目录

数学

欧拉函数/欧拉降幂

2018 年 7 月 29 日
17:23

一、参考链接

https://blog.csdn.net/yxuanwkeith/article/details/52387873 欧拉函数的性质

https://blog.csdn.net/ez_yww/article/details/76176970 扩展欧拉定理

https://blog.csdn.net/popoqqq/article/details/43951401 BZOJ 3884 欧拉降幂例子

https://www.nowcoder.com/acm/contest/142/A 欧拉降幂例子 2

二、题目：

1、2018 牛客多校第四场 A

题意：每一步在所有的 1 后插 0，所有的 2 后插 1，之后删除第一个数。给定初始数列，问多少轮后所有数消失。

做法：考虑每个数的影响。设到某个数时（即前面的数全被删除掉），已经删了 ans 轮。

分类讨论。

若当前数为 0： ans=ans+1

若当前数为 1： ans=ans+ans+2

若当前数为 2： ans=3*2^(ans+1)-3 //仅此为打表得出 前两个易理解

前两种情况的很显然。主要需要处理一下第三种情况。因为欧拉定理，每次系数模的是 phi(p)，而在递归过程中 p 在不断变化，对此有两种做法：（主要看第 2 种，比较经典！）

1、维护模所有从 p 到 1（通过不断执行 phi(p）形成的链）的意义下，ans 的值（实际上也只有 28 个）

2、倒着做，就可以通过递归，先处理出之前的答案在模**当前需要的**模数下的值。

dfs 函数形式为 solve(int lo,int M) 其中 lo 为当前执行到的字符串的位置（以此来确定执行哪一种操作），M 为当前需要模的数。对于当前数为 2 的，使用该式子：

return (mul(3,powMM(2,solve(lo-1,phi(M))+1,M),M)-3+M)%M; 另两个比较简单 不再赘述。

答案即为 solve(len-1,MOD)

类欧几里德

2018 年 8 月 20 日
15:38
参考链接见备忘录 ACM 项中

1、BZOJ 3817 （例 1）

链接：https://www.lydsy.com/JudgeOnline/problem.php?id=3817
题意：

给定N,R
求$\sum_{d=1}^{n}(-1)^{\lfloor d^2R \rfloor}$

其中 R 可能为无理数。
做法：
利用 [x]%2=[x]-[x/2]*2 。所求即转化

$$\sum_{d=1}^{n}(4\lfloor\frac{dx}{2}\rfloor-2\lfloor dx\rfloor+1)$$

即为

$$n+4\sum_{d=1}^{n}\lfloor\frac{dx}{2}\rfloor-2\sum_{d=1}^{n}\lfloor dx\rfloor$$

将求和全部看成

$$\sum_{d=0}^{n}\lfloor\frac{bx+c}{a}\rfloor$$

的形式.
然后可以用类欧几里得算法来递归求解.

2、BZOJ 2987 （例 2）

链接 https://blog.csdn.net/CreationAugust/article/details/50778646

通用部分：

对于 x\y 为 0 的情况，都可以直接 [范围/a(or b)] 来求解 于是都转化为 x>=1 y>=1 的情况

做法 1： （现推式子 其中对于初始题目的输入 ans=Σ[(c-bx)/a]） 将-b 转化为 b 按照之前推导 f 的类似方法即可，在下图中也有展示。

求$Ans=\sum_{x=1}^{n}\lfloor\frac{C+Bx}{A}\rfloor$

下面要求 $C<A$ & $B<A, A, B, C\geq 0$ 如果不满足可以通过这些来转换：

1. $if\ \ B<0\ \ t=\lfloor\frac{A-1-B}{A}\rfloor, Ans-=\frac{t*n*(n+1)}{2}, B+=t*A$
2. $if\ \ C<0\ \ t=\lfloor\frac{A-1-C}{A}\rfloor, Ans-=n*t, C+=A*t$ ps:t为C变为$<A$的累加次数
3. $if\ \ C\geq A\ \ Ans+=\lfloor\frac{C}{A}\rfloor*n, C\%=A$
4. $if\ \ B\geq A\ \ Ans+=\lfloor\frac{B}{A}\rfloor*\frac{n*(n+1)}{2}, B\%=A$

然后来推式子：

$$Ans=\sum_{x=1}^{n}\lfloor\frac{C+Bx}{A}\rfloor \tag{1}$$

$$=\sum_{x=1}^{n}\sum_{k}[kA\leq C+Bx] \tag{2}$$

$$=\sum_{k=1}^{\lfloor\frac{C+Bn}{A}\rfloor}n-\lfloor\frac{Ak-C}{B}\rfloor+1 \tag{3}$$

$$=\sum_{k=1}^{\lfloor\frac{C+Bn}{A}\rfloor}n-\lfloor\frac{Ak-C+B-1}{B}\rfloor+1 ps: Ak-C\leq Bx \tag{4}$$

那么$n=\lfloor\frac{C+Bn}{A}\rfloor, B'=A, C=B-C-1, A'=B$

于是问题规模被不断缩小，复杂度$O(\log n)$

做法 2：

推到 Σ[(c+bx)/a]后， 这里 x 是从 1 开始。而之前通用的 f 函数，下标是从 0 开始 只要转化为 Σ[(c+b+bx)/a] 并将 x 的最大限制-1 即转化为了对 f 的求解。

3、 BZOJ 2187（例 3）

题意

分数 给你4个正整数a，b，c，d，求一个最简分数 p / q满足 a / b < p / q < c / d，若有多组解，输出q最小的一组，若仍有多组解，输出p最小的一组。
1 < = a，b，c，d < = 10^9

分析

又是一道经典的类欧题目。推一波做法：

若$\frac{a}{b}$与$\frac{c}{d}$之间存在一个整数，则返回这个整数即可。

$$若a=0，那么\frac{p}{q}<\frac{c}{d}=>q>\frac{pd}{c}$$

$$由于q要最小，所以p=1,q=\lfloor\frac{d}{c}\rfloor+1$$

$$若a<=b且c<=d则\frac{d}{c}<\frac{q}{p}<\frac{b}{a}$$

$$否则的话，\frac{a\%b}{b}<\frac{p}{q}-\lfloor\frac{a}{b}\rfloor<\frac{c}{d}-\lfloor\frac{a}{b}\rfloor$$

其中可以把分数倒过来，是因为对于大小在一个给定区间内的分数，最小化分子等价于最小化分母。

4、2018 牛客暑期训练营第十场 H

H. Rikka with Ants

题目大意：在坐标平面上有两只蚂蚁，它们从 $(1,0)$ 开始走，每一步它们会从 (x,y) 走到 $(x+1,y)$ 或 $(x,y+1)$。第一只蚂蚁不能穿过 $y=\frac{a}{b}x$，第二只蚂蚁不能穿过 $y=\frac{c}{d}x$，每只蚂蚁会尽可能先往上走，再往右走。求两只蚂蚁经过的公共整点的数量。

题解：显然 $\frac{a}{b}=\frac{c}{d}$ 时，公共整点有无穷个。否则，直线 $y=x$ 上两只蚂蚁走过的区间分别为 $[\lfloor\frac{ax-a}{b}\rfloor,\lfloor\frac{ax}{b}\rfloor]$ 和 $[\lfloor\frac{cx-c}{d}\rfloor,\lfloor\frac{cx}{d}\rfloor]$，那么答案即为 $\sum_{x=1}^{+\infty}\max\{\min\{\lfloor\frac{ax}{b}\rfloor,\lfloor\frac{cx}{d}\rfloor\}-\max\{\lfloor\frac{ax-a}{b}\rfloor,\lfloor\frac{cx-c}{d}\rfloor\}+1,0\}$。不妨设 $\frac{a}{b}>\frac{c}{d}$，有答案为 $\sum_{x=1}^{+\infty}max\{\lfloor\frac{cx}{d}\rfloor-\lfloor\frac{ax-a}{b}\rfloor+1,0\}$。为了后面求类欧方便，把式子变换成 $\sum_{x=0}^{+\infty}max\{\lfloor\frac{cx+c}{d}\rfloor-\lfloor\frac{ax}{b}\rfloor+1,0\}$。注意到，当 $\frac{cx+c}{d}<\frac{ax}{b}-1$ 时，有 $\lfloor\frac{cx+c}{d}\rfloor-\lfloor\frac{ax}{b}\rfloor+1\le 0$，因此不用计算贡献；而 $\frac{cx+c}{d}\ge\frac{ax}{b}-1$ 时，有 $\lfloor\frac{cx+c}{d}\rfloor-\lfloor\frac{ax}{b}\rfloor+1\ge 0$，可以计算贡献。设 $X=\lfloor\frac{bc+bd}{ad-bc}\rfloor$，那么答案即为 $\sum_{x=0}^{X}\lfloor\frac{cx+c}{d}\rfloor-\lfloor\frac{ax}{b}\rfloor+1$，使用类欧计算即可。

时间复杂度 $\mathcal{O}(\log\max\{a,b,c,d\})$。

需要注意的是，过程中求解 f 的变量等等要用 int128 不然会溢出

FWT 快速沃尔什变换

2018 年 8 月 22 日
20:57

已有几道相关的题目记录在了备忘录-ACM-题目记录 36 附近的地方

1、2018 牛客暑期第 8 场 H

链接

H. Playing games

题目大意：给 $n \le 5 \times 10^5$ 个数（范围 5×10^5），问最多能选多少个使得它们异或和为 0。

题解：记 n 个数的和为 sum。从反面思考，即为最少选多少个数使得它们的异或和为 sum。我们以每个数为一行，构造一个模 2 意义下的矩阵，易知它有一组大小不超过 19 的基，也就是说我们可以用不超过 19 个向量表示 sum。

设 $f(x) = \sum_{i=0}^{2^{19}} \mathrm{cnt}_i x^i$，在异或卷积的意义下定义多项式乘法，那么 $[x^j]f^i(x)$ 就表示用 i 个数（可能重复）表示 j 的方案数。于是我们可以依次计算 $[x^{\mathrm{sum}}]f^0(x), [x^{\mathrm{sum}}]f^1(x), \cdots$，直到它大于 0 为止。虽然之前我们说选取的数可以重复，但是**最少**的能表示出 sum 的数肯定是不会重复的，所以这并没有影响。

由于乘了几次以后系数就会变得很大，所以要取模，恰好被模数整除的概率很小，如果实在不放心可以多取几个模数。

如果每次都逆 FWT，复杂度为 $\mathcal{O}(n \log^2 n)$。但事实上我们没有必要这样做，每次乘法后我们只需要知道 x^{sum} 的系数，这一项我们很容易 $\mathcal{O}(n)$ 求出，而不需要将整个多项式逆 FWT。

时间复杂度 $\mathcal{O}(n \log n)$。

2 个 log 的常数小的做法也可以通过（就是不像上文中所说 O（n）求，而是每次都 DWT 出来整个的结果），不过只能用模版中的最速的那个来做。
1 个 log 的做法，就是利用了可以 O(n)的求出某个 FWT 后的数列，DWT 后某一个位置的值。（分析式子就可以发现，某个位置的贡献是+1 还是-1 只与下标 i 的二进制下 1 的个数有关，因为没有一个 1，其都会出现在 x-y 中一次） （**新姿势！**）

2、2018 牛客暑期第 9 场 A

链接

A. Circulant Matrix

题目大意：给出一个线性方程组 $\sum_{j=0}^{n-1} a[i \oplus j] \cdot x[j] = b[i]$，求解 X。

题解：考虑递归求解。

$n = 1$ 时直接计算即可。

$n > 1$ 时，我们将相邻的两个方程两两合并，可以得到一个 $n/2$ 阶线性方程组：

$$b[2i] + b[2i+1] = \sum_{j=0}^{n/2-1} (a[2(i \oplus j)] + a[2(i \oplus j)+1])(x[2j] + x[2j+1])$$

于是我们可以解出所有的 $x[2j] + x[2j+1]$。

再将所有的 $x[2j+1]$ 代换成 $(x[2j] + x[2j+1]) - x[2j]$，可以发现，常数项可以用 FWT 计算，系数则变成了 $a[2j] - a[2j+1]$，具体式子就不写了。这样又得到一个 $n/2$ 阶线性方程组。

时间复杂度 $\mathcal{O}(n \log^2 n)$。

coldwater's comment:

然而题目就是给了 a 和 x 求 FWT 的结果 b，求解 x，直接 FWT 就完事儿了。

只需要看下面这个 coldwater's comment 即可。实际上就是一个裸的对 FWT 求逆的过程。模意义下，FWT 过程中可以任意取模，对结果都没有影响。
这里给出 a 和结果 x，想要逆过来求 b，只需要求 x 和 a^(MOD-2) 的卷积即可。
（a*b=x 所以 b=x/a 模意义下即是 x*a^(MOD-2)）

