

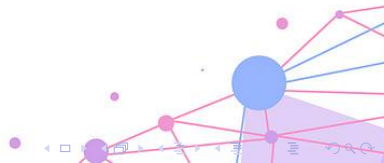


杂题选讲

任轩笛

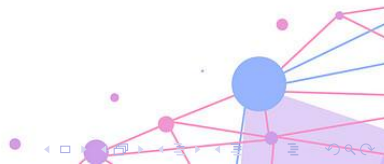
绍兴市第一中学

2018 年 3 月 18 日



前言

挑了一些我觉得比较有趣的题目。



前言

挑了一些我觉得比较有趣的题目。
部分题目比较老，可能做法也很多，欢迎大家踊跃发言。

前言

挑了一些我觉得比较有趣的题目。
部分题目比较老，可能做法也很多，欢迎大家踊跃发言。
希望这节课能给大家带来欢乐。

Binary Cards

Codeforces #469 (Div.1) E

题意

给出 n 个需要表示的数，你需要用最少的 2^k 或 -2^k ，使得能拼出所有需要表示的数。输出方案。

范围

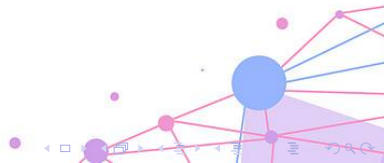
$n, |A_i| \leq 100000$ 。



Binary Cards

Codeforces #469 (Div.1) E

首先观察到同一个数只会选最多一次，因为两个2不如一个2一个4优。



Binary Cards

Codeforces #469 (Div.1) E

首先观察到同一个数只会选最多一次，因为两个2不如一个2一个4优。

然后可以发现2和-2不会同时选，不如选2和-4或者4和-2。

Binary Cards

Codeforces #469 (Div.1) E

首先观察到同一个数只会选最多一次，因为两个2不如一个2一个4优。

然后可以发现2和-2不会同时选，不如选2和-4或者4和-2。

考虑缩小问题规模。如果当前所有数都是偶数，直接全都除以2。

Binary Cards

Codeforces #469 (Div.1) E

首先观察到同一个数只会选最多一次，因为两个2不如一个2一个4优。

然后可以发现2和-2不会同时选，不如选2和-4或者4和-2。

考虑缩小问题规模。如果当前所有数都是偶数，直接全都除以2。否则一定要在-1和1中选择一个。

Binary Cards

Codeforces #469 (Div.1) E

首先观察到同一个数只会选最多一次，因为两个2不如一个2一个4优。

然后可以发现2和-2不会同时选，不如选2和-4或者4和-2。

考虑缩小问题规模。如果当前所有数都是偶数，直接全都除以2。否则一定要在-1和1中选择一个。

直接枚举选哪个，暴搜下去，每一层都把数字去个重。

Binary Cards

Codeforces #469 (Div.1) E

首先观察到同一个数只会选最多一次，因为两个2不如一个2一个4优。

然后可以发现2和-2不会同时选，不如选2和-4或者4和-2。

考虑缩小问题规模。如果当前所有数都是偶数，直接全都除以2。否则一定要在-1和1中选择一个。

直接枚举选哪个，暴搜下去，每一层都把数字去个重。

这样第 k 层数字个数不会超过 $O(\frac{A_i}{2^k})$ 。

Binary Cards

Codeforces #469 (Div.1) E

首先观察到同一个数只会选最多一次，因为两个2不如一个2一个4优。

然后可以发现2和-2不会同时选，不如选2和-4或者4和-2。

考虑缩小问题规模。如果当前所有数都是偶数，直接全都除以2。否则一定要在-1和1中选择一个。

直接枚举选哪个，暴搜下去，每一层都把数字去个重。

这样第 k 层数字个数不会超过 $O(\frac{A_i}{2^k})$ 。总的复杂度即为 $O((n + A_i) \log n)$ 。

不上升序列

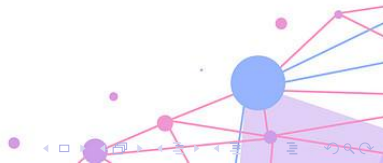
hihoCoder挑战赛29 D

题意

给出一个整数序列 A ，每次可以花费1的代价给某个数 $+1$ 或者 -1 ，求最少需要多少代价使其变成一个不上升序列。

范围

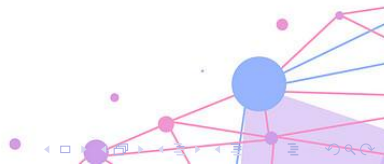
$n \leq 500000$ 。



不上升序列

hihoCoder挑战赛29 D

假设求的是不降序列。不升序列同理。



不上升序列

hihoCoder挑战赛29 D

假设求的是不降序列。不升序列同理。

设 $f(x)$ 表示最后一个数 $\leq x$ 时的最优答案，由于每次加的代价都是线性的，它显然形如一条每段斜率都是整数的折线。

不上升序列

hihoCoder挑战赛29 D

假设求的是不降序列。不升序列同理。

设 $f(x)$ 表示最后一个数 $\leq x$ 时的最优答案，由于每次加的代价都是线性的，它显然形如一条每段斜率都是整数的折线。

在最右边加入一个值 A 时，将 $f(x)$ 加上 $|x - A|$ ，再对 $f(x - 1)$ 取个min。

不上升序列

hihoCoder挑战赛29 D

假设求的是不降序列。不升序列同理。

设 $f(x)$ 表示最后一个数 $\leq x$ 时的最优答案，由于每次加的代价都是线性的，它显然形如一条每段斜率都是整数的折线。

在最右边加入一个值 A 时，将 $f(x)$ 加上 $|x - A|$ ，再对 $f(x - 1)$ 取个min。

两条这种形状的折线相加时，只要把拐点直接取个并就行了。

不上升序列

hihoCoder挑战赛29 D

假设求的是不降序列。不升序列同理。

设 $f(x)$ 表示最后一个数 $\leq x$ 时的最优答案，由于每次加的代价都是线性的，它显然形如一条每段斜率都是整数的折线。

在最右边加入一个值 A 时，将 $f(x)$ 加上 $|x - A|$ ，再对 $f(x - 1)$ 取个min。

两条这种形状的折线相加时，只要把拐点直接取个并就行了。

$f(x) = \min(f(x), f(x - 1))$ 显然就是把折线最右边斜率 ≥ 1 的部分都删去。

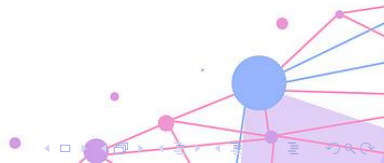




不上升序列

hihoCoder挑战赛29 D

那么具体的算法就是：每次插入两个拐点，再把最大的拐点删掉。用一个大根堆维护拐点的横坐标就行了。



不上升序列

hihoCoder挑战赛29 D

那么具体的算法就是：每次插入两个拐点，再把最大的拐点删掉。用一个大根堆维护拐点的横坐标就行了。

最后用 $f(0) = \sum |A_i|$ 来还原出纵坐标即答案。

Sum of Powers

CS Academy #32 G

题意

考虑所有的正整数可重集 $\{a_1, a_2, a_3 \dots a_k\}$, 满足 $a_1 + a_2 \dots + a_k = n$, 求所有 $a_1^m + a_2^m \dots + a_k^m$ 的和。

范围

$n, m, k \leq 4096$

Sum of Powers

CS Academy #32 G

设 $f_{i,j}$ 表示 i 个物品和为 j 的方案数。转移要么放一个 1，要么把所有物品都 +1。

$$f_{i,j} \rightarrow f_{i+1,j+1}$$

$$f_{i,j} \rightarrow f_{i,j+i}$$

Sum of Powers

CS Academy #32 G

设 $f_{i,j}$ 表示 i 个物品和为 j 的方案数。转移要么放一个 1，要么把所有物品都 +1。

$$f_{i,j} \rightarrow f_{i+1,j+1}$$

$$f_{i,j} \rightarrow f_{i,j+i}$$

考虑 x^m 变成 $(x+1)^m$ ，对于所有 $0 \leq i \leq m$ 维护 $\sum x^i$ ，乘组合数转移，复杂度 $O(n^4)$ 。

Sum of Powers

CS Academy #32 G

• 设 $f_{i,j}$ 表示 i 个物品和为 j 的方案数。转移要么放一个1，要么把所有物品都+1。

$$f_{i,j} \rightarrow f_{i+1,j+1}$$

$$f_{i,j} \rightarrow f_{i,j+i}$$

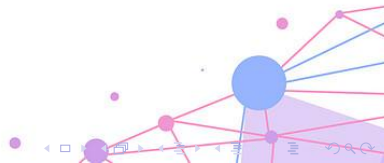
考虑 x^m 变成 $(x+1)^m$ ，对于所有 $0 \leq i \leq m$ 维护 $\sum x^i$ ，乘组合数转移，复杂度 $O(n^4)$ 。

