# CODECHEF
# 题目选做与分析

*2016* 集训队作业

杨乐（中山纪念中学）

# 目录

# 1 [2015-08]Future of draughts

## 题目简述

给出标号从 1 到 $N$ 的 $T$ 个无向无权图。在第 $i$ 次询问中，给出三个参数 $L_i, R_i$ 和 $K_i$，接着会同时在标号在 $L_i$ 到 $R_i$ 之间的无向图上进行游戏。

具体而言，对于标号在 $L_i$ 到 $R_i$ 之间的每个无向图，先**随机**地指定棋子的起始位置。然后，在每一步中，会在这些无向图中，选中一个**随机非空集合**，然后在所有被选中的无向图中，随机地将棋子移动一步。当所有无向图中棋子都回到它的起始位置时，本次询问结束。注意，在训练中至少需要进行一次行动。

对于一共的 $Q$ 次训练，想知道有多少种可能的方案，使它在 $K_i$ 步之内结束。由于答案可能非常大，输出它对 1000000007 取模的结果。

## 题目分析

**本题缩写：CLOWAY**
**题目类型：**线性代数，多项式乘法
**题目模型：**计数问题，无向图

## 解法一：容斥原理

设 $F[i][j]$ 为仅考虑第 $i$ 张图，在上面形成长度为 $j$ 的环的方案数。$F$ 数组的推导可以使用简单的递推，总时间复杂度为 $O(TN^2K)$。

接下来考虑多个图合并答案的情况。考虑第 $i$ 张图到第 $j$ 张图，恰好 K 步结束的答案 $H[i][j][k]$。设图 $X$ 在 $K$ 步的过程中，有效的步数是 $W$（存在没有被选取的情况），这时的方案数是

$$\binom{K}{W} \times F[X][W]$$

那么图 $X$ 贡献的总方案数为

$$G[X][K] = \sum_{i=0}^{K} \binom{K}{i} \times F[X][i]$$

直观上，答案

$$H[i][j][k] = \prod_{X=i}^{j} G[X][k]$$

但显然这样的表达式是错误的，错误在于它没有考虑到可能存在一步，这一步中所有图都没有移动，而这不符合题目的限定。所以类似于容斥原理，在总方案中去掉不合法的方案：令合法的方案数为 $P[K]$，那么递推式为：

$$P[K] = H[K] - \sum_{X=0}^{K-1} \binom{K}{X} \times P[X]$$

这样就可以在 $O(T^2K^2)$ 的时间复杂度内预处理所有可能的答案。但时间复杂度不足以通过所有的数据。

曾想过使用多项式的乘法与除法来优化处理 $G$ 与 $P$ 的时间复杂度，但本人尚未找到解决的方案，如果大家有解决的办法，可以与我交流细节处理问题。

## 解法二：线性代数

### 闭合回路：矩阵的迹 (trace)

我们先考虑只询问一个无向图的问题。用一个邻接矩阵 $G$ 来表示图的连通情况：$G_{i,j}=0$ 表示没有连通而 $G_{i,j}=1$ 则代表连通。接下来记 $P(k)_{i,j}$ 为 $i$ 与 $j$ 之间长度为 $k$ 的路径的数量，显然满足 $P(0)_{i,i}=1$。而通过递推表达式

$$P(k)_{i,j}=\sum_t P(k-1)_{i,t}G_{t,j}$$

可以得到

$$P(k)=P(k-1)\cdot G=G^k$$

而长度为 $k$ 的回路总数为 $\sum P(k)_{i,i}$，也就是矩阵对角线上的和。一个矩阵 $M$ 的对角线上的和 $tr[M]$ 也被称为**矩阵的迹**。所以回路总数等于矩阵 $G^k$ 的迹 $tr[G^k]$。

### 矩阵的特征向量 (eigenvector) 与特征值 (eigenvalue)

为了进一步了解矩阵的迹，下面来解释其在线性代数中的特性：

**特征向量 (eigenvector)**：一个非零列向量 $v$ 称作矩阵 $A$ 的一个特征向量，仅当存在 $\lambda$ 使得 $Av=\lambda v$，换句话说，$v$ 在 $A$ 的变换中下方向不变。

**特征值 (eigenvalue)**：在上面的情况中，$\lambda$ 称作 $v$ 对应的特征值。让我们尝试计算一个大小为 $N\times N$ 的矩阵 $A$ 的特征值，由定义得：

$$\begin{aligned}Av&=\lambda v\\ \lambda v-Av&=0\\ (\lambda I-A)v&=0\end{aligned}$$

其中 $I$ 为单位矩阵。要使 $v$ 不为 0（特征向量不能为 0），只能使矩阵 $\lambda I-A$ 的行列式为 0。从而问题变为求解方程

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

而 $det(\lambda I-A)$ 又是一个 $N$ 次的关于 $\lambda$ 的多项式，也称作该矩阵的**特征多项式**。显然这个方程存在 $N$ 个解，这些都是矩阵 $A$ 的特征值。我们把特征值的集合称作**矩阵的谱 (spectrum)**，记作 $sp(A)$（https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors）。

存在定理：**矩阵的迹等于它的特征值之和**。

这对我们理解问题有帮助，因为特征值存在更多的有趣的性质。例如，如果 $\lambda$ 是矩阵 $A$ 的一个特征值，那么 $\lambda^k$ 是矩阵 $A_k$ 的一个特征值（为什么？）。所以 $A^k$ 的谱为集合 $\{\lambda^k : \lambda \in sp(A)\}$，$tr[A^k] = \sum_{\alpha \in sp(A)} \alpha^k$。

### 强乘积图 (strong product of two graphs) 与它的特征值

在解决一个图的情况后，我们着手于多个图结合的情况，为此引入强乘积图 (strong product of two graphs) 的定义：

在图论中，两个图 $G$,$H$ 的强乘积图 $G \boxtimes H$ 是这样的一个图：

- 图中的点集是二元组 $(x, y)$，其中 $x$ 来自 $G$，$y$ 来自于 $H$
- 两个不同的顶点 $(u, u')$ 与 $(v, v')$ 在 $G \boxtimes H$ 中是相连的，当且仅当满足下面 3 种条件之一：
  1. $u$ 与 $v$ 相邻，$u' = v'$
  2. $u = v$，$u'$ 与 $v'$ 相邻
  3. $u$ 与 $v$ 相邻，$u'$ 与 $v'$ 相邻

根据定义，在多个图上分别计算实质上等价于在它们的**强乘积图**上计算。那么题目就转变为求一些图的强乘积图所对应的**矩阵的迹**。

下面来观察两个图的强乘积图的特征值的性质。我们可以证明，$G \boxtimes H$ 的**谱**满足

$$sp(G \boxtimes H) = \{(\alpha + 1)(\beta + 1) - 1 | \alpha \in sp(G), \beta \in sp(H)\}$$

### 强乘积图的迹

由上可得**强乘积图的迹**满足：

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

又

$$tr((G + I)^k) = \sum_{\alpha \in sp(G)} (\alpha + 1)^i$$

由此可以推广到**多个图的强乘积图的迹**：

$$tr[(G_l \cdots G_r)^k] = \sum_{i=0}^{k} \binom{k}{i} (-1)^{k-i} tr[(G_l + I)^i] \cdots tr[(G_r + I)^i]$$

令

$$P(i,j,k)=\prod_{x=i}^{j} tr[(G_x+I)^k]$$

则

$$tr[(G_l\cdots G_r)^k]=\sum_{i=0}^{k}\binom{k}{i}(-1)^{k-i}P(l,r,i)$$

将整个式子化成卷积的形式，可得询问 $(l,r,k)$ 的答案 $ans(l,r,k)$ 满足

$$ans(l,r,k)=tr[(G_l\cdots G_r)^k]=k!\sum_{i=0}^{k}\frac{P(l,r,i)}{i!}\frac{(-1)^{k-i}}{(k-i)!}$$

对于卷积这部分的操作，可以使用快速傅里叶变换 (FFT) 来优化转移时间复杂度，所以这部分的时间复杂度为 $O(T^2KlogK)$。

但由于题目所给的模数 MOD=$10^9+7$，并不能直接套用数论变换 (NTT)；但同时可以注意到计算后的每一项实际不大于 $K\text{MOD}^2$，所以可以使用中国剩余定理 (CRT) 处理。具体实现时，在三个不同的模数下（例如 167772161, 998244353, 1004535809）分别计算结果，接着使用中国剩余定理合并出结果。由于三个模数的乘积大于 $K\text{MOD}^2$，故能正确地求出结果。

### 处理 $tr[(G_i+I)^k]$ 与 $P(i,j,k)$

还剩下需要预处理的是 $tr[(G_i+I)^k]$；$P(i,j,k)$ 可以在前者处理后在 $O(T^2K)$ 的时间复杂度内计算得到。但直接计算矩阵 $(G_i+I)^k$ 的总时间复杂度为 $O(TN^3K)$，明显通过不了全部的测试数据。

**方法一：“大步小步”** 我们若只考虑计算矩阵 $A\cdot B$ 的迹，可以在 $O(N^2)$ 的时间复杂度内计算得到，因为只需要计算目标矩阵位于对角线上的位置的结果。由此启发，可以将 $(G_i+I)^k$ 中的 $k$ 表示为 $aP+b$，那么该矩阵则是 $(G_i+I)^{aP}$ 与 $(G_i+I)^b$ 的乘积，通过预处理所有的 $(G_i+I)^{xP},(xP\le k)$ 与 $(G_i+I)^y,(y<P)$，可以在 $O(T(\frac{k}{P}+p)N^3+TN^2K)$ 时间复杂度内计算得到。明显当 $P=\sqrt{K}$ 时总时间复杂度最小，可以通过全部测试数据。

**方法二：特征多项式与 Cayley-Hamilton theorem** 回到线性代数上，我们有更优美（复杂）的理论与算法。

为了方便叙述，令 $H=G+I$，我们的计算目标是 $tr[I],tr[H],tr[H^2],\ldots,tr[H^k]$。

由 Cayley-Hamilton theorem 可以得到，如果 $c_0+c_1x+c_2x^2+\cdots+x^N$ 是矩阵 $H$ 的特征多项式，则 $c_0I+c_1H+c_2H^2+\cdots+H^N=0$（注意 0 指的是一个零矩阵而非数字 0）。

所以，我们可以推导得到（在等式两端同时乘以 $H^{k-N}$）：

$$\begin{aligned} c_0 I + c_1 H + c_2 H^2 + \cdots + H^N &= 0 \\ c_0 H^{k-N} + c_1 H^{k-N+1} + c_2 H^{k-N+2} + \cdots + H^k &= 0 \\ tr[c_0 H^{k-N} + c_1 H^{k-N+1} + c_2 H^{k-N+2} + \cdots + H^k] &= 0 \\ c_0 tr[H^{k-N}] + c_1 tr[H^{k-N+1}] + c_2 tr[H^{k-N+2}] + \cdots + tr[H^k] &= 0 \end{aligned}$$

最后一步的推导利用了**矩阵的迹是线性映射（linear map）**的性质。

由此，我们只需暴力计算出 $tr[I], tr[H], tr[H^2], \ldots, tr[H^N]$，再求出 $H$ 的特征多项式，使用递推来计算出 $tr[H^(N+1)], \ldots, tr[H^k]$。所以这部分的总时间复杂度为 $O(TN^4 + TNK)$。

**EXTRA：特征多项式** 要使用上述的方法前提是求出某一个矩阵 $H$ 的特征多项式。根据定义，它等于 $det(xI - H)$，是一个 $N$ 次的**首一多项式（monic polynomial）**(首项为 1)，它们的根的集合就是**矩阵 $H$ 的谱**。假设矩阵 $H$ 的特征值分别为 $\lambda_1, \lambda_2, \ldots, \lambda_N$，那么它的特征多项式就是 $(x - \lambda_1)(x - \lambda_2) \cdots (x - \lambda_N)$。

举个例子，假设 $N = 3$，那么它的特征多项式就像这个形式：

$$(x-\lambda_1)(x-\lambda_2)(x-\lambda_3) = x^3 - (\lambda_1+\lambda_2+\lambda_3)x^2 + (\lambda_1\lambda_2+\lambda_1\lambda_3+\lambda_2\lambda_3)x - \lambda_1\lambda_2\lambda_3$$

$(x_1 + x_2 + x_3), (x_1x_2 + x_2x_3 + x_1x_3), (x_1x_2x_3)$ 称作 $x_1, x_2, x_3$ 这三个变量的**初等对称多项式（elementary symmetric polynomial）**，这些多项式与一个多项式的系数与根有着很大的联系。

**$k$ 阶初等对称多项式（elementary symmetric polynomial of order k）**记作 $e_k(x_1, \ldots, x_N)$，它是所有的在 $\{x_1, \ldots, x_N\}$ 中大小为 $k$ 的子集的乘积之和。如果将集合中的数赋值为特征多项式的根（即特征值），那么，显然地，特征多项式的系数满足：

$$\begin{aligned} e_1(\lambda_1, \ldots, \lambda_N) &= -c_{N-1} \\ e_2(\lambda_1, \ldots, \lambda_N) &= +c_{N-2} \\ e_3(\lambda_1, \ldots, \lambda_N) &= -c_{N-3} \\ e_4(\lambda_1, \ldots, \lambda_N) &= +c_{N-4} \\ &\cdots \\ e_N(\lambda_1, \ldots, \lambda_N) &= (-1)^N c_0 \end{aligned}$$

那么，我们希望计算 $e_i(\lambda_1, \ldots, \lambda_N)$。

计算的关键在于使用 $tr[I], tr[H], tr[H^2], \ldots, tr[H^N]$（已经在前面计算过了）。由于 $tr[H^k]$ 等于所有特征值的 $k$ 次方之和，所以运用**牛顿恒等式（Newton's identities）**可以推导出：

$$ke_k(x_1, \ldots, x_N) = \sum i = 1^k (-1)^{i-1} e_{k-i}(x_1, \ldots, x_N) p_i(x_1, \ldots, x_N)$$