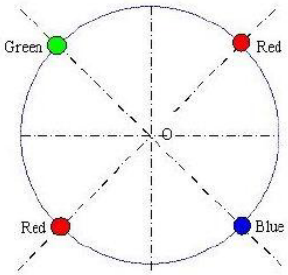
Polya 定理

2018 年 9 月 3 日
23:06

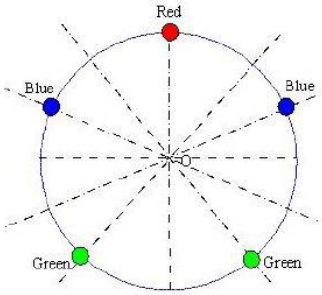
1、POJ 1286

Necklace of Beads

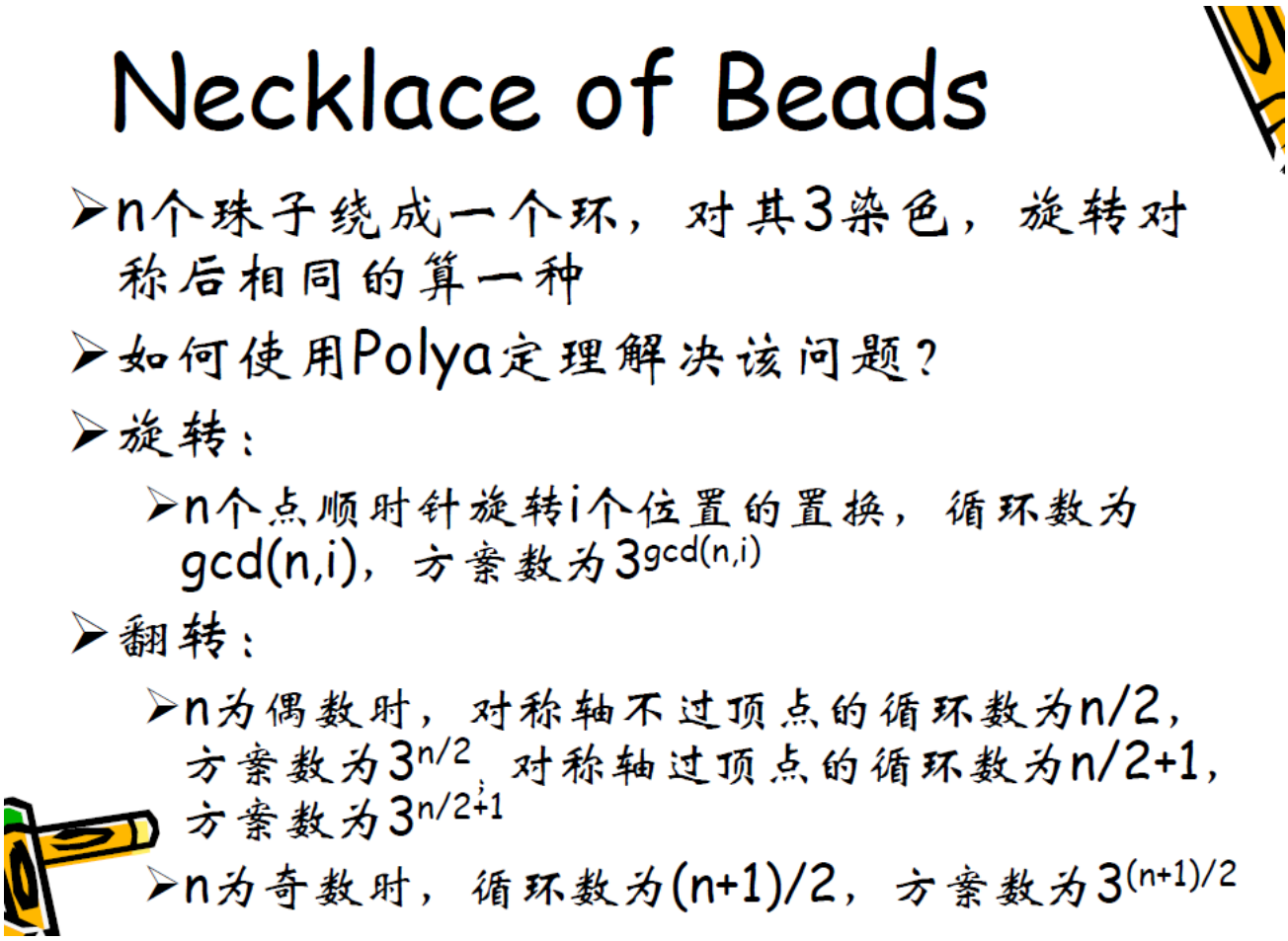
- POJ1286
- 题目大意:
  - 将三种不同颜色的珠子串成有n个珠子的项链,旋转/翻转后相同的算同一种,求方案数



The form with n=4



The form with n=5



2、HDU 1812

题意：给定 n、c。n*n 的方格，染 c 种颜色，旋转相同的算一种染法。求总共染色方案数。

做法：总置换数为 8：以左右的中间、上下的中间对称。以左上到右下的对角线对称，以左下到右上的对角线对称。旋转 0、90、180、270 度。接下来只需要数每一个的循环数。其中以左右 or 上下的中间对称，如果 n 为奇数，则是以中间的列为对称轴。左右两侧循环。故总共 n+（n*n-n）/2 个。若 n 为偶数，是以中间的线为对称轴，左右两侧循环，故总共 n*n/2 个。以斜对角线为对称轴，不管怎样都是(n*n-n)/2 个。

旋转只需要按与中心的距离分组，就变成了若干个大小为 4i 的圈同时旋转 ki。（k=0、1、2、3）。对每一层圈，循环数为 gcd（4i，ki）。（旋转的结论）求和即为旋转 k 时的循环总数。

3、UVA10601

题目：有 12 条边，分别有特定的颜色，组成一个立方体，问有多少种

做法：正方体 polya 的分析见奥赛经典-组合数学-P36。以某轴旋转，相当于与该轴垂直的面直接旋转，与该轴垂直且相交的直接绕交点旋转（注意到要使整个图形还在原位置，这样则只能转 0or180）。其余按照画图自己分析即可。
另外关于总共 24 种置换是由于：对于某个点可以移动到 8 个位置，而其连出的 3 条棱也有 3 种位置关系。故总共 3*8。（略感 fake）

4、UVA 11255

题目：a、b、c 各三种珠子，串成一个 a+b+c 大小的项链，可旋转、翻转，问有多少种方法
做法：burnside，旋转循环数是 gcd(n,i)，翻转如果是奇数则为（n+1）/2，若是偶数：过点：n/2+1，不过点 n/2。使用 burnside 其实与 polya 的区别就在于同一个循环中要求相等的总共染色方法数需要自己计算。对于这里显然就是一个组合数。（旋转操作：每个循环大小相等，翻转操作除了偶过顶点这种情况外全是 2，对于这种情况直接枚举这两点的颜色即可）

5、POJ 2154

题意：将正n边形的n个顶点用n种颜色染色，问有多少种方案（答案mod p，且可由旋转互相得到的算一种）

先说说Pólya定理

设Q是n个对象的一个置换群，用m种颜色涂染这n个对象，一个对象涂任意一种颜色，则在Q作用下不等价的方案数为：

$$L=\frac{1}{|Q|}\sum_{q\in Q}m^{\lambda(q)}$$

|Q|为置换群中置换的个数，$\lambda(q)$为将置换q表示成不相杂的轮换的个数，其中包括单轮换，m为颜色数。

分析可以知道本题方案的表达式为：

$$L=\frac{1}{n}\sum_{0\le k\prec n} n^{\gcd(k,n)}$$

$$=\frac{1}{n}\sum_{d|n} n^{d}\sum_{0\le k\prec n}[d=\gcd(k,n)]$$

$$=\sum_{d|n} n^{d-1}\sum_{0\le k\prec n}[\gcd(k/d,n/d)=1]$$

$$=\sum_{d|n} n^{d-1}\sum_{0\le k\prec n/d}[\gcd(k,n/d)=1]$$

$$=\sum_{d|n}\varphi(n/d)n^{d-1}$$

6、POJ 2888

题意：

要用 $N(\le 10^9)$颗珠子连成环形的手镯，共有 $M(\le 10)$种不同的珠子，规定 K 个条件，每个条件规定某两种珠子不能相邻，旋转后相同的方案视作相同，问有多少种本质不同的方案，对 9973 取模（$\gcd(N,9973)=1$）。

做法：旋转操作共 n 种置换，每个置换 gcd(n,i)个循环，每个循环大小 n/gcd(n,i)。注意到实际上连续的 gcd(n,i)个恰好对应的就是所有循环中的某个元素。使用 burnside 只需要这 k（k=gcd(n,i)）个组成的环符合。使用 01 矩阵表示可行关系后，就是该矩阵的 k 次幂的对角线上的和。为了加速求解，依然使用欧拉函数来加速。（类似上一题）

7、HDU 2481

题意：外面有一圈 N 个结点，中心有一个结点与 N 个结点都相连，总共就是 2*N 条边，删除 N 条边，使 N+1 个点连通，旋转相同视为等价，问有多少种情况。

做法：

首先先不考虑旋转的情况。对应的就是 BZOJ1002.

首先就存在线性递推的式子，这可以通过基尔霍夫矩阵的复杂推导得出。（真的很复杂 orz 可见 vfk 博客）
也可以考虑递推。先考虑一条链的情况。dp[n][0]表示 n 在的连通块没有与中心点连接，前 n-1 个点的每一个连通块与 n 连通或与中心连通。Dp[n][1]表示 n 在的连通块已与中心点连接，前 n-1 个点的每一个连通块与 n 连接或与中心连通的方案数。
很容易得到递推 dp[n][0]=dp[n-1][0]+dp[n-1][1] （必须要连进前一个连通块）
Dp[n][1]=dp[n-1][0]+dp[n-1][1]*2 （如果前一个连通块没有与中心点连接，而该点与中心点连接，就必须连接这两个连通块。如果前一个连通块已经与中心点连接，则该点可与前一个连通块连接，或单独连向中心点。）
考虑外圈 n 个点时的答案。（这里不考虑旋转）。对于某相邻的两点 a、b，显然就是 a、b 直接相连的可行方案数+a、b 不直接相连的可行方案数。
对此 分情况讨论：
①如果 a、b 不直接相连。那么将环拆成链。枚举 a 在的链的大小为 k，余下 n-k，这 k 个里一定有一个连向中心。于是方案数即为 k*dp[n-k][1]
②如果 a、b 直接相连。枚举 a、b 在的连通块的大小（不含中心点），对于一个大小 k 的连通块，有 k-1 个位置可以放 a、b，有 k 个位置可以与中心点相连。由乘法原理即为 k*(k-1)*dp[n-k][1]
综上所述，答案即为 Σk*k*dp[n-k][1]
最后使用的时候还是要使用线性递推，而这就可以使用矩阵快速幂加速了。
不过还需注意，这道题中模的数不是个质数，可能存在不存在逆元的情况。于是过程中要 MOD n*模数 （因为最后要除以 n 不过这样因为模的数太大，过程中就要使用二分乘法（见代码模版，实际上就是类似快速幂，二进制下的枚举 b 的各位，过程中不断修改 a 以及返回的值））

8、SPOJ - TRANSP2

题意：给定一个 2^(a+b)长的序列的 a、b，初始将其依次放进一个 2^a*2^b 的矩阵中。现在想交换该序列的顺序（一次操作只能交换序列中的两个元素）。使得将其依次放进一个 2^b*2^a 的矩阵中，恰好为之前矩阵的转置。
做法：
考虑在矩阵中行列二元组连起来构成的二进制表示：
e.g.行的二进制表示共 3 位，列的二进制表示共 4 位 初始在 (011,0110)连起来为 0110110
转置即为(0110,010) 连起来为 0110010 。
即每个位置通过循环左移 a 位 or 循环右移 b 位能得到的位置构成了一个循环。一个循环可以通过|循环|-1 次操作达成目标。

到这一步时，实际上是转化为了长度为 a+b 的 0\1 串，任意方向循环移动 a 个位置，通构集合个数

将位置的 a+b 位表示，抽象为一个环上有 a+b 个点。每个点取 0or1，整体即组成了一个位置的表示。

上述循环，即意为，对于两种染色的环，若能通过若干次 顺时针转 a，or 逆时针转 b（这两种实际可划归为：任意方向转 a）得到，则算是一个循环里的。

而我们之前已经可以通过旋转的结论，得知这样总共会有 gcd(a,a+b)=gcd(a,b)个循环，每个循环的大小为(a+b)/gcd(a,b)。

最后就转化为了 (a+b)/gcd(a,b)个元素的环，每个位置有 2^(gcd(a,a+b))种取值，的旋转不同染色方案数。

9、HDU 2865

题意：

有一个n个珠子的环，中心还有一颗珠子，用k种颜色来染。要求相邻珠子的颜色不同，中心珠子的颜色不能和外围任意一颗珠子的颜色一样，考虑旋转，问本质不同的珠子个数?

