并且点 $(i, 0)$ 的颜色为 $a[i]$.

现在我们对于每一对颜色相同的点, 都连一条圆心在 x 轴上的圆弧. 问有多少对不同颜色的圆弧相交?

1.3.2 数据规模和约定

$1 \leq n, a[i] \leq 10^5$

时间限制: 5s.

1.3.3　分析

直接做并不好做. 那么我们不妨考虑容斥, 我们可以用总数减去不相交的点对数来得到答案. 而不相交的点对有两种情况, AABB 和 ABBA. 其中相同的字母代表相同的颜色.

AABB 的情况

这种情况的数量为

$$\sum_{i=1}^{n}\sum_{j\neq a[i]}\binom{cnt[i][j]}{2}$$

其中 $cnt[i][j]$ 表示 i 之前颜色为 j 的点的数量. 显然, 我们可以简单地扫一遍来统计答案.

ABBA 的情况

这种情况我们可以使用三种复杂度不同的算法来统计答案.

以 A 为基础统计 B　此时我们已经确定了 A 是什么, 想要统计满足条件的 B 的数量.

我们设 $left[i]$ 表示 i 左边的 A 的数量, $right[i]$ 表示 i 右边的 A 的数量, 那么答案就等于

$$\sum_{i<j\wedge a[i],a[j]\neq A} left[i]\times right[j]$$

这可以通过在扫一遍的同时维护 $left[i]$ 的前缀和来 $O(n)$ 得到.

以 B 为基础统计 A　不妨设总共有 k 个 A, 其中第 i 个 A 左边有 $sum[i]$ 个 B.

那么答案就等于

$$\begin{aligned}
\sum_{i<j}\binom{sum[j]-sum[i]}{2} &= \sum_{i=1}^{k}\sum_{j=i+1}^{k}\binom{sum[j]-sum[i]}{2} \\
&= \frac{1}{2}\sum_{i=1}^{k}\sum_{j=i+1}^{k}\left((sum[j]-sum[i])^2-(sum[j]-sum[i])\right) \\
&= \frac{1}{2}\sum_{i=1}^{k}\sum_{j=i+1}^{k}\left(sum[j]^2-2sum[i]sum[j]+sum[i]^2-(sum[j]-sum[i])\right) \\
&= \frac{1}{2}\sum_{i=1}^{k}(k-1)sum[i]^2-\sum_{i<j}sum[i]sum[j]-\frac{1}{2}\sum_{i<j}(sum[j]-sum[i]) \\
&= \frac{1}{2}\sum_{i=1}^{k}(k-1)sum[i]^2-\frac{1}{2}\left(\left(\sum_{i=1}^{k}sum[i]\right)^2-\sum_{i=1}^{k}sum[i]^2\right)-\frac{1}{2}\sum_{i<j}(sum[j]-sum[i]) \\
&= \frac{k}{2}\sum_{i=1}^{k}sum[i]^2-\frac{1}{2}\sum_{i=1}^{k}sum[i]^2-\frac{1}{2}\sum_{i<j}(sum[j]-sum[i])
\end{aligned}$$

于是直接分类统计即可.

时间复杂度: $O(n)$.

直接统计　此时我们可以将圆弧看作是二维平面的点, 然后直接上扫描线就可以了.

时间复杂度: $O(k\log k)$, 其中 k 为 (平面) 点数.

平衡规划　这里我们可以采用平衡规划的思想, 把三种算法结合在一起, 从而得到一个复杂度较优的方案.

我们把点的个数大于 k 的颜色称为大颜色, 其余的称之为小颜色.

那么对于 A 和 B 中有至少一个大颜色的情况, 我们可以利用上面的前两种算法来统计答案, 注意去重. 由于大颜色的数量不会超过 $\frac{n}{k}$, 因此, 这一部分的复杂度是 $O(\frac{n^2}{k})$.

而对于 A 和 B 都是小颜色的情况, 此时颜色对数不超过 $O(nk)$, 因此这一部分的复杂度为 $O(nk\log n)$.

分析可得, 当 $k=\frac{\sqrt{n}}{\log n}$ 的时候, 复杂度最优, 为 $O(n\sqrt{n\log n})$. 题解给出的阈值是 $k=300$.

1.4　Card Shuffle

1.4.1　问题描述

有 n 张牌, 初始的排列顺序为 $1,2,\ldots,n$, 之后有 m 次操作, 第 i 次操作给定三个参数 a_i,b_i,c_i, 然后你需要:

1. 从牌堆顶端拿走 a_i 张牌;
2. 再从牌堆顶端拿走 b_i 张牌;
3. 将第一步拿走的 a_i 张牌放回到剩下的牌堆上面;
4. 从牌堆顶拿走 c_i 张牌;
5. 将第二步你拿起的 b_i 张牌一张一张放到牌堆顶;
6. 最后，将剩下的 c_i 张牌放回到牌堆顶.

1.4.2　数据规模和约定

- $1 \leq n, m \leq 100000$
- 保证数据合法

时间限制: 6s.

1.4.3　分析

用 Splay 直接模拟就可以了.

1.5　Sereja and Equality

1.5.1　问题描述

定义两个长度为 n 序列 a 和 b 相似当且仅当它们离散化之后对应相同的序列, 并且用符号 $\sim$ 来表示.

定义两个长度为 n 序列的相似程度 $f(a, b)$ 为

$$\sum_{1 \leq l \leq r \leq n} (a[l \ldots r] \sim b[l \ldots r]) \wedge (\text{count}(a) \leq m)$$

其中 $\text{count}(a)$ 表示 a 的逆序对数.

现在你需要求出

$$\left(\sum_{a,b} f(a, b) \right) \bmod (10^9 + 7)$$

其中 a 和 b 取遍所有前 n 个自然数所构成的排列.

1.5.2　数据规模和约定

- 多组数据 (至多 10000 组)
- $1 \le n \le 500$
- $1 \le m \le 10^6$

时间限制: 1s.

1.5.3　分析

不妨设 $S(n,k)$ 表示从 n 个元素中选 k 个所得到的所有排列所构成的集合. 那么, 我们要求的式子可以化简为

$$\begin{aligned}
\sum_{a,b\in S(n,n)} f(a,b) &= \sum_{a,b\in S(n,n)} \sum_{1\le l\le r\le n} (a[l\dots r] \sim b[l\dots r]) \wedge (\text{count}(a) \le m) \\
&= \sum_{1\le l\le r\le n} \sum_{a\in S(n,r-l+1)} (\text{count}(a) \le m) \left(\sum_{b\in S(n,n)} (a \sim b[l\dots r]) \right)^2 \\
&= \sum_{len=1}^{n} \sum_{a\in S(n,len)} (\text{count}(a) \le m) \left(\sum_{b\in S(n,n)} \sum_{l=1}^{n-len+1} (a \sim b[l\dots l+len-1]) \right)^2 \\
&= \sum_{len=1}^{n} \sum_{a\in S(n,len)} (\text{count}(a) \le m)(n-len+1) \left(\binom{n}{len} (n-len)! \right)^2 \\
&= \sum_{len=1}^{n} (n-len+1) \sum_{a\in S(n,len)} (\text{count}(a) \le m) \left(\frac{n!}{len!} \right)^2
\end{aligned}$$

这么做的本质是交换了求和顺序, 也就是说从先枚举排列, 再计算相似度变成了先枚举相似部分, 再计算有多少对排列包含了这个相似部分.

因此, 如果我们设 $f[i][j]$ 表示 i 个元素所构成的逆序对数小于等于 j 的排列数, 那么, 我们就可以将上式写成

$$\sum_{len=1}^{n} (n-len+1) f[len][m] \left(\frac{n!}{len!} \right)^2$$

从而我们可以 $O(n)$ 得到答案.

而数组 f 显然可以通过一个 $O(n^3)$ 的 DP 预处理出来.

时间复杂度: $O(n^3 + tn)$.

1.6 Game of Numbers

1.6.1 问题描述

给定两个长度为 n 的数组 $a[]$ 和 $b[]$, 每一次我们要进行一个操作: 选出两个数对 (i,j) 和 (x,y), 使得

1. $a[i] < b[j]$
2. $a[x] > b[y]$
3. $\gcd(a[i], b[j], a[x], b[y]) \neq 1$

1.6.2 数据规模和约定

- 多组数据 (不超过 10 组)
- $1 \leq n \leq 400$
- $1 \leq a[i], b[i] \leq 10^9$

1.6.3 分析

首先, 我们可以把所有的 (i,j) 按照 $a[i]$ 与 $b[j]$ 的大小关系分成两个集合 X 和 Y, 然后连边做最大匹配. 但是这样的话边数是 $O(n^4)$, 无法承受.

我们可以考虑添加辅助点来辅助连边. 不妨对于所有的 gcd 的质因子, 我们都建一个点来表示这个质因数. 然后所有 X 集合中有这个质因数的点连到它, 再让它连到 Y 中所有有这个质因数的点. 这样就大大减少了边数.

时间复杂度: $O(\text{MaxFlow}(n^2, n^2 \log n))$.

第二章　试题泛做 (2)

2.1 Observing the Tree

2.1.1 问题描述

给定一棵 n 个结点的带点权有根树. 接下来你要处理 m 个询问, 每个询问形如:

- c x y a b: 将 x 到 y 路径上第 k 个结点的权值加上 $a+(k-1)b$
- q x y: 询问 x 到 y 的路径上所有点的权值和
- l x: 将所有结点的权值还原到第 x 次修改后的状态

2.1.2 数据规模和约定

- $1 \leq n, m \leq 10^5$
- $1 \leq x, y \leq n$
- 强制在线

2.1.3 分析

这题是 FIBTREE 的简化版, 可以用和 FIBTREE 类似的方法做.
时间复杂度: $O(n \log^2 n)$.

2.2 Fibonacci Numbers on Tree

2.2.1 问题描述

给定一棵 n 个结点的带点权有根树. 接下来你要处理 m 个询问, 每个询问形如:

- A x y: 将 x 到 y 路径上第 k 个结点的权值加上 $Fib[k]$, 其中 $Fib[k]$ 表示第 k 个斐波那契数.

- QS x y: 询问以 x 为根时 y 的子树中的所有点的权值和
- QC x y: 询问 x 到 y 的路径上所有点的权值和
- R x: 将所有结点的权值还原到第 x 个询问后的状态

答案对 10^9+9 取模.

2.2.2 数据规模和约定

- $1 \le n, m \le 10^5$
- $1 \le x, y \le n$
- 强制在线

2.2.3 分析

我们暂时先不考虑 R 操作.

由于

$$Fib[n] = \frac{1}{\sqrt{5}}((\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n)$$

因此, 我们不妨在 $\bmod 10^9+9$ 意义下求出 $\sqrt{5}$, 然后就可以把 $Fib[n]$ 表示为 $\gamma(\alpha^n - \beta^n)$ 的形式.

对于原问题中的每个 A 操作, 我们可以利用树链剖分把要操作的路径分成许多连续的段, 然后在线段树中将一段区间中的第 i 个数字加上 $k\alpha^i$, 注意到这个标记是可加的, 可下传的, 并且我们可以 $O(1)$ 算出一个标记对区间和的影响, 因此我们可以通过线段树的标记来实现这种修改操作.

对于 QC 询问, 我们可以和上面类似地, 利用树链剖分来做.

而对于 QS 询问, 首先留意到我们可以不妨以 1 为整棵树的根, 那么以任意的 x 为根时, y 的子树要么是以 1 为根时的子树, 要么是它的补集并上 y 自己, 因此原问题中的 x 是无关紧要的.

又由于这涉及到子树和, 所以我们需要一个 DFS 序, 那么我们不妨把上面的树链剖分和 DFS 序结合起来, 构造一个每条重链和每棵子树都是序列中连续一段的序, 而这是非常容易实现的.

现在考虑 R 操作, 那么我们只需要简单地把线段树可持久化就可可以了.

时间复杂度: $O(n + m\log^2 n)$.

2.3 Rectangle Query

2.3.1 问题描述

给定一个笛卡尔平面, 你需要维护 q 个操作

- I x_1 y_1 x_2 y_2: 插入一个左下角坐标为 (x_1, y_1), 右上角坐标为 (x_2, y_2) 的矩形;
- D k: 删除插入的第 k 个矩形;
- Q x_1 y_1 x_2 y_2: 询问插入的矩形中有多少个和矩形 $(x_1, y_1), (x_2, y_2)$ 有至少一个公共点.

2.3.2 数据规模和约定

- $1 \leq q \leq 10^5$
- $1 \leq x, y \leq 10^9$
- 保证数据合法

时间限制: 2s.

2.3.3 分析

首先注意到, 二维相交的充要条件是在两个一维上分别相交. 并且我们可以把删除一个矩形看作是加入一个权值为-1 的矩形.

但即使做了这样的转化, 二维相交仍然是一个非常难以处理的问题.

正难则反, 我们不妨考虑求与一个矩形不相交的矩形的个数. 我们可以形象地考虑这个问题, 如果我们把当前询问的矩形画在平面上, 那么我们实际上就是要求它外面那一圈里面的矩形的个数.

我们来考虑怎么计算这个个数, 实际上不相交的个数就等于左边, 右边, 上面, 下面的矩形个数, 减去左上, 左下, 右上, 右下的矩形的个数. 下面称我们加上的为第一类询问, 减去的为第二类询问.

以求左边的矩形个数为例, 来介绍一下第一类询问的处理方法. 此时, 我们所插入的一个矩形 $(x_1, y_1), (x_2, y_2)$ 可以抽象为一个一维的, 坐标为 x_2 的点, 然后对于询问时一个矩形 $(x_1, y_1), (x_2, y_2)$, 询问就变成了求区间 $[1, x_1 - 1]$ 中的点的个数.

以求左上角的矩形个数为例, 来介绍一下第二类询问的处理方法. 此时, 我们所插入的一个矩形 $(x_1, y_1), (x_2, y_2)$ 可以抽象为一个坐标为 (x_2, y_1) 的点, 然后对于询问时的一个矩形 $(x_1, y_1), (x_2, y_2)$, 询问就变成了求横坐标在 $[1, x_1 - 1]$ 中, 纵坐标在 $[y_2, \infty)$ 中的点的个数. 这是一个带插入, 删除的二维询问问题, 我们可以用 CDQ 分治或者二维线段树来解决.

时间复杂度: $O(n \log^2 n)$.

第三章　试题泛做 (3)

3.1 Fibonacci Number

3.1.1 问题描述

给定 c 和 p, 求最小的 n, 使得 $Fib[n] \equiv c \pmod{p}$, 其中, $Fib[n]$ 表示 Fibonacci 数列的第 n 项.

如果无解, 那么输出-1.

3.1.2 数据规模和约定

- 多组 (不超过 100 组) 数据
- $11 \leq p \leq 2 \times 10^9$ 且 p 是质数
- $0 \leq c \leq p - 1$
- $p \bmod 10$ 是完全平方数

3.1.3 分析

首先, 观察 Fibonacci 数列的通项公式

$$Fib[n] = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right)$$

不妨尝试在模意义下把根号开出来. 我们可以发现, 5 一定是 p 的一个二次剩余.

Proof. 根据二次互反律

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$$

因此我们不妨考虑先求 $\left(\frac{p}{5}\right)$, 而 $\left(\frac{p}{5}\right) = p^{\frac{5-1}{2}} \bmod 5 = p^2 \bmod 5$, 注意到由于 $p \equiv 1 \pmod{10}$, 因此 $p^2 \bmod 5 = 1$.

于是

$$\left(\frac{5}{p}\right)\left(\frac{p}{5}\right) = (-1)^{\frac{5-1}{2}\frac{p-1}{2}} = (-1)^{p-1}$$

即

$$\left(\frac{5}{p}\right) = (-1)^{p-1} = 1$$

因此, 5 是 p 的一个二次剩余.