其中，$p_i(x_1,\ldots,x_N)$ 被定义为 $x_1^i+x_2^i+\cdots+x_N^i$，正是 $tr[H^i]$。
所以递推计算 $e$ 的时间复杂度为 $O(N^2)$，是十分优秀的。

## 归纳总结

本题是一道冗长的线性代数题目，需要缜密的分析与清晰的实现思路：

1. 对于每一个无向图 $G$，令 $H=G+I$，计算出它的 $tr[H^i](0\le i\le k)$。
2. 计算出 $P(i,j,k)$。
3. 计算出 $ans(i,j,k)$。

对于第一步，可以使用“大步小步”法或特征多项式法；
对于第三步，要使用 FFT 优化卷积的计算。
总时间复杂度为 $O(TN^4+TNK+T^2KlongK+Q)$；
而空间复杂度为 $O(TNK)$。

# 2 [2015-08]Simple Queries

## 题目简述

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
$$\left(\sum_{1\le i<j<k\le |S|} S_iS_jS_k\right)(\mathbf{mod}\ 10^9+7)$$
- 2 $x$ $y$：将下标为 $x$ 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 $r$ 范围内的不同元素个数

数组下标从 1 开始。数据保证数组总是非空。

## 题目分析

**本题缩写：DISTNUM**
**题目类型：**数据结构，容斥原理，二维数据结构
**题目模型：**区间询问与修改

## 无插入与修改

我们先考虑没有插入与修改的情况：

### 二维平面：询问 5

对于每一个数 $A_i$ 找到最大的 $p_i$ 满足 $A_i = A_{p_i}$（若不存在则令 $p_i = 0$）。

若查询区间为 $[l,r]$，那么满足 $l \le i \le r, p_i < l$ 的 $i$ 的个数则是询问的答案：所有 $p_i \ge l$ 的 $i$ 在区间中已重复，不必重复统计。

在二维平面上标记出所有的 $(i,p_i)$ 点，每次询问转化为在矩形 $[l,r][0,l-1]$ 内的点数。解决二维平面上的问题，可以使用二维数据结构维护。

#### 容斥原理：询问 1

题目中要求相乘的三个数不能够重复，故容斥原理能很好的解决重复问题。

记数集为 $|S|$，所有数的一次方，二次方，三次方之和分别是 $S_1, S_2, S_3$，所求的 Sum 满足：

$$Sum = \frac{S_1^3 - 3S_1S_2 + 2S_3}{6}$$

容斥原理的核心是：加一个大的 $S_1^3$，减去其中不符合的 $3S_1S_2$，再补上额外减去的 $2S_3$。

但是简单的维护前缀和是不够的，这样会导致同一个数可能会被重复计算。

解决方法是把这些信息全部放到二维平面上，类比询问 5 回答即可。

#### 平衡树：修改操作

在修改 $A_i$ 时，可能会修改若干个 $p_i$: 给每一个权值建造一棵平衡树。在修改权值时，在对应权值的平衡树查询前驱与后继，修改 $p_i$ 而后再修改二维数据结构。

### 插入与删除操作

本题当中存在的插入与删除操作表面上要求使用在线算法解决，而实际上可以预先处理所有操作使问题转变为离线。

中心思想：预先处理所有数的相对顺序（包括已删除的）。

用一棵平衡树维护当前的序列————遇到插入操作直接在对应位置插入；遇到删除操作不直接删除该元素，而是把它标记为已删除。这要求平衡树的每个节点记录该子树内有多少个尚未删除的节点，这样在插入操作时可以保证插入位置的正确性。

把所有数的相对顺序确定后，插入与删除操作就可以转变为修改操作。

同时，还需要把询问操作的区间相应的进行修改。（这一步可以使用线段树简单地做到）

### 二维数据结构

#### 二维数据结构：二维线段树/树状数组套线段树

在用上述两种方法实现时，显然地树状数组套线段树较为简单。

具体的操作方法不再详述。

#### 二维数据结构：二维树状数组

二维树状数组的空间复杂度太大 ($O(N^2)$)，不适合这道题的解答。

如果使用哈希算法，则寻址的总时间复杂度太高了。

**二维数据结构：分块**

将整个平面分为 $\sqrt{N} \times \sqrt{N}$ 块，分别处理块内信息与块外信息。

每次询问一个二维区间时，包含的有若干个整块与若干个不完整的部分。对于整块，必然是一个矩形区域，所以维护一个二维树状数组能够方便地回答区间总和；对于不完整的部分，由于每一行或每一列最多同时存在一个点，直接枚举该部分的点就足够了。

# 3 [2015-07]Easy exam

## 题目简述

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 $a_1^F \times a_2^F \times \cdots a_L^F$ 的期望值。假设答案为分数 $P/Q$，其中 $P$ 和 $Q$ 是整数。令整数 $X$ 等于 $(P * Q^{-1})$ **mod** 2003，其中 $Q^{-1}$ 是 $Q$ 关于 2003 的乘法逆元。输出一行包含整数 $X$。

## 题目分析

**本题缩写：EASYEX**
题目类型：数学，组合数学，计数问题，卷积，递推。
方法技巧：分解与转化问题，由特殊到一般

## 分解问题

分析 $a_i$，它是由 $N$ 个独立的事件组成的。给这 $N$ 个独立事件分别记变量 $X_j$，代表在第 $j$ 次投掷中对于第 $i$ 面的投掷情况：当 $X_j = 1$ 时，代表第 $j$ 次掷出了第 $i$ 面，而 $X_j = 0$ 则代表没有掷出。显然地，$X_j = 1$ 的概率为 $1/K$，而 $a_i = \sum X_j$。

## 转化问题

题目所求的 $E(\prod a_i^F)$ 将每个 $a_i$ 按每次投掷分解后，转变为

$$E\left((A_1+A_2+\cdots+A_N)^F \times (B_1+B_2+\cdots+B_N)^F \times \cdots \times (L_1+L_2+\cdots+L_N)^F\right)$$

（其中 $A, B, \cdots, F$ 为第 1 面，第 2 面，...，第 $L$ 面的投掷情况）

## 从简单开始：$F = 1$

当 $F = 1$ 时，所求的期望为

$$E\left((A_1+A_2+\cdots+A_N) \times (B_1+B_2+\cdots+B_N) \times \cdots \times (L_1+L_2+\cdots+L_N)\right)$$

由多项式乘法的定义可知展开式中每一项包含 $L$ 项的乘积，分别来自 $A, B, \cdots, L$。假设其中的一项为 $A_x B_y C_z \cdots L_w$，它对答案产生贡献的需要满足两个要求：他们必须全部取 1；不能产生矛盾情况。

什么情况会产生矛盾情况？$x = y$，就是一种。它代表着在第 $x$ 次投掷中，既投出了第 1 面，又投出了第 2 面。所以由此得出: 不出现矛盾需要满足 $x, y, z, \cdots, w$ 互不相等。

接下来的就是简单的组合数学和计数运算：$F = 1$ 的答案可直接计算得到：

$$\frac{\binom{N}{L} \times L!}{K^L}$$

## 由特殊到一般：$F > 1$

当 $F > 1$ 时，多项式的展开式中就不止 $L$ 项的乘积了。准确地说，每一项包含 $L * F$ 项乘积，但其中可能会有些项是重复出现的。（建议写出 $N = F = 3$ 的展开式来推导它的性质）

假设某一项拥有 $P$ 个不同的项（$L \le P \le L * F$）：那么和 $F = 1$ 类似的，它也必须满足每一项的标号不一样。所以所有拥有 $P$ 个不同的项对答案的贡献为

$$\frac{\binom{N}{P} \times P! \times co(P)}{K^P}$$

其中 $co(P)$ 为拥有 $P$ 个不同的项的情况数。产生不同的情况数的原因是 $A, B, \cdots, L$ 贡献的项数不同，以及 $A, B, \cdots, L$ 内部贡献项数的方法多样性。例如 $A$ 贡献 1 项，$B$ 贡献 2 项与 $A$ 贡献 2 项，$B$ 贡献 1 项的情况不同；$A$ 中 $A_1$ 贡献 1 项，$A_2$ 贡献 3 项与 $A_1, A_2$ 均贡献 2 项又会产生不同 (但 $A_1$ 贡献 1 项，$A_2$ 贡献 2 项与 $A_1$ 贡献 2 项，$A_2$ 贡献 1 项则是一样的，不能被统计入 $co(P)$ 内 [为什么])。

- 解决 $A, B, \cdots, L$ 内部的问题：$A, B, \cdots, L$ 中均有 $F$ 个乘式，逐个进行考虑：设 $G[i][j]$ 为考虑到第 $i$ 个乘式 $(1 \le i \le F)$，前面有 $j$ 项不同的项。$G$ 满足转移 $G[i][j] = G[i-1][j-1] + G[i-1][j] \times j$。于是每一个块中贡献 $i$ 项的方案数即为 $G[F][i]$。
- 解决块与块之间的问题：类似于背包合并问题。使用多项式乘法，用 FFT 进行优化即可。

## Trick:Mod=2003

当 $P >= 2003$ 时就可不必计算了。所以多项式乘法时只保留前 2002 项就可以快速地计算了。

## 复杂度分析

总时间复杂度为 $O(F^2 + (FL)Log(FL)LogL)$。
总空间复杂度为 $O(F^2 + FL)$。

# 4 [2015-05]Chef and Balanced Strings

## 题目简述

定义一个字符串为“平衡的”当它内部出现的每种字母的数量均为偶数。

题目给出一个字符串 $S$，并在线地给出 $Q$ 个询问，询问在 $S[L\dots R]$ 这个子串中，所有平衡子串的个数，或长度总和，或长度的平方和。

## 题目分析

**本题缩写：CBAL**
**题目类型：**强制在线，预处理，分块
**题目模型：**字符串
**方法技巧：**容斥原理

## 模型转化

题目中定义的“平衡子串”为内部的每种字母的数量均为偶数，故定义与出现了多少次并没有直接关系，而是仅与奇偶性存在关联。记录 $H[i][j]$ 为前 i 位字母 j 出现的奇偶情况；这样，若 $S[L\dots R]$ 为“平衡子串”等价于 $H[L-1]=H[R]$。又由于奇偶情况可以用 0 与 1 表示，所以可以将 $H[x]$ 表示为一个 26 位的二进制数，判断就能够在 $O(1)$ 完成。

由此，题目可以转化为给出数组 $H$，以及 $Q$ 个询问，询问在 $L\dots R$ 区间中，满足 $H[x]=H[y]$ 的 $(x,y)$ 对数，或 $\sum(y-x)$，或 $\sum(y-x)^2$。

## 离线算法：莫队

令 $ans0, ans1, ans2$ 分别为以上三种答案的值。

考虑我们已经求得某个区间的答案，现在往区间 $L\dots R$ 内增加一个数 $H[R+1]=x$，考虑 $ans0, ans1, ans2$ 的变化：

- $x$ 会和区间内的每个 $x$ 增添一对，会有以下的变化：
- 区间内若有 $A[q]=x$，统计 $q$ 的个数 $f[x]$。$ans0=ans0+f[x]$。
- 统计 $q$ 的总和 $s[x]$。由于对于每个 $q$ 都会增加 $R+1-q$，由此得到 $ans1=ans1+(R+1)\times f[x]-s[x]$。
- 统计 $q$ 的二次方总和 $s2[x]$。由于对于每个 $q$ 都会增加 $(R+1-q)^2$，由此得到 $ans2=ans2+(R+1)^2\times f[x]-2\times(R+1)\times s[x]+s2[x]$。

那么，在区间头与尾增添一个数或删除一个数就可以进行类似的操作，从而动态地维护 $ans0, ans1, ans2$ 的变化。

运用莫队将询问进行排序而后操作，就可以在 $O(N\sqrt{N})$ 时间内求解。

## 离线转在线：预处理

由于这道题目是对于每个数分开处理的，所以不能采用数据结构来优化它的计算。所以只能用预处理某些区间的值，化成更小的单位进行运算。

具体地，把数组按每 $B$ 个数进行分块。预处理所有左端点或右端点下标是 $B$ 的倍数的 $ans$，这个部分的计算可参照前面的方法。

对于询问 $L, R$，若它们处于同一个块内可以直接计算，时间复杂度为 $O(B)$。

若询问 $L, R$ 不在同一个块中，则记它们所在的块分别为 $i, j$。它的答案 $F(L, R)$ 满足

$$F(L,R) = F(L,jB)+F((i+1)B,R)-F((i+1)B,jB)+G(L,(i+1)B-1,jB,R)$$

通过画图，可以很直接的观察得到：区间 $L, R$ 的答案由两块比它小的块而合并得到，减去重复的区域的答案。但还有一部分没有统计入内：左端点在左方、右端点在右方的询问。但对于这部分可以使用类似的方法计算。

因为预处理的时间复杂度为 $O(N^2/B)$，所以当 $B = \sqrt{N}$ 时总时间复杂度达到平衡。

## 复杂度分析

总时间复杂度为 $O(N\sqrt{N})$。

总空间复杂度为 $O(N\sqrt{N})$。

# 5 [2015-05]Counting on a directed graph

## 题目简述

给定一个 $N$ 个点（从 1 到 $N$ 标号）$M$ 条边的有向图 $G$。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，和一条点 1 到点 $Y$ 的路径，且两条路径除了点 1 之外没有公共点。

## 题目分析

**本题缩写：GRAPHCNT**
**题目类型：**支配树（Dominate Tree）
**题目模型：**有向图

## 解题步骤

二元组 $(X,Y)$ 不合法，仅当点从 1 到 $X$ 必须要经过某个点 $P$，而 1 到 $Y$ 也必须要经过这个点。提到必须要经过某个点的问题，直观地可以使用支配树来解决。