如果维护 x^m ，有 $(x+1)^m = x^m + mx^{m-1}$ ，用第二类斯特林数还原出答案， $O(n^3)$ 。

Sum of Powers

CS Academy #32 G

但是这些都太慢了。



Sum of Powers

CS Academy #32 G

但是这些都太慢了。

考虑算每个东西对答案的贡献。体积为*i*的东西贡献到的次数是

$$\sum_{j=1}^{\infty} [\text{包含至少 } j \text{ 个 } i \text{ 的方案数}]$$

Sum of Powers

CS Academy #32 G

但是这些都太慢了。

考虑算每个东西对答案的贡献。体积为 i 的东西贡献到的次数是

$$\sum_{j=1}^{\infty} [\text{包含至少 } j \text{ 个 } i \text{ 的方案数}]$$

也就是

$$\sum_{j=1}^{\infty} f_{k-j, n-i*j}$$

$O(n^2)$ 。

Perpetual Subtraction

Codeforces #470 (Div.1) E

题意

一开始有一个数 x ，每次把 x 变成 $[0, x]$ 内随机一个整数。
给出一开始数字为 $0 \sim n$ 的概率以及轮数 m ，求 m 轮后剩下数字是 i 的概率。
对998244353取模。

范围

$n \leq 10^5, m \leq 10^{18}$ 。

Perpetual Subtraction

Codeforces #470 (Div.1) E

以 $n = 4$ 为例，转移矩阵

$$M = \begin{pmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

要求的就是 $V * M^m$ 。

Perpetual Subtraction

Codeforces #470 (Div.1) E

矩阵的对角化：若 n 阶方阵 M 有 n 个线性无关的特征向量，则它可以被表示成 SJS^{-1} 的形式。

Perpetual Subtraction

Codeforces #470 (Div.1) E

矩阵的对角化：若 n 阶方阵 M 有 n 个线性无关的特征向量，则它可以被表示成 SJS^{-1} 的形式。

其中 S 是 M 的 n 个特征向量排成 n 列形成的矩阵。 J 是一个对角矩阵，对角元素是对应的特征值。

Perpetual Subtraction

Codeforces #470 (Div.1) E

矩阵的对角化：若 n 阶方阵 M 有 n 个线性无关的特征向量，则它可以被表示成 SJS^{-1} 的形式。

其中 S 是 M 的 n 个特征向量排成 n 列形成的矩阵。 J 是一个对角矩阵，对角元素是对应的特征值。

$$V * M^m = V * (SJS^{-1})^m = VSJ^mS^{-1}$$

Perpetual Subtraction

Codeforces #470 (Div.1) E

矩阵的对角化：若 n 阶方阵 M 有 n 个线性无关的特征向量，则它可以被表示成 SJS^{-1} 的形式。

其中 S 是 M 的 n 个特征向量排成 n 列形成的矩阵。 J 是一个对角矩阵，对角元素是对应的特征值。

$$V * M^m = V * (SJS^{-1})^m = VSJ^mS^{-1}$$

乘对角阵显然是 $O(n)$ 的，如果它的 S 有特殊性质使得 VS 和 VS^{-1} 能快速计算，这题就做完了。

Perpetual Subtraction

Codeforces #470 (Div.1) E

矩阵的对角化：若 n 阶方阵 M 有 n 个线性无关的特征向量，则它可以被表示成 SJS^{-1} 的形式。

其中 S 是 M 的 n 个特征向量排成 n 列形成的矩阵。 J 是一个对角矩阵，对角元素是对应的特征值。

$$V * M^m = V * (SJS^{-1})^m = VSJ^mS^{-1}$$

乘对角阵显然是 $O(n)$ 的，如果它的 S 有特殊性质使得 VS 和 VS^{-1} 能快速计算，这题就做完了。

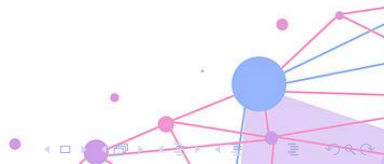
然而特征值和特征向量并不好算（因为一元五次及以上的方程没有求根公式），我们可以把它扔进Mathematica试试。



Perpetual Subtraction

Codeforces #470 (Div.1) E

Mathematica告诉我们：



Perpetual Subtraction

Codeforces #470 (Div.1) E

Mathematica告诉我们：

$$M = SJS^{-1}$$

$$S_{i,j} = \binom{n-i}{n-j}$$

$$J_{i,i} = \frac{1}{n-i+1}$$

$$S_{i,j}^{-1} = \binom{n-i}{n-j} (-1)^{i+j}$$

Perpetual Subtraction

Codeforces #470 (Div.1) E

Mathematica告诉我们：

$$M = SJS^{-1}$$

$$S_{i,j} = \binom{n-i}{n-j}$$

$$J_{i,i} = \frac{1}{n-i+1}$$

$$S_{i,j}^{-1} = \binom{n-i}{n-j} (-1)^{i+j}$$

乘 S 和乘 S^{-1} 都相当于卷一个组合数，FFT即可。

小L的计算题

THUPC 2017 I

题意

给出 $A_1 \sim A_n$, 设

$$F_k = \sum_{i=1}^n A_i^k$$

求 $F_1 \sim F_n$ 。模998244353。

范围

$n, k \leq 200000, A_i \leq 10^9$ 。

小L的计算题

THUPC 2017 I

牛顿恒等式：对于一个首一多项式 $F(x) = \sum_{i=0}^n C_{n-i}x^i$ ($C_0 = 1$)，设 P_i 表示它的 n 个根的 i 次方和，对于所有正整数 d ，有等式

$$\sum_{i=0}^{d-1} C_i P_{d-i} + C_d * d = 0$$

小L的计算题

THUPC 2017 I

牛顿恒等式：对于一个首一多项式 $F(x) = \sum_{i=0}^n C_{n-i}x^i$ ($C_0 = 1$)，设 P_i 表示它的 n 个根的 i 次方和，对于所有正整数 d ，有等式

$$\sum_{i=0}^{d-1} C_i P_{d-i} + C_d * d = 0$$

知道 C 求 P ：多项式求逆， $O(n \log n)$ 。

小L的计算题

THUPC 2017 I

牛顿恒等式：对于一个首一多项式 $F(x) = \sum_{i=0}^n C_{n-i}x^i$ ($C_0 = 1$)，设 P_i 表示它的 n 个根的 i 次方和，对于所有正整数 d ，有等式

$$\sum_{i=0}^{d-1} C_i P_{d-i} + C_d * d = 0$$

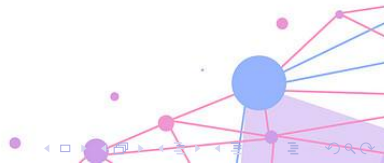
知道 C 求 P ：多项式求逆， $O(n \log n)$ 。

知道 P 求 C ：多项式 \exp ， $O(n \log n)$ 。

小L的计算题

THUPC 2017 I

这题把 $A_1 \sim A_n$ 想象成一个多项式的 n 个根，要求所有根的 k 次方和，即 P 。



小L的计算题

THUPC 2017 I

这题把 $A_1 \sim A_n$ 想象成一个多项式的 n 个根，要求所有根的 k 次方和，即 P 。

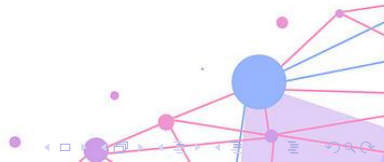
分治FFT展开 $\prod_{i=1}^n (x - A_i)$ 得到 C ，多项式求逆得出 P 即可。

题意

在 n 个点之间给出 m 条带权无向边，权值是 $[0, p = 17)$ ，问有多少种方案选择一些边（不选重边），使得整张图连通，并且边权和为 x ？对于 $x \in [0, p)$ 都要求答案。

范围

$n \leq 17, m \leq 10^5$ 。复杂度为 $O(2^n n^2 p)$ 。

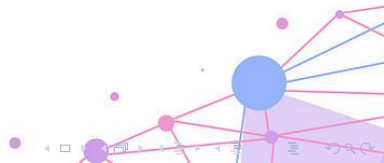




星空

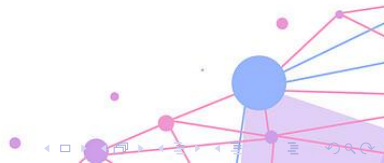
by 杨家齐

首先注意到的是这里是循环卷积，而模数998244353是有17次单位根的，那么肯定先DFT，中间全都用点值算，最后再IDFT回去。



首先注意到的是这里是循环卷积，而模数998244353是有17次单位根的，那么肯定先DFT，中间全都用点值算，最后再IDFT回去。

考虑总数减掉不连通的方案数。



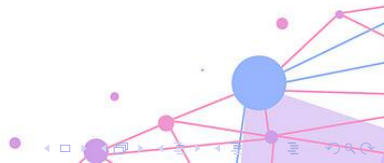


星空

by 杨家齐

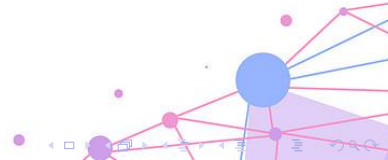
首先注意到的是这里是循环卷积，而模数998244353是有17次单位根的，那么肯定先DFT，中间全都用点值算，最后再IDFT回去。