数据范围：$n \leq 10^9, k \leq 10^9$

做法：

显然中间先选 k 种之一放在那里，余下的就是一个环染 k-1 种颜色。这是高中数列递推的一个常见的问题。易得 An=an-1+an-2 使用矩阵快速幂很快就可以求得。不过需注意要从 2、3 开始递推，1 是不符合递推式子的。

接下来开始判旋转相等。对于一个循环大小为 k 的，实际上相等于 n/k 个连续的 k 个点（将这 k 个点视为一个）组成的环。每 k 个与中心组成的恰好就是 ak 的情况数。并且 n/k 个都是相等的。在用欧拉函数优化一下，即为

Σd|n φ（n/d）*f(d) 之后再乘以 k（中心选择的颜色），除以 n（一共 n 种旋转）即可。

10、HDU 4187

题意：将一个圆盘均分成 360000 份。给定 n 个点（按表盘上平分的坐标给出）。每个点染 m 种颜色，问有多少种旋转不同的染色方案。

做法：主要在于转化为 polya 的条件。这里由于不是均分，需要考虑哪些情况的坐标旋转可以重合。容易发现，只与点之间的距离有关。求出距离序列，对其求“最小覆盖”（kmp 中 n-next[n]）如果是整除 n 的，即可以完全覆盖上。完全覆盖上，显然就可以用最小覆盖划分整个圆，每最小覆盖个放在一组里。

就转化为了 n/长度个 颜色^长度种染色的旋转不同的染色方案，就可以直接 polya 了

群论

2018 年 9 月 6 日
15:41

1、

POJ 3128

题意：给你一个置换 P,问是否存在一个置换 M，使 M^2=P

来自 <https://www.cnblogs.com/simplekinght/p/6647563.html>

思路：资料参考 《置换群快速幂运算研究与探讨》 https://wenku.baidu.com/view/0bff6b1c6bd97f192279e9fb.html

结论一： 一个长度为 l 的循环 T，l 是 k 的倍数，则 T^k 是 k 个循环的乘积，每个循环分别是循环 T 中下标 i mod k=0,1,2… 的元素按顺序的连接。

结论二：一个长度为 l 的循环 T，gcd(l,k)=1，则 T^k 是一个循环，与循环 T 不一定相同。

结论三：一个长度为 l 的循环 T，T^k 是 gcd(l,k)个循环的乘积，每个循环分别是循环 T 中下标 i mod gcd(l,k)=0,1,2… 的元素的连接

考虑某个置换的平方。对于其中长度为奇数的轮换，平方以后这个轮换仍然为一个轮换只是元素顺序换了。一个长度为偶数的轮换，平方以后就变为两个大小相等的轮换了。因此，对于给定的置换，当中所有长度为奇数的轮换，可以直接当做是它原先平方产生的。而长度为偶数的轮换，必须一一配对，当做原先拆出来的。满足这个条件，就是平方。

来自 <https://www.cnblogs.com/simplekinght/p/6647563.html>

换言之，就是奇数长度的循环，平方之后仍是一个奇数长度的循环，只是顺序略有变化。
偶数长度的循环，平方之后会化为两个长度相等的循环。
现在我们有平方后的结果。对于奇数长度的循环，直接将其认作是原本就是这些数组成的奇数循环即可。
而对于偶数长度的循环，其一定是原本由 2*|其长度|的偶数循环平方得到的，这些需要进行配对。于是平方结果中，所有偶数长度的循环集合（按长度分集合），每个集合中元素的个数一定得是偶数

2、POJ 3590

题意：

对每一个置换 T，都存在一个 T^k = e。现在让你求一个 n 元置换，使得它的阶最大，即当 T^k = e 时，k 最大。若同时存在多个这样的 T，那么输出其中排序最小的。

题解：由于每一个置换都可以分解成若干个轮换，那么这些轮换的阶的最小公倍数就是该置换的阶。

所以题目可以变成这样：给你一个整数 n，求 n1+n2+n3```+ni = n。 并且 n1,n2,```ni 的最小公倍数最大。

1.求最小公倍数并不难，动态规划解决。

2.那么求得最小公倍数之后怎么保证置换排序最小呢？

我们不妨令某个最小公倍数为 lcmMax， 那么将 lcmMax 因式分解之后得到 lcmMax = p1^k1*p2^k2*``pi^ki

并且 p1^k1+p2^k2+…+pi^ki <= n。这个是显然的，因为 lcmMax = p1^k1*p2^k2*``pi^ki <= n1*n2*n3```*ni

而 n1+n2+n3```+ni = n，所以 p1^k1+p2^k2+…+pi^ki <= n。

3.用次用 pi^ki 个元素构成一个轮换，那么就能保证该置换 T 的阶最大。

那么你可能会问，剩下的元素怎么办呢？其实全部让它们为一阶轮换就 OK 了，因为一阶轮换并不影响最后 T 的阶。

来自 <https://blog.csdn.net/tsaid/article/details/7389140>

以上题解中间有一段不太明白是在说什么。反正转换成最小公倍数的问题后，dp 是很显然的。对于每个结果，将 ans 质因数分解，即化为了若干个 pi^ki 长度的循环（余下部分用长度为 1 的循环代替）。这样易证可以构造出方案。（其实方案的存在是显然的，根据 dp 就知道了）要求输出排序最小的，实际上很容易发现，一定要让较短的循环在前面，如果有长度为 1 的那就更优先了。（相当于从 1-n 按顺序放，只有长度>1 的循环部分需要串一位）。并且可以证明，拆分成 pi^ki 是使得余下的 1 最多的

方案（并且不会取等，当且仅当此时能取得）。余下长度非 1 的部分，按先放长度短的循环的策略，贪心的放即可。

3、POJ 1282

题意：n 个房间，n 个人。每个房间有一个有 p 个格子的转盘，第一天每个人进入自己编号的房间。之后重复一下过程：每个人下一天进到当天所在的房间的转盘指的房间，转盘转一个格子。问最少多少天又回到最开始的状态。

做法：

参见《庆典的日期(解题报告)》。大致来讲，就是：先前做的群论置换问题，每个位置下一次的转移都是固定的。而这里 p>1，不再是固定的。不过注意到，整体的置换中，连续 p 个的结果是相同的。于是将总次数按模 p 的结果分别考虑。预处理出前 p 轮的置换的结果。

枚举模 p 的余数。并且逐个位置考虑。在总次数模 p 余 k 的情况下。初始在 i 的，需要 pt+k 轮才能回到 i。k 是固定的。先考虑最后 k 轮，根据最开始的预处理，可以确定出在 pt 轮后 i 需要在的位置。那么接下来只要找最小的 t，使得 t 次连续 p 个操作后，i 会移动到对应的位置。而显然，将 p 次操作视作一次后，也相当于一个置换。那么 i 一定在其中的一个轮换中。在 0 次的时候，i 就在自己的位置。我们枚举 i 在的轮换（每次按连续 p 转移），就能得到轮换的长度，以及 i 想要移动到目标位置需要的操作次数模轮换的长度为几。（轮换的长度就是轮换的循环节）。这样就能得到 n 个同余方程，使用中国剩余定理即可求出其最小正整数解。

数据结构

线段树

2018 年 7 月 29 日
17:23

1、

2018 牛客多校第四场 J

线段树加速拓扑（建图？）。

题意：正常 hash，x 的 h(x)就是 x%p，但有可能冲突。这时候就修改为 h'(x)=(h(x)+1)%p,直到没有冲突。但题目给的不是插入数的顺序，而是结果的 hash 表。请输出字典序最小的插入顺序。

做法：显然某个数与其应在的位置之间的“区间”的数都需要在该数之前被插入。因此构成了一个拓扑顺序的问题。考虑到区间都是连续的，使用线段树优化这个拓扑问题。线段树每个结点维护该区间内的点的状态（使用度数记录）。初始为从所有叶子节点，有向的不断向上指的有向的树。每次多限制，就将对应的一侧或两侧“区间”内的结点（就是线段树上好几个结点）指向该被限制的叶子。这样在线段树上拓扑序（相同拓扑序选择字典序最小）即可。

2、

2018 牛客多校第四场

扫描线

题意：给定许多个点，以及这些点出现的概率。对于平面 x>0,y>0 上的整点，只要存在某个给定的(xi,yi)使得该点(x,y)满足 x<=xi&&y<=yi，则该点即被选择。求被选择的点的数目的期望。

做法：稍加分析即可以转化为扫描线。

首先，对所有给定的点按 x 分块。离散化 y。注意到[xi+1,x(i+1)]这个区间中，每个 y 值的点的期望被 x(i+1)这一列上的点划分成了许多部分。我们就是利用这一点进行扫描线（从右向左扫）。每一点的期望为 1-π(1-pi) (其中 pi 为其右上的点的概率)

使用线段树维护这个期望，线段树叶子结点个数即为离散化 y 后 y 的不同个数。每次更新[1,yi]区间（因为我们要看的是某点其右上方点的（1-pi）之积 ，从右往左扫，已经保证了在右侧，只更新[1,yi]的，保证了只更新其实际影响到的。）

我们一路从右扫过来，对于 xj,j>i+1 的情况已经更新完。接下来只需要执行 x(i+1)这一列上的 y 在线段树中的区间更新。之后加和即可。对于每一列，其期望和为 ∑1-π(1-pi) 求和即为 ymax-sum[1] (sum[1]为线段树根节点的 sum 值) 由于 xi+1 到 x(i+1)区间内，相同的 y 值期望相同，再乘上(x(i+1)-xi)即可。

3、

2018 牛客多校第五场 D

线段树+BIT（主要是性质，利用到的数据结构方面的都很简单）

题意:

题目描述

给定一个 [1,n] 之间所有偶数的排列 b，其中 n 是偶数

现在有一个数组 a=[1,3,5.....n-1]

要求归并 a 和 b，使得他们归并后逆序对数量最少

1<=n<=200000

做法：
首先单独考虑 b 自身内部构成的逆序对数。单增的奇数数列无逆序对形成。
有一个小结论，从小到大插这些奇数，最优的插的位置一定不减的递增。
证明如下：设插入到某个位置，其前面有 p1 个比其大的，后面有 p2 个。前面有 q1 个比其小的，后面有 q2 个。则这个位置构成的逆序对数就是 p1+q2 个。注意到从小到大的奇数，比其小的数，和比其大的数都是确定的。比其小的为(n-1)/2 个。而 q1+q2=(n-1)/2。那么逆序对数就转化为了 p1-q1 +(n-1)/2。后部分是独立于选的位置的。将所有的 b 的位置分别用 1，-1 表示比其大，比其小。那么所要找的就是一个最小的前缀和位置。（包括最前方不选的地方，共 m+1 个，其中 m 为偶数个数）。这个使用线段树很容易维护。并且每次考虑完 n，要考虑到 n+2 时，原线段树中只有 n+1 这个偶数所在的位置开始的后缀中的数，其前缀和要-2。对此线段树区间修改一下既可。由此可证最小的前缀不减的单增。因此独立考虑这些数，只需要算当前最优位置的逆序对个数即可，因为选出来的位置就已经是单调递增的了。

4、

2018 牛客多校第六场 I
题意：给若干个小区间。每次去掉一个点。问每次会导致多少个小区间第一次被去掉其中的点。最后再对每个区间输出第一次被去掉其中点的操作的 ID
做法：离散化坐标。线段树，维护左端点在区间内

树状数组

2018 年 8 月 3 日
11:12

1、POJ 2155

题意：n*n 的 01 矩阵（n<=1000），单次修改一个矩形（矩形内的点异或 1），单次查询一个点的 01 值。

做法：裸的二维 BIT（之前还以为有多高大上 orz）。处理方法实际就是两维的循环，lowbit。所以显然复杂度是两个 log。将(x1,y1) (x2,y2) (分别为左下，右上）的修改，转化为 (x1,y1) (x2+1,y2+1) (x1,y2+1) (x2+1,y1)的前缀异或即可。

2、树状数组求已出现数中不大于 k 的最大值

做法：

该段为重点！加深了对树状数组的理解。修改某个位置，能转移到 x 的集合为：lowbit(x)以前（更高）的位与 x 一致。与 x 相等，或 lowbit(x)这一位为 0，余下（更低）的位取遍 01 子集的数的集合。

亦可以表示为 [x-lowbit(x),x]。因此对于任意一个数 x，所有<=x 的都能转移到它，而这些能转移到它的数，可以按上述办法，对于 x 不断-lowbit(x)，分块到不同的区间中。（e.g. 第一个区间中的数最大为 [x-lowbit(x),x] 第二个区间中的数略小与上述的数 设 y=x-lowbit(x) 则第二个区间即为[y-lowbit(y),y] 其恰为 x 去掉倒数两个最低位的 1，到 x 去掉最后一个 1 之间的数）。

理解了以上之后，就只需从后往前逐个检查这些区间有无被更新过（总共最多该数 2 进制下 1 的个数次）。确定了某个区间后，继续二分的找（每次 x-lowbit(x)/2）即可最终确定。

CDQ 分治

2018 年 8 月 10 日
10:40

1.BZOJ 3262

题意:

有 n 朵花,每朵花有三个属性:花形(s)、颜色(c)、气味(m),用三个整数表示。
现在要对每朵花评级,一朵花的级别是它拥有的美丽能超过的花的数量。
定义一朵花 A 比另一朵花 B 要美丽,当且仅 Sa>=Sb,Ca>=Cb,Ma>=Mb。
显然,两朵花可能有同样的属性。需要统计出评出每个等级的花的数量