□

不妨假设已经求出了 $s \equiv \sqrt{5} \pmod p$. 设 $\alpha = \frac{1+\sqrt{5}}{2}, \beta = \frac{1-\sqrt{5}}{2}$, 那么我们可以发现, $\alpha\beta = -1$, 即 $\beta = -\frac{1}{\alpha}$. 设 $x = \alpha^n$, 那么

$$\alpha^n - \left(\frac{1}{\alpha}\right)^n \equiv \sqrt{5}c \pmod p$$

以 n 为奇数为例, 此时上式化为

$$x + \frac{1}{x} \equiv \sqrt{5}cx \pmod p$$

$$x^2 - \sqrt{5}cx + 1 \equiv 0 \pmod p$$

$$x \equiv \frac{\sqrt{5}c \pm \sqrt{5c^2 - 4}}{2}$$

因此我们只需要求出 x(有两个), 然后求出它关于 α 的离散对数 (这个值要满足我们关于 n 的奇偶性的假设) 并更新答案, 就可以了.

怎么求满足 $s^2 \equiv 5 \pmod p$ 的 s 呢? 下面以求满足 $x^2 \equiv n \pmod p$ 的 x 为例, 来介绍两种求二次剩余的算法.

Baby Step Giant Step

我们可以先求出 p 的原根 g, 由于如果一个数 p 有原根, 那么它就有 $\varphi(\varphi(p))$ 个原根, 并且检查 g 是不是原根只需要对于 $\varphi(p)$ 的每一个质因数 p_i, 检查 $g^{\frac{\varphi(p)}{p_i}}$ 是否不为 1. 因此, 我们只需要暴力枚举并检验就可以得到原根了.

之后我们求出 k, 使得 $g^k \equiv n \pmod p$, 然后 $g^{\frac{k}{2}}$ 就是答案了.

时间复杂度: $O(\sqrt{p})$.

Cipolla 算法

我们先求出一个 a, 使得 a^2-n 是模 p 的一个二次非剩余.

然后我们扩展域 $\mathbf{F}_p = \mathbb{Z}/p\mathbb{Z}$. 我们引入一个新的元素 $I = \sqrt{a^2-n}$, 从而得到一个新域 $F_p^2 = F_p(\sqrt{a^2-n})$.

最后我们计算出 $x = (a+I)^{\frac{p+1}{2}}$, 就是满足条件的 x 了.

Proof. 我们将说明为什么 $x = (a+I)^{\frac{p+1}{2}}$ 满足条件.

事实上

$$\begin{aligned} x^2 &= (a+I)^{p+1} \\ &= (a+I)(a+I)^p \\ &= (a+I)\left(\sum_{i=0}^{p}\binom{p}{i}a^i I^{p-i}\right) \end{aligned}$$

注意到当 $i \neq 0, p$ 时, $\binom{p}{i} \bmod p = 0$, 因此

$$\begin{aligned} x^2 &= (a+I)(a^p I^0 + a^0 I^p) \\ &= (a+I)(a+I^p) \end{aligned}$$

由于 I^2 为一个二次非剩余, 因此, $I^{p-1} = (I^2)^{\frac{p-1}{2}} = -1$. 于是上式可进一步化简为

$$\begin{aligned} x^2 &= (a+I)(a-I) \\ &= a^2 - I^2 \\ &= a^2 - (a^2-n) = n \end{aligned}$$

□

求满足 $\left(\frac{a^2-n}{p}\right) = -1$ 的 a 可以用随机来实现, 因为一个数成为二次剩余的概率大概是 $\frac{1}{2}$.

时间复杂度: $O(\log p)$.

3.2 Chef and Balanced Strings

3.2.1 问题描述

给定一个长度为 n 的字符串 s.

定义 $s_{l,r}$ 表示字符串中第 l 个字符到第 r 个字符所构成的字符串.

定义一个字符串是平衡的当且仅当这个字符串中每一种字符出现的次数都是偶数.

有 q 个询问, 每一个询问给出三个数 l, r, k, 其中 $k \leq 2$, 询问 $s_{l,r}$ 的平衡子串的长度的 k 次方和.

3.2.2　数据规模和约定

有多组数据, 保证所有数据的 n 的和与所有数据的 q 的和小于等于 10^5.

字符集为所有的小写字母.

询问强制在线.

3.2.3　分析

不妨定义 $mask[i]$ 的二进制第 j 位表示 $s_{1,i}$ 中第 j 个字符出现次数的奇偶性, 为 0 表示出现了偶数次, 为 1 表示出现了奇数次. 并且令 $mask[0] = 0$.

那么如果说没有强制在线这个限制的话, 我们用一个数组统计区间 $[l-1, r]$ 中某个二进制出现的次数, 然后套一个莫队就可以了.

如果需要在线回答询问的话, 我们可以利用分块, 假设每块的大小为 $size$.

对于 $i = 0 \ldots n$, 如果 $i \bmod size = 0$, 那么称它为关键点. 我们对于每个关键点 i, 维护一个数组 $sum[i][j][k]$, 表示 i 到 j 中, 所有平衡串长度的 k 次方和, 显然这个数组可以用 $O(\frac{n^2}{size})$ 的时间得到.

考虑如何回答一组询问 l, r, k.

如果 $r - l \leq size$ 的话, 那么我们可以直接暴力, 这种情况的复杂度为 $O(size)$.

否则, 不妨令 l 之后的第一个关键点为 x, r 之前的第一个关键点为 y, 并且设答案 ans.

一开始令 $ans = sum[x][r][k] + sum[y][l][k] - sum[x][y][k]$. 这实际上对应于总的答案减去所有左端点在区间 $[l, x)$ 中, 右端点在区间 $(y, r]$ 的平衡串的信息.

那么我们之后只需要暴力统计所有左端点在区间 $[l, x)$ 中, 右端点在区间 $(y, r]$ 的平衡串的长度的 k 次方和就可以了, 这种情况的时间复杂度也为 $O(size)$.

因此总的时间复杂度为 $O(\frac{n^2}{size} + q \times size)$, 因此取 $size = \lfloor \sqrt{n} \rfloor$, 这样总的复杂度为 $O((n+q)\sqrt{n})$.

3.3　Counting on a directed graph

3.3.1　问题描述

给定一个 n 个点 (从 1 到 n 标号)m 条边的有向图. 请你统计无序对 (x, y) 的个数, 其中 (x, y) 满足存在一条从点 1 到点 x 的路径, 和一条从点 1 到点 y 的路径, 且两条路径除了点 1 之外没有公共点.

3.3.2　数据规模和约定

- $1 \leq n \leq 10^5$
- $0 \leq m \leq 5 \times 10^5$
- 图没有自环或重边

3.3.3　分析

对于一个点 x, 定义 $dom(x)$ 为从起点走到 x 所必须要经过的点的集合. 那么不难发现, 这个集合中的点连起来可以构成原图中的一条链.

那么定义 $idom(x)$ 为 $dom(x)$ 中最靠近 x 的点. 实际上, 如果我们令 x 和 $idom(x)$ 连一条边, 那么这些边将会构成一棵树.

我们不妨把这棵树称为必经点树 (Dominator Tree). 很显然的是, 如果我们已经构出了必经点树, 那么我们可以非常容易地得到答案.

于是本题的难点转化为怎么构必经点树.

关于这个问题, Lengauer 和 Tarjan 提出了一个在 $O(m\alpha(m))$ 的复杂度内求出必经点树的算法.

它的做法是这样的, 我们先构建出原图的一棵 DFS 树, 记它为 T. 下面称一个点 x 比另一个点 y 小, 意味着 x 的 DFS 编号 $dfn[x]$ 小于 y 的编号 $dfn[y]$.

对于一个点 x, 定义在 T 中, 从起点到 x 的路径为树枝边, 称其他边为非树枝边.

定义 x 的半必经点 $sdom(x)$ 为深度最小的, 可以**仅**经过比 x 的 DFS 编号大的点到达 x 的点 (不考虑这个点本身以及 x).

显然, $sdom(x)$ 是 x 在 T 中的一个祖先 (否则我们就可以把 $sdom(x)$ 往上移, 直到它是 x 的祖先).

为了求解 $sdom(x)$, 我们不妨枚举 x 的所有前导 (存在一条边到 x 的点)y, 然后维护一个点 $temp$.

- $dfn[y] < dfn[x]$: 则 (y, x) 为树枝边或前向边, 此时令 $temp = \min(temp, y)$;
- $dfn[y] > dfn[x]$: 则 (y, x) 为横叉边或后向边, 此时 $\forall y$ 在 T 中的祖先 z, 如果 $dfn[z] > dfn[x]$, 则令 $temp = \min(temp, sdom(z))$.

最后 $sdom(x) = temp$, 也就是说 $sdom(x)$ 取上面所有情况中 DFS 编号最小的点.

上述结论被称之为半必经点定理.

怎么理解上面这个过程呢? 很明显, 如果这条边 (y, x) 是树枝边, 那么 y 肯定是 x 的祖先 (否则这就不是一棵合法的 DFS 树了), 因此直接取 DFS 编号较小的点即可.

如果 (y,x) 是横叉边或后向边, 那么当 $dfn[z] > dfn[x]$ 时, 从 $sdom(z)$ 走到 z 再走到 x 肯定是合法路径, 而 $dfn[z] < dfn[x]$ 的点走到 x 是不合法的, 因此在满足上面条件的点中取个 min 即可.

求出了 $sdom(x)$ 之后怎么求 $idom(x)$ 呢? 我们有如下结论: 考虑 T 中从 $sdom(x)$ 到 x 的路径, 不妨令 $rdom(x)$ 表示路径上拥有最小的 $sdom$ 的点.

那么 $idom(x)$ 的取值有两种情况

- $sdom(x) = sdom(rdom(x))$, 此时 $idom(x) = sdom(x)$;
- 否则, $idom(x) = idom(rdom(x))$.

定理的第一部分是显然的, 因为在这种情况下, 不存在跨过 $sdom(x)$ 的到 x 的路径了.

第二部分我们可以这么理解, 首先设 $idom(x) = sdom(x)$, 然后执行下面这个过程

1. 找到 T 中 x 到 $idom(x)$ 这条路径上 $sdom$ 最小的点 y;
2. 检验是否有 $sdom(y) = idom(x)$, 如果是, 那么退出, 否则令 $idom(x) = sdom(y)$, 并继续整个过程.

这个做法的正确性是显然的.

这个算法的实现可以借助并查集. 我们先考虑怎么求 $sdom(x)$: 不妨按照逆 DFS 序枚举每一个点 x, 那么我们可以枚举它的前导 y, 然后我们要知道的就是 y 到它祖先中大于 x 的点的最小 $sdom$, 那么我们可以用并查集维护一个森林, 每做完一个点就把它和它的父亲连一条边, 令边权为它的 $sdom$, 那么我们要求的就是某一个点到它祖先的整条路径上的最小 $sdom$, 这可以在并查集路径压缩的时候加以维护.

现在考虑怎么求 $rdom(x)$. 我们仍然按照逆 DFS 序做, 只不过在每个点 x 枚举所有 $semi(y) = x$ 的 y, 然后求出 y 的 $rdom$. 此时的 $rdom$ 所要求的最小值实际上和我们求 $sdom$ 时所维护的并查集是一致的, 因此我们只需要利用上面的并查集即可.

3.4 The Street

3.4.1 问题描述

平面上有 n 个点, 每个点 i 存有两个信息 $info[i]$ 和 $tag[i]$. 你需要回答 m 个询问, 询问有三种类型

1. 1 l r a b: 对于区间 $[l,r]$ 中的点 i, 令 $info[i] = \max(info[i], f[i])$, 其中 $f[i]$ 表示以 a 为首项, b 为公差的数列的第 i 项;
2. 2 l r a b: 对于区间 $[l,r]$ 中的点 i, 令 $tag[i] += f[i]$, $f[i]$ 的定义类似;

3. 3 i: 求 $info[i] + tag[i]$.

3.4.2　数据规模和约定

- $1 \leq n \leq 10^9$
- $1 \leq m \leq 3 \times 10^5$
- 保证数据合法
- 对于询问 1, $|a|, |b| \leq 10^9$
- 对于询问 2, $|a|, |b| \leq 10^4$

时间限制: 3s.

3.4.3　分析

显然, 询问 1 和询问 2 可以分开维护. 而询问 2 的维护比较简单, 因此下面我们主要讨论询问 1 的处理.

实际上, 我们要维护的是一个点最上面的线段是哪条. 这可以用线段树来维护: 设 cur 表示当前所加入的线段, $max[id]$ 表示线段树中编号为 id 的结点所维护的一条线段. 不妨假设当前递归到了线段树的区间 $[l, r]$, 如果 $max[id]$ 和 cur 的交点不在区间 $[l, r]$ 中, 那么要么 $max[id]$ 在 cur 上面, 要么反过来, 这两种情况我们都可以在当前区间更新答案并退出; 而如果 $max[id]$ 和 cur 的交点在区间 $[l, r]$ 中, 我们设 $[l, r]$ 的两个儿子结点分别为 $[l, mid]$ 和 $[mid + 1, r]$, 我们可以发现, 我们只需要递归交点所在的那一个子区间就可以了, 因为对于另一个子区间, 实际上会满足 $max[id]$ 覆盖 cur, 或 cur 覆盖 $max[id]$ 的情况, 这样我们就只需要递归 $\log n$ 层了.

时间复杂度: $O(m \log n)$.

3.5　Chef and Graph Queries

3.5.1　问题描述

给定一个有 n 个点, m 条边的无向图 G. 有 q 组询问 $[l, r]$, 对于每一组询问, 你需要给出只保留第 l 条边到第 r 条边时, 图 G 中有多少连通块.

3.5.2　数据规模和约定

- $1 \leq n, m, q \leq 2 \times 10^5$
- 可能包含自环和重边

- 保证数据合法

时间限制: 8s.

3.5.3　分析

首先, 我们考虑所有询问的 r 相同的情况: 我们可以把所有的边以编号为权值构一棵最大生成树, 设这样子所得到的连通块数为 cur, 然后对于每一个 l, 当它为左端点时, 答案就是 cur + 生成树中编号小于 l 的边的数量.

那么接下来的做法就很显然了: 我们可以枚举 r 并用 LCT 维护当前的生成树, 然后再维护树中的边按照编号的前缀和, 就可以得到答案了.

时间复杂度: $O((n+m)\log n)$.

第四章　试题泛做 (4)

4.1 Counting D-sets

4.1.1 问题描述

考虑以曼哈顿距离为测度的空间 $\mathbb{R}_n$.

定义一个点集 $A \subset \mathbb{R}_n$ 的直径为点集中最远点对的距离.

给定 n 和 d, 你需要统计 $\mathbb{R}_n$ 中有多少个本质不同的直径为 d 点集? 如果一个点集可以由另一个点集平移得到, 那么我们称它们是本质相同的.

4.1.2 数据规模和约定

- 多组数据 (不超过 10 组)
- $1 \le n \le 1000$
- $1 \le d \le 10^9$

时间限制: 1s.

4.1.3 分析

设直径小于等于 d 的点集数为 $f(d)$, 我们可以按照有多少维是碰不到边界的为容斥条件, 从而得到下式

$$f(d) = \sum_{i=0}^{n} (-1)^i \binom{n}{i} 2^{d^i + (d+1)^{n-i}}$$

然后 $f(d) - f(d-1)$ 就是答案了.

时间复杂度: $O(d \log n)$.

4.2 Counting The Important Pairs

4.2.1 问题描述

给定一个有 n 个点, m 条边点图. 求有多少对边 (a,b), 使得割掉 a 和 b 这两条边之后图不连通.

4.2.2 数据规模和约定

- $1 \le n \le 10^5$
- $1 \le m \le 3 \times 10^5$
- 保证没有自环, 重边

时间限制: 3s.

4.2.3 分析

如果整个图一开始就不连通, 那么答案就是 $\binom{m}{2}$.

下面考虑图一开始连通的情况: 我们不妨构建出图的一棵 DFS 树, 留意到此时 DFS 树只有返祖边. 接下来, 对于每一条边 i 维护一个集合 $edge[i]$. 不妨设 $i=(u,v)$, 当 i 为树边时, 令 $edge[i]$ 表示所有的断开边 i 之后, 连接了 u 和 v 所在连通块的非树边; 当 i 为非树边时, $edge[i]=\{i\}$.

我们可以发现, 对于任意一对边 i 和 j, 它对答案有贡献当且仅当 $edge[i]=edge[j] \vee edge[i]=\varnothing \vee edge[j]=\varnothing$.