考虑总数减掉不连通的方案数。随便选的方案数很简单，直接背包就行了。



首先注意到的是这里是循环卷积，而模数998244353是有17次单位根的，那么肯定先DFT，中间全都用点值算，最后再IDFT回去。

考虑总数减掉不连通的方案数。随便选的方案数很简单，直接背包就行了。然后子集枚举1号点所在的连通块，减一下。复杂度 $O(3^n p)$ 。



星空

by 杨家齐

做法一：设 f_S 是在 S 中随便选的方案数， g_S 是 S 连通的方案数（ $g_0 = 0$ ）。

星空

by 杨家齐

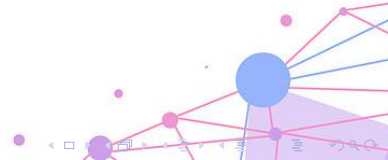
做法一：设 f_S 是在 S 中随便选的方案数， g_S 是 S 连通的方案数（ $g_0 = 0$ ）。

显然有 $f = e^g$ 。 $g = \ln f$ 。这里的乘法定义为子集卷积。

做法一：设 f_S 是在 S 中随便选的方案数， g_S 是 S 连通的方案数（ $g_0 = 0$ ）。

显然有 $f = e^g$ 。 $g = \ln f$ 。这里的乘法定义为子集卷积。

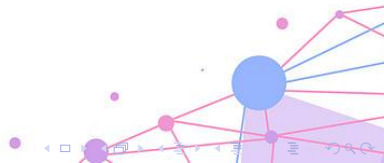
那么考虑子集卷积的过程，先做莫比乌斯变换，然后把占位多项式用 $O(n^2)$ 的做法求个 \ln ，再莫比乌斯反演回去就可以了。复杂度 $O(2^n n^2 p)$ 。



星空

by 杨家齐

做法二：考虑斯特林数，如果枚举一个连通块划分，强制不同连通块之间没有边，同一个连通块不一定要连通，一个 n 个连通块的图会在 k 个连通块的地方算 $S_{n,k}$ 次。



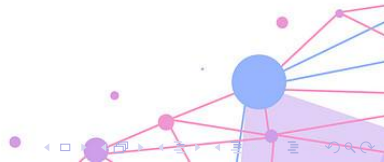
星空

by 杨家齐

做法二：考虑斯特林数，如果枚举一个连通块划分，强制不同连通块之间没有边，同一个连通块不一定要连通，一个 n 个连通块的图会在 k 个连通块的地方算 $S_{n,k}$ 次。

根据

$$\sum_{i=1}^n (-1)^{i-1} (i-1)! S_{n,i} = [n=1]$$

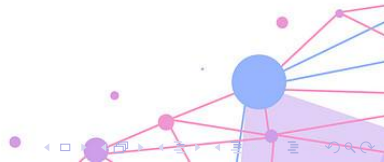


做法二：考虑斯特林数，如果枚举一个连通块划分，强制不同连通块之间没有边，同一个连通块不一定要连通，一个 n 个连通块的图会在 k 个连通块的地方算 $S_{n,k}$ 次。

根据

$$\sum_{i=1}^n (-1)^{i-1} (i-1)! S_{n,i} = [n=1]$$

我们想让一个 i 个连通块的图被枚举到 $(-1)^{i-1} (i-1)!$ 次。那么只要乱序枚举除1号点以外点的连通块就行了。



做法二：考虑斯特林数，如果枚举一个连通块划分，强制不同连通块之间没有边，同一个连通块不一定要连通，一个 n 个连通块的图会在 k 个连通块的地方算 $S_{n,k}$ 次。

根据

$$\sum_{i=1}^n (-1)^{i-1} (i-1)! S_{n,i} = [n=1]$$

我们想让一个 i 个连通块的图被枚举到 $(-1)^{i-1} (i-1)!$ 次。那么只要乱序枚举除1号点以外点的连通块就行了。

换句话说，设 f_S 表示包含1号点的方案数， g_S 为不包含1号点的方案数，且系数带了个 -1 。



做法二：考虑斯特林数，如果枚举一个连通块划分，强制不同连通块之间没有边，同一个连通块不一定要连通，一个 n 个连通块的图会在 k 个连通块的地方算 $S_{n,k}$ 次。

根据

$$\sum_{i=1}^n (-1)^{i-1} (i-1)! S_{n,i} = [n=1]$$

我们想让一个 i 个连通块的图被枚举到 $(-1)^{i-1} (i-1)!$ 次。那么只要乱序枚举除1号点以外点的连通块就行了。

换句话说，设 f_S 表示包含1号点的方案数， g_S 为不包含1号点的方案数，且系数带了个 -1 。那么我们要求的是 $f * (1 + g + g^2 \dots) = \frac{f}{1-g}$ 。乘法定义为子集卷积。

做法二：考虑斯特林数，如果枚举一个连通块划分，强制不同连通块之间没有边，同一个连通块不一定要连通，一个 n 个连通块的图会在 k 个连通块的地方算 $S_{n,k}$ 次。

根据

$$\sum_{i=1}^n (-1)^{i-1} (i-1)! S_{n,i} = [n=1]$$

我们想让一个 i 个连通块的图被枚举到 $(-1)^{i-1} (i-1)!$ 次。那么只要乱序枚举除1号点以外点的连通块就行了。

换句话说，设 f_S 表示包含1号点的方案数， g_S 为不包含1号点的方案数，且系数带了个 -1 。那么我们要求的是 $f * (1 + g + g^2 \dots) = \frac{f}{1-g}$ 。乘法定义为子集卷积。

那么同样先做莫比乌斯变换，把占位多项式求逆，再莫比乌斯反演回去。 $O(2^n n^2 p)$ 。

Chef attic window

Codechef WINDOW

题意

T 组数据, 给出 n, A, B, K, L , 求

$$\sum_{x=1}^n \binom{\lfloor \frac{Ax}{B} \rfloor}{K+1} \binom{x}{L}$$

范围

$T \leq 50, n, A, B \leq 10^{18}, K, L \leq 10$

Chef attic window

Codechef WINDOW

$$\binom{n}{m} = \frac{n^m}{m!}$$

组合数是一个关于 n 的 m 次多项式。

Chef attic window

Codechef WINDOW

$$\binom{n}{m} = \frac{n^m}{m!}$$

组合数是一个关于 n 的 m 次多项式。相当于要求

$$\sum_{x=1}^n \left\lfloor \frac{Ax}{B} \right\rfloor^a x^b$$

Chef attic window

Codechef WINDOW

$$\binom{n}{m} = \frac{n^m}{m!}$$

组合数是一个关于 n 的 m 次多项式。相当于要求

$$\sum_{x=1}^n \left\lfloor \frac{Ax}{B} \right\rfloor^a x^b$$

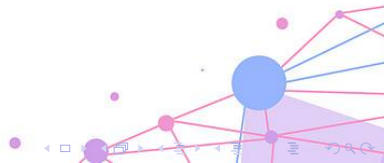
类欧几里德算法，考虑求

$$\sum_{x=1}^n \left\lfloor \frac{Ax+B}{C} \right\rfloor^a x^b$$

Chef attic window

Codechef WINDOW

$$\sum_{x=1}^n \left\lfloor \frac{Ax + B}{C} \right\rfloor^a x^b$$



Chef attic window

Codechef WINDOW

$$\sum_{x=1}^n \left\lfloor \frac{Ax + B}{C} \right\rfloor^a x^b$$

$$\left\lfloor \frac{Ax + B}{C} \right\rfloor = \left\lfloor \frac{A}{C} \right\rfloor + \left\lfloor \frac{(A \bmod C)x + B}{C} \right\rfloor$$

Chef attic window

Codechef WINDOW

$$\sum_{x=1}^n \left\lfloor \frac{Ax + B}{C} \right\rfloor^a x^b$$

$$\left\lfloor \frac{Ax + B}{C} \right\rfloor = \left\lfloor \frac{A}{C} \right\rfloor + \left\lfloor \frac{(A \bmod C)x + B}{C} \right\rfloor$$

A 可以先模掉 C 。然后注意到

$$\lfloor x \rfloor^a = \sum_{y=1}^N [y \leq x] (y^a - (y-1)^a)$$

设 $S_a(n) = \sum_{i=1}^n i^a$, 原式

Chef attic window

Codechef WINDOW

$$\sum_{x=1}^n \left\lfloor \frac{Ax + B}{C} \right\rfloor^a x^b$$

$$\left\lfloor \frac{Ax + B}{C} \right\rfloor = \left\lfloor \frac{A}{C} \right\rfloor + \left\lfloor \frac{(A \bmod C)x + B}{C} \right\rfloor$$

A 可以先模掉 C 。然后注意到

$$[x]^a = \sum_{y=1}^N [y \leq x] (y^a - (y-1)^a)$$

设 $S_a(n) = \sum_{i=1}^n i^a$, 原式

$$= \sum_{x=1}^n x^b \sum_{y=1}^m [Cy \leq Ax + B] (y^a - (y-1)^a)$$

Chef attic window

Codechef WINDOW

$$\sum_{x=1}^n x^b \sum_{y=1}^m [Cy \leq Ax + B](y^a - (y-1)^a)$$

Chef attic window

Codechef WINDOW

$$\sum_{x=1}^n x^b \sum_{y=1}^m [Cy \leq Ax + B](y^a - (y-1)^a)$$

$$= S_b(n) * m^a - \sum_{x=1}^n \sum_{y=1}^m [Cy - B - 1 \geq Ax] x^b (y^a - (y-1)^a)$$