来自 <https://vjudge.net/problem/description/38388?1533479541000>

做法：相当于三维(x,y,z)，对每个点求严格小于等于它的点。按 x 排序（除了 x 外，y、z、操作编号也要尽可能有序，至少保证查询操作都在后面，否则会出错）。这样就保证之后扫描 x 是有序的了，即降了一维，转化为二维的问题。CDQ 分治，维护 y 的大小为从小到大。这样 CDQ 分治的合并操作中，就能保证 y 是从小到大的了。再使用树状数组维护 z 即可。

2、

BZOJ 3295

权限题。

题意：给出初始数列，为一个 1——n 的排列。给定 m 个不同的 1——n 的数，依次从这个序列中删除，问整个序列在每次删除前的逆序对个数。

做法：使用 CDQ 分治，首先就要离线，并将问题转化为一个几维的问题。对于这道题，问删除前逆序对的个数，就相当于倒着往一个数列里插数，每次问插完之后的逆

序对数。三维（t,x,y）表示：t 时刻加入，在位置 x，数为 y 的操作。首先排序 t。对于 x 使用 CDQ 分治处理，对于 y 使用树状数组维护。具体分治的操作为：在合并时，第一遍合并得到 x 递增的序列，在此过程中按照之前做的 CDQ 分治的题目一样处理，左侧区间的点加入 BIT，右侧遇到询问的点就查询 BIT 更新答案。这样我们就维护了左侧时间区间内，满足在右侧时间区间内 x 左侧，比其大的逆序对。但这样只考虑了一半，还有一半是左侧时间区间内，满足在右侧时间区间的 x 右侧，比其小的这样的情况。对此，只需要对已经得到的 x 递增的序列，倒序扫一遍，再使用 BIT 维护一下即可。由于是最外维按 t 递增排序，根据 t 的值就知道某个 CDQ 结点是在左侧区间还是在右侧区间。

3、

BZOJ 2726

题意：

常见的斜率优化中的分组问题。大概可以描述为 转化为了 dp[i]=min{-T[i]X[j] + Y[j] }的问题。

这样就转化为了维护下凸包，使得 k 递增的问题。其中每次最优在 第一个 k> T[i]时取得。如果只是这样，那就用单调队列维护一个下凸包即可。但是这道题目里的 T 并不是单调递增的。

做法：（代码在模板中有） 这里写的不够好 可以看下面的“4、”

使用 CDQ 分治处理。最外维通过排序使得 T 单调递增。在 CDQ 内部，每次先处理左侧部分（处理的过程维护使得 X 为第一关键字，Y 为第二关键字单调递增）。之后开始考虑左侧的下凸包对右侧的贡献。注意这里枚举右侧的点，必须要按照 T 递增，才能在下凸包上整体 O(n)的走。因此这个操作必须在 CDQ(mid+1,r)之前进行。这样我们就完成了考虑左侧区间对右侧的贡献，接下来再递归处理右侧区间即可。

另： 二分的方法： https://www.cnblogs.com/HocRiser/p/8681536.html

横坐标单调递增，所以依旧是维护下凸包，每次在下凸包上二分

4、

BZOJ 1492

题意：

转化为 dp 方程

f[i]=max(f[i-1],x[j]*a[i]+y[j]*b[i])

做法：（代码在模版中有）

考虑斜率优化，f[i]=max(x[j]*a[i]+y[j]*b[i]),y[j]=(-a[i]/b[i])*x[j]+f[i]/b[i]，-a[i]/b[i]看作斜率，表示成了关于平面 x,y 上的斜率关系。bi 不变，要让 fi 最大，就要让截距最大。

很容易发现，满足的点都在(x,y)的上凸壳上，所以我们维护这个凸壳。

但注意到每一点的-a[i]/b[i]在 x 单调递增的时候并不满足单调。于是需要使用 CDQ 分治来维护。先在外面按斜率从小到大排。注意到转移只能从 id 小的向大的转移。在同一层 CDQ 内，再按 id 划分为左右（cdq 中的[l,r]表示的就是 id 从[l,r]）。之后分治左侧区间，求解出了左侧 id 范围的 dp 值。此时右侧区间尚未处理，但是满足其中的 id 都比左侧区间的 id 大，并且斜率优化中的 k 单调递增，那么就可以同步在左侧的上凸包和右侧的 id 值上贪心的走，O(n)就可以处理完左侧向右侧的转移。之后再递归的处理右侧区间即可。

5、

HDU 5730

链接：https://vjudge.net/contest/244989#problem/D

题意：求递推式$f_n = \sum_{i=1}^{n} a_i f_{n-i}$，模313

做法：

裸的 CDQ 分治+FFT 的题目。注意到递推实际上是卷积的形式，就可以考虑 FFT。当前区间[l,r] 先递归处理[l,mid] ，进行完后，[l,mid]的 dp 值便全都算出来了。就按照 CDQ 分治的套路来考虑左侧区间对右侧区间的影响。这只需要做一个卷积即可。之后再递归的处理右侧区间。

6、

BZOJ 3237

给定一个连通的无向图和若干个小集合，每个小集合包含一些边。对于每个集合，你需要确定将集合中的边从原来的无向图中删除后该图是否保持连通。

一个图是连通的当且仅当任意两个不同的点之间存在一条路径连接他们。

做法：

建立结构体记录每次操作的边集。对每条边建立结构体，记录边的两点、边被加入的时间戳。

初始时，遍历所有边，不再任何一个边集的边直接用并查集维护上（即合并）。

开始 CDQ 分治。其中终态：l=r 时，需要除了这个边集的边全都连上。这时只需要枚举这个边集中的所有边的对应两点，如果两点不再一个连通分量中，则说明去掉这组边集会导致整个图不连通。不然就是联通的。

考虑初态。想要先处理[l,mid]。那么就需要先把[mid+1,r]的边集中的边（不在[l,mid]的边集中）全部连上。之后开始递归处理[l,mid]。之后再撤销[mid+1,r]的加边操作。转而把[l,mid]的边集中的边（不在[mid+1,r]的边集中）全部连上。开始递归处理[mid+1,r]。

其中维护在不在某个边集中，用之前初始看是不是在某个边集中的时间戳方法即可。并查集连边与撤销使用**可持久化并查集**（这里用的是简易暴力版，整个知识点尚未学习）即可。

7、

HDU5126

题意：Q 次操作，三维空间内 每个星星对应一个坐标，查询以（x1,y1,z1）(x2,y2,z2)为左下顶点 、右上顶点的立方体内的星星的个数。

做法：与裸的静态四维偏序差不多。将询问化为 8 个差分。四维的 CDQ，简单来说就是先按一维排序。之后每次[l,r]将该维打标记。按第二维排序，再进入下一个 CDQ，按第三维排序，BIT 维护第四维。对更新 or 查询，进一步加上了对第一维标记的要求。

整体二分

2018 年 8 月 10 日
23:47

1、第 k 小（静态&动态）

静态对应 POJ 2104 动态对应 ZOJ 2112

做法：对于整体二分来说，就是一个入门题目。node 记录增加、查询操作。其中增加操作通过设置+1\-1 就可以完成对增加还是删除的界定。分治的时候其实类似于 CDQ 分治，但这里想向下一层，必须得在该层进行完时才能进行。对于每一层，就是维护一个当前对应的 node 区间编号，以及可能的答案的区间[l,r]。在此区间里二分得(l+r)/2。以将整个 node 区间进一步划分。这样一直进行下去，l==r 时，对应的 node 区间的查询点的答案就是 l。

KD-TREE

2018 年 9 月 10 日
22:10

1、BZOJ 2648

题意：

这天，SJY 显得无聊。在家自己玩。在一个棋盘上，有 N 个黑色棋子。他每次要么放到棋盘上一个黑色棋子，要么放上一个白色棋子，如果是白色棋子，他会找出距离这个白色棋子最近的黑色棋子。此处的距离是 曼哈顿距离 即(|x1-x2|+|y1-y2|) 。现在给出 N<=500000 个初始棋子。和 M<=500000 个操作。对于每个白色棋子，输出距离这个白色棋子最近的黑色棋子的距离。同一个格子可能有多个棋子。

来自 <https://vjudge.net/problem/description/18926?1536376103000>

做法：

KDtree 入门题目。

每个 kdtree 结点，维护：[l,r]表示该节点维护的区间,d[0],d[1] 区间 mid（也是这个结点的支配点）的 x 和 y。Min[0],Max[0] 该区间 x 坐标的范围。Min[1] Max[1]该区间 y 坐标的范围。显然每个区间只有一个支配点，并且[1,n]中每个点也恰好只在一个结点中作为支配点。KD-tree 每一层分别按 x、按 y 为第一优先级来排序。

以上为静态建树过程。

先考虑之后操作的插入点操作。新插进的点，在原树中一定能沿着某条路径走到叶子（可能对应的区间非只有 1 个元素，但是要走到的某个儿子没有结点，即最初建树时没有范围在其中的点）。此时，按该层的排序方式，新的点一定在一侧（这里一侧含等号，即加入的点和原先的某个点重合，但没有影响，无非就是新加一个点）。沿路径更新这些点，并且在最终的叶子节点下加入了新插进来的点。并在原数组中加入该点。

考虑查询操作。这里实际上是使用 A*的估值来“二分搜索”。

估值函数为：对于(x,y) 和一个 x 区间、y 区间。如果 x 不在 x 区间内，则值加上 x 到该区间的最短距离，y 同理。优先搜索估值函数低的。可以证明，这样单次复杂度均摊为 sqrt(n)。（很玄学啊 orz，但据说一般都可以接受）。每次用支配点尝试更新答案。之前插入点的复杂度显然小于此，于是整体 O(nsqrt(n))

2、BZOJ 2850

题意：平面上若干个点，每个点有一个权值，若干次询问，每次询问 ax+by<=c 的点的权值之和

做法：依旧 KD-tree 维护区间（这里的模版代码写的更好一些），额外增加每个结点维护所有儿子节点的权值和之和。修改之处只有估值函数，以及判断。这里的估值，只是判断该子树是否有可能有符合的点。如果某子树所有点均符合，则一定有 ax+by 的四种极值全符合。只有部分符合的就必须再进一步 dfs，并且把该支配点的权值加入。

3、BZOJ 4066

题意：强制在线，2 种操作，一种是给定(x,y)，该点权值+v；一种是询问某矩形范围内点的权值之和。

做法：非强制在线的可以 CDQ 分治处理，强制在线就可以使用 KD-tree 维护各个矩形。判断依旧只需判断某矩形是完全包含在查询范围内，或者是完全不包含在查询范围内即可。注意到由于操作次数较多，需要若干次（大约 10000）之后就重构 KD-tree

笛卡尔树

2018 年 9 月 15 日
22:07

首先，记录一下笛卡尔树的具体内容。 链接

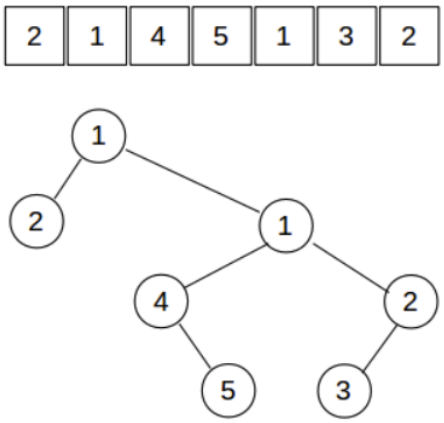
简单来说，笛卡尔树（Cartesian Tree）是一种特殊的二叉树，每棵子树的根节点都是整个子树的极值，并且对于这棵树进行中序遍历后的结果就是原序列（也就是所有点中序遍历正好对应了原数组的一段区间）。这样的树就是笛卡尔树。

性质:

要构造笛卡尔树，我们要先了解其性质。

1. 结点一一对应于数列元素。即数列中的每个元素都对应于树中某个唯一结点，树结点也对应于数列中的某个唯一元素。
2. 中序遍历（in-order traverse）笛卡尔树即可得到原数列。即任意树结点的左子树结点所对应的数列元素下标比该结点所对应元素的下标小，右子树结点所对应数列元素下标比该结点所对应元素下标大。
3. 树结构存在堆序性质，即任意树结点所对应数值大/小于其左、右子树内任意结点对应数值。

举个例子，以下就是对于数列 $\{2,1,4,5,1,3,3\}$ 构造的笛卡尔树（终于有一张原创的图了！不容易啊～）：



先看第一条性质：结点一一对应于数列元素，很显然满足。再看第二条，对这棵树进行**中序遍历**

笛卡尔树与 RMQ

RMQ（区间最值问题）指的是给出一个序列，要求以 $log_2 n$ 的级别求出某个区间内的极值。一般的 RMQ 求法是：定义 $F(i,j)$ 表示第 i 个元素往前推 2^j 个元素这段区间里的极值，这样便于修正（具体RMQ的做法这里不展开讲）。是不是感觉和上面的笛卡尔树有些联系？

一颗笛卡尔树里 i 元素到 j 元素的极值，就是他们的最近公共祖先的值。（请看上图仔细体会）

笛卡尔树的构造

知道了笛卡尔树的性质，我们就可以开始构造笛卡尔树了。对于上面那道题目，我们只需要维护一棵笛卡尔树，然后用单调栈存储从根节点到最后一个点的一段链（栈里除了第一个点以外每一个点都是上一个点的右儿子）。如果新加入一个元素 k，判断两种情况：