构造图 $G$ 以结点 1 为根的支配树。二元组 $(X,Y)$ 是合法的仅当它们位于以 1 为根的不同的两棵子树中。

## 复杂度分析

时间复杂度为构造支配树的复杂度，$O(N\alpha(N))$。

空间复杂度为 $O(N)$。

# 6 [2015-04]Black-white Board Game

## 题目简述

给出 $N$ 个约束，每个约束形式为一个二元组 $(L_i, R_i)(1 \leq L_i \leq R_i \leq N)$。一个长度为 $N$ 的序列 $P$ 是合法的，仅当其所有的 $i$ 都满足 $L_i \leq P_i \leq R_i$。在所有合法的序列中，记其中逆序对为偶数的序列个数为 $A$，为奇数的个数为 $B$，判断 $A$ 与 $B$ 的大小关系。

## 题目分析

**本题缩写：BWGAME**
**题目类型：**矩阵，行列式，可合并堆
**题目模型：**矩阵

## 模型转化

由题目中的逆序对可以很自然的联想到矩阵的行列式。如果把约束表示成一个 $N \times N$ 的矩阵，将第 $i$ 行的 $L_i$ 至 $R_i$ 列均置为 1 而其余的均置为 0，那么该矩阵的行列式的正负就是 $A$ 与 $B$ 的大小关系。这与行列式的定义直接联系（http://baike.baidu.com/view/111348.htm）。

## 计算行列式

接下来考虑的则是如何计算这个矩阵的行列式。可以使用最朴素的高斯消元法，将整个矩阵消成一个上三角形的形式。现在需要做的是优化这一步的运算。

找出所有 $L_i = 1$ 的行，若在其中找出 $R_i$ 最小的一行作为主元，则在消元后余下的行仍为完整的一段，且均变成了以 $R_i + 1$ 开头的约束。于是我们对每一列 $x$ 存储满足 $L_i = x$，以 $R_i$ 为键值建立一个堆。顺序处理每一列，而后将剩余的信息与后面合并。实现合并操作可以使用左偏树。

## 复杂度分析

总时间复杂度为 $O(NLogN)$。
总空间复杂度为 $O(N)$。

# 7 [2015-04]Little Party

## 题目简述

给出 $M$ 个拥有 $N$ 个变量的集合，每个集合中每个变量可以为真或假。找到一些长度和最小的关键子集能**恰好**覆盖这些集合。每一个集合都包含所有变量。

## 题目分析

**本题缩写：LPARTY**
**题目类型：**位运算，搜索
**题目模型：**集合覆盖

## 解题思路

找出所有可能在答案中出现的关键子集，枚举它们的选取情况。

但这样做会出现许多的冗余情况。去除所有被其他情况完全包含的冗余情况，剩余的情况数最多为 32（$N = 5$ 的情况下）。

接下来则使用搜索解决，加上一些剪枝就可以通过。

- 最优化剪枝：当前长度已经超过
- 可行性剪枝：当前的情况已不能达到满足题目条件
- 最优化剪枝：若当前子集加入不会影响集合的变化则不加入

## 复杂度分析

总时间复杂度为 $O(2^{2^N})$。
总空间复杂度为 $O(2^N)$。

# 8 [2015-03]Counting on a Tree

## 题目简述

给出大小为 $N$ 的树，每条边带正整数边权，而两点 $(x, y)$ 之间的距离 $D$ 定义为 $x$ 到 $y$ 的路径上的所有边权的最大公约数。题目要求统计有多少对点距离为 1，并给出 $Q$ 个对边权的修改操作，统计在每个修改操作后的点对数量。

## 题目分析

**本题缩写：TREECNT2**
**题目类型：**树，容斥原理，并查集
**题目模型：**树，最大公约数
**题目技巧：**变与不变

## 容斥原理

先对题目中的边权 E 作出变换：将其 $E = p_1^{c_1} \times p_2^{c_2} \times \cdots \times p_k^{c_k}$ 替换为 $E = p_1 \times p_2 \times \cdots \times p_k$，这并不会对结果产生影响。

由于题目中统计的 GCD=1 的情况，可以很自然的联想到容斥原理：对于每一个 X，统计 GCD 是 X 的倍数的方案数，其中 X 中不能含有同一个因子 (这也是对边权做出变换的原因)。

## 并查集

考虑计算某一个 X 的方案数。在这个情况下，只有边权 E 是 X 的倍数才能成为路径。将边权是 X 的倍数的所有边两端的顶点合并，则每一块中顶点都可以两两互达。所以这一部分可以使用并查集实现。

## 修改：变与不变

但题目中还有 $Q(Q \leq 100)$ 个修改操作。对每次修改操作重新计算明显是效率低下的。于是我们把边分为两类：从头到尾都没有参与修改的边权、存在修改的边。

这样，对于每个询问，那些没参与修改的边权处理是一样的，所以只处理一次；存在修改的边有可能不相同，在前面的基础上分别修改后就可以达到。

## 复杂度分析

总时间复杂度为 $O((N+Q)LogC + QC)$。
总空间复杂度为 $O(Q + NLogC + C)$。

# 9 [2015-02]Devu and Locks

## 题目简述

找到满足数字和不超过 $M$，且能整除 $P$ 的 $N$ 位数的个数（允许拥有前导 0）。输出答案模 998244353 后的总数。

## 题目分析

**本题缩写：DEVLOCK**
**题目类型：**数位 DP，计数问题，组合数学，容斥原理卷积，FFT
**题目模型：**数位 DP
**方法技巧：**合并问题，降维处理

## 从简单开始：数位 DP

直观地，可以使用简单的数位 DP 来解决。以转移第 $i$ 位为阶段进行划分，记录当前的余数 $j$ 与数字和 $k$ 就可以得到转移（具体不再详述）。

## 合并问题

在数位 DP 转移中，每一位的转移是类似的，但唯一的区别是 $g = 10^i \bmod P$（在第 $i$ 位加 1 则会给余数加 $g$）。显然地，在所有 $g$ 相等的 $i$ 处的转移都是相同的，于是预处理出 $cnt_g$（$cnt_g$ 为 $g = 10^i \bmod P$ 的 $i$ 的数量），以 $g$ 为阶段进行成段的转移，实现对问题的合并，去掉 $N$ 这一维的影响。

## 总体思路

合并问题后，转移方程为：

$$F[g][j][k] = \sum F[g-1][(j - g * x)\mathbf{mod}P][k-x] \times W[cnt_g][x]$$

其中 $W[cnt_g][x]$ 为在 $cnt_g$ 个位置中增加数字和为 $x$ 的方案数。
至此，题目的总体思路就清晰了：

- 预处理 $cnt_g$
- 预处理 $W[cnt_g][x]$
- DP 方程的转移

下面分三部分讨论这几部分的做法。

## 预处理 $cnt_g$

很明显 $g = 10^i \bmod P$ 有循环性质，暴力找出它的周期计算即可。

## 预处理 $W[cnt_g][x]$：容斥原理

本题的亮点。

如果不考虑限制，这是一个经典的组合问题，方案数为 $\binom{cnt_g+x-1}{x}$。但由于存在题目的限制，每一位不能超过 9，不能直接套用。

解决这个问题存在几种方法：

- 生成函数：待补
- 多项式乘法：待补
- 容斥原理：我们选择不符合的位置个数作为容斥的变量。设 $N = cnt_g, F[M][K]$ 为数字和为 $M$ 时存在**不少于**$K$ 个不符合位置的方案数（显然地 $W[cnt_g][x] = F[x][0]$）。如何计算 $F[M][K]$：首先确定 $K$ 个不符合的位置，满足它们的限制，再把剩下的数字和随意分配（这部分与经典组合问题类似）。最终可以得到 $F[M][K] = \binom{N}{K} \times \binom{N+M-10K-1}{N-1}$。
- 由容斥原理得 $W[cnt_g][x] = F[x][0] - F[x][1] + F[x][2] - F[x][3] + \cdots$。

容斥原理的总复杂度为 $O(PM^2)$，但由于 $cnt_g$ 相等时所求是相等的，而且不同的 $cnt_g$ 值不多，故只需要几次计算就行了。

## DP 方程的转移：降维处理

DP 转移方程与卷积的形式十分接近，但有两点不同：

- 存在两维，且两维都是卷积形式
- 其中 $j$ 这一维存在 $g$ 的系数

因为两维中 $j$ 这一维比较小，于是将两维的信息顺序存储在一维数组中，二维信息对应相乘时能落在一维数组的对应位置。具体地可以构造 $(2M+1)$ 进制，达到两维都能相加的效果。至于 $j$ 这一维存在 $g$ 的系数可以通过调整位置来正确地进行卷积计算。

也可以使用 $P^2$ 次单独的乘法处理。

## 复杂度分析

总时间复杂度为 $O(M^2 + MP^2Log(MP))$。

总空间复杂度为 $O(MP)$。

# 10 [2015-02]Payton numbers

## 题目简述

给出了 Payton 数，以及它们的乘法，0，单位元，单位，整除，质数的定义，判断一个 Payton 数是否为质数。

## 题目分析

**本题缩写：CUSTPRIM**
**题目类型：**抽象代数（群论，环论，域，模，线性空间），欧几里得整环，Miller-Rabin 算法
**题目模型：**抽象代数

## 解题步骤

令 $\omega$ 为满足方程 $\omega^2 = \omega - 3$ 的解，即 $\omega = \frac{1+\sqrt{-11}}{2}$，那么对于每一个三元组 $(a, b, c)$ 都有到域 $Z[\omega]$ 的映射 $\phi(a, b, c) = (33 - 2a - c) + (b - a)\omega$。

于是问题就转化为了判断域 $Z[\omega]$ 下的数 $a + b\omega$ 是否为素数。定义共轭 $(a + b\omega)' = (a + b - b\omega)$，如果令 $Nx = xx'$，那么就有如下结论：

1. 如果 $x$ 不是整数，那么 $x$ 是质数当且仅当 $Nx$ 是质数。
2. 如果 $x$ 是整数，那么 $x$ 是质数当且仅当 $x$ 是质数且要么 $|x| = 2$，要么 $|x| \neq 11$ 且 $-11$ 在模 $x$ 域下没有二次剩余。

可以直接用欧拉判别法和 Miller Rabin 算法来分别判断二次剩余和质数。

## 复杂度分析

总时间复杂度为 $O(LogA)$。
总空间复杂度为 $O(1)$。

# 11 [2015-01]Ranka

## 题目简述

给出一个 $9 \times 9$ 的棋盘，一人执黑，一人执白，交互落子。在每一轮操作中，每人可以把棋子置于空的位置上，或是跳过这一轮。如果这位玩家选择落子，则要判定：

- 若在这一步后，对手的一个联通块（四联通）无气，则将这个联通块中的棋子全部取出。
- 若在这一步后，你的一个联通块无气，则这一步是不合法的。

为了避免循环，我们规定不能出现重复的状态。一个状态由当前执子玩家与棋盘状态组成。如果在一步操作后，这个状态在之前曾经出现，这一步操作是不合法的。

请给出一种合法的包括 $N(N \leq 10000)$ 步的双方的操作方式。

## 题目分析

**本题缩写：RANKA**
题目类型：构造，Gray 码

## 劫

直观地，我们能构造出一种无尽的落子方式：如下图，双方可以在两者之间相互转化。这种情况在围棋术语中称作“劫”，显然地，正式的围棋中也是禁止黑白双方如此往复落子，规定后手不能立即在提子处落子，必须在其它地方下一手后在该处落子。

那么“劫”有什么用处呢？将每个劫根据中间处是黑子还是白子，看作 0 或 1。如果在整个棋盘上放置 P 个“劫”，那么就可以根据状态的 01 情况表示出至多 $2^P$ 种情况。

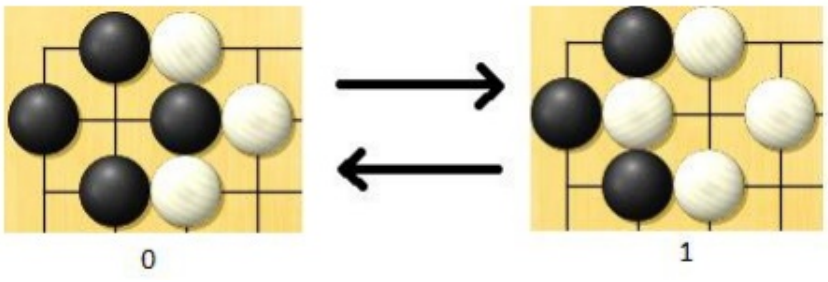


图 1: 劫

## Gray 码

但需要注意的是，状态之间的转变是通过一步的落子实现的，也就是说是通过二进制中其中一位由 0 变 1 或 1 变 0 得到下一个状态。这样就不能简单地遍历所有状态；这时候 Gray 码可以解决这个问题。(https://en.wikipedia.org/wiki/Gray_code)

简单地说，Gray 码是通过每次改变一个位置来不重复地遍历 $000\cdots00$ 到 $111\cdots11$ 的所有二进制数码。从下面这幅图可以简单地看出它的构造方式：

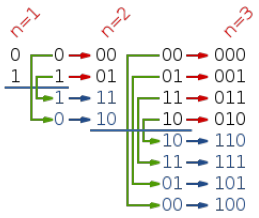


图 2: Gray Code

## 构造

整个棋盘大概可以放置 8 个“劫”。这样只利用这劫可以达到的步数大概为 $256 \times 1.5$（两步中间有的需要跳过）。通过利用余下的空白的格子，一个一个放置，可以到达 $256 \times 1.5 \times 30$ 步左右，可以达到题目的要求。