Chef attic window

Codechef WINDOW

$$\begin{aligned}& \sum_{x=1}^n x^b \sum_{y=1}^m [Cy \leq Ax + B](y^a - (y-1)^a) \\&= S_b(n) * m^a - \sum_{x=1}^n \sum_{y=1}^m [Cy - B - 1 \geq Ax] x^b (y^a - (y-1)^a) \\&= S_b(n) * m^a - \sum_{y=1}^m (y^a - (y-1)^a) S_b \left(\left\lfloor \frac{Cy - B - 1}{A} \right\rfloor \right)\end{aligned}$$

Chef attic window

Codechef WINDOW

$$\sum_{x=1}^n x^b \sum_{y=1}^m [Cy \leq Ax + B](y^a - (y-1)^a)$$

$$= S_b(n) * m^a - \sum_{x=1}^n \sum_{y=1}^m [Cy - B - 1 \geq Ax] x^b (y^a - (y-1)^a)$$

$$= S_b(n) * m^a - \sum_{y=1}^m (y^a - (y-1)^a) S_b \left(\left\lfloor \frac{Cy - B - 1}{A} \right\rfloor \right)$$

变成了若干个A, C互换了位置的子问题。

Chef attic window

Codechef WINDOW

$$\begin{aligned}& \sum_{x=1}^n x^b \sum_{y=1}^m [Cy \leq Ax + B](y^a - (y-1)^a) \\&= S_b(n) * m^a - \sum_{x=1}^n \sum_{y=1}^m [Cy - B - 1 \geq Ax] x^b (y^a - (y-1)^a) \\&= S_b(n) * m^a - \sum_{y=1}^m (y^a - (y-1)^a) S_b \left(\left\lfloor \frac{Cy - B - 1}{A} \right\rfloor \right)\end{aligned}$$

变成了若干个A, C互换了位置的子问题。

具体实现的时候在同一层把所有(a, b)的答案都算出来，可以通过预处理部分和降低复杂度。复杂度 $O(K^3 \log n)$ 。

Query on a tree VII

SPOJ QTREE7

题意

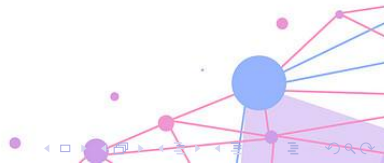
给出一棵 n 个点的无根树，每个点有点权。

点的颜色有黑白两种，初始所有点都为黑色。

需要支持：单点反色，单点修改点权，询问一个点所在的同色连通块中的最大点权。

范围

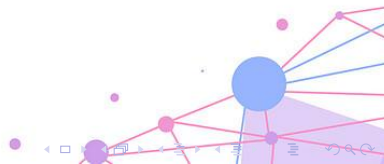
$n, m \leq 10^5$ 。



Query on a tree VII

SPOJ QTREE7

这里讲一种基于LCT的做法。

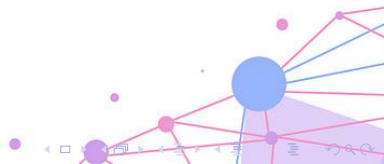


Query on a tree VII

SPOJ QTREE7

这里讲一种基于LCT的做法。

定义两棵树：黑树和白树。黑点在黑树上与它父亲相连，白点在白树上与它父亲相连。



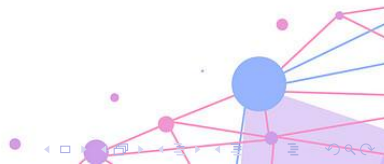
Query on a tree VII

SPOJ QTREE7

这里讲一种基于LCT的做法。

定义两棵树：黑树和白树。黑点在黑树上与它父亲相连，白点在白树上与它父亲相连。

修改一个点的颜色，在两棵树中都只影响它和它父亲的那条边。



Query on a tree VII

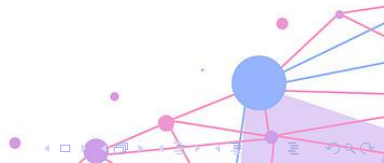
SPOJ QTREE7

这里讲一种基于LCT的做法。

定义两棵树：黑树和白树。黑点在黑树上与它父亲相连，白点在白树上与它父亲相连。

修改一个点的颜色，在两棵树中都只影响它和它父亲的那条边。

任意一棵树中的任意一个连通块除开最浅点，都是一个极大同色连通块。只要找到询问点的最浅同色祖先，询问下子树信息就行了。



Query on a tree VII

SPOJ QTREE7

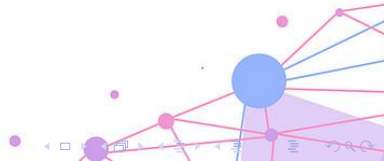
这里讲一种基于LCT的做法。

定义两棵树：黑树和白树。黑点在黑树上与它父亲相连，白点在白树上与它父亲相连。

修改一个点的颜色，在两棵树中都只影响它和它父亲的那条边。

任意一棵树中的任意一个连通块除开最浅点，都是一个极大同色连通块。只要找到询问点的最浅同色祖先，询问下子树信息就行了。

考虑用splay维护两棵树的dfs序，在link/cut的时候把splay也对应的切割拼接一下就可以了。



Query on a tree VII

SPOJ QTREE7

这里讲一种基于LCT的做法。

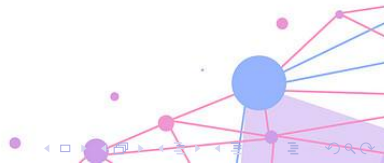
定义两棵树：黑树和白树。黑点在黑树上与它父亲相连，白点在白树上与它父亲相连。

修改一个点的颜色，在两棵树中都只影响它和它父亲的那条边。

任意一棵树中的任意一个连通块除开最浅点，都是一个极大同色连通块。只要找到询问点的最浅同色祖先，询问下子树信息就行了。

考虑用splay维护两棵树的dfs序，在link/cut的时候把splay也对应的切割拼接一下就可以了。

复杂度 $O(n \log n)$ 。



题意

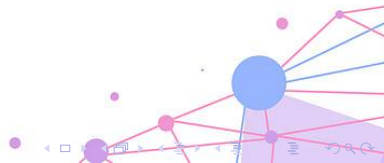
一棵树，树有边权，每个点有黑白两种颜色，要求支持：

1 u : 询问点 u 所在同色连通块中，最远两点的距离，即直径。

2 u, v, c : 将 $u \sim v$ 链上颜色都修改为 c 。

范围

$n, q \leq 100000$ 。

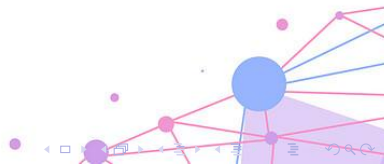




Jabby's shadows

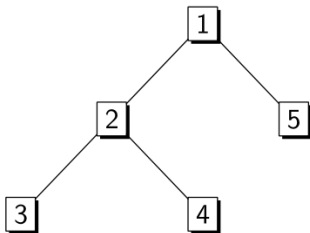
BZOJ3914

帶修改樹上直徑：括号序列。



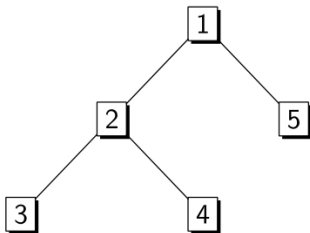
带修改树上直径：括号序列。

- 考虑把树的括号序列弄出来，则两个点之间的距离就是它们之间的括号序列去掉匹配括号之后的长度。



带修改树上直径：括号序列。

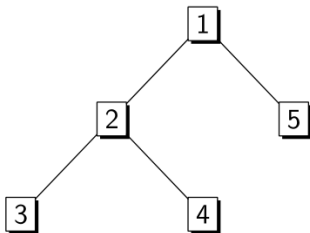
- 考虑把树的括号序列弄出来，则两个点之间的距离就是它们之间的括号序列去掉匹配括号之后的长度。



[1[2[3][4]][5]]

带修改树上直径：括号序列。

- 考虑把树的括号序列弄出来，则两个点之间的距离就是它们之间的括号序列去掉匹配括号之后的长度。



[1[2[3][4]][5]]

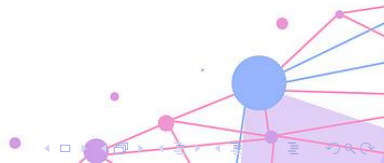
树的直径就是选一个子串，使得去掉匹配括号后长度最大。



Jabby's shadows

BZOJ3914

用 (A, B) 表示 A 个右括号, B 个左括号。



Jabby's shadows

BZOJ3914

用 (A, B) 表示 A 个右括号, B 个左括号。

两个串合并时在中间会产生 $\min(B_1, A_2)$ 对匹配括号。也就是

$$(A_1, B_1) + (A_2, B_2) = (A_1 + \max(A_2 - B_1, 0), B_2 + \max(B_1 - A_2, 0))$$