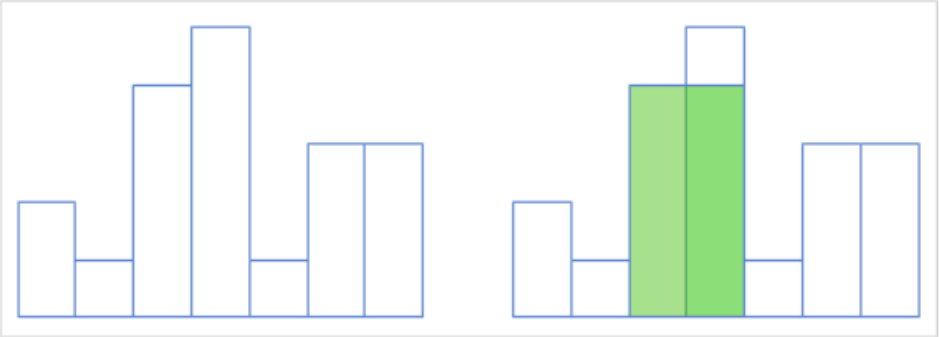
1. 它比栈顶还大。因为是单调栈，可以直接加到栈顶。
2. 它不比栈顶大。我们只需要在栈里找到它插入的位置，把比它大的一段链全都变成它的左子树，然后再把它插入。这样，在中序遍历的时候仍然是先遍历其左子树再遍历它自身，满足第二条性质；因为左子树都比它大，显然满足性质 1、3。

根据这种方法，很容易构造出笛卡尔树。详见代码。

1、最大子矩形面积 （HDU 1506）

题目描述

有一系列的长条状的矩形，宽度都为1，相邻的竖立在x轴上，求最大的子矩形。



输入输出格式

输入格式

第一行输入一个整数 n $(1 \leqslant n \leqslant 100000)$。

第二行输入 n 个整数表示每个矩形的高度 h $(1 \leqslant h \leqslant 10^9)$。

输出格式

输出最大子矩形的面积。

样例输入

```
7
2 1 4 5 1 3 3
```

样例输出

```
8
```

解题思路

可以很容易看出，这题可以用单调栈做：可以对于每个点，可以构造出 $left_i$ 和 $right_i$ 分别表示以当前的高度，向左、向右可以取到哪里。

这是单调栈的做法。其实另一种思路是 RMQ+递归：在 1～n 这段区间里找出最小值 A_i，然后以 i 为中心，把区间分成两半，左右分别和刚才一样找最小值、递归处理。这种方法就很想**笛卡尔树**了。

具体的操作为：建出笛卡尔树后，每个结点都是作为最小的最大区间就是以其为根的子树。这样只需要 dfs 一遍，得到每个子树的大小，对每个结点尝试更新答案即可。

2、HDU 6044

题意：对于有 n 个元素的全排列的合法性定义为：有 n 个区间，对于第 i 个区间[li,ri]有 li<=i<=ri,对于任意 1<=L<=i<=R<=n，**当前仅当 li<=L<=i<=R<=ri** 时 P[i]=min(P[L],P[L+1],...,P[R])。现在给出序列和相应的区间，问区间是否合法？

题解：

首先要理解题意：当前仅当 li<=L<=i<=R<=ri 时 P[i]=min(P[L],P[L+1],...,P[R])

因此对于 P[i]一定有 **P[i]>P[li-1]且 P[i]>P[ri+1]**，进一步说区间[li,ri](除了[1,n])一定被某个区间[lj,rj]包含,且 j=li-1 或 j=ri+1

即**区间 j 可分成[lj,j-1]和[j+1,rj]**

我们把 n 个区间按 L 升序 R 降序进行排序(**这样得到的区间 LR 正是前序遍历的区间,区间由大到小**)。**这样得到的实际上就是笛卡尔树的 dfs 序（前序遍历）。**得到的第 1 个区间一定要是[1,n](1 比任何数都小)，否则不合法，输出 0；设这个区间对应的是第 i 个数，因此区间可再分为[1,i-1]和[i+1,n],看是否有这 2 个区间，如果没有则不合法，输出 0...直到区间不可再分。

现在再来考虑方法数：设 f(i)为区间 i 内的方法数，u,v 分别为左右子区间，i 内一共有 ri-li+1 个数，除去中间一个，要从中选 i-li 个数放入左区间，剩下的放入右区间，因此答案为：f(i)=f(u)*f(v)*C(ri-li,i-li)

来自 <https://blog.csdn.net/qq_31759205/article/details/76146845>

3、HDU 6305

题意：定义RMQ(A,l,r)为：序列A中，满足A[i] = max(A[l],A[l+1],...,A[r])的最小的i。如果对于任意(l,r)都满足RMQ(A,l,r)=RMQ(B,l,r)则为A和B是RMQ Similar。现在出A序列，B序列的每个数都是0~1之间的实数，问满足与A是RMQ Similar的所有B序列中所有数之和的期望。

题解：不难看出如果A和B是RMQ Similar，则A和B的笛卡尔树同构。考虑B中的每个数是0~1之间的实数，因此出现相同数字的概率为0，可以假设B是每个数都不相同排列。设A的笛卡尔树每个子树的大小为sz[u]，则任一B排列与A同构的概率是$\prod_1^n \frac{1}{sz[i]}$，因为B中每个数满足均匀分布，因此期望值为$\frac{1}{2}$，和的期望为$\frac{n}{2}$，因此满足与A同构的B中所有数之和的期望为$\frac{n}{2\prod_1^n sz[i]}$

其中概率为此是因为：考虑笛卡尔树中的某一节点。条件概率，其在这个子树中的概率为 1，作为根节点意味着其是这个子树中数中的最大的。则概率即为 1/siz 综上所述，所有点的概率乘起来即为总概率。

Pb_ds-红黑树

2018 年 9 月 17 日
20:25

声明/头文件（使用时，红黑树就当成是一个平衡树就好了）

```
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree<pt,null_type,less<
pt >,rb_tree_tag,tree_order_statistics_node_update> rbtree;
```

来自 <https://www.cnblogs.com/keshuqi/p/6257895.html>

使用方法

```
/*
定义一颗红黑树
pt 关键字类型
null_type 无映射(低版本 g++为 null_mapped_type)
less<int>从小到大排序 (小的 rank 小)
rb_tree_tag 红黑树（splay_tree_tag）
tree_order_statistics_node_update 结点更新
插入 t.insert();
```

删除 t.erase();
Rank:t.order_of_key();
第 K 值:t.find_by_order();
前驱:t.lower_bound();
后继 t.upper_bound();
a.join(b)b 并入 a 前提是两棵树的 key 的取值范围不相交
a.split(v,b)key 小于等于 v 的元素属于 a，其余的属于 b
T.lower_bound(x) >=x 的 min 的迭代器
T.upper_bound((x) >x 的 min 的迭代器
T.find_by_order(k) 有 k 个数比它小的数
*/
（用此树 500ms 水过 bzoj3224）

来自 <https://www.cnblogs.com/keshuqi/p/6257895.html>

额外的，多次插入相同的数是没有意义的，计算 rank，查询 rank 中的 rank 都是从 0 开始。并且重复的只计算 1 次（似乎可能重复的都没有真的插进去）
于是就可以很容易的查询区间第 k 大，以及区间中某数是第几大。（这里的区间必须就是红黑树维护的区间）

具体示例：

1、ZOJ 4053 (2018 青岛网络赛 G)

题意：给一个数列，每次删除其中一个数，求出每一段的逆序对数量的最大值。强制在线。

题解：考虑新分裂出的两段，短的那段暴力计算，长的那段把短的那段以及删除掉的数的贡献从中扣除，复杂度 $O(n \log^2 n)$。

每次都选小的那一段进行暴力，那么总共暴力的长度就是 nlog。每次暴力使用红黑树维护。具体操作为：
原本长区间的总逆序数-短的逆序数-删除掉的数的贡献-跨左右区间的逆序对个数。
①式原本就已经维护了 ②式在建短区间的红黑树的过程中 nlog 的得到 ③式分别查询建完新区间，删完原区间两侧比其大、小的数的个数 log 得到 ④式枚举短区间，查询另一个区间的大？小于其的个数 nlog。
总共操作的区间的总长为 nlog 因此总复杂度 nlog^2
需注意的是，由于模版中计算名次等操作不算重复的元素，因此事先对 a 数组修改。
a[i] -> a[i]<<20|(a[i]目前为止出现的次数) 这样与 a[i]相等的范围就是 [a[i]<<20+1,(a[i]+1)<<20] （右侧等号取不到）

乍一看维护逆序对树状数组也可以，然而因为总共要建 n 个，使用固定区间的树状数组会远远的超过内存限制，因此只能使用各种树状结构（e.g.红黑树、主席树）

图论

虚树

2018 年 7 月 29 日
17:23

1、

BZOJ 3572（提交在 BZOJ 上）

题意：

给定一棵树，每次给一些关键点。树上每个点都被离他最近的关键点支配，距离相同取编号小的关键点。每次询问要求输出每个关键点支配的点个数。

做法：

又是一个通过给出∑k 的范围，间接暗示用虚树做的题目（尽管求出虚树之后还是得想的很清楚才行）。

建出虚树后，先求出此时虚树上的这些点每个点被哪个点支配。使用两次 DFS 即可。（一次用 i 子树的结点更新 i，一次用 i 的父节点更新 i）

然后进一步统计虚树上的边上的点。

对于边（u,v）

如果 bel[u]=bel[v]则"边"上的点全部计入 ans[bel[u]]

（注：这里边对应到原树中未必是边。对应的应是（设 u 为深度较浅的点。则应为 u 的含 v 的子树的子节点的子树中去除 v 的子树的部分））

如果 bel[u]！=bel[v]，那么就说明中间部分存在分界点。（这里找分界点也是只在（u,v）这条链上找就可以了。因为其他的点（不再链上的点，因为在链上的是显然的）到 u 和到 v，除了链上的部分分别是向下或向上走有不同外，其他部分是公有的部分）。加到答案中时，实际上就转化为了 bel[u]=bel[v]的情况，不过这里的 u、v 是原树中的点（说这个实际上是想说要加的也是两个子树的差）

不过上述的考察，实际上没有考虑完。其考察的只有虚树上的点，以及两虚树上点之间子树的情况。对于只在某一个虚树结点下的点没有考虑。（e.g. u 在虚树上，其某儿子结点对应的子树没有任何一个点在虚树上，那么这个子树就没有被考虑到。）对于这种情况，其中的点显然都被该结点对应的支配点（因为该节点也有可能是由 LCA 加入的）所支配。考虑这样的情况，只需要记录虚树中每个点的子树中有多少被考虑的了（临时数组 g 存初始的 siz，更新过程中不断减去被考虑的部分，余下的就应该加入这个点对应的关键点）

2-SAT

2018 年 8 月 2 日
10:10

1、
BZOJ 3495
前（后）缀优化建图（代码有保存在模版中作为例题）
题意:
有 n 个点， m 条边和 K 个国家（国家里的点已知）。
每个国家只能选一个点作为首都，并且要保证最后所有边的两端至少有一个点是首都，问是否存在方案。

做法:
每个点是首都或不是首都，只有两个状态，所以是 2-SAT 问题。m 条边的限制很容易转化，就是每个国家只能选一个点为首都比较奇怪。
其实这是典型的前后缀优化建图，这里以前缀优化建图为例:
首先我们先增加 n 个点，令 i 的新增节点为 i+n 。然后对于一个国家，
假设有 w 个点，那么该国家中的第 i 个点（设为 A[i] ）的新增点 A[i]+n
表示该国家中 1~ i 的点是否有被选的节点，即代表前缀 i 中是否有被选的节点。
那么我们可以得到以下关系式:
1. A[i−1]+n 选了， A[i]+n 必定选了； A[i]+n 没选， A[i−1]+n 必定没选（前缀之间的关系）。
2. A[i] 选了， A[i−1]+n 必定没选； A[i−1]+n 选了， A[i] 必定没选（一个国家只能有一个首都）。

后缀优化建图原理是一样的，只不过这道题并不需要后缀优化。
最后刷 2-SAT 判断是否存在解即可。

树形 DP

2018 年 8 月 4 日
22:17

1、

2018 牛客暑期多校第 6 场 G

题意：给定一棵树。n 个点重构为一个完全图，任意两点之间的边定义为树上两点之间的长度。求这个新图中任意两点最大流之和。

做法：最大流显然转化为最小割。即求 S 到 T 中边的和的最小值。由于总和一定，也就是求 S 中的和与 T 中的和两者之和的最大值。显然除了 s 和 t 外，其余点都在同一个集合中时其和最大。那么我们就是要 min｛s 到其他点距离之和，t 到其他点距离之和｝。

这个可以通过两次 dfs 进行树上 DP 来处理。求出所有的之后排序，从小到大，第 i 小的贡献为 n-I (其在 n-i 个点对中作为答案进行贡献）。总和会超过 long long,使用两个 long long 模拟一下大树即可。