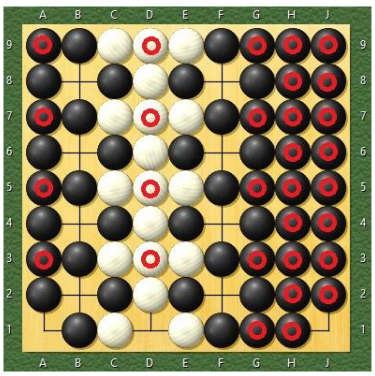

图 3: 余下的位置

## 复杂度分析

总时间复杂度为 $O(N)$。总空间复杂度为 $O(N)$。

# 12 [2015-01]Xor Queries

## 题目简述

给出一个一维数组和若干个不同类型的询问：

- $0\,x$ ：在数组的最后添加 $x$
- $1\,L\,R\,x$ ：在区间 $[L,R]$ 中找到 $y$，使 $x$ **xor** $y$ 最大
- $2\,k$ ：删除数组的最后 $k$ 个数
- $3\,L\,R\,x$ ：统计区间 $[L,R]$ 中有多少个数 $y$ 小于或等于 $x$
- $4\,L\,R\,k$ ：找出区间 $[L,R]$ 中从小到大的第 $k$ 个数

## 题目分析

**本题缩写：XRQRS**
**题目类型**：可持久化数据结构，位运算，TRIE 树
**题目模板**：区间操作，可持久化

## 题目解法

这道题是十分经典的可持久化数据结构类型。将每一个数按照其二进制插入 TRIE 中，维护前缀的 TRIE 树总和。对于每一个询问，在两棵 TRIE 上由上至下的走就可以了。

## 复杂度分析

总时间复杂度为 $O(NLogN)$。
总空间复杂度为 $O(NLogN)$。

# 13 [2014-12]Course Selection

## 题目简述

给出 $N$ 项课程，每项课程都需要在 $M$ 个学期中的某一个学期中完成：对于课程 $i$，在第 $j$ 个学期中完成可获得学分 $X_{i,j}$。

同时，存在 $K$ 个限制，规定课程 $A_i$ 必须在 $B_i$ 之前的学期完成。

最大化学分的总和。

## 题目分析

**题目代号：RIN**
**题目类型：**网络流

## 解法详解

题目给出的模型是很显然的网络流中的差分模型。这种模型的特点是：

- 每个事物有很多项选择，多者选其一；不同的选择有不同的代价
- 最小化/最大化代价
- 对于事物 $i$ 的选择 $C_i$ 与事物 $j$ 的选择 $C_j$ 有约束 $C_i + T \leq C_j$ 或 $C_i + C_j \leq T$（$T$ 为常数）。

这类题目的网络流建模十分简单：将事物 $A$ 的选择 $A_1, A_2, \ldots, A_n$ 用边顺次连接，再将 $S$ 与 $T$ 分别接上头和尾。而每一项选择的代价则作为连接它与前一项的边权。这样可以保证最小割时至少会割掉一条边。

对于约束，则利用最小割的定义，在两排选择的点中相互连边。具体的题目具体分析：在这道题目中，若规定 $A$ 必须在 $B$ 之前，则把 $A_i$ 把一条边连向 $B_{i+1}(i < m)$。这样可以保证在 $A$ 选取第 $i$ 学期时，$B$ 不会选取在第 $i$ 学期之前。

## 复杂度分析

总时间复杂度为 $O(NM \times (NM + KM)^2)$。

总空间复杂度为 $O(NM + KM)$。

# 14 [2014-11]Chef and Churu

## 题目简述

给出一个长度为 $N$ 的数组 $A$ 与 $N$ 个函数 $F$，其中第 $i$ 个函数 $F_i$ 定义为数组 $A$ 标号中为 $L_i$ 到 $R_i$ 的元素总和。

接下来给出 $Q$ 个操作，有两种形式：

- 更改数组 $A$ 某个位置的值
- 查询标号为 $U$ 到 $V$ 的函数值总和

## 题目分析

**题目代号：FNCS**
**题目类型**：数据结构，分块，预处理

## 解题分析

对于这类型的问题我们通过平衡修改与查询的时间复杂度解决。

**解法 1：修改复杂度 $O(1)$，查询复杂度 $O(N^2)$**

**解法 2：修改复杂度 $O(1)$，查询复杂度 $O(NlogN)$**

上面的两种解法都是通过直接修改数组 $A$ 中的元素，查询时枚举每一个函数的取值累加而得。解法 2 中使用线段树或树状数组优化区间求和的询问。

**解法 3：【无修改】查询复杂度 $O(logN \times \sqrt{N})$**

对于查询操作，我们使用分块算法：将连续的 $\sqrt{N}$ 个查询的总和记录下来，这样回答查询操作的时候就可以查询不超过 $\sqrt{N}$ 个整块与不超过 $\sqrt{N}$ 个小段了。

**解法 4：修改复杂度 $O(\sqrt{N})$，查询复杂度 $O(logN \times \sqrt{N})$**

解决了查询操作，接下来我们考虑如何在修改时维护一整块的信息。

假设数组 $A$ 在 $x$ 位置加上了 $y$，对于块 $P$ 的影响：假如块 $P$ 中存在 $i$ 个询问包含位置 $x$，那么块 $P$ 的总和则要加上 $y \times i$。每次查询时对每一个块都更新，这样修改的复杂度为 $O(\sqrt{N})$。

接下来的问题是如何预处理每个位置在每个块中出现的次数。这十分简单：扫描每一个块，对于每一个询问，在其 $L_i$ 加 1，$R_i + 1$ 处减 1，最后扫描就可以在 $O(N)$ 时间复杂度内得出块中的答案。

## 复杂度分析

分块算法是解决数据结构题目中朴素但十分实用的算法。
总时间复杂度为 $O(NlogN + N\sqrt{N} + QlogN\sqrt{N})$。
总空间复杂度为 $O(N\sqrt{N})$。

# 15 [2014-11]Sereja and Order

## 题目简述

给出 $N$ 个程序，每个程序都要在两台电脑上分别运行。第 $i$ 个程序需要在第一台电脑上运行 $A_i$ 秒，在第二台电脑上运行 $B_i$ 秒。一个电脑不能同时运行两个程序，一个程序也不能同时在两台电脑上运行。用最少的时间完成所有程序都在两台电脑上运行的任务。给出具体的操作步骤。

## 题目分析

**题目代号：SEAORD**
**题目类型：**构造，AD-HOC

## 最优值：上界与下界

一种构造问题的题目形式与本题相仿：“构造一种排列/组合/分配方式，使得按某一种方式的估价值最大/最小”。抛开具体的构造方案不谈，先考虑题目中估价值是否能计算出它的上界/下界。

对于本题来说，按照程序的运行限制，它的上界 $T$ 必然满足：

- 对于单个程序，$A_i + B_i \leq T$；
- 对于第一台电脑，$\sum A_i \leq T$；
- 对于第二台电脑，$\sum B_i \leq T$；

令 $T$ 为这三者的最大值，那么估价值的最优值 $Ans$ 满足 $Ans \geq T$。我们可以大胆的猜想，$Ans = T$。

## 具体方案：随机

下面根据 $T$ 的值来讨论不同的构造方案：

- $T = \mathbf{max}\{A_i + B_i\}$
  不再赘述这种情况这么直观的构造方法了。
- $T = \sum A_i$ 或 $T = \sum B_i$
  通过尝试了几组数据，可以得到：如果 $N$ 很大，那么很简单能得到一组合法解；如果特殊构造的 $N$ 很小的数据，则枚举可能的情况也不需要太长的时间。所以使用随机算法，随机数组 $A$,$B$ 的运行顺序，再判断这种顺序是否能在 $T$ 内成功运行。

## 复杂度分析

每次随机的时间复杂度为 $O(N)$；空间复杂度为 $O(N)$。

# 16 [2014-10]Children Trips

## 题目简述

给出一棵 $N$ 个节点的无向带权树，每条边上权值 $D$ 满足 $D=1$ 或 $D=2$。给出 $Q$ 个询问，形式形如 $X,Y,S$，顺次取出从 $X$ 到 $Y$ 树边上的权值，构成序列 $Seq$，将 $Seq$ 分割成尽量少的段数，使每一段的权值和不大于 $S$。

## 题目分析

**题目代号：TRIPS**
**题目类型：**树，根号算法，倍增，二分，分类讨论

## 解题步骤

可以发现，当 $S$ 很大时，形成的段数很少，从而通过二分模拟行进的过程在 $O(Ans\times \mathbf{log}N)$ 的时间复杂度内得到答案。

同时，当 $S$ 很小的时候，通过倍增算法记录每一个节点 $i$ 向上跳 $2^j$ 步到达的节点 $UP[i][j]$。通过 $UP$ 数组可以在 $\mathbf{log}N$ 时间内得到答案，但是对于每一个 $S$ 都会计算一遍整个数组。

所以，对于 $S>\sqrt{N}$ 时，使用第一类算法处理；对于 $S\leq\sqrt{N}$ 时，使用第二类算法。通过根号算法，设计两种不同的针对性方法来平衡总时间复杂度。

## 复杂度分析

时间复杂度为 $O(N^{1.5}\mathbf{log}N)$，空间复杂度为 $O(N\mathbf{log}N)$。

# 17 [2014-10]Union on Tree

## 题目简述

给出一棵 $N$ 个节点的无向有权图，每条边的长度为 1。给出 $N$ 组询问：在每一组的询问中，给出 $K$ 个二元组 $(X, R)$，代表节点 $X$ 控制与其距离不超过 $R$ 的节点；同时询问有多少个节点至少被一个节点所控制。

## 题目分析

**题目代号：BTREE**
**题目类型：**树，虚树，树形动态规划，树分治

### 从特殊开始：$K = 1$

$K = 1$ 是最简单的一种情况：询问一棵树中离点 $X$ 距离不大于 $R$ 的节点个数。假设这棵树随机生成，每个节点的深度均在 **logN** 以内；假设节点 $Y$ 满足要求，以及 $X, Y$ 的 LCA 为节点 $Z$，那么它们应该满足的条件则是 $Dep_X + Dep_Y - 2Dep_Z \leq R$。显然只有根节点到 $X$ 的路径上的节点可以成为它们的 LCA。我们从根节点往目标节点 $X$ 遍历每一个节点 $Z$，依次累加 LCA 恰好为 $Z$ 时的节点数。具体的实现方法是对于每一个节点维护一个桶 $T$，$T[u]$ 记录的是在以它为根节点的子树中深度不超过 $u$ 的节点数。因为树的深度约为 **logN**，所以每一次的询问时间复杂度为约为 $O(\mathbf{log}N)$。

但对于普通的情况，需要使用树分治来保证其时间复杂度。

### 虚树：$K > 1$

为了处理 $K > 1$ 的情况，引入虚树这个工具。对于每一次询问，只关注它的查询点与它们之间的路径上的节点。将“导出树”中的度数为 2 的节点去除，就能得到规模为 $O(K)$ 的虚树。在虚树上计算出每一个节点的控制距离 $R$，并考虑每一条虚树上的边 $(u, v)$：

- $R_u + R_v \leq D_{u,v}$：两点的控制距离没有在边上相交，可以直接运用 $K = 1$ 的算法，统计两点的控制点数。
- $R_u + R_v > D_{u,v}$：两点的控制距离相交，两部分有相交的部分。设 $u$ 为 $v$ 的祖先，$u \to p$ 由 $u$ 控制，而 $q \to v$ 由 $v$ 控制。节点 $u$ 多出的控制部分位于 $q$ 的子树中，可以使用子树查询得出；而 $v$ 多出的控制部分则位于 $p$ 点及其上方。这部分节点的计算可以使用折衷的算法：先计算从 $p$ 出发控制的部分，再一次减去位于 $q$ 子树中的点数。

## 复杂度分析

这道题目中心思想是虚树；但在计算的过程中支持的操作需要用到树分治与可持久化线段树，所以代码量较大，细节较多。

总时间复杂度为 $O(N\mathbf{log}N + \sum K\mathbf{log}N)$。

总空间复杂度为 $O(N\mathbf{log}N)$。

# 18 [2014-09]Rectangle Query

## 题目简述

给定一个二维笛卡尔平面，你需要支持下类三种操作：

- $I\,x_1\,y_1\,x_2\,y_2$：插入一个左下角在 $(x_1, y_1)$，右上角在 $(x_2, y_2)$ 的矩形。
- $D$ index：删除第 index 条插入的矩形。（从 1 标号）
- $Q\,x_1\,y_1\,x_2\,y_2$：询问有多少之前加入的矩形，和左下角在 $(x_1, y_1)$，右上角在 $(x_2, y_2)$ 的矩形有至少一个公共点。

注意，添加操作前后可能会出现一些位置相同的矩形，统计时，你需要视它们为不同的矩形。

## 题目分析

**题目代号：QRECT**
**题目类型**：按时间处理，数据结构，容斥原理

## 按时间处理：线段树

因为题目中存在插入，删除与查询操作，那么可以维护一棵以时间为关键字的线段树：每个区间内存贮这一段时间内完整“存在”的矩形。在实行查询操作时，在线段树从上往下行走，每个区间中的矩形都会影响该询问。但是对于每个询问都这样操作明显会超时，所以对于每个区间同时存贮修改与查询，同时处理。

## 容斥原理

问题现在转化为，若干个插入对若干个询问的影响。考虑到两个矩形的相交形式存在很多种，故改为计算与其不相交的矩形个数。

与一个查询矩形 $A$ 不相交的矩形 $B$ 可能位于其左方，右方，上方，下方。但在计算过程中，严格位于 $A$ 的左上方，左下方，右上方，右下方的矩形会被计算两次，需要把这部分多减去的矩形重新加上。