用 (A, B) 表示 A 个右括号， B 个左括号。

两个串合并时在中间会产生 $\min(B_1, A_2)$ 对匹配括号。也就是

$$(A_1, B_1) + (A_2, B_2) = (A_1 + \max(A_2 - B_1, 0), B_2 + \max(B_1 - A_2, 0))$$

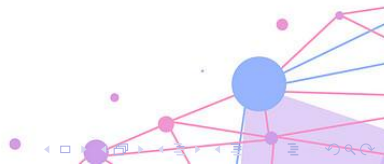
$$A' + B' = \max(A_1 + B_1 + B_2 - A_2, A_1 - B_1 + A_2 + B_2)$$



Jabby's shadows

BZOJ3914

$$A' + B' = \max(A_1 + B_1 + B_2 - A_2, A_1 - B_1 + A_2 + B_2)$$



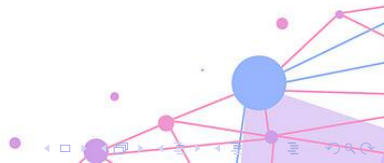


Jabby's shadows

BZOJ3914

$$A' + B' = \max(A_1 + B_1 + B_2 - A_2, A_1 - B_1 + A_2 + B_2)$$

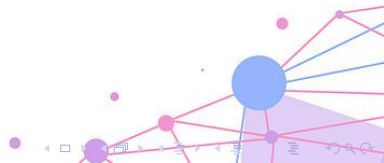
维护最大 $A + B$ ，后缀最大 $A + B$ ，后缀最大 $A - B$ ，前缀最大 $B - A$ ，前缀最大 $A + B$ ，以及整个区间的 A, B ，这些信息就都可以互相计算了。



$$A' + B' = \max(A_1 + B_1 + B_2 - A_2, A_1 - B_1 + A_2 + B_2)$$

维护最大 $A + B$ ，后缀最大 $A + B$ ，后缀最大 $A - B$ ，前缀最大 $B - A$ ，前缀最大 $A + B$ ，以及整个区间的 A, B ，这些信息就都可以互相计算了。

如果树的形态不变，用线段树维护括号序列即可。

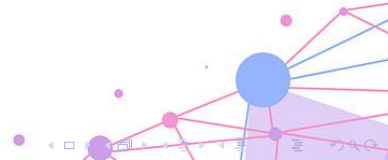


$$A' + B' = \max(A_1 + B_1 + B_2 - A_2, A_1 - B_1 + A_2 + B_2)$$

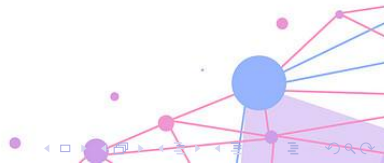
维护最大 $A + B$ ，后缀最大 $A + B$ ，后缀最大 $A - B$ ，前缀最大 $B - A$ ，前缀最大 $A + B$ ，以及整个区间的 A, B ，这些信息就都可以互相计算了。

如果树的形态不变，用线段树维护括号序列即可。

如果是单点修改颜色，套用QTREE7的做法，用splay维护括号序列即可。

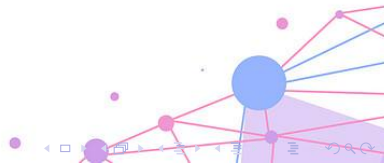


链修改颜色有一种Memphis学长原创的算法，姑且称之为Link-Cut-Memphis (LCM)。



链修改颜色有一种Memphis学长原创的算法，姑且称之为Link-Cut-Memphis (LCM)。

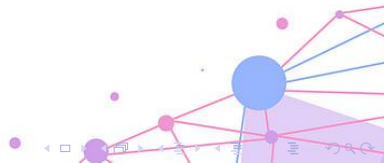
仍然维护黑树和白树两棵树，黑点仍然在黑树上与它父亲相连，白点仍然在白树上与它父亲相连。



链修改颜色有一种Memphis学长原创的算法，姑且称之为Link-Cut-Memphis (LCM)。

仍然维护黑树和白树两棵树，黑点仍然在黑树上与它父亲相连，白点仍然在白树上与它父亲相连。

如果白树中一个白点 u 与它的白儿子 v 在LCT中以重边相连，则在黑树中连接 (u, v) 。黑树同理。

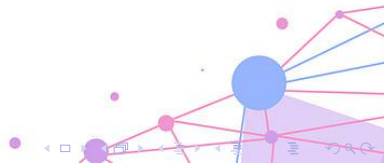


链修改颜色有一种Memphis学长原创的算法，姑且称之为Link-Cut-Memphis (LCM)。

仍然维护黑树和白树两棵树，黑点仍然在黑树上与它父亲相连，白点仍然在白树上与它父亲相连。

如果白树中一个白点 u 与它的白儿子 v 在LCT中以重边相连，则在黑树中连接 (u, v) 。黑树同理。

因此黑树的结构实际上是：



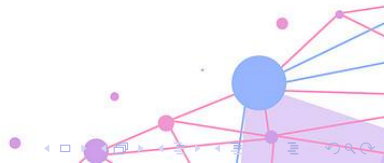
链修改颜色有一种Memphis学长原创的算法，姑且称之为Link-Cut-Memphis (LCM)。

仍然维护黑树和白树两棵树，黑点仍然在黑树上与它父亲相连，白点仍然在白树上与它父亲相连。

如果白树中一个白点 u 与它的白儿子 v 在LCT中以重边相连，则在黑树中连接 (u, v) 。黑树同理。

因此黑树的结构实际上是：

- 1、若干条白色的链（白树中的重链，这里称之为链树）。



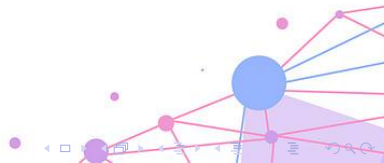
链修改颜色有一种Memphis学长原创的算法，姑且称之为Link-Cut-Memphis (LCM)。

仍然维护黑树和白树两棵树，黑点仍然在黑树上与它父亲相连，白点仍然在白树上与它父亲相连。

如果白树中一个白点 u 与它的白儿子 v 在LCT中以重边相连，则在黑树中连接 (u, v) 。黑树同理。

因此黑树的结构实际上是：

- 1、若干条白色的链（白树中的重链，这里称之为链树）。
- 2、黑色连通块（称之为块树）。



链修改颜色有一种Memphis学长原创的算法，姑且称之为Link-Cut-Memphis (LCM)。

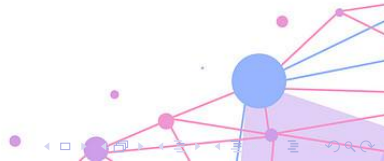
仍然维护黑树和白树两棵树，黑点仍然在黑树上与它父亲相连，白点仍然在白树上与它父亲相连。

如果白树中一个白点 u 与它的白儿子 v 在LCT中以重边相连，则在黑树中连接 (u, v) 。黑树同理。

因此黑树的结构实际上是：

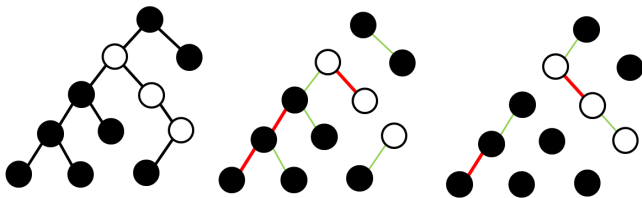
- 1、若干条白色的链（白树中的重链，这里称之为链树）。
- 2、黑色连通块（称之为块树）。

可以发现链树下面会挂若干个块树，但块树下面一定不会有链树。



Jabby's shadows

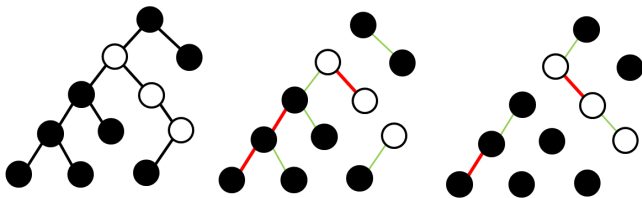
BZOJ3914



上面是某时刻一种可能的原树/黑树/白树，红边表示重边。

Jabby's shadows

BZOJ3914

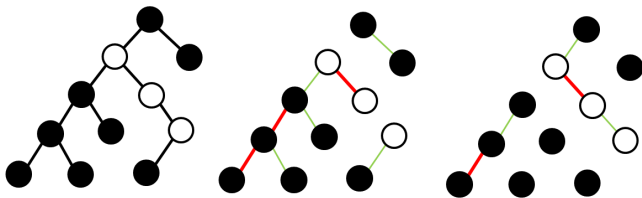


上面是某时刻一种可能的原树/黑树/白树，红边表示重边。

每次链修改颜色时，把每段颜色相同的链一起做，做完后所有段颜色都会变得相同。相当于Access操作。因此颜色段数是均摊 $O(n \log n)$ 的。

Jabby's shadows

BZOJ3914



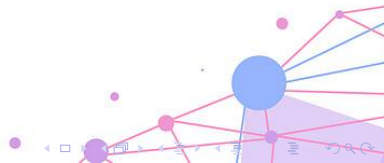
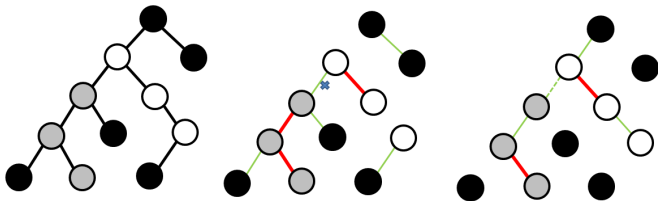
上面是某时刻一种可能的原树/黑树/白树，红边表示重边。