Proof. 后两种情况显然是对的.

对于第一种情况, 设我们删掉了 i 和 j 这两条边, 那么如果它们都是非树边或者它们中有一条树边, 一条非树边, 结论显然是正确的.

下面我们考虑 i 和 j 都是树边的情况, 此时, 这两条边会把生成树分成三块, 设分别为 A,B,C, 那么我们不妨设块和边按照 $A-i-B-j-C$ 的顺序连接. 那么如果 $edge[i]=edge[j]$, 即连接 A 和 BC 的边的所构成的集合 ($edge[i]$) 与连接 AB 和 C 的边的所构成的集合 ($edge[j]$) 相等, 那么此时一定有所有的 $edge[i]$ 中的边都是连接 A 和 C 的, 因此 B 就不能够与 A, C 连通, 因此图就不连通.

而如果 $edge[i] \neq edge[j]$, 不妨设 $a \in edge[i] \wedge a \notin edge[j]$, 那么此时边 a 一定能够连接 A 和 B, 那么我们把 a 选上, 然后再用 $edge[j]$ 中的任意一条边连接 AB 和 C, 就可以使得这个图成为连通图. □

这样我们就可以得到一个朴素算法: 直接用 bitset 表示集合, 然后暴力, 时间复杂度 $O(\frac{nm}{32})$, 无法接受.

注意到我们想要知道的仅仅是 $edge[i]$ 与 $edge[j]$ 是否相等, 因此我们可以考虑对集合进行某种哈希: 对于一条非树边 e, 我们不妨用一个 64 位的二进制数 $tag[e]$ 来表示它, 然后对于所有的树边 i, 令 $tag[i] = \oplus_{e \in edge[i]} tag[e]$, 其中 $\oplus$ 表示按位异或. 然后, $tag[i] = tag[j] \iff edge[i] = edge[j]$, 并且 $tag[i] = 0 \iff edge[i] = \varnothing$.

因此我们就可以统计每个 tag 出现的次数并统计答案了.

时间复杂度: $O(n + m)$.

第五章　试题泛做 (5)

5.1 Evil Book

5.1.1 问题描述

有 n 个敌人, 打败第 i 个敌人可以获得 m_i 点魔法力量, 同时需要付出 c_i 点努力.

然后有一本魔法之书, 它可以让你以 x 点魔法力量的代价, 将某个敌人的 m_i 和 c_i 都除以 3(可以使用多次, 可以重复使用).

初始时你没有魔法力量, 然后你需要给出一开始至少需要多少努力才能够获得 666 点魔法力量?

5.1.2 数据规模和约定

- 多组数据 (不超过 5 组)
- $1 \leq n \leq 10$
- $10 \leq x < 666$
- $0 \leq c_i, m_i \leq 10^9$

时间限制: 3s.

5.1.3 分析

我们肯定可以在打一个人之前减弱他, 设对于第 i 个人我们用了 k 次攻击, 那么 k 会满足 $\frac{m_i}{3^k} > kx$, 这实际上告诉我们 k 的可能的取值数量不会太多. 经过计算可以得到, 它只会有 4 种不同的取值. 这样总的状态数是 4^k, 然后我们再加一下剪枝什么的就可以通过本题了.

时间复杂度: $O(4^k)$.

5.2 Dynamic GCD

5.2.1 问题描述

给定一棵有 n 个结点的树, 每个点有一个权值. 点 i 的初始权值为 $val[i]$.

有 q 个询问, 分为两类

- F u v: 查询 u 到 v 路径上的点的权值的 gcd
- C u v d: 将 u 到 v 路径上的点的权值加上 d

5.2.2 数据规模和约定

- $1 \le n, q \le 5000$
- $1 \le val[i] \le 10000$
- $0 \le d \le 10000$

时间限制: 2s.

5.2.3 分析

由于 gcd 运算是可加的, 因此我们可以链剖, 然后把树的问题转化为序列问题.

我们不妨将 $val[]$ 进行差分, 令 $dif[i] = val[i] - val[i-1]$, 并且令 $dif[1] = val[1]$. 那么, $val[i] = \sum_{j=1}^{i} dif[j]$.

于是, 将一个区间 $[l, r]$ 加上一个数的操作就变成了把 $dif[l]$ 加上一个数, 并把 $dif[r+1]$ 减去一个数的操作.

如何询问某个区间的 gcd 呢? 留意到 $\gcd(a, b) = \gcd(a, a-b)$, 因此, 区间 $[l, r]$ 的 gcd 就等于 $\gcd(dif[l+1 \ldots r], val[l])$, 其中 $val[l]$ 可以用 $dif[]$ 的前缀和来得到.

因此, 线段树中只需要维护 $dif[]$ 的区间和以及区间 gcd 就可以了.

时间复杂度: $O(n \log^2 n \log V)$, 其中 V 是权值的范围, $O(\log V)$ 是 gcd 的复杂度.

第六章　试题泛做 (6)

6.1　Easy Exam

6.1.1　问题描述

一天, 大厨考了概率论. 本来没什么特别的, 但考试容易得出奇. 做到最后, 大厨才发现了这么一个题目:“假设你有一个 K 面骰子, 上面分别写着 1 到 K. 还有两个整数 L 和 F($0 < l \leq k$). 你掷了 n 次骰子, 记掷出数字 i 的次数为 a_i. 请你求出 $(a_1 a_2 \cdots a_l)^f$ 的期望值.” 读完这个题, 大厨感觉自己要被退学了, 就再也没办法称为一个厨师了. 为了以后你能吃到大厨做的美味佳肴, 请你帮帮大厨.

假设答案为 $\frac{p}{q}$, 其中 p 和 q 是整数. 你需要输出 $p \times (q^{-1} \pmod{2003})$.

本题有多组数据.

6.1.2　数据规模和约定

- $1 \leq t \leq 50$
- $0 < n, k \leq 10^9$
- $0 < l \times f \leq 50000$
- $0 < f \leq 1000$

6.1.3　分析

题目中要求期望的式子不太优美, 我们考虑令 $x_{i,j} = 1$ 表示第 j 个骰子的值为 i,$x_{i,j} = 0$ 则表示值 $\neq i$, 那么

$$a_i = \sum_{j=1}^{n} x_{i,j}$$

化简一下式子

$$E[\prod_{i=1}^{l} a_i] = E[\prod_{i=1}^{l}(x_{i,1} + x_{i,2} + \cdots + x_{i,n})] \tag{6.1}$$

$$= E[\sum \prod_{i=1}^{l} x_{i,a_i}] \qquad (\forall i \neq j, a_i \neq a_j) \tag{6.2}$$

$$= \sum E[\prod_{i=1}^{l} x_{i,a_i}] \tag{6.3}$$

$$= \sum P(\prod_{i=1}^{l} x_{i,a_i}) \tag{6.4}$$

现在, 题目中要我们求的期望变成了

$$\prod_{i=1}^{l}(\sum_{j=1}^{n} x_{i,j})^f$$

考虑将它展开之后的每一项, 我们需要确定每一项为 1 的概率.

我们枚举展开式中不同的骰子个数 p, 那么它对答案的贡献为

$$\binom{n}{p} \cdot p! \cdot count_p$$

其中 $count_p$ 等于多项式 $(\sum_{i=0}^{f} S(f,i)x^i)^l$ 的 p 次项系数, 其中 $S(n,k)$ 表示第二类 Stirling 数.

我们先枚举这 p 个骰子的顺序, 这部分的贡献为 $\binom{n}{p} \cdot p!$. 之后我们根据因式中每一个 $x_{i,j}$ 的 i 来将它们分类, 那么整个因式总共被分成了 l 类, 每类有 f 项. 显然, 每个骰子只会出现在 (至多) 一类中 (一个骰子不能同时有两个值).

那么我们枚举第 i 类总共有多少个不同的数 a_i, 对于这一类, 方案数等于把 f 项划分成 a_i 个等价类 (按照每一项属于哪个骰子) 的方案数, 而我们可以发现后者就是第二类 Stirling 数.

那么总的方案数就等于

$$\sum_{a_1+a_2+\cdots+a_l} S(f, a_i)$$

这个式子是一个卷积的形式, 因此可以像上面那样通过求多项式某一项的系数得到.

注意到 $\binom{n}{p} \cdot p!$, 也就是排列数, 在 $p > 2003$ 之后就会变成 0(题目中的模数 2003 比较小), 因此我们在求多项式的幂的时候不需要计算后面的项, 这样就可以暴力卷积.

题目中的 f 比较小, 因此可以直接算 Stirling 数.

题目中的排列数的 p 比较小, 因此也可以暴力计算.

时间复杂度 $O(f^2 + 2003^2 \log l)$.

6.2 Deleting numbers

6.2.1 问题描述

给定一个有 n 个元素的数组 $a[]$, 你每次可以按照如下规则删除一些数: 指定一个 v 和 t, 设 $k = \max\{x|v + tx \leq n\}$, 那么 v 和 t 需要满足 $a[v] = a[v + t] = a[v + 2t] = \cdots = a[v + kt]$, 然后第 $v, v + t, v + 2t, \ldots, v + kt$ 这些数都会被删除.

你的任务是删除所有数, 你的目标是最小化删除次数.

6.2.2 数据规模和约定

$1 \leq n, a[i] \leq 10^5$

时间限制: 1s.

6.2.3 分析

如果我们考虑删除之后的坐标变化的话, 这道题会变得非常复杂. 因此我们不妨考虑最简单的情况: 每次 (至少) 把最后一个数删掉. 那么我们可以每次从后往前找到一个最长的等差数列, 然后删掉.

6.3 Music & Lyrics

6.3.1 问题描述

给定 n 个模式串 S 和 m 个文本串 T, 求每个模式串在所有文本串中出现次数之和.

6.3.2 数据规模和约定

- $1 \leq n \leq 500$
- $1 \leq |S| \leq 5000$
- $1 \leq m \leq 100$
- $1 \leq |T| \leq 50000$

时间限制: 3s.

6.3.3 分析

我们先构出所有模式串的 AC 自动机, 然后再拿每个文本串在自动机上面跑一下, 并统计次数, 然后再把这个次数沿着 fail 边更新一下.

时间复杂度: $O(n|S| + m|T|)$.

6.4 Prime Distance On Tree

6.4.1 问题描述

给定一棵有 n 个结点的树, 其中每条边的长度都是 1. 求距离为质数的点对的数量.

6.4.2 数据规模和约定

$1 \leq n \leq 50000$.

时间限制: 5s.

6.4.3 分析

这道题直接点分治, 然后对于每一个 i, 求出长度为 i 的路径数, 最后统计答案就可以了.

点分治的时候注意要把儿子自身的影响减掉, 然后 FFT 的时候注意要传一个长度参数到 FFT 函数里面, 否则复杂度会退化为 $O(n^2 \log^2 n)$.

时间复杂度: $O(n \log^2 n)$.

第七章　试题泛做 (7)

7.1 Stepping Average

7.1.1 问题描述

给定一个 n 个点的数组 $a[]$, 你每次可以取两个数并把它们取出来, 然后把它们的平均数加进去, 最后会剩下一个数, 你需要让这个数尽可能地接近 k.

7.1.2 数据规模和约定

- $n = 1000$
- 数据随机

时间限制: 5s.

7.1.3 分析

不妨假定我们最后的合并顺序一定是 $a[n]$ 和 $a[n-1]$ 合并, 得到的新数和 $a[n-2]$ 合并, 然后一直进行下去. 那么 $a[i]$ 对答案的贡献就是 $\frac{a[i]}{2^i}$. 我们不妨枚举前 30 个数 (因为后面的数对答案已经几乎没有贡献了), 然后再枚举后面哪个数和它交换, 如果交换后更优就交换. 这样可以在 Codechef 上拿到 1 分.

时间复杂度: $O(\lambda n)$, 其中 λ 是我们设定的阈值 (在本题中 $\lambda = 30$).

7.2 Lucky Days

7.2.1 问题描述

给定一个线性递推数列 s, 其中 $s[1] = a$, $s[2] = b$, $s[i] = (x \cdot s[i-1] + y \cdot s[i-2] + z) \bmod p, \forall i \geq 3$.

我们同时定义一个数 c 为我们的幸运数.

你需要回答 q 组询问, 每一组询问给出一个区间 $[l,r]$, 你需要回答区间 $[l,r]$ 中有多少个数 i, 使得 $s[i]=c$.

7.2.2　数据规模和约定

- 多组数据 (不超过 2 组)
- $2 \le p \le 10007$ 并且 p 是一个质数
- $0 \le a,b,x,y,z,c < p$
- $1 \le q \le 20000$
- $1 \le l \le r \le 10^{18}$

时间限制: 5s.

7.2.3　分析

不放令下标从 0 开始.

定义初始向量 $\mathbf{v}=(a,b,1)$, 转移矩阵 $A=\begin{pmatrix}0 & y & 0\\ 1 & x & 0\\ 0 & z & 1\end{pmatrix}$, 那么, 问题就转化为对于区间 $[l,r]$, 有多少个 i 使得 $\mathbf{v}A^i=(c,t,1)$, 其中 t 为任意数.

首先, 这个数列肯定是循环的, 那么如果我们能够求出循环节以及一个循环节中哪些 i 是满足条件的, 那么问题就解决了.

我们考虑用 BSGS 来解决这个问题, 不妨先枚举 t, 假设分块阈值为 $size$, 那么, 我们实际上就会得到 $\mathbf{v}A^{i\cdot size+j}=(c,t,1)$, 此时, 如果 A 没有有逆矩阵, 即 $\det A=-y=0$, 那么数列 s 就变成了一个一次的递推数列, 它的循环节显然不超过 p, 因此我们可以直接暴力讨论掉这种情况.

而如果 A 有逆矩阵, 那么上式就可以化为 $\mathbf{v}A^{i\cdot size}=(c,t,1)A^{-j}$. 留意到循环节不超过 p^2, 于是我们可以预处理出所有的 $\mathbf{v}A^{i\cdot size}$, 然后枚举 j 去查询.

时间复杂度: 如果采用哈希表的话, 那么预处理的时间复杂度为 $O(\frac{p^2}{size}+p\cdot size)$, 令 $size=\sqrt{p}$, 那么此时的复杂度为 $O(p\sqrt{p})$ 最小.

但是, 这样做的常数很大, 在 Codechef 上会 TLE.

然后我就去参考了 yyt16384 的代码, 发现上面的推导过程中我们犯了思维定势的错误, 如果我们把 BSGS 的式子设成 $\mathbf{v}A^{i\cdot size-j}=(c,t,1)$, 那么我们就会得到 $\mathbf{v}A^{i\cdot size}=(c,t,1)A^j$, 这样就不需要求逆矩阵了. 注意到此时算法成立仍然依赖于 A 有逆矩阵.

除此之外, 我们还可以摆脱矩阵的束缚: 我们用一个有序对 (u, v) 表示上式中的向量的前两维, 并且定义 $succ(u, v)$ 为将有序对 (u, v) 递推一项后得到的有序对. 那么, $\mathbf{v}A$ 就相当于 $succ(\mathbf{v}_1, \mathbf{v}_2)$.

令 $p_{00} = succ^{size}(0, 0)$, $p_{01} = succ^{size}(0, 1)$, $p_{10} = succ^{size}(1, 0)$, 那么对于任意一个有序对 cur, 我们可以利用 p_{00}, p_{01}, p_{10} 来求出 $succ^{size}(cur)$. 下面设有序对 p 的第一项为 $p[1]$, 第二项为 $p[2]$.

那么实际上, 我们要求的式子可以化为

$$\begin{aligned}
succ^{size}(cur) &= (cur[1], cur[2], 1) \cdot A^{size} \\
&= \left(cur[1] \cdot (1, 0, 1) + cur[2] \cdot (0, 1, 1) + (1 - cur[1] - cur[2]) \cdot (0, 0, 1)\right) A^{size} \\
&= cur[1] \left((1, 0, 1) \cdot A^{size}\right) + cur[2] \left((0, 1, 1) \cdot A^{size}\right) + (1 - cur[1] - cur[2]) \left((0, 0, 1) \cdot A^{size}\right) \\
&= cur[1] \cdot p_{10} + cur[2] \cdot p_{01} + (1 - cur[1] - cur[2]) \cdot p_{00}
\end{aligned}$$

这样, 我们就可以不使用矩阵乘法, 而仅仅采用有序对来实现整个算法, 这将加快整个算法并且减小代码量.