树上 DP。先 DP 每个点到其子树所有点的距离之和。再 dp 每个点到其非子树中的点距离之和。二者求和即可。

树链剖分

2018 年 8 月 7 日
16:04

1、

Wannafly 挑战赛 21 E

题意：给出一棵以 1 为根节点的树。2 种操作，第一种 add u v val 将 u 到 v 的路径上经过的边权值都加 val

第二种 ask u 询问 u 的子树（含 u）所有路径（即两点都在子树中）的权值和

做法：

解法 1 考虑当我们询问点 x 的时候，x 的子树内每条边的贡献。对于在 x 的子树内的点 p，记 p 到父亲的边的权值为 a_p，则根据乘法原理，在这次询问中这条边被经过的次数为 $size_p \times (size_x - size_p)$ 即产生的贡献为

$$a_p \times size_p \times (size_x - size_p) = a_p \times size_p \times size_x - a_p \times size_p \times size_p$$

注意到 $size_x$ 是我们询问的时候已知的，因此每个询问就相当于在问这 2 个东西

$$\sum_{x \to p} a_p \times size_p$$

$$\sum_{x \to p} a_p \times size_p \times size_p$$

预处理每个点的 $size$ 值，对这棵树做轻重路径剖分，然后线段树维护即可解决询问。至于如何实现链加？轻重路径剖分之后这就已经是线段树套路了，不作赘述。稍微注意下根结点 $a_0 = 0$ 。时间复杂度 $O(N(log_2 N)^2)$

杂

2018 年 8 月 26 日
14:29

1、
BZOJ 3569
https://www.lydsy.com/JudgeOnline/problem.php?id=3569
题意：多次询问一个无向连通图当图中某 k 条边消失时这个图是否联通 强制在线
做法：若是可以离线，就可以采用 CDQ 分治的做法（CDQ 部分里有记录 其中的 BZOJ 3237）。这里强制在线，一般的处理方式显然是不可以的。这里采用的就是一种比较神奇的随机化权值方法。

首先，通过 DFS 找到这个连通图的一个生成树。
之后，遍历所有边，给不在生成树边的边随机一个权值。注意到，不在生成树中的边，一定是连接生成树中某两个不相邻的结点。
这样，非树边的权值就已经处理出来了。并且对每个结点，维护其连出的所有非树边的权值的异或和。
接下来考虑通过树形 DP，对树上的每个结点，维护该子树所有结点的上述权值的异或和。对于某个树上的结点，该结果表示的实际上是：将整个生成树划分为两个部分，由该子树连向另一部分的所有非树边的权值的异或和（对于两点都子树内的边，其权值异或了 2 次，结果即为 0）。并且对某条树边，给予其的权值设定为：深度较大的点的上述值。（树中某边连得两点深度一定不同）。那么，想要最终该子树与其余部分不连通，就必须要将该树边，与所有连出的非子树边全部选出。注意到：按照上述方式给树边权值，就等价于，某些边的权值异或和为 0。
因此，我们只需要判断选出的边集中，是否存在一个子集使得其中的边权异或和为 0。而这就可以通过线性基 O（32|S|）的处理出来。

字符串

后缀自动机

2018 年 7 月 29 日
17:23
前言：
最基础的 SAM 已经算是比较理解了。这里主要谈一下对广义后缀自动机的插入时每次都新建结点的认识。
在此之前，一直觉得如果 maxlen<minlen 就是不合理的状态。实际上，这只是一种冗余的，即使用多个结点表示同一个状态的处理方法。并且这些状态都会通过 slink，连续的串起来，亦即并不会影响在 SAM 上走，或者统计不同的子串个数。
之前一直认为如果已有 trans[u][c]并且 maxlen[trans[u][c]]==maxlen[u]+1 那么就直接使用（即返回）trans[u][c]就是更好的选择。
对此的例子是，先插入字符串"a",之后再插入"a".
但是实际上，像这样处理依旧可能会导致 maxlen<minlen 状况的产生。对于 maxlen[trans][c]]>maxlen[u]+1 的，就会把 trans[u][c]分解成 [minlen[trans[u][c]],maxlen[u]+1] 和余下长度，两部分，其中新建的点 z 会 slink 到前者。

而这样做，对于先插入 bbabba，再插入 a。就会产生矛盾。因为对于第一个串建立完 SAM 后，a,ba,bba 是在同一个状态的，并且是由空结点（0 也就是根）指向的。而之后再插第二个字符串时，maxlen[trans[u][c]]=3　而当时处在根节点，maxlen 显然为 0 而 0+1！=3。于是要把 trans[u][c]拆分成[1] 和[2,3]的两个长度的结点。之后再由长度为 1 的指向新建的单独的"a"的结点。就导致新建的单独的“a”这个结点，minlen 为 2，maxlen 为 1。

会导致上述这种情况产生，很大程度上是因为我们在 SAM 的插入过程中，存在默认“新插入一个字符后，由于字符串整体长度边长，取新的整个字符串，一定会导致新的状态产生”。而这一点在插入一个新的字符串，从根从头再来时就不成立了。通过比较复杂的判断，是可以使得所有相同状态用同一个结点表示的。但这就比较麻烦，实际上也没有必要。对于无脑新建结点这样的插入，虽然会产生 maxlen<minlen 的状况，但也使这样的点通过 slink 连了起来，并且只会有 maxlen+1=minlen 这一种情况。实际上对 SAM 的任何处理都不会有影响。

1、

BZOJ 3926

题意：

给定一棵树，每个节点有一个颜色，问树上有多少种子串（定义子串为某两个点上的路径），保证叶子节点数<=20。n<=10^5

做法：

直接摘录陈立杰的博客 其中关键就是要转化为某 Trie 树的一条直立的链。也就转化为了一个子串的问题。 以及 trie 建 SAM 时，通过 DFS 进行。

首先题目中有一个关键条件是说叶子的数量不超过 20 个。
我们不妨对每个叶子都以它为根建一个 Trie。
那么注意到整个树的任何一个子串，都是某个 Trie 上从一个点到它的一个子孙的路径。
那么，我们可以把这 20 个 Trie 合并成一个大 Trie，然后求这个大 Trie 的子串数量就可以了（Trie 的子串指的是从 Trie 中一个点到它的一个子孙）。
这可以使用比较经典的后缀自动机或者后缀数组实现。

来自 <http://wjmzbmr.com/archives/zjoi-2015-day-1%E9%A2%98%E8%A7%A3/>

2、

BZOJ 3473

题意：

给定 n 个字符串，询问每个字符串有多少子串（不包括空串）是所有 n 个字符串中至少 k 个字符串的子串？

做法：
建立广义后缀自动机。之后遍历每个串的前缀（即沿着串走），遍历每个位置的 slink，更新对应结点的属于的字符串数、上次被哪个字符串标识(防止 abab 这样的情况会两次更新 ab 属于的字符串数，产生问题)。之后对广义 SAM 拓扑排序。从根开始 dp。dp 每个结点对应的最长字符串，其“所有”（不止是这个结点的[minlen,maxlen]）后缀出现在>=k 个字符串的个数。（这里显然随着长度变短，就更容易出现在 k 个字符串中）。topo 排序后，由根向下 dp 即可。之后对每个字符串求解答案时，只要在 SAM 上沿着该字符串一直走，在每个结点加上该 dp 值即可（因为每个字符串的所有子串，都可以表示为其所有前缀的所有后缀，沿着字符串在 SAM 上走，走到的就是其所有前缀）。
注意这里一定要建立完整个广义后缀自动机之后再开始 dp。（对于这道题根据题意也只能这样做）因为建完之后 trans 才是稳定的。不然随着每次加，相同状态可能被多个结点表示。trans 是不稳定的，dp 也就是错误的。
并且沿着某个字符串，从头往下走，走到的某个结点，该节点的 maxlen 一定就是该字符串走到的前缀的长度。证明如下：因为能走到，所以 maxlen>=前缀的长度。如果该节点 maxlen>前缀的长度，说明 sx 和 s 都在该结点。其中 x 为 sx 的后缀。（s 表示当前的字符串的前缀，sx 表示该节点的 maxlen）。而因为当前是从该字符串的头走到这里的。那么显然 s 就比 sx 这个结尾的位置要多。与某个 SAM 定义中某节点表示的字符串，出现的次数相同，矛盾。
综上所述，maxlen=前缀的长度。
事实上，似乎可以证明，从 SAM 的根节点走某个字符串，只要一直都只通过 trans 走，没有 slink 过，走到的结点的 maxlen，一定与走的距离（字符串长度）相等。

DP

DP（1）

2018 年 7 月 29 日
17:23

1、

UVA live 6923 （Regionals 2014 :: Asia - Dhaka G）

题意：n 个人围成一个环。重新排列 n 个人。一个人是快乐的，当且仅当他新的位置与原来的位置的距离小于等于 1。求有多少种排列方式，使得至少 k 个人是快乐的
做法：首先，环一定是要拆成链的。拆成链后，对链进行 DP。保证能将链拼起来，就需要枚举起始状态（0）的状态，其需要与 n 的值相等（亦即加只加 n 时状态等于枚举的 0 的状态的）。Dp[i][j][k]表示，到第 i 个位置，有 j 个快乐， i 和 i+1 的状压为 k 的方案数。我们枚举的就是 dp[0][0][k]中的第三维 k，最后加的就是 dp[n][i][k] （必须起始枚举的与最终的相同才能“链接”起来）。对每种的结果，根据第二维乘对应的阶乘（不快乐的人全排列）求和，就得到一个暂时的“ans”数组。但是对这个数组还需要进行**容斥**。倒过来进行 ans[i]-=Σans[j]*C(j,i)。
这样得到的才是真正的 ans（这里为什么这样还不清楚）

状压 DP

2018 年 8 月 7 日
20:28

1、

CF gym 101666　G

题意：m 个人，给出 n 笔支付关系。问最少用多少次交易，就能代表所有支付。（即这些交易的结果，与进行这 n 笔支付的结果是相同的）
做法：这些交易之后，每个人都有净支出与净收入。注意到如果有 x 个人，它们的净结果和为 0，通过对它们建立一个树的结构就可以建立完其等价的支付。而树中只有 x-1 条边。于是就相当于我们想把这些人分成尽可能多组，每一组其净结果和都为 0。（注意到至少分成 1 组，即所有人）。那么答案就是 m-分的组数。求解这个使用状压 DP。（并且注意到划分为 s1、S2……之后，S1、S1∪S2、S1US2∪S3…的净结果和也全为 0，利用这个加速 DP，只需要 DP 前缀即可）。使用 DP[i]表示，i 压缩的人中，最多串成多少组（即下面定义的 L 函数，dp 方程图片中也展示了出来）

- For each set (bitmask) S, let $L(S)$ be the largest integer i such that we have a chain

$$\emptyset \subsetneq S_1 \subsetneq \cdots \subsetneq S_i \subseteq S$$

  with each S_i (but not necessarily S) having zero balance.
- Then for $S \neq \emptyset$,

$$L(S) = \max_{p \in S} L(S \setminus \{p\}) + \begin{cases} 1 & \text{if } S \text{ is zero-balance.} \\ 0 & \text{else.} \end{cases}$$

- Since $|S| \leq n$, this takes n steps to compute from previous results, which gives a $\mathcal{O}(n2^n)$ DP algorithm.
- The answer is given by $n - L(\{1, \ldots n\})$.

这里预处理 sum[i] (i 状态下净结果和）时，技巧是遍历 1 - （（1<<m）-1)的过程中，每一次对于 I 从 i^lowbit(i)递推。dp 时，就只能枚举 0——m-1 来看了。总复杂度 n*2^n

树形 DP

2018 年 8 月 8 日
10:07

1、
CF 461B
题意：给出一棵 n 个节点的无根树，每个节点要么是 0，要么是 1，现在要把树分成若干区域，要求每个节点都必须属于一个区域，而且一个区域中只能有一个节点为 1。问一共有多少种分法?
做法：
Dp[i][0] 表示 i 在其子树中，与其连通的部分不包含 1 的方案数
Dp[i][1]表示 i 在其子树中，与其连通的部分包含 1（且仅一个）的方案数
并且以上 DP 要保证其子树中的所有连通块都合法（即包含一个且仅一个 1）

初始时，每个点初始化为 dp[i][col[i]]=1 其中 col[i]表示 i 这个点的颜色。 最终所求的答案显然就是 dp[1][1]

DP 过程，利用了一个子树与整个树中非其子树的部分所连接的仅有其与其父亲连的这条边。

这样实际 DP 就很显然了。

遍历 i 的每一个儿子。出现在等号右侧的都是更新之前的 i 的状态，左侧为 i 的新的状态。设当前遍历的儿子为 s

即有

Dp[i][1]= dp[i][1]*(dp[s][1]+dp[s][0])+dp[i][0]*dp[s][1]

Dp[i][0]=dp[i][0]*(dp[s][0]+dp[s][1])

解释为： 已考虑一些儿子的状态。要转移的状态为 i 已在一个含 1 的连通块时。从之前 i 已在含 1 连通块的 dp 值转移时，该儿子在含 1 连通块的时候不连与其儿子的边，该儿子不在含 1 连通块时必须连其儿子的边。从之前 i 尚未在含 1 连通块的 dp 值转移时。其儿子必须在含 1 连通块里。且连这条边。Dp[i][0]类似的进行考虑。乘起来是因为乘法原理。

2、

HDU 4313

题意：给定 n（n<=100000）个节点的树，每条边有一定的权值，有 m 个点是危险的，现在想将树分成 m 块使得每块中恰好只有一个危险的点，问最小的花费是多少。

做法：

做法 1：树形 DP。DP[i][0]表示 i 所在的子树符合，并且 i 在的连通块内没有危险的点的最小花费。

DP[i][1]表示 i 所在的子树符合，并且 i 在的连通块内恰有 1 个危险的点的最小花费。

边界条件: 对于叶子节点 如果是危险结点 DP[i][0]=INF,DP[i][1]=0。 如果不是危险结点，DP[i][0]=DP[i][1]=0 (对于不含危险结点的连通块显然可以视为含危险结点来转移，不过设为 0 还是 INF 对答案都没有影响)

转移方程:

对于某点，如果该点为危险结点. DP[i][0]=INFF DP[i][1]=∑min(dp[son][0],dp[son][1]+edg.val)

如果该点非危险结点 DP[i][0]=∑min(dp[son][0],dp[son][1]+edg.val).