## 复杂度分析

总时间复杂度为 $O(N\mathbf{log}^2N)$。
总空间复杂度为 $O(N\mathbf{log}N)$。

# 19 [2014-09]Fibonacci Numbers on Tree

## 题目简述

在数学中，斐波那契数列 $F[N]$ 由以下递归关系确定：$F[N] = F[N-1] + F[N-2]$，以及边界条件 $F[1] = 1, F[2] = 1$。

给出一棵 $N$ 个节点的点权有根树，节点从 1 到 $N$ 编号并且节点 1 是根节点。初始，每个节点的权值均为 0，接下来处理 $M$ 个询问，形式如下：

- $A\,x\,y$：将 $x$ 到 $y$ 路径上第 $k$ 个节点的权值增加 $F[k]$（$x$ 节点的权值增加 F[1]，$x$ 到 $y$ 路径上的第二个节点增加 $F[2]$）
- $QS\,x\,y$：视 $x$ 为根，询问以节点 $y$ 为根的子树中所有权值和。
- $QC\,x\,y$：询问 $x$ 到 $y$ 路径上所有节点的权值和。
- $R\,x$：将所有节点的权值还原到第 $x$ 次询问后的状态。如果 $x = 0$，那么回到初始状态，也就是将所有节点的权值清 0。

## 题目分析

**题目代号：FIBTREE**
**题目类型**：数据结构，线段树，轻重剖分，可持久化，数学

## 斐波那契数列

根据递推式，可以得到斐波那契数列 $F$ 的递推式：

$$F[N] = \frac{3+\sqrt{5}}{5+\sqrt{5}}\Big(\frac{1+\sqrt{5}}{2}\Big)^N + \Big(1 - \frac{3+\sqrt{5}}{5+\sqrt{5}}\Big)\Big(\frac{1-\sqrt{5}}{2}\Big)^N$$

在模 $10^9+9$ 意义下，$\sqrt{5}$ 可以找到其对应的数，能够方便的计算出某一项的值。

## 轻重树链剖分，线段树，可持久化

对于同时存在链操作与子树操作的问题，一般使用轻重树链剖分；而通过 DFS 时优先遍历其重儿子得到的 DFS 序，既可实现链操作，又可以实现子树的操作。

题目中同时支持还原到第 $x$ 次询问后的状态，必须要使用可持久化的数据结构。注意在下传标记时要注意新建节点。

## 复杂度分析

总时间复杂度为 $O(N\log^2 N)$。
总空间复杂度为 $O(N\log^2 N)$。

# 20 [2014-07]Game of Numbers

## 题目简述

给出两个长度为 $N$ 的数组 $A_1, A_2, \ldots, A_N$ 与 $B_1, B_2, \ldots, B_N$。

维护两个二元组的集合 $S1, S2$，初始时集合均为空。每次操作，他会选择两个数对 $(i, j), (p, q)$，满足 $(i, j)$ 不在 S1 中，$(p, q)$ 不在 $S2$ 中，$B_j > A_i$，$B_p < A_q$，$\mathbf{GCD}(A_i, B_j) \neq 1$,$\mathbf{GCD}(A_q, B_p) \neq 1$，且 $\mathbf{GCD}(A_i, B_j)$ 与 $\mathbf{GCD}(A_q, B_p)$ 不互质。如果这样的数对存在，将会把 $(i, j), (p, q)$ 分别加入到集合 $S1, S2$ 中。

问最多进行多少次操作。

## 题目分析

**题目代号：GNUM**
**题目类型：**网络流，建模优化

### 解题步骤

这道题是网络流中的匹配模型。

直观地可以得到每次的加入操作实际是将两个二元组匹配起来。我们将二元组 $(u, v)$ 分为两类，$A_u < B_v$ 的放在左边，而 $A_u > B_v$ 的放在右边，每次匹配的两个二元组必定是一左一右。至此转化为二分图的最大匹配。

但由于两边的节点数量达到了 $N^2 = 16000$ 数量级别，不能直接连边。我们把二元组 $(u, v)$ 使用 $\mathbf{GCD}(A_u, B_v)$ 代表它的权值，因为两边取的二元组 $\mathbf{GCD}$ 不为 1，所以必定存在同一个质因数。我们可以在二分图中央建立一列辅助节点，代表每一个可能出现的质因数。每一个二元组都往中间它含有的质因数连边，这样左右就能够间接的连接起来。

这样做的优点是由于每个数最多拥有 9 个不同质因数，既节省的枚举时间，又减少了边数，节约空间开销。

## 卡常数

假设 $\mathbf{GCD}(A_u, B_v) = \prod P_i^{C_i}$，可以用 $\prod P_i$ 来代替，因为两者是等价的；而如果存在多个相同的 $\prod P_i$，可以将它们合并成一个节点。

## 复杂度分析

总时间复杂度为 $O(N^2\mathbf{log}A + \mathbf{MAXFLOW}(N^2 + N\mathbf{log}A, N^2\mathbf{log}A))$。

总空间复杂度为 $O(N^2\mathbf{log}A)$。

# 21 [2014-07]Sereja and Equality

## 题目简述

按以下的方式定义两个长度为 $N$ 的数组 $A, B$ 相似：对于所有的 $i(1 \leq i \leq N)$，满足 $CA(A_i) = CB(B_i)$。其中 $CX(x)$ 等于满足 $X_j < x | 1 \leq j \leq n$ 的 $j$ 的数目。

对于两个排列 $P1, P2$，定义函数 $F(P1, P2)$ 等于满足 $P1[l \dots r]$ 相似于 $P2[l \dots r] | (l, r)(1 \leq l \leq r \leq N)$ 并且 $P1[l \dots r]$ 包含不超过 $E$ 个逆序对的二元组 $(l, r)$ 的数目。

让 $P1, P2$ 取遍所有 $N$ 个数的排列，求出所有 $F(P1, P2)$ 的总和。

## 题目分析

**题目代号：SEAEQ**
**题目类型：**计数问题，组合数学

### 解题步骤

可以直观地得到两个数组相似等价于它们的相对顺序一致。所以它们的逆序对数是相同的，题目中对 P1 的约束实际上对两个数列均有效。

将 $P1[l \dots r]$ 的相对顺序求出，**简化**到一个 $r - l + 1$ 的排列。我们称长度与相对顺序均相同的序列为**本质相同**的序列，那么对于一个长度为 $L$ 的排列 $P$，有 $\binom{N}{L}$ 个序列简化后能得到 $P$，并且这些序列之间是**本质相同**的。由于只有本质相同的序列之间才会贡献答案，所以得到**单个排列**$P$ 对答案的贡献值为

$$(N - L + 1) \times \binom{N}{L}^2 \times [(N - L)!]^2$$

接下来我们只需考虑在长度为 $i$ 的排列中，有多少个排列的逆序对值在 $j$ 以内。这可以通过简单的递推得到结果。

## 复杂度分析

总时间复杂度为 $O(N^3 + QN)$，空间复杂度为 $O(N^3)$。

# 22 [2014-06]Two Companies

## 题目简述

给出一棵大小为 $N$ 的树。现在有若干条路径，我们用 $(S,T,E)$ 来代表一条从结点 $S$ 出发到结点 $T$ 的价值为 $E$ 的路径。每条路径属于 $A$ 公司或 $B$ 公司。

从所有的路径中选取若干条路径，需要满足每一个结点不会被同时被 $A$ 公司与 $B$ 公司的路径覆盖。

最大化路径的价值总和。

## 题目分析

**题目代号：TWOCOMP**
**题目类型：**网络流，最大独立子集
**题目模型：**树上路径相交

## 解题步骤

如果公司 $A$ 的第 $i$ 条路径 $A_i$ 与公司 $B$ 的第 $j$ 条路径 $B_j$ 相交，那么最多在它们两者中选取一个。这让我们联想到一个经典网络流模型：对于公司 $A$ 与 $B4$ 分列两排：对于公司 $A$ 建立 $S \to A_i$ 权值为 $E_i$ 的边，对于公司 $B$ 建立 $B_j \to T$ 权值为 $E_j$ 的边。而若 $A_i$ 与 $B_j$ 相交，则建立 $A_i \to B_j$ 权值无穷大的边。答案则为权值之和减去 $S$ 到 $T$ 的最大流。由于两个相交的公司一定会被割去一条，故一定保证合法；而最小割则使得答案达到最大。

由于 $M$ 比较小，所以可以直接判断公司两两是否相交。如果 $M$ 再大一点可以通过建立虚拟节点连接两旁的节点。

## 复杂度分析

总时间复杂度为 $O(N + M^2\mathbf{log}N + \mathbf{MAXFLOW}(M, M^2))$。
总空间复杂度为 $O(N + M^2)$。

# 23 [2014-05]Dynamic Trees and Queries

## 题目简述

给定一个大小为 $N$ 的有向树，每个结点分配一个权值，执行以下 $M$ 个询问，询问有 4 种类型，强制在线。

1. 新建结点 $Y$，令其权值为 $V$，父亲结点为 $X$。
2. 以结点 $X$ 为根节点，将子树中所有结点的权值加上 $D$
3. 删除以结点 $X$ 为根节点的子树
4. 回答以结点 $X$ 为根节点的子树的权值和

## 题目分析

**题目代号：ANUDTQ**
**题目类型：**数据结构

## 解题步骤

解决子树操作的方法有 dfs 序与欧拉序列；前者适用于静态的树形态；而后者则方便处理动态的树形态。

这道题目中存在插入与删除操作，所以采用欧拉序列来维护树的形态，同时还可以维护树上信息。具体的操作都十分基础，就不再详述。

## 复杂度分析

均摊时间复杂度为 $O(N\mathbf{log}N)$。
空间复杂度为 $O(N)$。

# 24 [2014-05]Sereja and Subsegment Increasings

## 题目简述

给出两个长度均为 $N$ 的整数序列 $A$ 与 $B$。序列中的元素可能为 $0, 1, 2, 3$。在一次操作中，可以选择 $A$ 中的连续一段子序列，将其中的元素增加 1 后模 4。

问让 $A$ 数组转换为 $B$ 至少要执行多少次操作。

## 题目分析

**题目代号：SEINC**
**题目类型**：差分，贪心，AD-HOC

## 解题步骤

遇到对数列连续一段进行的操作（加上一个数），可以使用差分。

具体来说，这道题可以由数组 $A$ 变为数组 $B$ 转化为数组 $C$ 变为全 0 的数组（其中 $C_i = (B_i - A_i)\mathbf{mod}\ 4$）。而如果将一个数组 $C$，按照“每次操作将一段全部减一”，那么最少的操作数为 $\sum \mathbf{max}(C_i - C_{i-1}, 0)$（通过贪心来证明）。

但由于这道题目是在模意义下进行，数组 $C$ 中的每个数可以增加 4 的倍数。那么题目就变成“对数组 $C$ 中的每个元素都可增加 4 的倍数，最小化 $\sum \mathbf{max}(C_i - C_{i-1}, 0)$”。直观地，每个数都最多增加 4。所以使用动态规划可以轻松解决（当然官方题解也给出了贪心的解法）。

## 复杂度分析

总时间复杂度 $O(N)$，总空间复杂度为 $O(N)$。

# 25 [2014-04]Chef and Tree Game

## 题目简述

给出一棵大小为 $N$ 的有根树 $G$，每条边为黑色或白色。Chef 与 Ksen 在树上玩一个游戏，Chef 每次割去一条黑边，并去掉其所连接的子树；Ksen 正好相反。当一个人不能行动时就输了。两人总是按最优策略行动。请输出当 Chef 先手以及 Ksen 先手时谁会赢。

## 题目分析

**题目代号：GERALD08**
**题目类型：**博弈，*超实数*，高精度运算

## 解题步骤

这道题目中需要使用*超实数*（http://www.matrix67.com/blog/archives/6333）来分析。这个游戏的学名叫做 **Blue-Red Hackenbush**(http://en.wikipedia.org/wiki/Hackenbush)，这个游戏的获胜条件的具体证明在这里（http://www.geometer.org/mathcircles/hackenbush.pdf）。

具体地，可以得到以下结论：首先我们需要计算 $F$ 函数

```
def F(n):
    ret=0
    for each vertex v which is child of n:
        ret1 = dfs(v)
        if edge connecting u and v is 0:
           find smallest positive integer p such that ret1+p > 1
             ret += (ret1+p)/(1<<(p-1))
        else:
          find smallest positive integer p such that ret1-p < -1
            ret += (ret1-p)/(1<<(p-1))
    return ret
```

计算完毕后，判断树的根节点 *root* 的 $F$ 值，可以得到：

```
if F(root) = 0: print "Ksen Chef"
else if F(root) > 0: print "Chef Chef"
else: print "Ksen Ksen"
```

在计算 $F$ 值的过程中涉及到实数的运算，并需要足够的精度。故不能简单地使用 double 或 long double 计算。观察题目中涉及的运算，来设计相应的结构来存储。

- 两个数的加法
- 一个数乘以 2 的某整数次幂

- 查询一个数的整数部分

对一个数 $z$，我们用其整数部分 $x$ 与其正小数部分 $y$ 来表示，满足 $z = x + y(y > 0)$。对于 $y$，维护若干个二元组 $(val, bit)$ 与 $exp$(三者均是整数，其中 $val > 0$)，满足 $y = \sum val \times 2^{bit+exp}$。接下来考虑如何实现它们的操作。

- **加法**：分为整数部分与小数部分分别处理：
  1. 整数部分：整数部分直接相加
  2. 小数部分：由于两个数的 $exp$ 不相同，所以在相加时要对二元组中的 $bit$ 进行进一步的调整。
- **乘法**：分为整数部分与小数部分分别处理：
  1. 整数部分：若是正整数次幂，则直接相乘；若是负整数，则一步一步的除以 2：若是偶数则直接相除；若是奇数则将多出的 0.5 加入小数部分。
  2. 小数部分：直接对 $exp$ 进行调整
- **调整**：由于上面的操作可能使小数部分大于 1 或出现冗余状态，所以要对小数部分进行调整：对于 $val > 1$ 的二元组 $(val, bit)$，转化为 $(val/2, bit + 1)$ 与 $(val\%2, bit)$ 的和。对于 $val = 0$ 的二元组则可以直接删除。而对于 $val > 0, bit + exp >= 0$ 的可以加入到整数部分。
- **整数部分**：直接查询 $x$