每次链修改颜色时，把每段颜色相同的链一起做，做完后所有段颜色都会变得相同。相当于Access操作。因此颜色段数是均摊 $O(n \log n)$ 的。

定义 $Expose(x)$ 是 $Access(x)$ 的变种，它的效果是把 x 到同色连通块最浅点的链打通成重链，并在修改同色重儿子时，在另一棵树中对应完成link/cut。

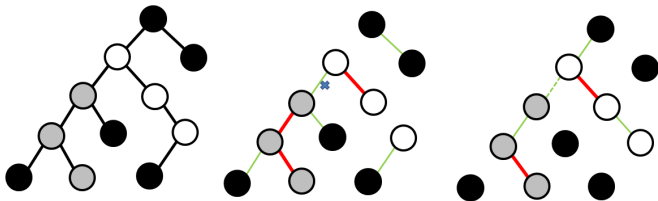
Jabby's shadows

BZOJ3914



Jabby's shadows

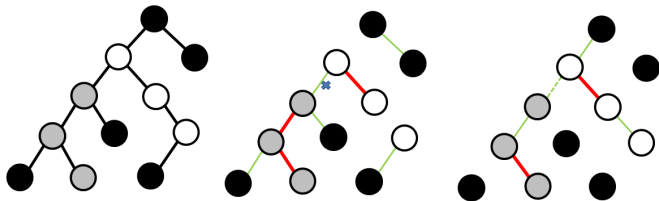
BZOJ3914



考虑我们要把那3个灰色点从黑色改成白色，已经对最深那个点进行过 $Expose()$ 。

Jabby's shadows

BZOJ3914

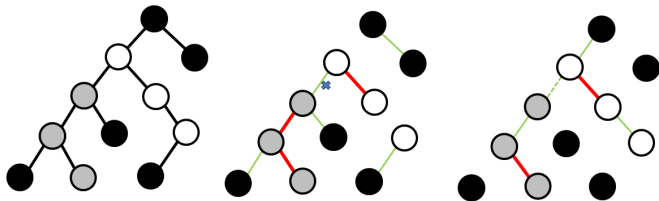


考虑我们要把那3个灰色点从黑色改成白色，已经对最深那个点进行过`Expose()`。

可以发现改色后黑树中块树的这部分会变成链树，而白树中链树的这部分会变成块树！

Jabby's shadows

BZOJ3914



考虑我们要把那3个灰色点从黑色改成白色，已经对最深那个点进行过 $Expose()$ 。

可以发现改色后黑树中块树的这部分会变成链树，而白树中链树的这部分会变成块树！

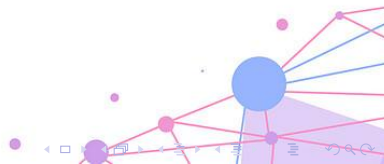
只要在这两棵树中分别处理下最顶上那条边就可以了（黑树中Cut，白树中Link）！



Jabby's shadows

BZOJ3914

于是就做完了，是不是很兹辞呢？

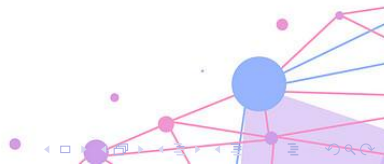




Jabby's shadows

BZOJ3914

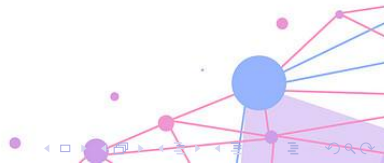
于是就做完了，是不是很兹辞呢？
复杂度的上界是 $O(n \log^2 n)$ 。



于是就做完了，是不是很兹辞呢？

复杂度的上界是 $O(n \log^2 n)$ 。

也就是说所有涉及树上同色连通块问题的题，只要能用dfs序维护，就能在外面套个链修改颜色。





Making Change

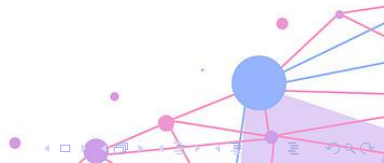
Codechef CHANGE

题意

有 n 种硬币，第 i 种面值是 D_i ，每种都有无限个。求拼出 C 的方案数。两种方案不同仅当某种硬币使用的数量不一样。

范围

$n \leq 50, D_i \leq 500, C \leq 10^{100}$ 。

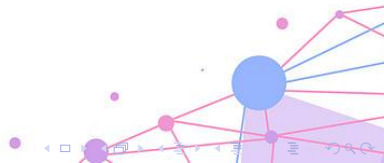




Making Change

Codechef CHANGE

部分分式分解是将有理函数分解成许多较低次数的有理函数和的形式，来降低分子或分母多项式的次数。



Making Change

Codechef CHANGE

部分分式分解是将有理函数分解成许多较低次数的有理函数和的形式，来降低分子或分母多项式的次数。举个例子：

$$\frac{1}{\prod_{i=1}^n (x - A_i)} = \sum_{i=1}^n \frac{B_i}{x - A_i}$$

其中 A_i 互不相同， B_i 是常数。

Making Change

Codechef CHANGE

部分分式分解是将有理函数分解成许多较低次数的有理函数和的形式，来降低分子或分母多项式的次数。举个例子：

$$\frac{1}{\prod_{i=1}^n (x - A_i)} = \sum_{i=1}^n \frac{B_i}{x - A_i}$$

其中 A_i 互不相同， B_i 是常数。

如何证明一定存在合法的 B_i ？如何构造？

Making Change

Codechef CHANGE

部分分式分解是将有理函数分解成许多较低次数的有理函数和的形式，来降低分子或分母多项式的次数。举个例子：

$$\frac{1}{\prod_{i=1}^n (x - A_i)} = \sum_{i=1}^n \frac{B_i}{x - A_i}$$

其中 A_i 互不相同， B_i 是常数。

如何证明一定存在合法的 B_i ？如何构造？考虑两边同乘 $(x - A_i)$ ，然后令 $x \rightarrow A_i$ 。

Making Change

Codechef CHANGE

部分分式分解是将有理函数分解成许多较低次数的有理函数和的形式，来降低分子或分母多项式的次数。举个例子：

$$\frac{1}{\prod_{i=1}^n (x - A_i)} = \sum_{i=1}^n \frac{B_i}{x - A_i}$$

其中 A_i 互不相同， B_i 是常数。

如何证明一定存在合法的 B_i ？如何构造？考虑两边同乘 $(x - A_i)$ ，然后令 $x \rightarrow A_i$ 。根据洛必达法则，右边只有 B_i 这一项，左边分子分母分别求导再取 A_i 代入，得到

$$B_i = \frac{1}{\prod_{j \neq i} (A_i - A_j)}$$

Making Change

Codechef CHANGE

部分分式分解是将有理函数分解成许多较低次数的有理函数和的形式，来降低分子或分母多项式的次数。举个例子：

$$\frac{1}{\prod_{i=1}^n (x - A_i)} = \sum_{i=1}^n \frac{B_i}{x - A_i}$$

其中 A_i 互不相同， B_i 是常数。

如何证明一定存在合法的 B_i ？如何构造？考虑两边同乘 $(x - A_i)$ ，然后令 $x \rightarrow A_i$ 。根据洛必达法则，右边只有 B_i 这一项，左边分子分母分别求导再取 A_i 代入，得到

$$B_i = \frac{1}{\prod_{j \neq i} (A_i - A_j)}$$

容易验证这 B_i 是满足条件的。

Making Change

Codechef CHANGE

部分分式分解是将有理函数分解成许多较低次数的有理函数和的形式，来降低分子或分母多项式的次数。举个例子：

$$\frac{1}{\prod_{i=1}^n (x - A_i)} = \sum_{i=1}^n \frac{B_i}{x - A_i}$$

其中 A_i 互不相同， B_i 是常数。

如何证明一定存在合法的 B_i ？如何构造？考虑两边同乘 $(x - A_i)$ ，然后令 $x \rightarrow A_i$ 。根据洛必达法则，右边只有 B_i 这一项，左边分子分母分别求导再取 A_i 代入，得到