7.3 Sine Partition Function

7.3.1 问题描述

给定 n, m, x, 你需要求

$$\sum_{k_1 + k_2 + \cdots + k_m = n} \sin(k_1 x) \sin(k_2 x) \cdots \sin(k_m x)$$

其中 $k_1, \ldots, k_m \in \mathbb{N}$.

7.3.2 数据规模和约定

- 多组数据 (不超过 10 组)
- $1 \le m \le 30$
- $1 \le n \le 10^9$
- $0 \le x \le 6.28 < 2\pi$

时间限制: 1.5s.

7.3.3　分析

我们可以把这个问题看作是把 n 个小球分配到 m 个盒子里, 然后再计算答案. 设 $f[i][j]$ 表示当前分配了 i 个小球, 最后一个盒子是第 j 个盒子的答案, 由于一旦有一个 $k_i = 0$, 那么这一项对答案的贡献就是 0, 因此我们不妨假设 $k_i > 0$.

此时我们可以发现, 对于 $f[i][j]$, 实际上它可以由两个状态转移过来: $f[i][j-1]$ 和 $f[i-1][j-1]$.

对于后者, 我们相当于新增加了一个盒子, 因此这一项对答案的贡献就是 $f[i-1][j-1] \cdot \sin(x)$.

而对于前者, 我们就需要在最后一个盒子里多放一个小球, 此时仅仅维护 f 是不够的, 我们再设一个数组 $g[i][j]$, 它是在 $f[i][j]$ 的基础上, 把最后一个盒子对答案的贡献由 sin 变成 cos 得到的. 那么, 考虑和差化积公式, 我们可以得到 $\sin(\alpha + x) = \sin(\alpha)\cos(x) + \cos(\alpha)\sin(x)$, 这就告诉我们向最后一个盒子里面放入一个小球对答案的贡献实际上是 $f[i-1][j-1] \cdot \cos(x) + g[i-1][j-1] \cdot \sin(x)$. 然后 $g[i][j]$ 的转移类似.

于是我们就得到了一个 $O(nm)$ 的做法. 显然, 这个做法可以利用矩阵乘法优化到 $O(m^3 \log n)$.

7.4　The Baking Business

7.4.1　问题描述

这道题差不多相当于让你写一个支持 SQL 中的按条件 SELECT 和 INSERT 两种操作的程序.

不过 INSERT 操作不超过 100 条.

7.4.2　数据规模和约定

操作数不超过 100000.

时间限制: 1~10s.

7.4.3　分析

由于 INSERT 操作很少, 我们可以直接维护前缀和数组, 然后询问的时候讨论一下就可以了.

第八章　试题泛做 (8)

8.1 Maximum Sub-rectangle in Matrix

8.1.1 问题描述

给定一个矩阵 $a_{n\times m}$, 你需要求出一个最大的非连续子矩阵, 使得它的权值和最大.

非连续子矩阵指的是: 选出一些行和一些列, 然后这些行和列交叉的格子就属于子矩阵.

8.1.2 数据规模和约定

- $1 \leq n, m \leq 300$
- $|a[i][j]| \leq 10^9$

时间限制: 2s.

8.1.3 分析

不妨考虑先随机一组答案: 对于每一行, 让它进入答案的概率为 $\frac{\text{行内正数个数}}{}$.

注意到如果我们知道如何选取行, 那么我们可以贪心出列的最优方案, 而如果我们知道怎么选取列, 我们同样可以得到行的最优方案.

然后我们可以迭代得到最优解: 我们先随机一组行的方案, 然后得到列的最优方案, 然后根据我们得到的列的方案再得到行的最优方案, 如此往复下去, 直到方案不再变化.

这样我们再卡一下时就可以得到比较优的答案了.

这个做法在 Codechef 上可以得到 0.998 分.

时间复杂度我也不知道.

8.2 Ranka

8.2.1 问题描述

构造一种在 9×9 的棋盘上下 n 步围棋的方法, 这里的围棋禁止全局同形再现, 并且执棋的一方可以选择不走.

8.2.2 数据规模和约定

$1 \leq n \leq 10000$.

时间限制: 1s.

8.2.3 分析

这题有很多种做法, 下面选择了两种比较有意思的做法来介绍.

做法 1

我们可以这么下

1. 黑方用棋子填满除了 $(1,1)$ 以外的所有点, 白方不动;
2. 白方下 $(1,1)$ 这一步, 把黑方的所有棋子都提出来;
3. 白方下除了 $(1,2)$ 以外的所有点, 黑方不动;
4. 黑方下 $(1,2)$ 这一步, 把白方的所有棋子都提出来;
5. 如此进行下去.

这样我们可以构造出有 $81 \times 81 \times 2$ 步的棋局, 可以通过本题.

但是这种做法不靠谱, 因为围棋规则中是禁止下到一个会导致双方到棋子都死掉到点的.

做法 2

我们可以利用 “劫”.

比如说, 这盘棋我们构造出了 8 个劫, 这样, 我们就可以用一位二进制来标记每一个劫是被哪方占领的, 从而得到 2^8 种棋局, 然后我们按照格雷码的顺序遍历这些棋局, 这样我们就可以走 1.5×2^8 步.

并且我们可以发现, 我们可以在棋盘上做一些小小的修改, 从而得到另外一种棋局, 这样就可以轻松走 10000 步了.

做法 3

还有一种做法是按照上三角来填充, 这里不再赘述.

8.3 Xor Queries

8.3.1 问题描述

给你一个初始时为空的整数序列 (元素由 1 开始编号) 以及一些询问

- 0 x, 在序列最后加入数字 x;
- 1 l r x, 在区间 $[l, r]$ 中找到数字 y, 并最大化 $x \oplus y$, 其中 $\oplus$ 表示按位异或;
- 2 k, 删除最后 k 个元素;
- 3 l r x, 在区间 $[l, r]$ 中统计小于等于 x 的元素个数;
- 4 l r k, 在区间 $[l, r]$ 中找到第 k 小的数.

8.3.2 数据规模和约定

- $1 \leq m \leq 5 \times 10^5$
- $1 \leq x \leq 5 \times 10^5$
- 保证数据合法

时间限制: 1.5s.

8.3.3 分析

把数字当作二进制字符串, 维护一个可持久化 Trie, 并且在 Trie 中存储一个结点下面的元素的个数 (即 size). 设 $trie[i]$ 表示前 i 个数字构成的 Trie, 那么显然我们维护的信息 (size) 是可减的, 因此我们可以很轻松地提取一个区间.

那么显然, 我们可以非常容易地解决 2 和 4 询问, 1 询问我们可以在 Trie 上贪心. 由于我们已经把 Trie 可持久化了, 因此我们同样可以解决 0 和 2 操作.

注意到我们只需要可持久化 Trie 就可以了, 不要思维定势, 觉得区间第 k 大一定需要用可持久化线段树 (其实这里的 Trie 就相当于一个可持久化线段树).

时间复杂度: $O(n \log n)$.

8.4 Children Trips

8.4.1 问题描述

给定一个 n 个结点的树, 其中每条边的长度为 1 或 2.

然后有 m 个人要在这棵树上走, 第 i 个人从 $s[i]$ 出发, 要到 $f[i]$ 去, 每天所能够走过的最大路程为 $p[i]$, 也就是说, 这个人走了 $p[i]$ 的距离 (可以不到 $p[i]$) 就必须要休息一次.

对于每个人, 你需要给出他最少需要多少天才能走完这段路.

8.4.2 数据规模和约定

- $1 \le n \le 10^5$
- 保证数据合法

时间限制: 8s.

8.4.3 分析

首先我们可以发现, 从 x 走到 y 和从 y 走到 x 的最小代价是一样的. 因此暴力的话, 我们可以这么做: 对于一对 (u, v), 我们可以利用倍增往上跳, 然后在 LCA 处进行讨论, 这样的话时间复杂度是 $O(\log n \sum k)$ 的, 其中 $\sum k$ 表示步数之和.

我们还可以考虑一下, 如果 p 是固定的该怎么做. 不妨设 $up[i]$ 表示点 i 往上走 p 步, 最多能走到哪里. 那么对于每一个询问, 我们可以在 LCA 处讨论它的答案: 我们可以用并查集动态维护对于以 i 为根的子树, 每个点最多向上能跳到哪里, 需要多少步 (这只需要在 DFS 退栈时更新一下就可以了), 这样就可以在 LCA 处得到答案了. 时间复杂度是 $O(n)$ 的.

然后我们可以利用平衡规划的思想, 把这两个算法结合一下, 得到一个复杂度较优的算法: 不妨设阈值为 k, 那么对于所有 $p[i] > k$ 的询问, 我们直接暴力用倍增走, 这样的复杂度是 $O(\frac{mn}{k} \log n)$; 而对于所有 $p[i] \le k$ 的询问, 我们枚举 p, 然后用第二种算法来 $O(n)$ 算, 这样的复杂度就是 $O(nk)$ 的.

总的复杂度为 $O(\frac{mn}{k} \log n + nk)$, 取 $k = \sqrt{n \log n}$ 即可使得复杂度最优, 为 $O(n\sqrt{n \log n})$.

考虑到算法实现的常数, 最后大约需要取 $k = \sqrt{n}$(Codechef) 或 $k = 0.038\sqrt{n \log n}$(Tsinsen).

第九章　试题泛做 (9)

9.1 Martial Arts

9.1.1 问题描述

给定一个左边 n 个点, 右边也是 n 个点的完全二分图. 其中每条边有两个权值 a 和 b, 你需要求出一个匹配, 使得匹配边的 a 权值和减去 b 的权值和最大, 并且, 在 $\sum a - \sum b$ 相等的情况下, 你需要最大化匹配边的 a 权值和.

并且, 你有一个对手, 他可以选择删除掉一条匹配边, 并且它的目标是先最大化 $\sum b - \sum a$, 再最大化 $\sum b$.

9.1.2 数据规模和约定

- $1 \leq n \leq 100$
- $0 \leq a[i][j], b[i][j] \leq 10^{12}$

时间限制: 5s.

9.1.3 分析

不妨假设我们已经求出了匹配集合. 然后, 如果 $a - b$ 最大的边的权值是负数, 那么对手是不会删边的. 并且对手如果要删边, 那么删掉的边一定是匹配集合中 $a - b$ 最大的, 并且在这个前提下 a 最小的边.

那么我们可以先把边按照这两种权值排序, 然后枚举最大的边是哪一条, 用权值小于等于它的边做最大匹配, 并强制让它在匹配集上.

我们可以发现, 这些问题实际上都可以归结为修改边权的问题. 一开始令所有边的权值为负无穷大. 那么加入一条边可以看作是把某条边的权值改为某个数, 而强制选一条边则可以看作是把它的权值设为正无穷大, 然后进行匹配.

如果直接这样做, 那么每次修改之后需要重新匹配, 这样的话时间复杂度就变成了 $O(n^5)$, 无法承受.

不妨通过改进 KM 算法来做到快速地在修改边权后维护最大匹配. 假设我们修改的边为 (x, y), 那么, 我们可以让 $label[x]$, 即 x 的标号, 上升到一个更高的位置 (从而让最大的那条边成为等边), 然后我们可以从 x 点进行增广, 并不断地使标号下降, 最终使得等边能够构成一个完美匹配.

这样的时间复杂度是 $O(n^4)$ 的.

9.2 Arithmetic Progressions

9.2.1 问题描述

给定 n 个整数 $a_1, a_2, \ldots, a_n$, Dexter 希望知道有多少种选择三个数的方法使得他们构成一个等差数列.

也就是说, 有多少种三元组 (i, j, k) 满足 $1 \le i < j < k \le n$ 并且 $a_j - a_i = a_k - a_j$.

例如三元组 $(2, 5, 8)$,$(10, 8, 6)$ 和 $(3, 3, 3)$ 是合法的, 因为他们构成了一个等差数列, 而三元组 $(2, 5, 7)$,$(10, 6, 8)$ 不是合法的.

9.2.2 数据规模和约定

- $3 \le n \le 100000$
- $1 \le a_i \le 30000$

时间限制: 3s.

9.2.3 分析

我们考虑 a_i 为等差中项的情况的答案: 这实际上相当于把左边的数构一个多项式并将它和右边的数构成的多项式卷积一下, 取 x^{2a_i} 这一项的系数.

然而直接这样做肯定会超时, 于是我们可以考虑分块: 我们把 i 每隔 k 个分一个块, 然后对于每一组 (a, i, b), 我们分 a 和 b 都不与 i 在一个块的情况, 以及 a 或 b 与 i 在一个块的情况. 对于第一种情况, 我们可以每隔 $\frac{n}{k}$ 个做一次 FFT, 对于后一种情况, 我们可以直接枚举块内的情况, 并维护由 i 左边的数和 i 右边的数构成的多项式的系数.

时间复杂度: $O(\frac{n}{k} m \log m + nk)$, 其中 $m = \max\{a_i\} \le 30000$, 因此当 $k = \sqrt{m \log m}$ 时算法的复杂度最优, 为 $O(n\sqrt{m \log m})$. 最后我选择了 $k = 3000$ 作为阈值.

第十章　试题泛做 (10)

10.1　Future of draughts

10.1.1　问题描述

给定 t 个图, 第 i 个图的点数为 n_i, 边数为 m_i.

有 q 组询问, 每组询问给出三个数 l_i, r_i, k_i, 表示你需要在编号为 l_i 到 r_i 之间的图上进行一次游戏.

首先, 你需要从每个图中选出一个起始点, 然后进行下面的操作至少一次, 至多 k_i 次: 对于每一个图, 要么不移动这个图中的点, 要么将图中的点移动一步. 并且定义成功的游戏为最后每个图中的点都回到起始位置的游戏.

你需要求出成功的游戏的种类数对 10^9+7 取模的值. 两次游戏是不同的, 当且仅当一开始至少有一个图的起始点不同, 或者是某次操作某个点所移动到的点不同 (不移动可以看作是移动到自己).

10.1.2　数据规模和约定

10.1.3　分析

基本思想

定义两个图 $G_1=\langle V_1,E_1\rangle$ 和 $G_2=\langle V_2,E_2\rangle$ 的 Strong Product 表示一个新图 $G_1\boxtimes G_2$, 它的点集为笛卡尔积 $V_1\times V_2$, 对于它的每一对点 (u_1,u_2) 和 (v_1,v_2), 它们之间有边当且仅当下面三个条件中至少有一个成立

1. $(u_1,v_1)\in E_1$ 且 $(u_2,v_2)\in E_2$
2. $u_1=v_1$ 且 $(u_2,v_2)\in E_2$
3. $(u_1,v_1)\in E_1$ 且 $u_2=v_2$

留意到上面这个运算是交换的, 并且在同构意义下是结合的.

下面将用图的记号 G 同时表示这个图和它的邻接矩阵.

不妨先考虑我们进行了恰好 k 次操作的情况. 实际上, 我们要求的答案就是

$$tr[(\boxtimes_{i=l}^{r} G_i)^k]$$

其中 $tr[]$ 表示矩阵的迹, 即矩阵的主对角线上的元素和.

迹与特征值

关于矩阵有如下定理

- $tr[A] = \sum sp(A)$
- $tr[A^k] = \sum sp(A^k) = \sum_{a \in sp(A)} a^k$

Strong Product 的特征值

实际上, $G = G_1 \boxtimes G_2$ 的特征值集合为 $\{(\alpha+1)(\beta+1)-1 \mid a \in sp(G_1), b \in sp(G_2)\}$, 下面给出证明.

Proof. 一个自然的想法是尝试构造特征值 $(\alpha+1)(\beta+1)-1$ 所对应的特征向量.