由于每棵子树中小数部分的二元组组数与子树大小呈正相关，所以加法操作时要将小的合并到大的中。合并的时候要实时对冗余状态进行删除。

## 复杂度分析

我们同时最多拥有 $N\mathbf{log}N$ 个二元组，又因为通过优化合并操作实现，总时间复杂度为 $O(N\mathbf{log}^2N)$，空间复杂度为 $O(N\mathbf{log}N)$。

# 26 [2014-04]Make It Zero 3

## 题目简述

给出长度为 $N$ 的数组 $A$ 与模数 $P$。每次可以对数组中连续的一段进行操作，选择三个整数 $(s,t,k)$，把位于 $s$ 与 $t$ 之间的元素均增加 $k$ 后对 $P$ 取模。

目标是用最少的步数把数组 $A$ 所有元素变为 0。

## 题目分析

**题目代号：LMATRIX3**
**题目类型：**差分，贪心，AD-HOC

## 解题步骤

本题与 **SEINC** 有某些相同的思路：使用差分思想。但在差分出数组 $C$ 后，需要进一步的分析。

题目的目标是把数组 $C$ 中的元素全部变为 0。若进行操作 $(s,t,k)$，那么对数组 $C$ 的影响是 $C_s$ 增加 $k$ 而 $C_{t+1}$ 减去 $k$。假如把每个位置看作一个点，每次操作把点 $s$ 与点 $t+1$ 连起来，那么操作成功的充要条件是每个联通块中的元素之和是 $P$ 的倍数。

证明如下：每次操作对联通块的元素之和没有影响（因为一加一减抵消了）。如果联通块内元素之和不是 $P$ 的倍数，不管通过怎么样的操作都不会使得它们全部变为 0。而若存在一个大小为 $K$ 的联通块，可以通过 $K-1$ 次操作将它们全部变为 0（类似于树形 DP 求出每次操作的具体值）。

所以本题转化为：给出一组数，将它们分配尽量多的集合中，满足每个集合中元素总和是 $P$ 的倍数。这部分需要用到比较复杂的解法，暂时没有弄清楚。

# 27 [2014-03]Chef and Graph Queries

## 题目简述

给定一个无向图 $G$，顶点从 1 到 $N$ 标号，边从 1 到 $M$ 标号。对于 $Q$ 个询问 $(L_i, R_i)$，回答在仅保留标号在 $L_i$ 与 $R_i$ 之间的边时，图 $G$ 有多少联通块。

## 题目分析

**题目代号：GERALD07**
**题目类型**：LCT，分块，莫队，并查集

## 经典解法：Link-Cut Tree

把边从 1 到 $M$ 顺次加入，维护一棵最大生成森林，其中每条边的权值就是它的序号。若某时刻已加入了序号为 1 到 $R$ 的边，则对于询问 $(L, R)$ 的答案则为 $N$-森林中权值不小于 $L$ 的边的数量（这可以使用 Kruskal算法来解释）。

## 直观解法：莫队算法

按照莫队算法中对所有询问排序。使用并查集回答联通块的数量。

对于左端点在同一组的询问，由于右端点单调递增，所以右端的新增边直接加入并查集即可；但左端点有可能左右摆动，所以对于这部分的边在并查集中应支持撤销操作，不进行路径压缩，记录原来的状态复原即可。

## 复杂度分析

经典解法 Link-Cut Tree：总时间复杂度为 $O(N\mathbf{log}N)$。
直观解法莫队算法：总时间复杂度为 $O(N\sqrt{N})$。

# 28 [2014-03]The Street

## 题目简述

在一条大街上有着 $N$ 个商店，商店内每一件纪念品的价格等于标价与本店税费的和。

初始时所有商店均无纪念品，且税费为 0。你需要维护以下三类操作。

1. 1 $u$ $v$ $a$ $b$（新增纪念品）：对于编号为 $u+i$ 的商店，新增一种价格为 $b+i\times a$ 价格的商品 $(0\leq i\leq v-u)$。
2. 2 $u$ $v$ $a$ $b$（调整税费）：对于编号为 $u+i$ 的商店，税费增加 $b+i\times a(0\leq i\leq v-u)$。
3. 3 $i$（查询操作）：询问在编号为 $i$ 的商店中，购买一件纪念品最高的花费为多少。

## 题目分析

**题目代号：STREETTA**
**题目类型：**数据结构

## 解题步骤

题目中涉及到对于区间的运算与单点的查询，很自然地可以想到使用线段树维护区间信息。

实现调整税费非常直观，在每个区间维护等差序列的信息就可以方便的下传。

而实现新增纪念品时，由于维护的是较大值，所以不是那么直观。假设现在区间内存在两种纪念品，该如何维护？把每一种纪念品看作一条直线，那么两条直线最多会有一个交点。如果交点在区间外，那么一种纪念品肯定会严格优于另一种；如果出现在区间内，则在两个子区间的一个，会出现严格优于的情况。递归进入那个相交的区间处理，可以在 $O(\mathbf{log}N)$ 的时间内处理完成。

## 复杂度分析

总时间复杂度为 $O(N\mathbf{log}N)$。
总空间复杂都为 $O(N\mathbf{log}N)$。

# 29 [2014-02]Graph Challenge

## 题目简述

给出 $N$ 个节点 $M$ 条边的有向无权图 $G$，节点按照 DFS 序编号。定义 $X$ 为节点 $Y$ 的**上级节点**，需要满足以下条件:

- 存在一条从 $X$ 至 $Y$ 的有向路 $v$：满足 $v_0 = X, v_k = Y$,$v_i$ 与 $v_{i+1}$ 之间存在一条有向边。
- 对于 $0 < i < k$，满足 $X < Y < v_i$：也就是路径上的节点的编号均比 $X$，$Y$ 要大。

同时定义 $X$ 是 $Y$ 的**超级上级节点**，当其满足节点 $X$ 是节点 $Y$ 的所有**上级节点**中编号最小的。

求出每个节点的**超级上级节点**。

## 题目分析

**题目代号：DAGCH**
**题目类型：**支配树（Dominator Tree）

### 解题步骤

题中所描述的**超级上级节点**就是在 Dominator Tree 中的**半必经点(Semi)**。由于题目中的 DFS 序已经给定，那么直接逆 DFS 序求出每个点的半必经点就可以了。

## 复杂度分析

总时间复杂度为构造支配树的时间复杂度，$O(N\alpha(N))$。
空间复杂度为 $O(N)$。

# 30 [2014-02]Count on a Treap

## 题目简述

维护一个大根堆 Treap，要求支持：

1. 0 $k$ $w$：插入一个关键字为 $k$，权值为 $w$ 的点。
2. 1 $k$：删除一个关键字为 $k$ 的点。
3. 2 $ku$ $kv$：返回关键字分别为 $ku$ 和 $kv$ 两个节点的距离。

保证任何时刻树上节点的关键字与权值都是两两不同的，并且不会删除当前不在 Treap 上的节点。

## 题目分析

**题目代号：COT5**
**题目类型**：数据结构

### 解题步骤

首先观察 Treap 的性质：Treap 上两个节点 $i, j$ 的 LCA 是关键字位于 $ki, kj$ 之间，而权值最大的点。维护一个平衡树便可以快速得到两个节点的 LCA 的具体位置。

再考虑如何求两点之间的距离。直观地，距离为两点的深度减去 LCA 深度的二倍。所以问题转变为求一个点的深度。

把关键字与权值看作两维，把点放在二维平面上。如果一个点与另一点形成的矩形中间没有任何点，那么它们是祖先关系。一个点的深度就是它祖先的数目；所以只用找出在一个点的上方，有多少个点与其中间没有其余的点。

观察可得，满足条件的点在左右两侧分别形成一条递增（递减）链。考虑用线段树维护链的信息：区间 $[l, r]$ 记录以 $l$ 开头对应的递增链的长度。现在考虑区间合并：区间 $[a, b]$ 与区间 $[c, d]$ 合并成区间 $[a, d]$，递增链肯定包括 $[a, b]$ 的递增链，而只包括 $[c, d]$ 的递增链中大于 v 的部分（v 为区间 [a,b] 递增链的最后的值）。而这也是类似的，只需要把 $[c, d]$ 分成两个部分，分情况讨论就可以了。

## 复杂度分析

时间复杂度为 $O(N\mathbf{log}^2N)$，空间复杂度为 $O(N)$。

# 31 [2014-01]Counting D-sets

## 题目简述

若给定一个在 $N$ 维空间中的整点点集，其直径定义为点集中最远的一对点的**切比雪夫距离**。

两组点集被认为是相等的，当其中一组可以通过平移得到另一组点集。准确的说，点集 $X$ 与点集 $Y$ 被认为是相等的，如果：

- 点数相同
- 存在一个 $N$ 元向量 $t = (t_1, \ldots, t_N)$ 使得 $X$ 中的每个点加上 $t$ 后等于 $Y$。

计算在 $N$ 维空间中有多少组互不相等的，直径为 $D$ 的点集。

## 题目分析

**题目代号：CNTDSETS**
**题目类型：**计数问题，容斥原理，组合数学

## 解题步骤

1. 题目要求计算直径为 $D$ 的点集。显然可以转化为计算直径不大于 $D$ 的点集，再减去不大于 $D-1$ 的答案。
2. 由于平移后点集只被计算一次，我们可以将其平移到一个 $[0, D]^N$ 的空间中。我们可以在这空间中随意添点，直径总在 $D$ 以内；但必须要保证 $N$ 维的每一维中都有一个点的坐标值为 0，这样情况才不会计算重复。（可以想象一下二维，三维的情况，如果点集与 $X$ $Y$ $Z$ 平面中有一个没有接触，它就可以通过平移靠近与之接触，所以会被重复计算）
3. 考虑至少有 $P$ 维没有接触的情况数。显然可以得到方案数为

$$\binom{N}{P} \times D^P \times (D+1)^{N-P}$$

对其进行容斥原理就能得到 $N$ 维都接触到的情况了。

## 复杂度分析

总时间复杂度为 $O(TN\mathbf{log}N)$。
预处理二项式系数，总空间复杂度为 $O(N^2)$。

# 32 [2014-01]Counting The Important Pairs

## 题目简述

给定一张连通的简单无向图（无重边，无自环），询问有多少对边的二元组，满足去掉两条边后，整张图会变得不联通？

## 题目分析

**题目代号：TAPAIR**
**题目类型：**树，随机化，割点，割边

## 解题步骤

首先在图中 DFS 搜索，生成一棵生成树。把每一条非树边 $i$ 连接的两点 $A_i, B_i$ 在树中的路径标记上 $i$。一条非树边的标记集合就是它自己；一条树边的标记集合就是所有标记。删除两条边 $i, j$ 后整个图会变得不联通，仅当其满足两条边上的标记集合相同。

证明如下：一条非树边可以看作一条返祖边，让“下边的节点”与“上边”连接。是若是割除一条树边与一条非树边，它们标记相同，则说明该树边的子树中只有一条返祖边与“上边”连接，两条被割断后则必定不联通；若割断两条树边，它们的标记集合相同，首先他们必定是直接祖先关系（不然不可能有相同的标记集合），割断后由于中间部分的节点没有与“上边”连接的返祖边，则会与其余部分不联通。同样地，如果标记集合不相同，则一定连通。

那么应该如何对树边进行标记？我们可以给每一条非树边一个随机的权值，用权值和或权值的异或值代表整个标记集合。在 $2^{64}$ 内进行随机，出错率则会很小了。

## 复杂度分析

总时间复杂度为 $O(N)$。
总空间复杂度为 $O(N)$。

# 33 [2013-12]Query on a tree VI

## 题目简述

给定一个 $N$ 个节点的树，节点从 1 到 $N$ 标号，并且为黑色或白色。根节点的标号为 1。

初始时，所有的节点均为白色。接下来给出 $M$ 个操作：

- 0 $u$：询问 $u$ 所在联通块中有多少节点。若两个节点之间的路径上的节点（包括这两点）颜色均相等，那么两点视为连通。
- 1 $u$：将节点 $u$ 反色。

## 题目分析

**题目代号：QTREE6**
**题目类型：**数据结构

### 解题步骤

这道题目有很多不同的解题思路：HLD，LCT，ETT 等等。其中使用 LCT 最为直观也最为简洁。

把每个点拆成黑点 $b$ 与白点 $w$。当点 $U$ 为黑色，就往它父亲 $V$ 连边 $U_b$,$V_b$；反之亦然。我们可以观察得到每个联通块中，除了最顶端的结点颜色可能不一致外，其余的都是相同的。所以使用 LCT 维护每一个联通块，最后特判顶端结点的颜色才输出答案。

而这道题目中需要维护的是子树信息（不同于往常只用维护链的信息）。所以我们需要记录一个点所有虚边儿子的和。这可以在 Access，Link，Cut 操作很方便的维护得到。

补充一句：如果是对子树进行**修改**的话，那么要用到 Toptree。

注意到这道题目是不用使用换根操作的，因为它一开始已经构造成一棵树的形态。Link，Cut，Query 操作都可以十分简洁地实现。

## 复杂度分析

总时间复杂度为 $O(N\mathbf{log}N)$。

总空间复杂度为 $O(N)$。

# 34 [2013-11]Gangsters of Treeland

## 题目描述

有一棵 $n$ 个节点的树，每个节点有一个颜色，初始每个节点颜色不相同，且以节点 1 为根。定义每个点的权值为这个点到根的路径上不同颜色的个数。现在进行 $m$ 次操作，每次操作为下列三种之一：