$$B_i = \frac{1}{\prod_{j \neq i} (A_i - A_j)}$$

容易验证这 B_i 是满足条件的。一般的，如果原式分子次数小于分母次数，拆出来的每个式子分子次数也小于分母次数。

Making Change

Codechef CHANGE

本题要求的就是

$$\frac{1}{\prod_{i=1}^n (1 - x^{D_i})}$$

在 x^C 前的系数。

Making Change

Codechef CHANGE

本题要求的就是

$$\frac{1}{\prod_{i=1}^n (1 - x^{D_i})}$$

在 x^C 前的系数。这等于

$$\frac{1}{(1 - x)^n \prod_{i=1}^n \sum_{j < D_i} x^j}$$

Making Change

Codechef CHANGE

本题要求的就是

$$\frac{1}{\prod_{i=1}^n (1 - x^{D_i})}$$

在 x^C 前的系数。这等于

$$\frac{1}{(1-x)^n \prod_{i=1}^n \sum_{j < D_i} x^j}$$

而

$$\sum_{j < D_i} x^j = \prod_{j=1}^{D_i-1} (x - \omega_{D_i}^j)$$

Making Change

Codechef CHANGE

本题要求的就是

$$\frac{1}{\prod_{i=1}^n (1 - x^{D_i})}$$

在 x^C 前的系数。这等于

$$\frac{1}{(1 - x)^n \prod_{i=1}^n \sum_{j < D_i} x^j}$$

而

$$\sum_{j < D_i} x^j = \prod_{j=1}^{D_i-1} (x - \omega_{D_i}^j)$$

题目保证了 D_i 两两互质，那么 $\omega_{D_i}^j$ 两两不同，那么 $\sum_{j < D_i} x^j$ 两两互质。

Making Change

Codechef CHANGE

考虑原式部分分式分解后的结果，一定能表示成

$$\frac{A(x)}{(1-x)^n} + \sum_{i=1}^n \frac{B_i(x)}{\sum_{j < D_i} x^j}$$

Making Change

Codechef CHANGE

考虑原式部分分式分解后的结果，一定能表示成

$$\frac{A(x)}{(1-x)^n} + \sum_{i=1}^n \frac{B_i(x)}{\sum_{j < D_i} x^j}$$

对于 $A(x)$ 和 $B_i(x)$ 分别求贡献，加起来即可。

Making Change

Codechef CHANGE

如何求解 $B_i(x)$?

$$\frac{1}{(1-x)^n \prod_{i=1}^n \sum_{j < D_i} x^j} = \frac{A(x)}{(1-x)^n} + \sum_{i=1}^n \frac{B_i(x)}{\sum_{j < D_i} x^j}$$

Making Change

Codechef CHANGE

如何求解 $B_i(x)$?

$$\frac{1}{(1-x)^n \prod_{i=1}^n \sum_{j < D_i} x^j} = \frac{A(x)}{(1-x)^n} + \sum_{i=1}^n \frac{B_i(x)}{\sum_{j < D_i} x^j}$$

两边同乘 $\sum_{j < D_i} x^j$, 然后令 $x \rightarrow \omega_{D_i}^1, \omega_{D_i}^2, \dots, \omega_{D_i}^{D_i-1}$, 使用洛必达法则,

$$B_i(\omega_{D_i}^t) = \frac{1}{(1 - \omega_{D_i}^t) \prod_{j \neq i} (1 - (\omega_{D_i}^t)^{D_j})}$$

Making Change

Codechef CHANGE

如何求解 $B_i(x)$?

$$\frac{1}{(1-x)^n \prod_{i=1}^n \sum_{j < D_i} x^j} = \frac{A(x)}{(1-x)^n} + \sum_{i=1}^n \frac{B_i(x)}{\sum_{j < D_i} x^j}$$

两边同乘 $\sum_{j < D_i} x^j$, 然后令 $x \rightarrow \omega_{D_i}^1, \omega_{D_i}^2, \dots, \omega_{D_i}^{D_i-1}$, 使用洛必达法则,

$$B_i(\omega_{D_i}^t) = \frac{1}{(1 - \omega_{D_i}^t) \prod_{j \neq i} (1 - (\omega_{D_i}^t)^{D_j})}$$

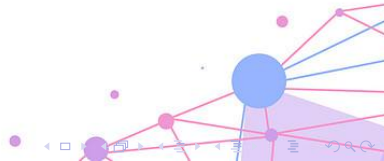
$B_i(x)$ 是一个 $D_i - 2$ 次的多项式, 已知 $D_i - 1$ 个单位根处的点值, IDFT一遍就能得到 $B_i(x)$ 的系数。然而这样需要用复数运算, 误差较大, 考虑别的方法。

Making Change

Codechef CHANGE

有引理：

$$\frac{1}{1 - \omega_d^t} = -\frac{1}{d} \sum_{i=0}^{d-1} (j+1) \omega_d^{tj}$$



Making Change

Codechef CHANGE

有引理：

$$\frac{1}{1 - \omega_d^t} = -\frac{1}{d} \sum_{i=0}^{d-1} (j+1) \omega_d^{tj}$$

证明：设

$$S(x) = \sum_{i=0}^{d-1} (i+1)x^i$$

Making Change

Codechef CHANGE

有引理：

$$\frac{1}{1 - \omega_d^t} = -\frac{1}{d} \sum_{i=0}^{d-1} (j+1) \omega_d^{tj}$$

证明：设

$$S(x) = \sum_{i=0}^{d-1} (i+1)x^i$$

$$(1-x)S(x) = \sum_{i=0}^{d-1} x^i - dx^d$$

Making Change

Codechef CHANGE

有引理：

$$\frac{1}{1 - \omega_d^t} = -\frac{1}{d} \sum_{i=0}^{d-1} (j+1) \omega_d^{tj}$$

证明：设

$$S(x) = \sum_{i=0}^{d-1} (i+1)x^i$$

$$(1-x)S(x) = \sum_{i=0}^{d-1} x^i - dx^d$$

当 x 取 ω_d^t 时， $\sum_{i=0}^{d-1} x^i = 0$ ， $x^d = 1$ ，故原式成立。

Making Change

Codechef CHANGE

有引理：

$$\frac{1}{1 - \omega_d^t} = -\frac{1}{d} \sum_{i=0}^{d-1} (j+1) \omega_d^{tj}$$

证明：设

$$S(x) = \sum_{i=0}^{d-1} (i+1)x^i$$

$$(1-x)S(x) = \sum_{i=0}^{d-1} x^i - dx^d$$

当 x 取 ω_d^t 时， $\sum_{i=0}^{d-1} x^i = 0$ ， $x^d = 1$ ，故原式成立。

那么 $B_i(x)$ 变成了 $n+1$ 个多项式的卷积。直接做是 $O(nd^2)$ 的，但这个卷积有特殊性，可以优化它。

Making Change

Codechef CHANGE

把 $(-\frac{1}{d})^n$ 提到外面，剩下部分每次要做的卷积形如

$$\left(\sum_{i=0}^{d-1} A_i \omega_d^i \right) \left(\sum_{i=0}^{d-1} (i+1) \omega_d^{iq} \right) = \sum_{i=0}^{d-1} C_i \omega_d^i$$

Making Change

Codechef CHANGE

把 $(-\frac{1}{d})^n$ 提到外面，剩下部分每次要做的卷积形如

$$\left(\sum_{i=0}^{d-1} A_i \omega_d^i \right) \left(\sum_{i=0}^{d-1} (i+1) \omega_d^{iq} \right) = \sum_{i=0}^{d-1} C_i \omega_d^i$$

$$C_i = \sum_{k=0}^{d-1} (k+1) A_{i-k*q} = \sum_{k=1}^{d-1} k A_{i-k*q} + \sum A_k$$

Making Change

Codechef CHANGE

把 $(-\frac{1}{d})^n$ 提到外面，剩下部分每次要做的卷积形如

$$\left(\sum_{i=0}^{d-1} A_i \omega_d^i \right) \left(\sum_{i=0}^{d-1} (i+1) \omega_d^{iq} \right) = \sum_{i=0}^{d-1} C_i \omega_d^i$$

$$C_i = \sum_{k=0}^{d-1} (k+1) A_{i-k*q} = \sum_{k=1}^{d-1} k A_{i-k*q} + \sum A_k$$

$$= \sum_{k=0}^{d-2} (k+1) A_{i-q-k*q} + \sum A_k = C_{i-q} - d * A_i + \sum A_k$$

(以上所有的下标运算都在模 d 域下进行)。

Making Change

Codechef CHANGE

把 $(-\frac{1}{d})^n$ 提到外面，剩下部分每次要做的卷积形如

$$\left(\sum_{i=0}^{d-1} A_i \omega_d^i \right) \left(\sum_{i=0}^{d-1} (i+1) \omega_d^{iq} \right) = \sum_{i=0}^{d-1} C_i \omega_d^i$$

$$C_i = \sum_{k=0}^{d-1} (k+1) A_{i-k*q} = \sum_{k=1}^{d-1} k A_{i-k*q} + \sum A_k$$

$$= \sum_{k=0}^{d-2} (k+1) A_{i-q-k*q} + \sum A_k = C_{i-q} - d * A_i + \sum A_k$$

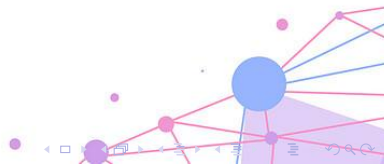
(以上所有的下标运算都在模 d 域下进行)。

那么先 $O(d)$ 求出 C_0 ，然后每次把下标 $+q$ 即可。一次卷积变成了 $O(d)$ ，求 $B_i(\omega_{D_i})$ 关于 ω_{D_i} 的系数的复杂度为 $O(nd)$ 。