DP[i][1]=min{DP[i][0]-min(dp[son][0],dp[son][1]+edg.val)+ dp[son][1] } (相当于，某个地方连进来一个含危险结点的。就只需把原本隔绝的去掉，改成含危险结点的即可。)

做法 2：

贪心。把所有边从大到小排序。逐条往里插（初始只有点没有边），并查集维护所有连通块。当恰好加入这条边会导致两个危险结点连通时（即两侧连通块各有一个危险结点），就在答案里加上这条边的权值（这条边即为这两个危险结点的连通块路径上的最短的边）。其中利用了树的性质，树上的连通块只可能两两相交，不存在三个交在一起的。即这些连通块也构成了一个树的结构。

树形依赖背包

2018 年 9 月 12 日
19:20

1、zzq 博客中的题目（洛谷 P2014）
https://www.cnblogs.com/zzqsblog/p/5537440.html

有一个树，每一个节点代表一个物品，每个物品有重量和价值，每个物品必须先选父亲才能选自己。求给定重量内最大价值。（亦即过根的最大连通块的权值）

这题的思路十分的厉害。我们把树的 dfs 序建出来，对于 dfs 序上每一个点，我们考虑如果自己选那么自己子树内就可以选，否则只有在这棵子树外面才可以选。

那么我们记 f[i][j]为 dfs 序中第 i 个点及以后的 dfs 序大小为 j 的联通块的最大权值，所以我们可以得出 f[i][j]=max(f[i+1][j-w[i]]+v[st[i]],f[i+sz[st[i]]][j])类似这样。

来自 <https://www.cnblogs.com/zzqsblog/p/5537440.html>

其中最关键的是，建出 dfs 序后，i+sz[i]就就可以转移到下一个"兄弟"结点。

斯坦纳树

2018 年 9 月 13 日
21:43

发现 d 非常小，所以我们显然可以使用斯坦纳树来求解。

斯坦纳树是用来解决这种问题的：**给定若干关键点，求使得关键点连通的最小代价。**

我们可以**令 f[i][opt] 表示以 i 为根时，关键点连通态为 opt 的最小代价。**（以二进制表示是否连通）

然后我们就可以用两种方法来更新 f[i][opt]：

1. 设定集合 x,y，x∪y=opt 且 x∩y=∅，那么我们显然就可以将用 x,y 合并来更新 opt，

2. 若 f[j][opt] 中 opt = f[i][opt]中 opt，那么我们就可以以连边方式合并两个集合，这种合并方式显然可以用最短路实现，使得答案更优。

然后我们就可以求出所有状态的 f[i][opt]，接下来再利用 DP，求解。

定义 **Ans[opt]表示连通态为 opt 时最小代价**，如果对应点同时连通或不连通则可以更新，枚举所有情况就可以求出答案了。

来自 <https://www.cnblogs.com/BearChild/p/6522904.html>

按照自己当前的理解。
如上文所说，斯坦纳树解决的就是这样一类可以额外加点的生成树，或是最小生成森林的问题的 dp 方式。
其 dp 方程使用一个状压记录连通状态（因此点数不能太多），使用一维记录当前连通状态的树（森林）的根的位置。Dp[st][lo]表示当前根在 lo，连通状态**至少**为 st 的最少（最多）花费
通过两种方式转移：（似乎也可以理解为，只有这样转移的才叫做斯坦纳树）
1、dp[st][lo]=dp[st1][lo]+dp[st^st1][lo] （如果是点权和需要再减去 lo 这个点的权，避免该权值计算两次）
2、dp[st][lo]=dp[st][lo2]+dis[lo][lo2]

其中第二种转移，显然可以在通过第一种转移求解后使用 dij 或 spfa 多源（只需要最初放入队列中）最短路来求解。
状态数 2^d，对于每个状态遍历所有的 n，之后枚举该状态的所有子集。因此第一种转移的总复杂度为

n*(ΣC(d,i)*2^(d-i)) =n*(1+2)^d=n*3^d 其中 d 为要求连通的点数。（考虑每次最小的集合（被某状态包含的子集）枚举其 1 的个数，对于每个 1 的个数有 C（d,i）种取法，包含其的集合有 2^(d-i)个 ）
对于第二种转移，使用最短路，每一轮循环内复杂度为 O(Elog(V))
因此总复杂度 n*3^d+2^d* Elog(n)

1、BZOJ 2595

题意：n*m 方格，每个格子有权值（想要选择需要花费该点的权值），若干格子权值为 0，为需要互相连通的结点。求最小花费。
做法：几个点之间，两两考虑的最短距离情况比较复杂，显然不能用最小生成树处理。而想要连通的点个数较少，因此即求使这几个点连通的斯坦纳树。Dp[i][j][st]表示当前在（i，j），连通状态至少为 st 的最小花费。完全按照斯坦纳树的方式转移即可。

2、BZOJ 4774

题意：
对于边带权的无向图 G = (V, E)，请选择一些边，使得 1 <= i <= d， i 号节点和 n - i + 1 号节点可以通过选中的边连通，最小化选中的所有边的权值和。
做法：
DP 出包含 $2d$ 个点的斯坦纳树。
然后再做一次子集 DP，因为有可能不成对的点不联通。 （即最终的答案是一个生成森林）
$g[S]$表示点对联通性为 S 的最小代价。
$g[S]=\min(g[S],g[sub]+g[Sxorsub])$
答案是 $g[2^d-1]$

来自 <https://blog.csdn.net/qaq__qaq/article/details/64905780>

DP 套 DP

2018 年 9 月 15 日
13:37

1、HDU 5548

给定点数为 1~K 的麻将牌各 4 张（这 4 张完全相同），问有多少种方案，从中选出一个 M 张牌组成的集合，能够和牌。“和牌”指：其中有两张完全相同的将牌，其他牌可以被三三分组，每组要么是“n-1 n n+1”型，要么是“n n n”型。注意：集合相同而和法不同仅算一次。

来自 <https://blog.csdn.net/wmdcstdio/article/details/53236823>

做法：从小到大看各种麻将牌。注意到，在看到第 i 种时，数量能影响到 i 的只有 i-2,i-1。因此只是简单的考虑转移的话，显然应该用 dp[已选的（前 i-1 种）牌数][当前看到第 i 种牌][i-2 开头的组的数量][i-1 开头的组的数量][是否成对子]来转移。并且，因为选出的是用集合不同来计数，而单纯的转移会使得 3 3 3 这样子的被分别用 3 个 I I I

或 3 个 i-1 I i+1 重复计数。解决办法就是转移的数组中，每个的数量都<=2（即%3，每满 3 个就自己的 3 个一组）。

具体的转移是很显然的，只需要从小到大枚举单前的牌，再枚举看上一个牌结束时的状态，枚举当前选的牌数。很显然，其中比较难处理的就是“上一个牌结束时的状态”。

在固定 dp 前两维的情况下，容易发现后面 3 维总共 3*3*2=18 种情况。这样对于每个前两维，能达到的状态就可以用一个不超过 2^18 的数状压记录（事实上不考虑总牌数的限制，不考虑当前具体是哪一种牌的限制，单纯考虑转移（状态集合的转移，不超过 2^18 的数记录的是该节点包含的状态集合，要看能转移到哪些状态集合，不过额外的需要记录**该位选了多少张牌**，来明确具体的状态转移）就行，转移的起点是 1，即只有一个 0 0 0 即 i-2 i-1 的数量均为 0，没有对子的状压结果）。参考括号中的话，很显然总的情况实际上非常的少。使用 bfs 搜索一下，可以得到总共只有 68 种状态集合（即 68 个数）。这样的话对于固定的前两维，对应的就是一个 68*3 的二维转移。（每个前两维，有若干个可能的状态集合，每个状态集合，根据该位选择的数的个数（0——2）有固定的转移，这个转移在 BFS 的过程中就已经计算出来了）。于是整体就是两个转移的拼接。

最后，对于这道题来说，结果剩的 i 的个数都是 0，可以通过枚举到 n+3,对于超过 n 的时候，每一位最多取 0 个来去掉不合法的结果。

2、HDU 4899（未完全理解）

题意:

给你一个只由AGCT组成的字符串S (|S| ≤ 15)，对于每个0 ≤ .. ≤ |S|，问
有多少个只由AGCT组成的长度为m(1 ≤ m ≤ 1000)的字符串T，使得$LCS(T,S)=i$?

几个比较好的博客：

https://www.cnblogs.com/candy99/p/6854038.html
https://blog.csdn.net/qpswwww/article/details/43882561

https://blog.csdn.net/gauss_acm/article/details/52840679
https://www.cnblogs.com/owenyu/p/6724616.html
个人理解：
正常 dp 的 lcs 做法是：dp[i][j]分别记录在 s 和 t 中看到了哪个下标。如果 s[i]==t[j]，则 dp[i][j]额外可能取 dp[i-1][j-1]+1，不然 dp[i][j]就是 max{dp[i-1][j],dp[i][j-1]}。于是，在使用一个额外的数组 d 记录 s 中前缀中最长的 lcs 后，对 d 差分，就变成了一个 01 序列。每个 1 表示 dp 到这个位置会+1。由于总长只有 15，因此这个状态数是 2^15。这个状态之间是通过每次加一个数来进行转移。可以预处理出这些状态间的转移。
之后对于原问题，dp[i][j]表示到第 i 个位置，状态为 j 的情况数。第一维可以滚动优化，直接按照上述转移进行即可。

其中Bi=max (B(i-1),Ai)的含义是什么呢？ Bi=max(Bi,1+A(i-1))呢?

B(i)表示T[1...j+1]与S[1...i]的LCS

B(i-1)表示T[1...j+1]与S[1...i-1]的LCS，Ai表示T[1...j]与S[1...i]的LCS，1+A(i-1)表示T[1...j]与S[1...i-1]的LCS+1

有没有发现最经典的LCS转移方程：

dp(i,j)=max(dp(i,j-1),dp(i-1,j)) s.t. S[i]!=T[j]

dp(i,j)=dp(i-1,j-1)+1 s.t. S[i]==T[j]

只是转移方程中j被j+1替换而已

网络流

最小费用最大流

2018 年 8 月 27 日
0:21

1、

HDU 6445 （2018CCPC 网络赛 1008）
题意：
给出 n 个点的竞赛图，其中有些点对(I,j)之间的边未连。要求你将这些边的方向确定下来，使得：
对四元组 (a,b,c,d) 有 a->b=b->c=c->d=d->a 就 ans++
若 a->b!=b->c && b->c!=c->d && c->d!=d->a 就 ans--

要求最大化最终的 ans。并输出对应的结果
做法：
这个题目实际上和 BZOJ 2597 基本上一样。
引用那个题的题解：
第一题：WC2007 石头剪刀布

这题是网络流，简直太难想到了，总之这道题有几个非常巧妙的转化。

首先是补集转化，我要统计一个完全有向图中的三元环的个数，就等于所有环的个数-非三元环的个数。非三元环的个数怎么统计呢？观察三元环的特点，发现一个三元环必然有一个点在这个环中入度为二（出度为二），由此我们便可以得到一个计算三元环的公式（设 d[i]为 i 的入度）

$C(3,N)-\sum C(2,d[i])$

$=n*(n-1)*(n-2)-\sum d[i] * (d[i]-1)/2$

$=n*(n-1)*(n-2)-\sum (d[i]^2+d[i])/2$

$=n*(n-1)*(n-2)-(1/2)*\sum d[i]^2 +m/2;$

其中（$n*(n-1)*(n-2)+m/2$）皆为常数，不影响最大化（$C(3,N)-\sum C(2,d[i])$），那么问题就转化成了最小化$\sum d[i]^2$了。