不妨假设 α 所对应的特征向量为 $\mathbf{a}$, β 所对应的特征向量为 $\mathbf{b}$, 那么我们有

$$G_1\mathbf{a} = \alpha\mathbf{a}$$

$$G_2\mathbf{b} = \beta\mathbf{b}$$

这实际上意味着 (对于 G_1 中的每一个点 j)

$$\sum_{i \sim j} \mathbf{a}_i = \alpha \mathbf{a}_j$$

对于 G_2, 我们也有

$$\sum_{i \sim j} \mathbf{b}_i = \beta \mathbf{b}_j$$

($i \sim j$ 表示 i 与 j 相邻, 下同)

我们将要说明, 我们所需要的特征向量为 $\mathbf{a} \otimes \mathbf{b}$(符号 $\otimes$ 表示向量的外积).

类似地, 对于 G 中的每一个点 (j_1, j_2), 我们有

$$\begin{aligned}\sum_{(i_1,i_2)\sim(j_1,j_2)}(\mathbf{a}\otimes\mathbf{b})_{i_1,i_2} &= \sum_{(i_1,i_2)\sim(j_1,j_2)}\mathbf{a}_{i_1}\mathbf{b}_{i_2}\\ &= \sum_{\substack{i_1\sim j_1\\ i_2\sim j_2}}\mathbf{a}_{i_1}\mathbf{b}_{i_2}+\sum_{i_1\sim j_1}\mathbf{a}_{i_1}\mathbf{b}_{j_2}+\sum_{i_2\sim j_2}\mathbf{a}_{j_1}\mathbf{b}_{i_2}\\ &= \sum_{i_1\sim j_1}\mathbf{a}_{i_1}\sum_{i_2\sim j_2}\mathbf{b}_{i_2}+\mathbf{b}_{j_2}\sum_{i_1\sim j_1}\mathbf{a}_{i_1}+\mathbf{a}_{j_1}\sum_{i_2\sim j_2}\mathbf{b}_{i_2}\\ &= \alpha\mathbf{a}_{j_1}\beta\mathbf{b}_{j_2}+\mathbf{b}_{j_2}\alpha\mathbf{a}_{j_1}\mathbf{a}_{j_1}\beta\mathbf{b}_{j_2}\\ &= (\alpha\beta+\alpha+\beta)(\mathbf{a}\otimes\mathbf{b})_{j_1,j_2}\end{aligned}$$

□

利用上面的结论, 我们可以得到

$$\begin{aligned}tr[(G_1\boxtimes G_2)^k] &= \sum_{\alpha\in sp(G_1)}\sum_{\beta\in sp(G_2)}((\alpha+1)(\beta+1)-1)^k\\ &= \sum_{\alpha\in sp(G_1)}\sum_{\beta\in sp(G_2)}\sum_{i=0}^{k}\binom{k}{i}(\alpha+1)^i(\beta+1)^i(-1)^{k-i}\\ &= \sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}\sum_{\alpha\in sp(G_1)}(\alpha+1)^i\sum_{\beta\in sp(G_2)}(\beta+1)^i\end{aligned}$$

不难将它推广到多个图的 Strong Product 的情况

$$tr[(G_1\boxtimes G_2\boxtimes\cdots\boxtimes G_t)^k]=\sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}\prod_{j=1}^{t}tr[(G_j+I)^i]$$

通往正解

不妨令

$$P(l,r,i)=\prod_{j=l}^{r}tr[(G_j+I)^i]$$

并且记

$$\begin{aligned} f(l,r,i) &= tr[(G_l \boxtimes G_{l+1} \boxtimes \cdots \boxtimes G_r)^k] \\ &= \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} P(l,r,i) \\ &= \sum_{i=0}^{k} (-1)^{k-i} \frac{k!}{i!(k-i)!} P(l,r,i) \\ &= k! \sum_{i=0}^{k} \frac{(-1)^{k-i}}{(k-i)!} \frac{P(l,r,i)}{i!} \end{aligned}$$

通过化简, 我们得到 $f(l,r,i)$ 可以写成一个卷积的形式, 因此在得到 $P(l,r,i)$ 之后, 我们可以用 FFT 来快速计算 $f(l,r,i)$.

由于这道题的模数 (10^9+7) 并不是 NTT 友好的, 因此我们需要用三个质数去做乘法, 然后利用中国剩余定理合并.

剩下的问题就是如何计算 $P(l,r,i)$, 那么问题实际上转化为计算 $tr[(G_i+I)^k]$.

计算 $tr[A], tr[A^2], \ldots, tr[A^k]$

一个直观的想法是我们可以直接计算矩阵 A 的 k 次方然后暴力计算. 然而这样做肯定是不能通过的.

实际上我们可以利用分块的思想来改进暴力算法: 我们先缓存这个矩阵的前 200 次方, 然后留意到 A^k 实际上可以变成 $A^{200a+b} = A^{200a}A^b$, 也就是说我们只需要计算两个矩阵的乘积. 并且由于我们只需要知道对角线上的元素的值, 因此这个计算是 $O(n^2)$ 而不是 $O(n^3)$ 的.

上面的分块做法是题解的评论中有人给出的做法, 实际上我们有一个更高级的做法.

考虑如果一个值 λ 是一个矩阵 A 的特征值意味着什么? 根据定义, 实际上意味着存在向量 $\mathbf{v}$, 使得

$$A\mathbf{v} = \lambda\mathbf{v}$$

即

$$(A - \lambda I)\mathbf{v} = 0$$

注意到这是一个存在性的叙述, 因此上面的式子意味着 $A-\lambda I$ 所对应的线性方程组有解, 也就是

$$\det(A - \lambda I) = 0$$

定义一个矩阵 A 的特征多项式

$$p(\lambda) = \det(A - \lambda I)$$

关于特征多项式有如下的 Caylay-Hamilton 定理

$$p(A) = 0$$

它的证明极其复杂, 下面略去.

考虑一个矩阵的特征多项式 $p(A)$, 实际上它相当于一个以 A 的所有特征值为根的方程, 即

$$(x - \lambda_1)(x - \lambda_2)(x - \lambda_3) \cdots (x - \lambda_n) = 0$$

将它展开, 得到

$$x^n - e_1(\lambda)x^{n-1} + e_2(\lambda)x^{n-2} + \cdots + e_n(\lambda) = 0$$

其中 $e_k(\lambda)$ 表示从 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 中选出 k 个数乘起来的所有乘积之和.

依照 Caylay-Hamilton 定理, 将矩阵 A 带入, 得到

$$A^n - e_1(\lambda)A^{n-1} + e_2(\lambda)A^{n-2} + \cdots + e_n(\lambda) = 0$$

注意到 $A \mapsto tr[A]$ 是一个线性变换, 因此

$$tr[A^n] - e_1(\lambda)tr[A^{n-1}] + e_2(\lambda)tr[A^{n-2}] + \cdots + e_n(\lambda) = 0$$

如果我们先在式子的两边同乘一个矩阵 A, 那么我们将会得到

$$tr[A^{n+1}] - e_1(\lambda)tr[A^n] + e_2(\lambda)tr[A^{n-2}] + \cdots + e_n(\lambda) = 0$$

这告诉我们可以在求出特征多项式与矩阵的前 n 次幂的迹之后用递推的方法得到矩阵的所有次幂的迹.

然而我们要怎么计算 $e_i(\lambda)$ 呢? 不妨令 $e_0(\lambda) = 1$. 我们不难发现, $e_i(\lambda)$ 可以用 $e_0(\lambda), e_1(\lambda), \ldots, e_{i-1}(\lambda)$ 容斥得到.

这里直接给出容斥后的结果

$$ke_k(\lambda) = \sum_{i=1}^{k} (-1)^{i-1} e_{k-i}(\lambda)p_i(\lambda)$$

其中 $p_i(\lambda)$ 表示所有特征值的 i 次方和, 即 $\lambda_1^i + \lambda_2^i + \cdots + \lambda_n^k$.

然后我们之前提到过,$tr[A^k] = \sum sp(A^k)$, 因此,$p_i(\lambda) = tr[A^i]$.

时间复杂度

$O(tn^4 + tn^k + t^2k \log k + q)$

10.2 Simple Queries

10.2.1 问题描述

给定一个含 n 个正整数的数组 a. 现有关于它的 q 个询问，询问有以下五种类型:

- 1 l r: 令 S 为由下标范围从 l 到 r 的不同的元素构成的有序集合. 你需要求出

$$(\sum_{1 \le i < j < k \le |S|} S_i S_j S_k) \pmod{10^9 + 7}$$

- 2 x y: 将下表为 x 的元素赋值为 y
- 3 x: 将下标为 x 的元素从数组中删除
- 4 z y: 在下标为 z 的元素之后插入元素 y, 若 z 等于 0, 则在数组最前端插入
- 5 l r: 输出下标在 l 到 r 范围内到不同元素个数

数组下标从 1 开始, 保证数组总是非空.
时间限制: 2 秒.

10.2.2 数据规模和约定

- $1 \le n, q \le 10^5$
- $1 \le a_i, y < 10^9 + 7$
- 保证操作合法

10.2.3 分析

不妨假设没有插入以及删除, 并且没有 1 操作, 我们要怎么做.

对于数组中的第 i 个元素, 定义 $prev[i]$ 为 a_i 上一次出现的位置, 如果这是 a_i 第一次出现, 则 $prev[i] = 0$.

如果我们把每个元素作为平面上的一个点 $(prev[i], i)$, 那么 5 操作实际上是询问横坐标在 0 到 $l - 1$, 纵坐标在 l 到 r 之间的点的个数 (实际上就是一个矩形).

而修改实际上相当于插入或删除一个点, 因此问题被转化为一个二维询问的问题, 用树状数组套线段树即可解决.

如果新增加 1 操作怎么办? 其实我们只需要在每个点 $(prev[i], i)$ 存一个形如 $(1, a_i, a_i^2, a_i^3)$ 的向量, 然后在询问的时候查询区间中向量的和即可.

现在考虑 2 操作和 3 操作, 我们可以用平衡树预处理出每个点的位置, 然后就可以消除 2 操作和 3 操作所造成的影响了.

线段树需要动态开点, 防止爆内存.

时间复杂度: $O((n+q)\log^2(n+q))$. 鉴于线段树常数巨大, 需要大量常数优化才能通过此题.

对于上面的二维平面询问的问题, 除了树套树外, 题解还给出了几种做法.

二维分块

留意到一个重要的事实: 除了 $x = 0$ 外, 每一个横坐标最多只有一个点.

对于包含 $x = 0$ 的询问可以特判掉 (单独用树状数组存). 下面的讨论将假定每一个横坐标最多只有一个点.

我们可以存一下对于每个横坐标 x, 是否有横坐标为 x 的点, 如果有的话存它所对应的是哪个点.

之后我们可以二维分块, 比如说, 将整个平面分成多个大小为 k^2 的正方形.

我们维护一个二维前缀和 $sum[i][j]$, 表示左下方的 $i \times j$ 个块中的点的和. 它的维护可以用二维树状数组实现.

之后考虑每次询问, 在查询完整块的值之后, 我们实际上会剩下一段横坐标是没有查询的, 那么我们暴力扫这一段横坐标, 然后看看它所对应的点是否在我们询问的矩形范围内.

时间复杂度: $O((n+q)(\sqrt{n+q} + \log^2(n+q)))$.

由于常数小, 这种方法可以跑得很快 (这也是出题人使用的算法).

二维分块 (cont'd)

然而我们还有另外一种分块的方法.

我们可以类似上面的做法, 只不过我们不再需要使用树状数组, 我们可以每隔 $\sqrt{n+q}$ 的时间重建一次前缀和数组.

时间复杂度和上面的差不多, 其实这应该是最容易实现的做法, 不过我没有看到有通过这道题的人使用了这种算法.

10.3 Dynamic Trees and Queries

10.3.1 问题描述

给你一棵初始有 n 个结点的树, 其中每个点有一个权值, 点的编号从 0 开始. 你需要支持 m 个询问, 询问有以下四种类型

1. 给定 k 和 v, 你需要加入一个新的点 (从 n 开始编号), 它的权值为 v, 并且它是点 k 的儿子;
2. 给定 k 和 v, 你需要把以点 k 为根的子树中的所有点的权值加上 v;
3. 给定 k, 你需要删除以 k 为根的子树中的所有点;
4. 给定 k, 你需要输出以 k 为根的子树中的点的权值和.

10.3.2 数据规模和约定

- $1 \leq n, m \leq 10^5$
- 强制在线
- 保证数据合法

时间限制: 2s.

10.3.3 分析

这道题直接用 ETT 做就可以了. 也就是说, 我们维护一个这样的 Splay: 对于树中的每个点 i, 我们用两个 Splay 中的点 $left[i]$ 和 $right[i]$ 来表示它, 使得 $left[i]$ 在 Splay 的编号为 i 的入栈时刻, $right[i]$ 在 Splay 的编号为 i 的出栈时刻.

那么询问就很好做了. 加入/删除/询问一个子树我们可以把以 $left[i]$ 和 $right[i]$ 为端点的区间 Splay 出来, 然后再做. 插入的话我们直接插进去就可以了.

时间复杂度: $O(n + m \log n)$.

10.4 Sereja and Subsegment Increasings

10.4.1 问题描述

给定两个长度为 n 的数组 a 和 b. 每一次操作你可以选择一个区间 $[l, r]$, 然后 $\forall_{i \in [l,r]} a[i] = (a[i] + 1) \bmod 4$. 你需要用最少的操作次数把 a 变成 b.

10.4.2　数据规模和约定

- 多组数据 (不超过 10 组)
- $1 \le n \le 10^5$
- $0 \le a[i], b[i] < 4$

时间限制: 1s.

10.4.3　分析

不妨设一个新的数组 c, 其中 $c[i] = (a[i] - b[i]) \bmod 4$. 那么, 我们相当于要把一个全 0 的数组通过区间加一的操作来变成 c.

在没有模的情况下, 答案显然是 $\sum_{i=1}^{n} \max(0, c[i] - c[i-1])$.

现在我们令 $c[0] = c[n+1] = 0$, 然后设 $d[i] = c[i] - c[i-1]$, 其中 $i = 1, 2, \ldots, n+1$. 那么, 我们就相当于, 对于任意的 $i < j$, 我们可以令 $d[i] += 4, d[j] -= 4$(相当于把 c 数组的一个区间加上 4), 然后我们要最小化 $\sum_{i=1}^{n+1} \max(0, d[i])$.

这可以通过贪心解决, 首先, 对于所有的 $-1 \le d[i] \le 1$, 我们可以发现, 此时改变 $d[i]$ 一定不是最优方案, 所以我们应该让它不动. 而对于 $2 \le d[i] \le 3$, 我们可以找到它前面的 $-2, -3$, 然后先尝试让 $d[i]$ 和 -3 匹配, 不行就和 -2 进行匹配, 然后更新数组 d, 最后求一遍答案.

上面这个问题还有一种做法: 不妨猜测 (.....) 对于每一个 $c[i]$, 我们最多只会把它加一次 4, 这样我们就可以用 DP 算出来最优值了.

时间复杂度: $O(n)$.

第十一章　试题泛做 (11)

11.1 Little Elephant and Boxes

11.1.1 问题描述

给定 n 个盒子, m 个物品. 打开第 i 个盒子有 $p[i]$ 的概率获得 $v[i]$ 元, 有 $1-p[i]$ 的概率获得 1 颗宝石. 购买第 i 个物品需要 $c[i]$ 元和 $d[i]$ 颗宝石. 求在你最大化买到的物品数的情况下, 你所能得到的物品数的期望.

11.1.2 数据规模和约定

- 多组数据 (不超过 5 组)
- $2 \le n \le 30$
- $1 \le m \le 30$
- $1 \le v[i], c[i] \le 10^7$
- $0 \le d[i] \le 30$
- $0 \le p[i] \le 1$

时间限制: 4s.

11.1.3 分析

这道题的一个比较暴力的做法是直接 $O(2^n)$ 枚举打开盒子后得到的东西, 然后用 DP 算出最多能够得到多少物品. 那么对于这种规模比较大的暴力题, 我们可以尝试使用 Meet in the Middle 算法.