Making Change

Codechef CHANGE

而观察推导过程可以发现 ω_{D_i} 换成其它 $\omega_{D_i}^t (1 \leq t < D_i)$ 仍然成立。



Making Change

Codechef CHANGE

而观察推导过程可以发现 ω_{D_i} 换成其它 $\omega_{D_i}^t (1 \leq t < D_i)$ 仍然成立。也就是

$$B_i(\omega_{D_i}^t) = \sum_{j=0}^{D_i-1} b_j(\omega_{D_i}^t)^j$$

Making Change

Codechef CHANGE

而观察推导过程可以发现 ω_{D_i} 换成其它 $\omega_{D_i}^t (1 \leq t < D_i)$ 仍然成立。也就是

$$B_i(\omega_{D_i}^t) = \sum_{j=0}^{D_i-1} b_j(\omega_{D_i}^t)^j$$

然而 $B_i(x)$ 是个 $D_i - 2$ 次多项式，这里的次数却是 $D_i - 1$ ，没法推广到一般的 x 。

Making Change

Codechef CHANGE

而观察推导过程可以发现 ω_{D_i} 换成其它 $\omega_{D_i}^t (1 \leq t < D_i)$ 仍然成立。也就是

$$B_i(\omega_{D_i}^t) = \sum_{j=0}^{D_i-1} b_j(\omega_{D_i}^t)^j$$

然而 $B_i(x)$ 是个 $D_i - 2$ 次多项式，这里的次数却是 $D_i - 1$ ，没法推广到一般的 x 。注意到 $\sum_{i=0}^{d-1} (\omega_d^t)^i = 0$ ，那么有

Making Change

Codechef CHANGE

而观察推导过程可以发现 ω_{D_i} 换成其它 $\omega_{D_i}^t (1 \leq t < D_i)$ 仍然成立。也就是

$$B_i(\omega_{D_i}^t) = \sum_{j=0}^{D_i-1} b_j(\omega_{D_i}^t)^j$$

然而 $B_i(x)$ 是个 $D_i - 2$ 次多项式，这里的次数却是 $D_i - 1$ ，没法推广到一般的 x 。注意到 $\sum_{i=0}^{d-1} (\omega_d^t)^i = 0$ ，那么有

$$(\omega_d^t)^{d-1} = - \sum_{i=0}^{d-2} (\omega_d^t)^i$$

Making Change

Codechef CHANGE

而观察推导过程可以发现 ω_{D_i} 换成其它 $\omega_{D_i}^t (1 \leq t < D_i)$ 仍然成立。也就是

$$B_i(\omega_{D_i}^t) = \sum_{j=0}^{D_i-1} b_j (\omega_{D_i}^t)^j$$

然而 $B_i(x)$ 是个 $D_i - 2$ 次多项式，这里的次数却是 $D_i - 1$ ，没法推广到一般的 x 。注意到 $\sum_{i=0}^{d-1} (\omega_d^t)^i = 0$ ，那么有

$$(\omega_d^t)^{d-1} = - \sum_{i=0}^{d-2} (\omega_d^t)^i$$

$$B_i(\omega_{D_i}^t) = \sum_{j=0}^{D_i-2} (b_j - b_{D_i-1}) (\omega_{D_i}^t)^j$$

Making Change

Codechef CHANGE

$D_i - 1$ 个点值可以唯一确定一个 $D_i - 2$ 次多项式。于是可以断言

$$B_i(x) = \sum_{j=0}^{D_i-2} (b_j - b_{D_i-1})x^j$$

Making Change

Codechef CHANGE

$D_i - 1$ 个点值可以唯一确定一个 $D_i - 2$ 次多项式。于是可以断言

$$B_i(x) = \sum_{j=0}^{D_i-2} (b_j - b_{D_i-1})x^j$$

放到最初的式子里看它对答案的贡献：

$$\frac{B_i(x)}{\sum_{j < D_i} x^j} = \frac{B_i(x)(1-x)}{1-x^{D_i}}$$

Making Change

Codechef CHANGE

$D_i - 1$ 个点值可以唯一确定一个 $D_i - 2$ 次多项式。于是可以断言

$$B_i(x) = \sum_{j=0}^{D_i-2} (b_j - b_{D_i-1})x^j$$

放到最初的式子里看它对答案的贡献：

$$\frac{B_i(x)}{\sum_{j < D_i} x^j} = \frac{B_i(x)(1-x)}{1-x^{D_i}}$$

分子次数为 $D_i - 1$ ，设 $w = C \bmod D_i$ ，则贡献显然是 $b_w - b_{w-1}$ 。

Making Change

Codechef CHANGE

$D_i - 1$ 个点值可以唯一确定一个 $D_i - 2$ 次多项式。于是可以断言

$$B_i(x) = \sum_{j=0}^{D_i-2} (b_j - b_{D_i-1})x^j$$

放到最初的式子里看它对答案的贡献：

$$\frac{B_i(x)}{\sum_{j < D_i} x^j} = \frac{B_i(x)(1-x)}{1-x^{D_i}}$$

分子次数为 $D_i - 1$ ，设 $w = C \bmod D_i$ ，则贡献显然是 $b_w - b_{w-1}$ 。

总的复杂度为 $O(n^2 d)$ 。

Making Change

Codechef CHANGE

还有一项是 $\frac{A(x)}{(1-x)^n}$, 其中 $A(x)$ 是一个 $n-1$ 次系数为常数的多项式。而

Making Change

Codechef CHANGE

还有一项是 $\frac{A(x)}{(1-x)^n}$, 其中 $A(x)$ 是一个 $n-1$ 次系数为常数的多项式。而

$$\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{n+i-1}{n-1} x^i$$

Making Change

Codechef CHANGE

还有一项是 $\frac{A(x)}{(1-x)^n}$, 其中 $A(x)$ 是一个 $n-1$ 次系数为常数的多项式。而

$$\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{n+i-1}{n-1} x^i$$

因此 x^i 前的系数是一个关于 i 的 $n-1$ 次多项式。

Making Change

Codechef CHANGE

还有一项是 $\frac{A(x)}{(1-x)^n}$, 其中 $A(x)$ 是一个 $n-1$ 次系数为常数的多项式。而

$$\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{n+i-1}{n-1} x^i$$

因此 x^i 前的系数是一个关于 i 的 $n-1$ 次多项式。

通过简单的 $O(n^2)$ 背包可以求出原问题在 $C = 1..n$ 时的答案。
 $B_i(x)$ 的贡献也是好求的, 相减得到 $C = 1..n$ 时 x^C 前的系数。拉格朗日插值即可。

Making Change

Codechef CHANGE

还有一项是 $\frac{A(x)}{(1-x)^n}$, 其中 $A(x)$ 是一个 $n-1$ 次系数为常数的多项式。而

$$\frac{1}{(1-x)^n} = \sum_{i \geq 0} \binom{n+i-1}{n-1} x^i$$

因此 x^i 前的系数是一个关于 i 的 $n-1$ 次多项式。

通过简单的 $O(n^2)$ 背包可以求出原问题在 $C = 1..n$ 时的答案。
 $B_i(x)$ 的贡献也是好求的, 相减得到 $C = 1..n$ 时 x^C 前的系数。拉格朗日插值即可。

总的复杂度为 $O(n^2 d)$ 。

FibonacciStringSum

SRM 701 900pts

题意

定义斐波那契串为：没有两个连续1的01串。

定义一个斐波那契串的权值为 $x^a y^b$ ，其中 x 是0的个数， y 是1的个数， a, b 是给出的。

求长为 n 的所有斐波那契串的权值和。

范围

$n \leq 10^9, a, b \leq 25$ 。

FibonacciStringSum

SRM 701 900pts

由于 n 是固定的, 即 $y = n - x$, 二项式定理展开来:

$$x^a(n-x)^b = \sum_{i=0}^b \binom{b}{i} (-1)^i n^{b-i} x^{i+a}$$

FibonacciStringSum

SRM 701 900pts

由于 n 是固定的，即 $y = n - x$ ，二项式定理展开来：

$$x^a(n-x)^b = \sum_{i=0}^b \binom{b}{i} (-1)^i n^{b-i} x^{i+a}$$

也就是只要对于 $a = 0..50$ ，求所有斐波那契串的 x^a 的和就行了。

FibonacciStringSum

SRM 701 900pts

由于 n 是固定的，即 $y = n - x$ ，二项式定理展开来：

$$x^a(n-x)^b = \sum_{i=0}^b \binom{b}{i} (-1)^i n^{b-i} x^{i+a}$$

也就是只要对于 $a = 0..50$ ，求所有斐波那契串的 x^a 的和就行了。

考虑用个dp算这个，设 $f_{i,j}$ 表示长为 i 的所有斐波那契串的 x^j 之和，如果 x 要加一的话就乘上组合数转移下。

FibonacciStringSum

SRM 701 900pts

由于 n 是固定的，即 $y = n - x$ ，二项式定理展开来：

$$x^a(n-x)^b = \sum_{i=0}^b \binom{b}{i} (-1)^i n^{b-i} x^{i+a}$$

也就是只要对于 $a = 0..50$ ，求所有斐波那契串的 x^a 的和就行了。

考虑用个dp算这个，设 $f_{i,j}$ 表示长为 i 的所有斐波那契串的 x^j 之和，如果 x 要加一的话就乘上组合数转移下。

矩乘即可。 $O(102^3 \log n)$ 。



祝各位选手省选顺利!