1. 将 $x$ 到当前根路径上的所有点染成一种新的颜色；
2. 查询以 $x$ 为根的子树中所有点权值的平均值。

## 题目分析

**题目代号：MONOPLOY**
**题目类型**：动态树，Link-Cut-Tree

## 解题步骤

观察题目中奇怪的距离定义与操作，可以联想到 **Link-Cut-Tree**。一个点到根的距离就是该点到根的路径上的虚边数目，而第一种操作就是**Link-Cut-Tree**中的 Access 操作。

用线段树来维护树上的信息。Access 操作中可能会把一条边由虚边变为实边（或相反），可以利用 DFS 序维护子树信息。

## 复杂度分析

总时间复杂度为 $O(N\mathbf{log}^2N)$。
总空间复杂度为 $O(N)$。

# 35 [2013-08]Music & Lyrics

## 题目描述

给出 $N$ 个单词，统计它们在 $M$ 首歌曲中一共出现了多少次。

## 题目分析

**题目代号：LYRC**
**题目类型：**字符串，AC 自动机

## 解题步骤

统计出现次数相当于将每首歌都与每个单词匹配。对于多串匹配，可以使用经典的 AC 自动机解决。关于 AC 自动机的有关习题，已经有很详细地对于经典题目的总结（http://notonlysuccess.me/?p=607）。

## 时间复杂度

总时间复杂度为 $O(\sum N \times |A| + \sum M)$，其中 $|A|$ 为字符集大小。
总空间复杂度为 $O(\sum N \times |A|)$。

# 36 [2013-08]Prime Distance On Tree

## 题目描述

给出大小为 $N$ 的树，统计有多少对节点之间的距离为质数。

## 题目分析

**题目代号：PRIMEDST**
**题目类型：**分治，卷积

## 解题步骤

求点对的数量可以很直观的想到分治。而合并两棵子树则需要 FFT 优化卷积的计算。由于卷积一次只能计算一对，所以使用边分治的效果比点分治要好。

但在处理极端数据上（比如菊花图），边分治的达不到“分治”的效果。所以要通过拆点转化树的结构，来达到“分治”的效果。具体地，对于儿子数量大于 2 的结点，建立虚拟结点，将每个点的度数控制在 3 以内。

## 复杂度分析

总时间复杂度为 $O(N\mathbf{log}^2N)$。
总空间复杂度为 $O(N)$。

# 37 [2013-06]Two k-Convex Polygons

## 题目描述

给出 $N$ 个数，从中选出两组数，每组 $K$ 个数，以这 $K$ 个数为边长能构造出一个多边形。

## 题目分析

**题目代号：TKCONVEX**
**题目类型：**构造，排序

## 解题步骤

若题目只要求选出一组数，可以采用贪心的策略：将这 $N$ 个数从小到大进行排序，判断相邻的 $K$ 个数是否构成一个多边形。

现在题目要求求出两组数，可以采取类似的策略：对数组排序后，首先判断是否能找到两个不相交的合法区间，找到了则输出这组解；否则我们枚举所有长度为 $2K$ 的区间，搜索区间中的数属于那一组数，并判断是否合法。

可以看出，当 $N$ 增大到一定程度时，第一种情况一定能找到一组解（构造一组不符合的解，数的规模以斐波那契数列的增长速度增长），所以第二种情况只有会在 N 较小时进行判断。故时间复杂度是可以保证的。

## 复杂度分析

时间复杂度：当 $N$ 较大时，时间复杂度为 $O(NLogN)$，当 $N$ 较小时，复杂度为 $O(N \times 2^N)$。

空间复杂度为 $O(N)$。

# 38 [2013-05]Queries on tree again!

## 题目描述

给出一个 $N$ 个结点，$N$ 条边的无向带边权图。图中存在一个奇环。给出两种操作：

1. 将点 $x$ 到点 $y$ 的路径上的边取反。若存在两条路径，对较短的一条操作（经过结点数较少一条）。
2. 查询点 $x$ 到点 $y$ 的路径上，连续一段边权和的最大值。若存在两条路径，对较短的一条操作。

## 题目分析

**题目代号：QTREE**
**题目类型：**数据结构

### 解题步骤

对于环套树的问题，解决的方式是比较简单的：维护每一棵外向树与整个环。对于树的情况，使用树链剖分解决；对于环的情况，维护一棵线段树即可。

## 复杂度分析

时间复杂度为 $O(NLog^2N)$，空间复杂度为 $O(N)$。

# 39 [2013-03]Little Elephant and Colored Coins

## 题目描述

一头利沃夫动物园里的小象非常喜欢硬币，但是最重要的是他喜欢彩色的硬币。

他有 $N$ 种类型的硬币，编号从 1 到，第 $i$ 种硬币价值 $V_i$ 美元，颜色是 $C_i$，对于每种硬币，他都有无限个。这头小象想用这些硬币恰好组成 $S$ 美元。组成这 $S$ 美元的硬币最多能有多少中颜色？如果不可能，输出 -1。

## 题目分析

**题目代号：LECOINS**
**题目类型：**动态规划

### 解题步骤

#### 简化版

考虑简化版的问题：除去颜色的限制，只考虑能否组成 $S$ 美元。

找到硬币中面值最小的，设其价值为 $M$。接下来在 $M$ 的同余系中讨论：$F[i](0 \leq i < M)$ 表示使用除去价值为 $M$ 的硬币能够组成最小面值，而且满足 $F[i]$ 模 $M$ 为 $i$。把每个 $i$ 抽象为一个顶点，那么每枚硬币可以看作一条有向边，在这个图中跑一次最短路就可以得到 $F$ 值。

现在判断能否组成 $S$ 美元：若 $F[S \bmod M]$ 比 $S$ 小，则可以通过面值为 $M$ 的硬币弥补组成；如果相等则不需补就可以组成；如果大于 $S$ 则肯定不能组成。

#### 正式版

拥有了颜色这一限制后，直接新增一维颜色数的状态。同时注意由于在转移时没有考虑面值为 $M$ 的硬币的影响，所以要注意细节：

- $F[S \bmod M] < S$，需要增加面值为 $M$ 的硬币，**可能**会对颜色总数产生影响（需要记录是否使用过同种颜色）。
- $F[S \bmod M] = S$，不需要增加额外硬币。

在是实现时可以把与面值为 M 的硬币相同颜色的放在最后转移，就不用额外的记录另一维了。

## 时间复杂度分析

时间复杂度为 $O(CNM + QN)$。

空间复杂度为 $O(CNM)$。

# 40 [2013-01]A New Door

## 题目简述

大厨非常擅长烹饪，他做的菜是城镇最好的，因此他打算装饰一下他的餐厅。顾客到店第一眼看到的是餐厅的门，所以大厨想从装饰门开始。他昨晚已经想出了一个新的设计，从门上挖去一些圆形，再在这些位置装上玻璃，让玻璃形成奇特的形状。

形式上，我们可以将大厨的一系列过程抽象为以下过程。令门为一个坐标平面上以 $(0,0),(W,0),(0,H),(W,H)$ 为四角的矩形。一开始门是白色的，而坐标平面其他部分是黑色的。每一步大厨会将一个圆的内部涂黑。在他涂完所有圆之后，他将门内的黑色部分（在代表门的矩形内的黑色部分）挖掉，并装上玻璃。

门的奇特程度可以用窗户在门内的边界的周长表示（请参考样例中第二组数据理解）。大厨在涂色之后已经很疲惫了，他请你算出这个数。

## 题目分析

**题目代号：ANDOOR**
**题目类型：**计算几何，扫描线
**题目模型：**圆的交点

## 解题步骤

题目中要求求出所有圆的交集的边长，考虑单独对一个圆进行讨论。

如果圆上的一段处于圆的交集上，它一定不被任意一个其他的圆包含。所以求出其余的圆在其上的交的圆弧，使用扫描线即可。

注意几点：如果圆在矩形外则不讨论；如果圆被另一个圆严格包含则不讨论；矩形的边界拆成四条直线，计算直线与圆的交点即可。

## 复杂度分析

时间复杂度为 $O(N^2LogN)$，空间复杂度为 $O(N)$。

# 41 [2012-12]Different Trips

## 题目简述

大厨正在 Byteland 享受一个完美的假期。Byteland 是一棵树，标号为 $1 \sim N$，其中城市 1 是根结点。

假如两个城市的度数相同，则他们被认为是相似的。

大厨会选择一些路径进行旅游：首先大厨确定城市 $A$，然后在 $A$ 到根的路径上确定一个城市 $B$，然后城市 $A$ 到 $B$ 就是一条可行的路径。

两条路径被认为是相似的当且仅当他们的长度相同且按顺序一一对应的城市都相似。

大厨想知道有多少种不相似的可行的路径。

## 题目分析

**题目代号：MATCH**
**题目模板：**树串
**题目类型：**字典树，后缀自动机

### 解题步骤

如果题中的树形态是一条链的话，那么就是求字符串中有多少不同的子串。可以使用后缀数组或后缀自动机解决。

由链形态转变为树形态，那么按照 BFS 序加入字符，构建整棵树后缀自动机即可。

TRICK: 按照后缀自动机的定义，应该是不能在同一个位置后添加同一个字母的，所以要先建立一棵字典树。但是直接按照题目给出的树形态添加也不会出错。这是为什么呢？

## 复杂度分析

总时间复杂度为 $O(N)$。
总空间复杂度为 $O(N)$。

# 42 [2012-11]Martial Arts

## 题目简述

两支分别有 $N$ 个选手的队伍参加这次比赛，分为主队与客队。作为组织者，Chef 要安排 $N$ 场比赛。一场比赛两支队伍都要有一名选手参赛。裁判会根据他们的表现给队伍加分。一支队伍的最终得分等于队伍中所有选手参加比赛后的得分总和。每一个选手都要恰好参加一场比赛。

Chef 对所有的选手都非常了解，他想安排每场比赛的配对来最大化主队和客队的得分差。 我们令主队最终得分为 $H$，客队最终得分为 $G$。那么 Chef 就是想要最大化 $H-G$。如果他能有多种方式达到目的，则他还想要最大化 $H$。

然而，他不是唯一的有隐藏动机的人。

当所有比赛的配对公布后，客队教练能编个理由——他们有个选手受伤了。

这意味着这个选手所在的比赛就取消了，这场比赛也不会算分。

当决定哪个选手不出战时，客队教练有着与 Chef 类似的目的。他的目的是最大化 $G-H$，其次最大化 $G$。

Chef 被客队教练的策略弄糊涂了。他来请教你怎样安排比赛最优。

## 题目分析

**题目代号：MARTARTS**
**题目类型：**KM 算法（最大权匹配）

## 解题步骤

假设没有客队的操作，那么就可以直接使用 KM 算法得到结果。

由于存在客队删除操作肯定是对权值最大的边操作，我们考虑按边权大小的顺序加入边。我们强制当前处理的这条边 $E$ 必定要被选取，再计算余下顶点的 KM 算法。

于是我们可以这样：按边权排序，初始置所有边权为负无穷，顺序加入边时把当前的边 E 权值修改为正无穷，表示一定要选取，再计算该图的 KM 值；计算后把权值 E 还原。

接下来就要考虑在 KM 算法中如何方便的计算修改边权后的 KM 的权值和。

由于 KM 算法的核心是标号数组，所以就要对标号数组进行修改，并重新寻找匹配路径。在取消 (X,Y) 之间的匹配时，对 X 与 Y 的标号都要修改成临界值（刚好能够进行下一次匹配的值）。

## 复杂度分析

总时间复杂度为 $O(N^4)$，总空间复杂度为 $O(N^2)$。

# 43 [2012-09]Annual Parade

## 题目简述

在魔法大陆上，每年春天都会举办一场游行。魔法大陆上有 $N$ 座城市，城市之间有 $M$ 条有向道路相连。

在游行期间，将会有一些（可能 0 个）英雄在魔法大陆上旅行。对于每一名英雄，他从城市 $begin[i]$ 开始，途径一些城市，最终在城市 $end[i]$ 结束。注意，$begin[i]$ 可能等于 $end[i]$，但他在路径中必须至少向另外的一个城市移动（即路径不能为自环）。他可以多次经过一条道路，但每次都需要付出费用。

游行的费用分成三部分：

1. 沿道路移动的总费用。（如果一条道路被 $k$ 人次经过，费用就将计算 $k$ 次）
2. 如果一名英雄的 $begin[i] \neq end[i]$，需要花费 $C$ 来送他回家。
3. 如果某个城市没有被任何一名英雄经过，需要花费 $C$ 来赔偿该市市民。

$C$ 的值每年都可能改变，而我们知道接下来 $K$ 年的这个值。你的任务是，计算每一年的最小花费。

## 题目分析

**题目代号：PARADE**
**题目类型**：网络流，增广路径

## 解题步骤

构建最小费用最大流模型：考虑把每做城市 i 拆成两个点 $(i, i')$：一条道路 $(X, Y, Z)$，连边 $(X, Y', INF, Z)$；另外连边 $(S, i, 1, 0)$，$(i', T, 1, 0)$，$(i', i, INF, 0)$。初始图中是没有增广路径的，答案为 $N \times C$。

观察新增一条增广路径它有什么含义：连接两条其余的增广路径，费用减少 $C$；连接一条增广路径的首与尾

# 44 [2012-08]A Game of Thrones

## 题目简述

布朗和提利昂大帝在争夺冰封铁王座的最后时刻，在两双方骑士的一直认同下，决定采用一个权利的游戏解决问题。当然是一个数字游戏。规则如下：

1. 一开始，有 $N$ 个数字写在一张纸上 (可能有相同数字)。
2. 双方轮流，布朗先手。
3. 在第一轮中，布朗先选择一个纸上的数字，把它称做游戏的当前数字。
4. 之后从第二轮开始执行当前回合的人都按如下操作：我们称现在的当前数字为 $u$。将 $u$ 从纸上擦去 (如果有多个擦去任意一个)。然后选择另一个纸上的数字 $v$ 作为当前数字，$v$ 要满足与 $u$ 刚好相差一个质因子。
5. 无法完成操作的人输。