首先我们求出 $f[i][j]$ 表示有 j 个宝石, 要买 i 个物品, 最少需要多少元. 然后我们按照阈值 λ 进行分块: 不妨先暴力枚举前 λ 个盒子的情况, 设当前枚举到的情况的钻石数为 a, 钱数为 b, 概率为 p, 然后枚举答案 j, 再枚举后 $n-\lambda$ 个盒子所能够得到的宝石数 k, 此时我们可以发现,

对于后 $n-\lambda$ 个盒子的所有状态中能够获得 k 颗宝石的那些状态, 它所对应的钱数应该在区间 $[f[j][a+k]-b, f[j+1][a+k]-b)$ 之间, 因为如果少了我们就买不了 j 个物品, 如果多了我们就不止买 j 个物品.

于是我们可以预处理后 $n-\lambda$ 个盒子的状态把它们按照 k 分类并按照钱数排序, 然后我们会发现, 枚举完前 λ 个盒子的状态后, 合法的后 $n-\lambda$ 个盒子的状态对应一个连续的区间, 于是我们可以用前缀和来优化算法.

时间复杂度: $O((n-\lambda)2^{n-\lambda}+nm2^{\lambda}(n-\lambda))$, 大约取 $\lambda=\max\left(1,\left\lfloor\frac{n}{3}\right\rfloor\right)$ 即可通过本题.

第十二章　试题泛做 (12)

12.1　Course Selection

12.1.1　问题描述

有 n 个课程, m 个学期, 在学期 j 学习第 i 门课程可以获得 $x[i][j]$ 的收益, 当收益为-1 时表示这个学期不开这门课程. 并且有 k 个约束, 表示某一个课程 i 是另一个课程 j 的先修课 (i 至少要比 j 提前一个学期学习).

你需要输出最大的收益.

12.1.2　数据规模和约定

- $1 \leq n, m \leq 100$
- $0 \leq k \leq 100$
- $-1 \leq x[i][j] \leq 100$

时间限制: 1s.

12.1.3　分析

这道题是非常经典的最小割模型.

我们先把最大化收益变成最小化代价, 然后在学期 j 学习课程 i 的代价就变成了 $sum[i] - x[i][j]$, 其中 $sum[i] = \sum_{j=1}^{m} x[i][j]$. 最后的答案就等于 $\sum_{i=1}^{n} sum[i] - \text{MaxFlow}$.

我们可以把课程看作是一条 s 到 t, 含有 m 条边的路径. 然后对于每一个 x 要在 y 前面的限制, 我们连一条边来表示不能让 x 在 y 后面, 之后跑一遍最小割就可以了.

时间复杂度: $O(\text{MaxFlow}(nm, O(nm + km))$.

第十三章　试题泛做 (13)

13.1 Social Network

13.1.1 问题描述

给定 m 个整数 $L_{x,y}$, 我们想要找到 n 个整数 P_i 和 n 个整数 Q_i, 使得对于这 m 个整数中的每一个,$W_{x,y} = L_{x,y} + P_x - Q_y$ 处在区间 $(S_{x,y}, T_{x,y})$ 内. 在满足以上约束的同时最大化 $\sum W_{x,y}$, 并给出一组合法的 P 和 Q.

13.1.2 数据规模和约定

- $1 \leq T \leq 10$
- $1 \leq n \leq 100$
- $1 \leq m \leq n^2$
- $1 \leq x, y \leq n$
- $|L_{x,y}| \leq 600$
- $|S_{x,y}|, |T_{x,y}| \leq 1000$
- $0 \leq P_i, Q_i \leq 10^6$

13.1.3 分析

对于线性规划, 如果它的一个原问题为如下形式: 在满足 $A\mathbf{x} \leq \mathbf{b} \wedge \mathbf{x} \geq \mathbf{0}$ 的条件下, 最大化 $\mathbf{c}^T\mathbf{x}$.

那么它的对偶问题具有如下形式: 在满足 $A^T\mathbf{y} \geq \mathbf{c} \wedge \mathbf{y} \geq \mathbf{0}$ 的条件下, 最小化 $\mathbf{b}^T\mathbf{y}$.

而如果原问题中没有 $\mathbf{x} \geq \mathbf{0}$ 这个限制, 那么对偶之后的问题就可以把 $A^T\mathbf{y} \geq \mathbf{c}$ 变成 $A^T\mathbf{y} = \mathbf{c}$, 也就是把不等号变成等号.

对于每一个形如 $S_{x,y} \leq L_{x,y} + P_x - Q_y \leq T_{x,y}$ 的限制, 我们将它重写为两个不等式,$P_x - Q_y \leq$ 和 $Q_y - P_x \leq$.

之后我们的问题具有如下特点:A 矩阵 (约束矩阵) 的每一行恰好包含一项正项, 一项负项.

这个问题不好解决, 我们考虑它的对偶问题. 对偶之后, 一个变量恰好出现在两个等式中, 并且一正一负.

是不是很像经典的线性规划转网络流构图? 没错! 我们只需要将对偶后的变量看作一条边, 一个等式看作一个点, 那么**等式**实际上和网络流的流量平衡是相似的.

注意到上面我使用了**等式**一词, 这里为什么从不等式变成等式了呢? 考虑对偶之后的约束 $A^T\mathbf{y} \geq \mathbf{c}$, 如果我们把它的左边加起来, 我们将会得到 0. 而 $\mathbf{c}$ 实际上形如 $Row[1], \ldots, Row[n], -Col[1], \ldots, -Col[n]($ 难得到). 因此我们将约束的右边加起来得到的也是 0. 因此所有的不等式都成了等式.

并且你也可以这样考虑: 如果一组解有负数, 那么我们可以通过把 P 和 Q 同时加上一个数来在把解变正的同时不改变目标函数的值. 因此原问题不需要有非负限制.

考虑到我们还要最小化目标函数, 我们只需要把上面的建模改造成费用流, 然后跑最小费用最大流即可.

如果建出来的图有负环, 那么说明原问题无解.

之后我们还需要构造一组解.

考虑在流网络中, 某一条边有流量, 这就意味着这个限制是有效的, 因此我们就需要增加一个方程. 把这些方程, 联立上原图中的不等式, 就得到了一个方程-不等式组, 我们可以把它看作一个差分约束问题, 只需要求解这个差分约束问题即可得到一组解.

时间复杂度 $O(MaxFlow(n, m) + n^3)$.

13.2　Push the Flow!

13.2.1　问题描述

给定一个有 n 个点, m 条边的无向连通图, 其中每个点属于至多一个简单环. 每一条边有一个容量. 你需要回答 q 个询问, 询问有两种类型

- $0\ s\ t$: 询问 s 到 t 的最大流;
- $1\ x\ y$: 将第 x 条边的容量修改为 y.

13.2.2　数据规模和约定

- $1 \leq n \leq 10^5$
- $0 \leq m, q \leq 2 \times 10^5$

- 保证数据合法

时间限制: 1s.

13.2.3 分析

题目中给的图大致相当于仙人掌图, 不过它的条件比仙人掌要强. 其实就算是仙人掌也是可以做的.

注意到最大流等于最小割, 那么我们的问题转化为找最小的割边.

先考虑没有修改的情况: 此时对于 s 到 t 的路径, 当它经过一条边的时候, 这条边对答案的贡献就是这条边的容量; 而当它经过一个环的时候, 那么首先, 这个环肯定有一条容量最小的边 e, 并且当它经过一个环的时候, 必定有一条路径经过 e, 另一条不经过 e, 因此我们可以把 e 的容量加到环的其他边上去, 然后删掉 e 这条边.

把环上的容量最小的边删掉之后, 我们就相当于要维护这个仙人掌的最大生成树. 我们考虑怎么修改边权: 如果它不在一个环上, 那么直接改可以了; 否则我们可以先把环上容量最小的边所造成的影响撤销, 然后找到新的容量最小的边, 并重新维护这个环的形态.

时间复杂度: $O((n+q)\log n)$.

13.3 Team Sigma and Fibonacci

13.3.1 问题描述

给定 n 和 m, 求

$$(\sum_{x,y,z\geq 0} 6xyz \times Fib[x] \times Fib[y] \times Fib[z]) \bmod m$$

其中, $Fib[k]$ 表示 Fibonacci 数列第 k 项, 并且 $Fib[0]=0$, $Fib[1]=1$.

13.3.2 数据规模和约定

- 各组数据的 m 之和不超过 10^6
- $0 \leq n \leq 10^{18}$
- $1 \leq m \leq 10^5$

时间限制: 6s.

13.3.3 分析

官方题解比较奇怪...

下面给出一种利用生成函数的做法, 不妨设数列 $i \times Fib[i]$ 的生成函数为 $g(x)$, 而答案的生成函数为 $f(x)$, 则

$$g(x) = \frac{x(x^2+1)}{(1-x-x^2)^2}$$

这可以通过对 Fibonacci 数列的生成函数 $\frac{x}{1-x-x^2}$ 求导, 再乘上 x 得到.

而

$$f(x) = 6g(x)^3 = \frac{6x^3(x^2+1)^3}{(1-x-x^2)^3}$$

而 $ans = [x^n]f(x)$, 因此问题转化为求一个有理函数的 n 次项的系数.

齐次递推关系与生成函数

事实上, 每一个形如 $f(x) = \frac{P(x)}{Q(x)}$ 的生成函数都对应一个齐次递推数列, 不妨假设这个数列的递推式为 $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$, 那么

$$\begin{array}{rccccccc}
f(x) = & a_0 + & a_1 x + & a_2 x^2 + & \cdots + & a_k x^k + & \cdots \\
c_1 x f(x) = & & c_1 a_0 x + & c_2 a_1 x^2 + & \cdots + & c_1 a_{k-1} x^k + & \cdots \\
c_2 x^2 f(x) = & & & c_2 a_0 x + & \cdots + & c_2 a_{k-2} x^k + & \cdots \\
\vdots & & & & & & \\
c_k x^k f(x) = & & & & & c_k a_0 x^k + & \cdots
\end{array}$$

用第一行减去下面的所有行, 可以得到

$$(1 - c_1 x - c_2 x^2 - \cdots - c_k x^k) f(x) = P(x)$$

其中 $P(x)$ 为某个次数小于 k 的多项式.

将这个过程反过来, 我们就可以从生成函数出发, 得到递推关系. 并且我们留意到一件非常重要的事情, 如果有理生成函数 $f(x)$ 的分母形如

$$Q(x) = 1 - c_1 x - c_2 x^2 - \cdots - c_k x^k$$

那么, $f(x)$ 所对应的数列具有如下递推式

$$a_n = \sum_{i=1}^{k} c_i a_{n-i}$$

是不是很像卷积的形式?

并且, 知道了 $P(x)$ 和 $Q(x)$, 我们同样可以轻松地推出数列的前 k 项, 知道了前 k 项和递推关系, 我们可以轻松地利用矩阵快速幂求出数列的第 n 项, 这样做的复杂度是 $O(k^3 \log n)$ 的.

矩阵快速幂的优化

下面讲一讲怎么把这个做法优化到 $O(k^2 \log n)$, 这个做法在郭晓旭的线性递推关系与矩阵乘法中提到.

显然, 上面的递推关系所对应的矩阵为

$$A = \begin{pmatrix} c_1 & c_2 & c_3 & \cdots & c_{k-1} & c_k \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

考虑它的特征多项式

$$p(\lambda) = |\lambda I - A| = \begin{vmatrix} \lambda - c_1 & -c_2 & -c_3 & \cdots & -c_{k-1} & -c_k \\ -1 & \lambda & 0 & \cdots & 0 & 0 \\ 0 & -1 & \lambda & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda & 0 \\ 0 & 0 & 0 & \cdots & -1 & \lambda \end{vmatrix}$$

展开得到 (不妨先利用第一行降维, 然后就剩下上三角行列式了), $p(\lambda) = \lambda^k - c_1\lambda^{k-1} - c_2\lambda^{k-2} - \cdots - c_k$.

根据 Cayley-Hamilton 定理, $p(A) = \mathbf{0}$.

因此, 矩阵 A 的所有幂都可以表示成 $I, A, A^2, \ldots, A^{k-1}$ 的线性组合. 于是我们可以用一个以 A 为自变量的 k 次多项式来表示矩阵 A 的一个幂, 不妨用 $poly_k$ 来表示这个多项式.

注意到 $A^{2k} = A^k A^k = poly_k^2$, 因此, 我们可以把矩阵快速幂的过程变成多项式乘法, 之后再利用递推关系把 $2k$ 次的多项式 $poly_k^2$ 降到 k 次多项式 $poly_{2k}$, 这样就可以在 k^2 的复杂度内完成一次矩阵乘法 (实际上变成了多项式乘法) 的过程.

最后我们会得到一个 $n-1$ 次多项式 $poly_n$, 它意味着

$$A^n = poly_n(A) = \sum_{i=0}^{k-1} b_i A^i$$

将上式右乘上我们矩阵快速幂的初始变量 $\mathbf{x}$, 得到

$$A^n\mathbf{x} = \sum_{i=0}^{k-1} b_i A^i\mathbf{x}$$

注意到 a_n 实际上等于 $A^n\mathbf{x}$ 的最后一维, 因此我们可以得到

$$a_n = \sum_{i=0}^{k-1} b_i a_i$$

此即我们要求的答案.

结论

由于询问太多, 因此本题在实现时最后还要缓存下模数小于 200 的答案的循环节.

时间复杂度: 单次询问 $O(k^2 \log n)$, 其中 $k = 13$.

第十四章　试题泛做 (14)

14.1　Devu and Locks

14.1.1　问题描述

给定 n, p, m, 对于每一个 $1 \leq i \leq m$, 求所有是 p 的倍数, 并且各位数字之和为 i 的 n 位十进制数的个数.

14.1.2　数据规模和约定

- $1 \leq n \leq 10^9$
- $1 \leq p \leq 16$
- $1 \leq m \leq 15000$

时间限制　15 秒

14.1.3　分析

我们先来看看暴力怎么做.

设 $f[i][j][k]$ 表示只考虑后 i 位, 模 p 为 j, 并且各位数字之和为 k 的方案数.

那么很显然, 对于所有 $10^i \bmod p$ 相同的 i, 那么在那一位的转移是相同的.

因此我们可以统计出 $cnt[i]$ 表示满足 $10^j \bmod p = i$ 的 j 的方案数.

我们现在考虑不同的 $cnt[i]$, 不妨设 $g[i][j]$ 表示总共有 $cnt[i]$ 个位给我们放, 我们在这些位所放的各位数字之和为 j.

求出 g 之后, 我们重新令 $f[i][j][k]$ 中的 i 表示考虑完前 i 个 $cnt[i]$ 的答案, 那么我们可以这么转移

$$f[i][j][k] = \sum_{t=0}^{\min(k,cnt[i])} f[i-1][j - t * 10^i \bmod p][k-t] \times g[i][t]$$

注意到上式是一个二维卷积的形式, 因此我们可以用二维 FFT 来加速转移.

下面的问题就是怎么求 g, 那么 $g[i][j]$ 相当于是把 j 个相同的元素放进 $cnt[i]$ 个不同的桶, 并且每个桶中的元素个数小于 10.

我们可以考虑容斥. 不妨设我们现在有 k 个桶的元素个数大于等于 10, 那么

$$g[i][j] = \sum_{k=0}^{k\leq \frac{j}{10}} (-1)^k \binom{cnt[i]}{k} \binom{j-10k+cnt[i]-1}{cnt[i]-1}$$

注意到, 上式 $\sum$ 中的项实际上表示先从 $cnt[i]$ 个桶中选出元素个数大于等于 10 个的桶, 然后剩下的任意放的个数.

时间复杂度: $O(pm^2 + p^2 m \log m)$.

14.2 Query on a tree VI

14.2.1 问题描述

给定一个 n 个点的树, 每个结点有标号 (从 1 到 n) 和颜色 (黑和白). 根结点的标号是 1.

初始时, 所有结点都是白色, 给定 m 个操作, 每个操作可能是

- 0 u, 询问 u 所在的连通块中有多少个结点, 其中, 两个结点被认为是连通的, 当且仅当连接这两个点的路径上的点的颜色都相同;
- 1 u, 将 u 反色 (即白变黑, 黑变白).

14.2.2 数据规模和约定

$1 \leq n, m \leq 10^5$

时间限制: 1s.