请你计算出谁将赢得游戏，如果两个人都采用最优策略。

注：对于两个数 $u$，$v$ 刚好相差一个质因子需满足：不妨设 $u < v$，则 $u|v$，且 $v/u$ 是一个质数。

## 题目分析

**题目代号：GTHRONES**
**题目类型：**网络流，Miller Rabin 算法

## 解题步骤

对题目中的每一个数字建立一个结点，拥有奇数个不同因子的放在左侧，拥有偶数个的放在右侧。对于匹配的数对之间连边。如果这个二分图中存在完备匹配，则后手获胜（后手的决策是对应的匹配点）；否则先手获胜。

题目中还要求求出先手可以胜出的选的数字。我们可以使用退流来判断每一条边是否可行；也可以利用匹配中的匹配树的性质，遍历匹配树找出所有可行边。

## 复杂度分析

时间复杂度为 $O(MAXFLOW(N, N^2))$，空间复杂度为 $O(N^2)$。

# 45 [2012-07]Dynamic GCD

## 题目简述

给你一棵 $N$ 个结点的树，每一个结点上有一个正整数权值，其中第 $i$ 个结点上的权值是 $v[i]$，你的程序必须维护 2 种操作：

1. 表示为 $Fuv$ 的查找操作：找出从 $u$ 到 $v$ 的唯一路径上所有点权值的最大公约数。（包括 $u$ 和 $v$）
2. 表示为 $Cuvd$ 的修改操作：将从 $u$ 到 $v$ 的唯一路径上所有点权值加上 $d$。（包括 $u$ 和 $v$）

## 题目分析

**题目代号：DGCD**
**题目类型**：数据结构，GCD，差分

### 解题步骤

注意到 $GCD(A,B) = GCD(A,B-A)$，我们对一个结点可以维护两个值：它本身的值，它与父亲的差值。对于查找操作，我们只需查询 LCA 位置的本身的值，与两条链上差值的 GCD 即可。修改的时候对差值的修改只需在端点上修改；而对本身值的修改则使用线段树的区间赋值操作即可。

## 复杂度分析

时间复杂度为 $O(NLog^2N)$，空间复杂度为 $O(N)$。

# 46 [2012-06]Expected Maximum Matching

## 题目简述

按以下方式随机生成一个二分图：左边第 $i$ 个点和右边第 $j$ 个点之间有边的概率为 $f[i][j]$。求这样生成的二分图的最大匹配的期望值。左边的点数为 $N(1 \leq N \leq 5)$，右边的点数为 $M(1 \leq M \leq 100)$。

## 题目分析

**题目代号：MATCH**
**题目类型：**概率，动态规划，二分图，Hall 定理
**题目模型：**二分图匹配

## 解题步骤

对于二分图的匹配问题，可以参考 Hall 定理：对于二分图 $G = (S, T, E)$，存在 $S$ 的完备匹配当且仅当对于 $S$ 的每一个子集 $U$，它的引出子集 $V$ 满足 $|U| \leq |V|$。引出子集 $V$ 包括与 $U$ 相连的所有点。

那么由于 $N$ 较小，可以考虑 $S$ 中所有子集是否存在完备匹配。设状态 $dp[i][S]$ 为考虑 $T$ 中的前 $i$ 个点，状态为 $S$ 的子集存在完备匹配。那么假如知道新加入的点 $i+1$ 连接的子集为 $U$，便可以得到新的状态 $S'$。假设子集 $(\phi), (A), (B)$ 存在完备匹配，而新加入点连接点 $(C, D)$，那么子集 $(\phi), (A), (B), (C), (D), (AC), (BC), (AD), (BD)$ 在新图中则存在完备匹配。举多几个例子可以得到如何得到新状态。

理论上最多会有 $2^5 = 32$ 个不同的子集，所以最多会有 $2^3 2$ 种不同子集情况。但由于其中存在很多矛盾的情况（例如只有 $(AB)$ 存在完备匹配是不可能的），经过 BFS 可以得到最多 $P = 4602$ 个状态。

## 复杂度分析

时间复杂度为 $O(2^N MP)$，空间复杂度为 $O((2^N + M)P)$。

# 47 [2012-05]Little Elephant and Boxes

## 题目简述

利沃夫动物园的小象有 $N$ 个盒子，每个盒子里有一些钱或者钻石。他不知道每个盒子里具体是什么，但他知道如果打开第 $i$ 个盒子，他有 $P_i/100$ 的概率能获得 Vi 美元的钱，而有 (1–$P_i$/100) 的概率能获得一个钻石。现在有 $m$ 个物品，分别编号为 0 到 $m$–1，第 $j$ 个物品需要花费恰好 $C_j$ 美元的钱和 $D_j$ 个钻石。小象非常聪明，当他获得了一定量的钱和钻石后，他总会买尽可能多的物品。注意，每个物品只能购买一次。现在请你帮助小象计算出，当他打开所有的盒子后，他期望能够买到的物品个数是多少？

## 题目分析

**题目代号：LEBOXES**
**题目类型：**折半搜索（Meet-in-the-Middle），动态规划

### 解题步骤

假如我们枚举每一个盒子的状态，可以使用动态规划解决：设 $F[i][j]$ 为得到 $i$ 个物品，最多使用 $j$ 颗钻石的前提下，花费的最小值。$F$ 可以在 $O(NM^2)$ 时间复杂度内得到。

由于 $N \leq 30$，不能直接枚举状态。考虑折半搜索：把盒子分成两个部分，枚举其中一个部分的状态，并预处理另一半的状态。具体来说，预处理保存一些状态 (Value, Dia, P) 分别表示总金钱，总钻石数及这个情况的概率。

假设枚举这一半的金钱为 A，钻石数为 B，枚举另一部分的钻石数 C，以及购买的物品数 X。则另一部分的金钱 D 应该满足 $F[X][B+C] \leq A+D < F[X+1][B+C]$（至少购买 X 个物品，而不足够去买第 X+1 个物品）。所以应该把所有钻石数相同的状态按照 Value 排序，则满足条件的是其中连续的一段状态。

## 复杂度分析

设前半段的个数为 A，后半段的个数为 B。
则时间复杂度为 $O(BN \times 2^B + M(BN)^2 \times 2^A)$。
空间复杂度为 $O(2^B + NM)$。
计算时前半段取 $N/3$，后半段取 $2N/3$，能达到较优的时间复杂度。

# 48 [2012-04]Substrings on a Tree

## 题目简述

Chefland 的科学家已经成功制造了一项新的发明！他们发现一种新的方式来表示一个长度为 $N$ 的字符串：考虑一棵有 $N$ 个节点的树，以 1 号点为根，对于每一个节点都有一个小写字母。由此我们得到一个" 树串"。科学家们还不知道怎么来读这个" 树串"，但是他们已经定义了" 树串" 的子串。一个字符串是该" 树串" 的子串，当且仅当这个串可以被获得通过树上一个节点到他一个子孙路径上所构成的字符串, 当然空串也属于子串。

现在 Chefland 的研究人员希望得到你的帮助。他们会给一个 $N$ 个节点的" 树串"，请输出这个" 树串" 有多少不同的子串。之后他们还会给你 $Q$ 次询问，对于第 $i$ 次询问，他们会给你一个包含 26 个小写字母的排列 $P$ 和一个数 $K$。$P$ 表示这 26 个小写字母的大小关系，从小到大给出。这次询问要求将所有不同的子串按照 $P$ 指定的字母大小关系按照字典序排序后，输出第 $K$ 小的字符串。

## 题目分析

**题目代号：TSUBSTR**
**题目模板：**树串 **题目类型：**字典树，后缀自动机

## 解题步骤

题中给出对“树串”的定义就是对树建立后缀自动机的模型。在建立后缀自动机后，就是简单的统计：设 $F[S]$ 为状态 $S$ 往后可以走出多少个不同的子串。按照拓扑序计算出 $F$ 后，可以在线性时间内在自动机上行走得到第 $K$ 个子串。

## 复杂度分析

时间复杂度为 $O(N)$。
空间复杂度为 $O(N)$。

# 49 [2012-04]Find a special connected block

## 题目简述

给出一个 $N \times M$ 的矩阵，每个格子上填着从 -1 到 $N \times M$ 中的一个数。

你的任务是找到一个联通块 (只能通过上下左右四个方向连接)，在这个联通块里，至少要包含 $K$ 个不同的正数，且不能有 -1。

我们还会给出选取每一个格子所需要的代价，我们要求你选出的连通块所包含的格子的代价和最小。

请输出这个最小值。

## 题目分析

**题目代号：CONNECT**
**题目类型：**随机化，斯坦纳树

## 解题步骤

考虑题中最紧的限制 $K \leq 7$。若题目中每个格子中的权值均在 K 以内的话，则能使用状态压缩 DP，类似于斯坦纳树的转移，得到答案。

但由于颜色总数可达 $N \times M$，所以采用一种折衷的办法：每次随机把每一种颜色的变为 1 到 $K$ 中的一种。这样只会把限制缩紧，变换后的图中的解一定满足原图。由于这样可能找不到最优解，所以要多次随机。

## 复杂度分析

令 $T$ 为随机总次数。
时间复杂度为 $O(TNM3^K)$。
空间复杂度为 $O(NMK)$。

# 50 [2012-03]Evil Book

## 题目简述

厨师 Ciel 从《邪恶之书》中获得了巨大的烹饪力量。然而，Ciel 必须贡献 666 点魔法力量给邪恶之书。

除了 Ciel，世界上还有 $N$ 个厨师。第 $i$ 位厨师有 $C_i$ 点烹饪力量和 $M_i$ 点魔法力量。Ciel 可以通过在一次烹饪战斗中打败第 $i$ 位厨师获得 $M_i$ 点魔法力量，同时需要付出 $C_i$ 点努力。在这场战斗之后，第 $i$ 位厨师的魔法力量 $M_i$ 变为 0。

如果她给邪恶之书 $X$ 的魔法力量，邪恶之书可以帮助她。也就是说，通过消耗 $X$ 点魔法力量，她可以选择某个 $i$ 并把第 $i$ 位厨师的烹饪力量 $C_i$ 和魔法力量 $M_i$ 都乘以 1/3。注意，只有当她拥有至少 $X$ 点魔法力量时才能请求邪恶之书的帮助。

初始时，Ciel 没有魔法力量。至少总共需要多少努力才能收集到至少 666 点魔法力量？

## 题目分析

**题目代号：EVILBOOK**
**题目类型：**搜索，模拟

## 解题步骤

注意到我们可以在与第 $i$ 位厨师战斗前才削弱他的力量。以削弱的次数为阶段搜索挑战的次数与顺序，注意最优化与可行性剪枝。

## 复杂度分析

时间复杂度为搜索的复杂度，空间复杂度为 $O(N)$。

# 51 [2012-02]Find a Subsequence

## 题目简述

给你一个长度为 $N$ 的数组 $A[0], A[1], ..., A[N-1]$ 和一个字符串 $B$，$B$ 是“12345”的排列。你需要找到一个长度为 5 的 $A$ 的子序列，该子序列中的元素互不相等，并满足他们的相对大小和 $B$ 一样。

这意味着，如果 $(i0, i1, i2, i3, i4)$ 是找到的子序列的下标 $(0 <= i0 < i1 < i2 < i3 < i4 < N)$，那么 $A[ik]$ 是五个数中第 $B[k]$ 小的。

## 题目分析

**题目代号：FINDSEQ**
**题目类型：**Ad-hoc，枚举，预处理

## 解题步骤

枚举 $i1, i3$，将 $[1, N]$ 分成三段。这样是因为位置若相互独立的话，讨论会变得更加简单。

接下来我们贪心的选取 $i0, i2, i4$ 的值。

设 $Bx < By < Bz (x, y, z \in \{0, 2, 4\})$。

- 首先考虑 $Bx$。不考虑 $By$ 与 $Bz$ 对它的限制，只考虑 $i1, i3$ 对它的限制，它尽量取小会给 $By$ 与 $Bz$ 带来更大的空间。
- 同理，$By$ 在 $i1, i3, Bx$ 的限制下取最小会给 $Bz$ 带来更大的空间。

所以我们需要找到区间某一段区间 $[x, y]$ 中比某一个数 $P$ 大的最小的数 $Q$。这个可以在 $O(logN)$ 时间内通过二分得到。

考虑**优化**它的做法：我们观察到三段区间中一段是前缀，一段是后缀，一段位于中部。对于前缀与后缀，通过预处理，回答的时间可以被优化至 $O(1)$。但对于中间就不能做到这么优的复杂度。

但如果把中间位置放到最后处理，只需判断一段区间 $[x, y]$ 是否存在权值在 $[P, Q]$ 之间的数。这也可以通过预处理 $O(1)$ 回答。但我们不能保证中间的值是最大的，不能最后处理。

但可以这样考虑，如果中间的值是最小的，可以反过来，依次处理 $Bz$ $By$，贪心它们的最大值；中间的值位于中间，可以贪心 $Bx$ 的最小，而贪心 $Bz$ 的最大，效果是一样的。所以无论三个数的相对顺序如何，都可以找到一种方法，先处理左右两段（通过预处理 $O(1)$ 回答），最后处理中央位置（只需知道个数而不需求出具体值）。若找到一组解，那么扫描一下就可以知道它的具体值了。

## 复杂度分析

**时间复杂度**为 $O(TN^2)$。**空间复杂度**为 $O(N^2)$。