14.2.3 分析

比较传统的做法是用一棵 LCT 来维护整棵树的信息, 然而这样的话我们就需要一棵能够维护虚边信息的 LCT, 比较难写.

下面介绍一种用 Euler Tour Tree 的做法.

首先我们要知道 ETT 是什么. ETT 是一棵维护整棵树的 DFS 回路的 Splay, 注意, 这里的 DFS 回路指的是 LCA 转 RMQ 的时候用到的那种回路 (好吧我也不知道叫什么.....).

它支持的操作有: 连边, 删边, 旋根 (我们用不到), 以及询问子树相关的信息.

我们可以维护两棵 ETT, 其中 $tree[0]$ 维护的是白色的信息, $tree[1]$ 维护黑色的信息. 对于 Splay 中的每个点我们维护一个 $size$ 和 val, 其中 val 表示这个 (Splay 中的) 点是否是它所代表的 (原树中的) 点在 DFS 回路的第一次出现, $size$ 表示以它为根的子树的 val 和.

不妨以 1 为根将树转化为有根树, 设结点 i 的父亲为 $fa[i]$, 并且令 $fa[1] = 0$. 我们可以把一个点的颜色和它在对应颜色的 ETT 中是否与父亲连通关联起来, 比如说, 结点 i 如果是黑色, 那么它在 $tree[1]$ 中就是与 $fa[i]$ 连通的, 而在 $tree[0]$ 中它就与 $fa[i]$ 不连通.

此时我们可以发现, 设点 i 的颜色为 $col[i]$, 点 j 为 $tree[col[i]]$ 的根结点 (深度最小的结点) 朝 i 这个方向走一步所到达的结点, 那么我们询问 i 的时候就相当于询问以 j 为根的子树在 $tree[col[i]]$ 中的大小, 求结点 j 可以通过倍增来实现, 而修改颜色就相当于一次删边, 一次加边.

并且这个做法可以很容易地推广到 QTREE7(在本题的基础上新增询问连通块中的权值最大的元素).

时间复杂度: $O(n \log n)$.

第十五章　试题泛做 (15)

15.1　Counting on a Tree

15.1.1　问题描述

给定一个包含 n 个结点的有标号有边权无根数, 从 1 开始标号.

你的任务是计算有多少无序对 $(s,t), s \neq t$, 满足连接 s,t 两点的路径上, 所有边权的最大公约数等于 1.

边的权值可能被修改, 给定 q 组询问, 其中第 i 组询问形如 a_i, c_i, 表示第 a_i 条边的权值修改为 c_i. 对于每一组询问, 输出改动后对应的答案.

15.1.2　数据规模和约定

- 边权小于等于 10^6
- $0 \leq q \leq 100$
- $1 \leq n \leq 10^5$
- 保证数据合法

15.1.3　分析

如果没有修改操作, 那么我们可以利用 Möbius 反演来计算答案.

设 $f(i)$ 表示路径上边权的最大公约数为 i 的答案, $g(i)$ 表示路径上边权的最大公约数为 i 的倍数的答案.

那么

$$g(i) = \sum_{i|j} f(j)$$

因此

$$f(i) = \sum i \mid j\mu(j)g(j)$$

由于边权的范围有限, 因此, 我们只需要对于每一个 i 求出 $g(i)$ 就可以了.

求 $g(i)$ 我们可以这么做, 令 ans 表示答案.

不妨令 $edge[i]$ 表示所有边权能够被 i 整除的边的集合. 在求 $g(i)$ 的时候, 我们只保留这些边. 然后我们用一个并查集来维护每个连通块的大小.

那么 $g(i) = \sum_j \binom{size[j]}{2}$, 其中 j 取遍所有的连通块. 所以我们只需要在并查集合并的时候顺便维护 $\binom{size}{2}$ 的和就可以了.

那么 $g(i)$ 对 ans 的贡献就是 $\mu(i)g(i)$.

现在考虑有修改的情况, 令 $ans[i]$ 表示第 i 次询问后 (下面称为时刻 i) 的答案.

不妨假设是前 q 条边被修改了. 我们还是考虑现在在求 $g(i)$, 那么我们可以在并查集中先把后面的 $n-1-q$ 条边连好, 之后对于每一个时刻 j, 我们用一个 q 维向量 $\mathbf{v}$, 其中 $\mathbf{v}_k$ 表示第 k 条边此时是否在 $edge[i]$ 中.

因此我们可以在并查集中不采用路径压缩, 仅仅使用按秩合并, 然后再维护一个栈, 这样我们就可以撤销最后一次操作了.

因此我们在做第 i 个时刻的时候, 把上次的影响撤销掉就可以了, 这样我们就可以快速得到 $g(i)$ 对 $ans[j]$ 的贡献.

时间复杂度: $O(m + (n+q^2)\log n)$, 其中 m 表示边权的最大值.

15.2 Random Number Generator

15.2.1 问题描述

给出线性齐次递推数列 a 的前 k 项以及 $c_{1\ldots k}$, 并且 a_i 满足

$$a_i = (c_1 a_{i-1} + c_2 a_{i-2} + \cdots + c_k a_{i-k}) \bmod 104857601$$

求 a_n.

15.2.2 数据规模和约定

- $1 \le n \le 10^{18}$
- $1 \le k \le 30000$

时间限制: 15s.

15.2.3　分析

一个直观的想法是直接套 SIGFIB 中我们求递推数列第 n 项的做法, 但是本题的 k 太大, 直接 $O(k^2 \log n)$ 做会超时.

不妨观察这个做法, 我们实际上是要把一个 $2k$ 次的多项式利用 $A^i = \sum c_j A^{i-j}$, 来把它的次数降到 k 一下.

其实我们可以发现, 如果令初始的多项式为 $A(x)$, 特征多项式

$$B(x) = x_k - c_1 x_{k-1} - c_2 x_{k-2} - \cdots - c_k$$

那么我们实际上就相当于要求 $A(x) \bmod B(x)$.

考虑到模运算关于乘法的分配率, 我们实际上要求的是

$$x^n \bmod B(x)$$

那么我们可以通过倍增得到它.

整个算法的时间复杂度是 $O(k \log k \log n)$ 的.

多项式求逆

我们可以利用牛顿迭代得到多项式 $A(x)$ 在模 x^n 意义下的逆元 $B(x)$.

很显然, 在模 x 意义下, $B(x)$ 就是 $A(x)$ 的常数项的逆元.

不妨假设我们已经得到了

$$A(x)B(x) \equiv 1 \pmod{x^n}$$

我们将设法得到

$$A(x)B(x) \equiv 1 \pmod{x^{2n}}$$

$$
\begin{aligned}
A(x)B(x) &\equiv 1 \pmod{x^n} \\
A(x)B(x) - 1 &\equiv 0 \pmod{x^n} \\
(A(x)B(x) - 1)^2 &\equiv 0 \pmod{x^{2n}} \\
A(x)^2B(x)^2 - 2A(x)B(x) + 1 &\equiv 0 \pmod{x^{2n}} \\
A(x)(A(x)B(x)^2 - 2B(x)) &\equiv -1 \pmod{x^{2n}} \\
A(x)(B(x) - A(x)B(x)^2) &\equiv 1 \pmod{x^{2n}}
\end{aligned}
$$

因此, 令 $B(x) = B(x) - A(x)B(x)^2 \bmod x^{2n}$ 迭代下去即可.

这个做法的时间复杂度是 $T(n) = O(n \log n) + T(\frac{n}{2})$, 也就是 $O(n \log n)$ 的.

多项式除法

给出多项式 $A(x)$, $B(x)$, 求出 $D(x)$, $R(x)$, 使得

$$A(x) = B(x)D(x) + R(x)$$

并且

$$\deg D = \deg A - \deg B$$

$$\deg R < \deg B$$

其中 $\deg A$ 表示 A 的最高次项的次数.

类比整数的带余除法, $D(x)$ 我们称之为商, 而 $R(x)$ 我们称之为余数.

下面这个做法我是从 Picks 的博客上看到的.

不妨令 $n = \deg A$, $m = \deg B$, 并设 $A^R(x)$ 表示 $x^{\deg A}A(\frac{1}{x})$, 相当于把 $A(x)$ 的系数反过来之后得到的一个新的多项式.

我们可以发现, 如果我们把最初的式子整个做一个系数反转的话, 我们将会得到

$$
\begin{aligned}
A^R(x) &= B^R(x)D^R(x) + x^{n-m+1}R^R(x) \\
A^R(x) &\equiv B^R(x)D^R(x) \pmod{x^{n-m+1}} \\
D^R(x) &\equiv A^R(x)[B^R(x)]^{-1} \pmod{x^{n-m+1}}
\end{aligned}
$$

注意到 $D(x)$ 的次数小于 $n-m+1$, 因此它是不会受到模 x^{n-m+1} 的影响的, 因此我们直接把 $D^R(x)$ 反转就可以得到 $D(x)$ 了.

然后我们把 $D(x)$ 代入 $A(x)=B(x)D(x)+R(x)$ 中, 就可以得到 $R(x)$ 了.

15.3 Fibonacci Numbers on Tree

15.3.1 问题描述

定义布尔函数为形如 $f:\{0,1\}^n\mapsto\{0,1\}$, 即一个定义在 n 位 01 向量上, 并且取值为 01 的函数.

- Z 集合为所有满足 $f(0,\ldots,0)=0$ 的布尔函数的集合;
- P 集合为所有满足 $f(1,\ldots,1)=1$ 的布尔函数的集合;
- D 集合为所有满足 $\neg f(x_1,x_2,\ldots,x_n)=f(\neg x_1,\neg x_2,\ldots,\neg x_n$ 的布尔函数的集合;
- A 集合为所有满足如下条件的集合: 若 $f(x_1,\ldots,x_{i-1},a,x_{i+1},\ldots,x_n)=f(x_1,\ldots,x_{i-1},b,x_{i+1},\ldots,x_n)$ 则 $f(y_1,\ldots,y_{i-1},a,y_{i+1},\ldots,y_n)=f(y_1,\ldots,y_{i-1},b,y_{i+1},\ldots,y_n)$ a, b 都位于向量的第 i 维, 上式对于任意的 $i,x_1,\ldots,x_n,y_1,\ldots,y_n,a,b$ 都成立.

现在给你一个由 Z,P,D,A,v,^,!,(,),\组成的合法表达式, 你需要求出这个表达式所表示的集合的元素个数对 1000003 取模的结果.

其中

- v 表示并集;
- ^ 表示交集;
- ! 表示补集, 即一个集合对于全体函数所构成的集合的补集;
- \表示差集, 例如 $A\backslash B$ 表示所有在 A 中且不在 B 中的元素所构成的集合.

并且括号的优先级最高, 其次是!, 最后是 v 和 ^ 和\, 并且它们的优先级是一样的.

15.3.2 数据规模和约定

- $1\leq n\leq 100$
- 总共有至多 100 个表达式
- 单个表达式至多 100 个字符

15.3.3 分析

我们不妨把所有的函数分成 0 到 15 共计 16 个集合, 其中集合编号的二进制高位到低位依次表示是否满足性质 Z, P, D, A. 例如, 集合 5=0101 表示不满足性质 Z, D 且满足性质 P, A 的函数所构成的集合.

很显然每个函数所处的集合是唯一确定的.

不妨第 i 种集合的元素个数为 $f[i]$.

显然, 用 Z, P, D, A 任意运算的集合肯定是由这 16 个集合中的某些并起来得到的. 因此我们可以用一个 16 位的二进制数来表示一个集合, 再把集合的运算对应到位运算上来做. 然后问题就相当于解析一个表达式, 这是一个经典问题, 可以通过维护符号栈和数值栈来解决.

接下来我们的问题就相当于为求 $f[i]$. 直接求 $f[i]$ 并不太简单, 我们不妨设 $g[i]$ 表示 i 这个二进制中的集合的交的大小, 比如说 $g[5] = g[0101]$ 就是集合 $P \cap A$ 的大小.

那么

$$f[i] = g[i] - \sum_{i \subsetneq j} f[j]$$

而求 $g[i]$ 就没有什么好的方法, 只能枚举了. 下面我们一一讨论所有的情况, 注意到 Z 和 P 在很多情况下是可以互换的, 这可以帮助我们简化推导过程.

- 全集: 2^{2^n}, 因为自变量有 2^n 种取值, 而对于每个自变量我们可以决定它所对应的函数值;
- Z 或 P: 2^{2^n-1}, 因为我们在全集的基础上确定了一个自变量的取值, 而 Z 和 P 的大小显然是一样的;
- D: $2^{2^{n-1}}$, 因为我们在全集的基础上确定了一半的自变量的取值 (另一半可以推出来);
- A: 2^{n+1}, 从题目中给的条件可以发现, 每一位对答案要么不贡献, 要么恰好使答案翻转, 所以所有 A 集合中的函数都是把自变量的某几维异或起来, 再异或上 0 或 1;
- $Z \cap P$: 2^{2^n-2}, 因为我们确定了两个自变量的取值;
- $Z \cap D$ 或 $P \cap D$: $2^{2^{n-1}-1}$, 因为我们在 D 的基础上确定了一对函数的值;
- $Z \cap A$ 或 $P \cap A$: 2^n, 因为 A 集合中的函数是由一些变量异或起来的, 而当我们决定完哪些变量需要异或起来之后, 我们可以知道它是否需要异或 1;
- $D \cap A$: 2^n, 和 $Z \cap A$ 的情况类似, 我们也可以在最后知道它是否要异或 1;
- $Z \cap P \cap D$: 和 $Z \cap D$ 一样, 因为 $Z \wedge D \implies P$;
- $Z \cap P \cap A$: 2^{n-1}, 首先, 作为一个 A 中的函数, 它最后异或的常数项必须是 0(因为 Z 性质), 同时, 注意到我们必须要选奇数个数出来进行异或;

- $Z \cap D \cap A$ 或 $P \cap D \cap A$: 2^{n-1}, 这和 $Z \cap P \cap A$ 的情况是类似的;
- $Z \cap P \cap D \cap A$: 2^{n-1}, 和 $Z \cap D \cap A$ 一样, 因为 $Z \wedge D \implies P$;

时间复杂度 $O(\log n + |s|)$.

15.4 Chef and Churu

15.4.1 问题描述

给你一个含 n 个元素的数组 $a[n]$, 以及 n 个函数, 第 i 个函数会返回 l_i, r_i 之间的函数和.

有 q 组询问, 每组询问形式为下面两者之一:

- 给出 x, y, 将 $a[x]$ 的值修改为 y;
- 询问编号在 x 和 y 之间的函数值的和.

15.4.2 数据规模和约定

- $1 \le n, q \le 10^5$
- $1 \le a[i] \le 10^5$

时间限制: 2.5s.

15.4.3 分析

不妨考虑分块维护函数, 设每个块中有 k 个函数. 设第 i 个块的函数值的和为 $sum[i]$, 并且设 $a[j]$ 对 $sum[i]$ 的贡献为 $coef[i][j] \times a[j]$, 我们在一开始的时候可以很容易地计算出 $sum[i]$ 和 $coef[i][j]$.

那么在修改 $a[j]$ 的时候, 我们可以暴力维护出它对每一个 $sum[i]$ 的影响. 而在询问一个区间中的函数值的和的时候, 对于两端的函数我们可以暴力 (我们只需再用树状数组维护一下数组 a 的前缀和即可), 而中间的可以直接用 sum 数组来更新答案.

现在我们来分析复杂度

- 修改的复杂度为 $O(\frac{n}{k} + \log n)$;
- 询问的复杂度为 $O(\frac{n}{k} + k \log n)$.

考虑到 n 和 q 同阶, 我们不妨用 n 代替 q, 因此总的复杂度是 $O(n(\frac{n}{k} + k \log n))$.

因此我们可以取 $k = \sqrt{\frac{n}{\log n}}$, 使得复杂度为 $O(n\sqrt{n \log n})$, 注意这里的 log 是可以放在根号里面的.

实践中我们还需要考虑常数, 因此我最后取了 $k = \left\lfloor 3.5\sqrt{\frac{n}{\log n}} \right\rfloor$.

第十六章　试题泛做 (16)

16.1 Find a special connected block

16.1.1 问题描述

有一个 $n \times m$ 的棋盘, 每一个点有一个颜色 $a[i][j]$ 和一个代价 $b[i][j]$, 你需要找到一个代价和最小的连通块, 使得这个连通块中包含至少 k 个颜色大于 0 的格子并且不能有-1.

16.1.2 数据规模和约定

- $1 \leq n, m \leq 15$
- $1 \leq k \leq 7$
- $-1 \leq a[i][j] \leq n \times m$
- $1 \leq b[i][j] \leq 100000$

时间限制: 7s.

16.1.3 分析

不妨考虑一下, 如果 $-1 \leq a[i][j] \leq k$ 该怎么做. 我们可以考虑经典的斯坦纳树: 设 $f[i][j][s]$ 表示当前走到格子 (i, j), 已经获得的状态为 s 的方案数, 这是比较容易转移的.

接下来我们可以对于 $a[i][j]$ 的每一种正的取值, 把它随机地映射到 $[1, k]$ 之间的整数, 这样的话正确的概率是 $\frac{k!}{k^k}$. 因为如果假设最优解选取的颜色是 $c_1, c_2, \ldots, c_k$ 的话, 那么 $c_{1\ldots k}$ 被映射到不同颜色的方案数是 $k!$, 而可能的映射方案数是 k^k, 而对于剩下的颜色, 我们并不关心它最终被映射到了什么颜色.

当 $k = 7$ 时, 大约随机 700 次即可得到最优解. 考虑到评测机效率等因素, 我在 Codechef 上设置的随机次数是 400 次, 可以通过本题.

时间复杂度: O(随机次数 $\times\, nm3^k$).

16.2 Graph Challenge

16.2.1 问题描述

给定一个有 n 个点, m 条边, 并且起点为 1 的控制流图 (即从起点可以到图中任意一个点的有向图)G. 对于每个点, 你需要求出有多少个点以它为半必经点.

16.2.2 数据规模和约定

- $1 \leq n \leq 10^5$
- $n-1 \leq m \leq 2 \times 10^5$
- 保证数据合法
- 没有重边

时间限制: 2s.

16.2.3 分析

这题是 GRAPHCNT 一题的一个子问题, 也就是求每个点的半必经点. 那么直接按照那道题的做法做就可以了.

时间复杂度: $O(m\alpha(m))$.

16.3 Count on a Treap

16.3.1 问题描述

你需要维护一棵笛卡尔树, 并回答 m 个询问, 询问分为以下三种类别

- 0 k w: 插入一个以 k 为 key, 以 w 为 val 的结点;
- 1 k: 删除以 k 为 key 的结点;
- 2 k_1 k_2: 询问以 k_1 为 key 的结点和以 k_2 为 key 的结点.

16.3.2 数据规模和约定

- $1 \leq m \leq 2 \times 10^5$
- $0 < k, w < 2^{32}$

- 保证数据合法, 并且所有结点的 key 和 val 互不相同

时间限制: 1s.

16.3.3　分析

我们离线处理这个问题, 并且离散化所有的 key 和 val, 设 n 为不同的 key 的数量.

首先, 我们可以发现, 两个点的 LCA 是很好求的, 因此我们可以把求距离转化为求深度, 然后就相当于统计祖先的数量.

不妨考虑一下, 当 $key[i] < key[j]$ 时 ($key[i] > key[j]$ 的情况类似), 点 i 是点 j 的祖先, 它的充要条件是什么. 实际上, 这只需要 $key[i] < key[j], val[i] > val[j]$, 并且 $key[i]$ 到 $key[j]$ 中间没有一个 val 比 i 大的结点. 这就相当于我们从 j 往前走, 维护一个单调栈, 每遇到一个 val 比栈顶的点的 val 大的点, 就把它加进去, 最后所得到的点的数量.

我们可以用线段树来维护这个数量. 设 $ls[id]$ 表示线段树的结点 id 的从左往右走, 所得到的点的数量, $rs[id]$ 表示从右往左走的数量, $mx[id]$ 表示区间中的 val 的最大值, $pos[id]$ 表示最大值的位置. 并且实现两个函数 $lQuery(id, cur)$ 和 $rQuery(id, cur)$, 分别表示当前栈顶的元素为 cur, 然后从左到右, 或者从右到左遍历整个区间, 最后区间内被加入到栈中的元素的数量.

不妨考虑怎么实现函数 $lQuery(id, cur)$, 设结点 id 的左儿子为 lch, 右儿子为 rch, 那么

1. $mx[lch] < cur$, 此时整个 lch 都不可能被加入到栈中, 因此返回 $lQuery(rch[id], cur)$ 即可;
2. $mx[lch] > cur$, 此时, 我们应该返回的是 $ls[id] - ls[lch[id]] + lQuery(lch[id], cur)$, 这就相当于我们把当前区间的栈的左半边换成是从 cur 开始的, 由于 $mx[lch] > cur$, 因此 lch 这一部分的栈一定是以 $mx[lch]$ 结尾的, 因此 rch 这边我们就不需要重新计算了.

注意到我们每次只会递归到一半的区间, 因此这个函数的时间复杂度是 $O(\log n)$ 的.

那么, 当我们修改一个点之后, 我们就需要更新某个区间的信息, 这可以利用上面的两个函数来轻松地实现.

当我们需要询问某个点 i 的深度时, 我们可以把它拆成统计左边的祖先数量和右边的祖先数量, 然后先在线段树上走到对应的区间, 再利用上面的两个函数来在某个区间中快速地得到答案.

时间复杂度: $O(n \log^2 n)$.

第十七章　试题泛做 (17)

17.1　Short II

17.1.1　问题描述

给定一个质数 p, 求有多少对 (a,b), 使得 $p<a, p<b$ 并且 $(a-p)(b-p)|ab$.

17.1.2　数据规模和约定

- 多组数据 (不超过 5 组)
- $1<p<10^{12}$

时间限制: 3s.

17.1.3　分析

题目所求的式子简单变形之后可以得到 $0<a,b$ 并且 $ab|(a+p)(b+p)$, 等价于 $ab|p(a+b+p)$.

设 x 为 $p|a$ 且 $p|b$ 的数对的数量, y 为 $p|b$ 但 $p\nmid a$ 的数对的数量, z 为 $p\nmid a$ 且 $p\nmid b$ 的数对的数量, 那么答案就等于 $x+2y+z$.

我们先来考虑如何求 x: 设 $a=a'p$, $b=b'p$, 则 $ab|p(a+b+p)$ 等价于 $a'b'p^2|p^2(a'+b'+1)$, 即 $a'b'|(a'+b'+1)$, 考虑到 $a'+b'+1\geq a'b'$ 以及这两者的增速之间的关系, 我们不难得到: 只有 5 对 (a',b'), 分别是 $(1,1),(1,2),(2,1),(2,3)$ 和 $(3,2)$, 于是 $x=5$.

再考虑如何求 y: 设 $b=b'p$, 则 $ab'|(a+b'p+p)$, 即 $a+b'p+p=kab'$, 即 $b'=\frac{a+p}{ka-p}$, 设 $d=ka-p$, 那么 $d|(a+p)$, 除此之外, $a|(d+p)$, 于是我们得到了一个神奇的式子 $ad|(a+p)(d+p)$, 并且由于 $p\nmid a$ 并且 $d|(a+p)$, 因此 $p\nmid d$. 因此我们可以建立双射 $(a,d)\mapsto(a,p\frac{a+p}{d})$, 从而得到 $y=z$.

最后考虑如何求 z: 考虑到 $a\nmid p\wedge b\nmid p$, 但 $ab|p(a+b+p)$, 因此 $ab|(a+b+p)$. 这意味着 $a+b+p\geq ab$, 即 $p+1\geq(a-1)(b-1)$. 此时如果 $a=b$, 那么 $a=b=1$, 否则不妨假设 $a<b$, 那么 $p+1>(a-1)^2$, 即 $a<w=1+\sqrt{p+1}$. 并且根据 $a+b+p=kab$ 我们可以得到 $b=\frac{a+p}{ka-1}$. 令 $d=ka-1$, 那么我们要找的数对 (a,d) 需要满足

1. $p \nmid a$
2. $d|(a+p)$
3. $a|(d+1)$
4. $a < b = \frac{a+p}{d}$

留意到 $bd = a + p < w + p$, 因此, b 和 d 之中至少有一个小于 $\sqrt{w+p}$, 那么我们考虑枚举小于 $\sqrt{w+p}$ 的那一项. 我们考虑枚举 d, 那么哪些 b 可能成为可行的 b 呢? 由 $d|(a+p)$ 我们可以得到 $a \equiv -p \pmod{d}$, 而由 $a|(d+1)$ 我们可以得到 $a \leq d+1$, 因此我们至多有两个可能的 a 需要检查, 因此这一步的复杂度是 $O(\sqrt{p})$ 的.

再考虑枚举 b, 此时我们不妨假定 $b < d$, 因为其它情况之前已经枚举过了. 此时我们依然有 $a \equiv -p \pmod{d}$, 并且由于 $a < b < d$, 因此此时我们至多有一个可能的 a 需要检查, 因此这一步的复杂度也是 $O(\sqrt{p})$ 的.

时间复杂度: $O(\sqrt{p})$, 注意本题常数卡得比较紧, 因此需要合并上面两种情况的循环, 只枚举一次.

17.2 Hypertrees

17.2.1 问题描述

定义 k-超图为每条边连接了 k 个点的图.

定义 k-超树为连通的, 并且断掉任意一条边均能够裂成至少两块的 k-超图.

现在你需要给出含有 n 个带标号点的本质不同的 k-超树的数量.

17.2.2 数据规模和约定

- 多组数据 (不超过 15 组)
- $3 \leq n \leq 17$

时间限制: 1s.

17.2.3 分析

先来考虑一个问题: 一个点双连通的 3-超树会长什么样? 实际上, 除了只有 3 个点的 3-超 BCC(此时它只有一条边), 这个 3-超 BCC 大概会类似于一个 (普通图中的) 环, 然后每条边都往外伸出去一个点, 那么这实际上对应于一个有 a 个点, b 条边, 并且 $a+b$ 等于 3-超 BCC 的点数的普通 BCC. 于是我们可以预处理处 $a+b \leq n$ 的点双的数量.

接下来我们可以枚举如何把 3-超 BCC 拼成一棵 3-超树, 注意这里需要用到记忆化来优化复杂度.

但是这样还是很慢, 于是我们可以打表.

17.3 Black-white Board Game

17.3.1 问题描述

给定 n 个区间 $[l_i, r_i]$, 你需要求出 1 到 n 的排列中, 第 i 个元素位于 l_i, r_i 之间, 并且逆序对数为偶数的排列的个数, 减去第 i 个元素位于 l_i, r_i 之间, 并且逆序对数为奇数的排列的个数.

17.3.2 数据规模和约定

- 多组数据 (至多 15 组)
- $1 \leq n \leq 10^5$

时间限制: 2s.

17.3.3 分析

首先注意到, 如果我们构造一个 01 矩阵 A, 使得它的第 i 行中, 第 l_i 至第 r_i 列为 1, 其余的为 0, 那么答案就是 $\det A$.

而求矩阵的行列式有一种非常经典的做法: 利用初等变换将矩阵变成下三角. 下面我们就考虑怎么把 A 消成下三角.

我们可以考虑一列一列的消, 并且在消的过程中保证任意一行的 1 都是连续的 (因此不妨用有序对 (l, r) 来表示一个第 l 列到第 r 列为 1, 其余列为 0 的行), 接下来将说明如何做到这一点.

不妨假设我们当前枚举到的是第 i 列, 那么我们可以找到形如 (i, j) 的行中 j 最小的那个, 如果没有, 那么显然, $\det A = 0$. 否则我们不妨假设这一行为 (i, j), 那么我们可以把它换到第 i 行来, 并且把其余的 $l = i$ 的行减去那一行. 我们可以发现, 这样减完之后, 原本为 (i, k) 的行现在变成了 $(j + 1, k)$, 它其中的 1 仍然是连续的.

下面的问题就是怎么样快速地找到形如 (i, j) 的行中 j 最小的那个, 我们不妨维护 n 个可并堆, 第 i 个堆中存放 $l = i$ 的所有行. 并且注意到我们在删除掉 (i, j) 之后, 会把所有的 $l = i$ 的行变成 $l = j + 1$ 的行, 这可以通过可并堆的合并操作来完成.

时间复杂度: $O(n \log n)$.

17.4 Little Party

17.4.1 问题描述

定义特征是固定向量的某些维所得到的一个集合, 比如说特征 $(0, x, 1, y, z)$ 就表示第一维为 0, 第三维为 1, 其余维任选所得到的所有向量的集合.

定义特征的大小为它所固定的维的数量, 比如说上面那个特征的大小就是 2.

给定 m 个 n 维的 01 向量, 你需要找出一些特征, 使得每个向量都至少属于一个特征, 并且每个特征中的向量都在这 m 个向量构成的集合中, 并最小化特征的大小和.

17.4.2 数据规模和约定

- 多组数据 (至多 120 组)
- $1 \leq n \leq 5$
- $1 \leq m \leq 1000$

时间限制: 1.2s.

17.4.3 分析

很显然, 这道题应该是 NP 的 (有点像 Exact Cover), 所以我们只能暴力做.

我们不妨枚举所有可能的特征, 然后再枚举所有可能的选特征的方法, 最后更新答案. 然而这样是跑不过去的. 因此我们可以把常见的优化, 比如说最优性剪枝, 可行性剪枝, 记忆化, 都加上去.

其实还有一个非常显然的优化, 注意到如果一个可行的特征 A 包含了另外一个可行的特征 B, 那么 B 是没有用的, 这样可以大大减少枚举的特征的数量.

实现上需要注意到最优性剪枝和记忆化之间的关系, 注意不要搞混了, 并且记忆化的时候可以采用 HashMap(或 std::unordered_map), 由于我们需要哈希的是一个 32 位无符号整数, 这样的话采用哈希表是非常快的.

第十八章　试题泛做 (18)

18.1　Across the River

18.1.1　问题描述

平面上有 n 个点, 第 i 个点的坐标为 (x_i, y_i). 有 m 种圆盘, 第 i 种圆盘的半径为 r_i, 价格为 c_i, 并且每一种圆盘都有无限多个. 如果你在第 i 个点上放第 j 个圆盘, 那么你需要花费 c_j 元, 并且在此之后以 (x_i, y_i) 为圆心, r_i 为半径的圆形区域将变成连通的. 两个连通区域如果有交, 那么它们是连通的.

你需要以最小的代价, 使得直线 $y = 0$ 和直线 $y = w$ 连通, 如果无解, 输出 impossible.

18.1.2　数据规模和约定

- $1 \le n, m \le 250$
- $2 \le w \le 10^9$
- $0 \le x_i \le 10^9$
- $1 \le y_i < w$
- $1 \le r_i \le 10^9$
- $1 \le c_i \le 10^6$

时间限制: 6s.

18.1.3　分析

不妨先将所有的圆盘按照 r_i 排序, 再去掉无用的圆盘 (对于圆盘 i, 如果存在圆盘 j 使得 $r_j \ge r_i$ 并且 $c_j \le c_i$, 那么显然圆盘 i 是没有用的), 然后对于每一个点, 我们把它拆成 m 个点, 点 (i, j) 表示第 i 个点放了圆盘 j 所构成的平面区域.

然后我们可以按照如下方式构出一个有向图:

1. 对于所有的 i, 在 (i,j) 和 $(i,j+1)$ 之间连一条费用为 $c_{j+1}-c_j$ 的边;
2. 对于所有的 (i,j) 和 $(a,b)(i\neq a)$, 如果它们有交, 那么由 (i,j) 向 (a,b) 连一条费用为 c_b 的边.

最后我们再以所有和直线 $y=0$ 有交的点为起点, 所有和直线 $y=w$ 有交的点为终点做一次最短路, 就可以得到答案了.

利用单调性很容易将图的边数减少到 $O(n^2m)$, 在此基础上留意到我们只需要知道起点和终点的最短路, 因此, 我们在第一次访问到一个终点的时候就可以终止 Dijkstra 算法的过程了, 这样可以极大地优化程序的效率.

时间复杂度: $O(n^2m\log(n^2m))$.