这个平方应该怎么得到？标准做法非常 NB，对于每条未定向边当做一个点，这个点向他的两个端点连边，而平方的费用怎么办？

拆边！对于一条容量为 C 的边，费用为 $flow^2*k$，那么我就把它拆为 C 条容量为一的边，费用分别为 k，3*k，5*k.....（2*c-1）k！

这道题貌似还要用到单位容量网络流做，等会在学把。

来自 <https://blog.csdn.net/jasonzhu8/article/details/5963048>

而对于这道题，实际上也是差不多的。最终可以转化为 A(n,4) - (ΣC(deg,2)) *(n-3)*8。（注意到所有四元组总个数为 A(n,4）,答案+1 的四元组不含出度为 1 的点（这里出度都是按这 4 个点内部来算），要答案-1 的四元组含 2 个出度为 2 的点。其余的

情况都只含 1 个出度为 2 的点。除了 A(n,4) 表示所有情况外的因式，都表示：对于固定的四元组（就是选定了 abcd 以及其翻转 or 旋转顺序的变换），总共有 8 种，所以乘 8。对于不+1 的，只包含了一个出度为 2 的点，也只被减了 1 次，恰好在答案中去掉这些情况。对于要-1 的，减了 2 次，恰好表示了-1 的操作。
这样之后就转化为了最小化平方和的问题。与 BZOJ 的这道题几乎一样。最小费用最大流即可。

题目记录

40 题

2018 年 3 月 2 日
17:41

1、
CF934　（round462 div2）E　A Colourful Prospect
利用欧拉公式
　f=e-v+2　/ 面=边-点+2
对于此题 还需要考虑连通块　即 f=e-v+c+1　 c 为连通块个数
图中的点和边 只要考虑交点即可　（边=2*点　单独的圆视为无点无边）

2、
Wannafly 交流赛 1 E 直径
https://www.nowcoder.com/acm/contest/69/E
关于 DFS 的一个好题。
求树上前 k 大距离（不同点对距离相等均计算）
代码：
https://paste.ubuntu.com/p/7MZ9yNxNkG/

3、
HDU4283 区间 DP
HDU2476 区间 DP　都需要加深理解　未完全理解

4、

UOJ 370 （Round #17 第一题）
非常好的 DP（很难想到与想清楚，前几组数据是在提示某错误的贪心想法，但可以用此剪枝）

当前状态为 now 想转移到到 st 通过与运算 需要一个 一定包含 now^st 中的 1 并且可以包含 now 的二进制 1 以外的 1 的数 即 st 中的 1 包含于该数不包含的子集
只要满足这个条件就一定可以转移到 st 甚至比 st 更好的状态 无需管 st 是否为最优
一直 dp 下去即可得到最优解
题解

5、
CF 954H
树形 DP 计数 注意 b\c 数组均为 “选择无关” （即该层选择的结点无关，故需在外乘组合数）

6、
牛客练习赛 14 B
二进制形式的 DP，记录每个点作为终点（or 起点）连续 2^l 个 k 的序列后最左（右）到的区间的下一个位置 这样就可以依 2 的幂次由小到大进行递推
最后求解答案时只需要从 2 的幂次从小到大转移即可

7、
牛客练习赛 14 E
点数<=1000 故最大距离也<=999 可以考虑 bitset 做
用数组存储距某点距离<=x 的点集（通过前缀或做即可 事实证明 bitset 总共 1e9 也是没有关系的 170000 左右 不会 MLE）
这样每次答案只需要一直或下去 再求当前 bitset 中 1 的个数即可
记录这个题是为了记录 bitset 这一几乎没有用过的 stl 数据结构

8、
华理 2018 校赛 K
很容易化式子为 ∑(aiy-bix)=0 鉴于 n 最大=35 将 n 折半 dfs 寻找每一半的可能和先做的一半用 map 记录 得数与可能的情况的值 这样第二次就可以 O(1)查询对答案的贡献了 这样做就达到了对所有子集的遍历（包括空集） 即完成目的

9、

SPOJ-LCS2

对于 SAM 来说，实在是太经典的一道题。topo 序从根到顶更新，e.g.某结点匹配长度 len (len 必定在[minlen,maxlen]之间 ） 则其父节点的匹配长度一定为 maxlen 利用此进行更新。

10、

ARC-094

将题解中的两种做法均进行了实现，对应两个 AC 的提交。

第一种构造的方法实在是巧妙。

第二种的策略是由于显然是各取尽可能小的，且在和一定的情况下，两数越接近积越大。二分答案，检查对应的中间若干个（只取 10 个就已经够了）的积即可。

11、

Wannafly 挑战赛 14-E

来自 <https://www.nowcoder.com/acm/contest/vip-index#list>

利用线性基，使用并查集维护每个“区间”的线性基，合并时只需要合并线性基即可。（学到了线性基）

12、

AGC023 -C

恰好 t 次操作时全部染黑的方案数=（<=t 次操作全部染黑的方案数）-（<=t-1 次操作全部染黑的方案数）

对于这道题，<=的方案数比较好计算（1\n-1 必取，中间 ai 之间差只能为 1 或 2，利用母函数很容易得到组合数公式）

先(n-1)!*(n-1)作为答案（假设每种排列都是 n-1），再从 n-1 逐个递减，一层层减去不合条件的即得到答案。

13、

Codeforces 980E

题意：给定一棵 n 个点的树，点 i 的权值为 2^i，去掉 k(1<=k<n)个点，使余下的点连通，且余下的权值最大，按递增顺序输出去掉的点

做法 1： BIT+DFS 序（learn from quailty）

显然，无论如何,n 这个点都到取（二进制大小的性质）。考虑最终要剩下的点，而非要去掉哪些。每次贪心的看当前最大的点是否可以选择。
这只需要维护每个点的"深度",这里深度表示包含这个点，需要向上经过多少个未标记的点才能达到之前已经被选择过的区域。（即当下为了选择该点需要付出的代价）。
利用 DFS 序使得任意子树 dfs 编号相邻的性质，使用 BIT 进行维护。初始时 in[i] +=dep in[i]+=-dep。之后选择某个点后，不断向根跳的过程中，每次将该结点对应的子树 dep--（做法为 in[i] += -1 out[i]+1 +=1）注意这里的+全是对 BIT 进行的。
做法 2：树链剖分+线段树（learn from zlc1114）
（我的第一道树链剖分）不同于 DFS 序，树链剖分 id 相邻的是同一条链上的点。利用线段树维护链上点之间的距离（点距离，即两点的路径上包含该两点有多少点）。……突然意识到，实际上就是在求 LCA 啊……但不同的是，每次选择某点，从该点往上需要 update 来去掉一些点。整体上就是求 LCA 的过程中顺便通过树链维护距离，这一点是倍增求 LCA 难以做到的。关于复杂度：树链剖分自带 log，使用线段树一个 log，总体 2 个 log

14、（未完全清楚）
Codeforces 204E
做法为广义后缀自动机。等价于将一个 Trie 压入后缀自动机，使得 Trie 上每个子串都可以在后缀自动机中找到。
对于本题，使用的方法为每次从 SAM 的起始节点开始加入每个串。在不考虑重复多余结点的时候，在一个 SAM 中进行字符串个数次插入后，一定可以保证后缀自动机中包含所有子串的状态。需要考虑的就是尽可能减少无意义结点的出现。而这只有在（设当前时刻 SAM 位置为 last）trans[last][c]已存在，且 maxlen[trans[last][c]]=maxlen[last]+1 时，才可以节省，此时即返回 trans[last][c]。为了证明这一点，先证明对于这种情况可以不加新的状态点。
即只需证明二者的 right 集合相同。Trans[last][c]每次出现，其前缀（除了 c 余下的部分）长度均至少为 maxlen[last]，即一定与当前插入状态的前缀相同。故当前新加的串与之一一对应，该 right 集合可以直接扩充这一新的 right。再考虑 maxlen[trans[last][c]]>maxlen[last]+1 的，而当前前缀并不保证为 maxlen[trans[last][c]]的前缀，right 集合因为这个新加进来的字符串，两者发生了变化。故加进来需要创建新的状态，且对应了原 SAM 中第三种情况的建图。
（写完自己都不知所云……等完全想清楚再来吧 orz）

15、
Poj2096
学到的：

1、期望可以通过概率进行转移
2、转移期望时，很可能可以转移到本状态，此时通过处理这种情况的期望步数，直接加入结果，并且对其他概率的求解中减去此种情况

16、

HDU6035

树形 DP。之前一直不知道树形 DP 是什么，今日方知原来原先也是写过几道树形 DP 的。

dfs 第一遍求子树结点个数。以任意时刻根节点的颜色进行考虑。一种是不跨越根节点，一种是跨越根节点仍在根节点的子树中，一种是跨越根节点到其余结点。

17、

Oj4th1311（2018 集训选拔 D）

CXIV、DXIV 没有公共前缀，但注意到他们有公共后缀，故可将字符串倒过来数。实际上状态只有 12 种 *VIX*VIX*VIX 而非自己所想的 2 的幂次种。关于状态的转移以及状态的抽象还是需要多想多练啊。

18、

Oj4th1305 （2018 集训选拔 B）

（以下用×表示未中奖，用〇表示中奖）注意到每一列要么全为×，要么出现〇才可中止该列。那么我们就可以将每一列转化为一个格子，用概率和对应的得分就能表示该格子的状态（全为×的概率以及第一个〇出现需要的次数的期望，以及相应概率）。那么原问题就变为了一个 1*n 的问题。并且等价于：给 n 对(p,e)
自己排列其顺序，使得 e1+p1e2+p1p2e3+p1p2p3e4+……+ p1p2……pn-1en 最小。只需要考虑每一对点对的位置关系，找到更优的判断方法就可以将其进行排序得出结果。
A+p0(eA+pA(eB+pB*G)) < B+p0(eB+pB(eA+pA*G))。消完剩下的就是 eA-eB+pAeB-pBeA<0，使用此进行排序即可。

19.

BZOJ3757 苹果树

求树上两点间路径上不同点的个数（完全裸的题是 SPOJ-COT2 BZOJ 这个题还有一些别的要求 虽然也不多 spoj 的提交见 VJ）

裸的树上莫队，似乎很神的一种做法。。

首先将树分块，块大小 B=sqrt(n)；

具体树的分块方法见大爷的博客：手把手教你块状树系列（即 BZOJ1086）

分块之后将询问排序，第一关键字按 l 结点所在块，第二关键字按 r 结点的 DFS 序；

然后处理如何从 l 到 r 的路径转移到 l'到 r 的路径；

首先莫队算法其实是对一个集合的维护，记录的数组是 l 到 r 路径上点的集合；

那么根据 VFleaKing 的证明，我们可以维护 l 到 r 路径上不包括 LCA(l,r)的点集；

转移直接将 l 到 l'路径上的所有点是否在集合中的状态取反就可以了；

时间复杂度和区间上的复杂度类似，都是 O(n√n)；

但是常数应该是大一些，因为块状树要深搜，LCA 要一个 log(这个是倍增，似乎不能用 tarjan 解决吧)

来自 <https://blog.csdn.net/ww140142/article/details/47315437>

VFleaKing 的证明：

用 S(v, u)代表 v 到 u 的路径上的结点的集合。

用 root 来代表根结点，用 lca(v, u)来代表 v、u 的最近公共祖先。

那么

S(v, u) = S(root, v) xor S(root, u) xor lca(v, u)

其中 xor 是集合的对称差。

简单来说就是节点出现两次消掉。

lca 很讨厌，于是再定义

T(v, u) = S(root, v) xor S(root, u)

观察将 curV 移动到 targetV 前后 T(curV, curU)变化：

T(curV, curU) = S(root, curV) xor S(root, curU)

T(targetV, curU) = S(root, targetV) xor S(root, curU)

取对称差：

T(curV, curU) xor T(targetV, curU)= (S(root, curV) xor S(root, curU)) xor (S(root, targetV) xor S(root, curU))

由于对称差的交换律、结合律：

T(curV, curU) xor T(targetV, curU)= S(root, curV) xorS(root, targetV)

两边同时 xor T(curV, curU)：
T(targetV, curU)= T(curV, curU) xor S(root, curV) xor S(root, targetV)
发现最后两项很爽……哇哈哈
T(targetV, curU)= T(curV, curU) xor T(curV, targetV)
（有公式恐惧症的不要走啊 T_T）
也就是说，更新的时候，xor T(curV, targetV)就行了。
即，对 curV 到 targetV 路径(除开 lca(curV, targetV))上的结点，将它们的存在性取反即可。
”
——vfk 博客

来自 <https://www.cnblogs.com/MashiroSky/p/5914599.html>

代码：链接

20.

HDU5840
分块？+树链剖分（提交在 HDU 上）
引用 qscqesze 的题解
“
首先对于每个询问我们可以拆开成为两个询问，一个是查询(u,lca(u,v))链上,deep[i]%k=a 的最大值，一个是查询(v,lca(u,v))链上，deep[i]%k=b 的最大值，就都变成了从 deep 值大的位置跑到 deep 值小的位置了。
然后根据 K 的大小来做这道题，block 是自己设定的一个值，一般为 sqrt(n)。
假设 K>=block，这一部分直接暴力去做就好了，把所有询问都离线下来，按照 dfs 顺序，拿一个栈维护一下从根到这个点经过了哪些点，这个点减去 k，就表示这个点往上跳了 k 步，直到跳到这个询问记录的 lca 位置。
这个的复杂度是 n + qsqrt(n)
假设 K<block，把询问按照 k 分类。
对于每一种 k，我们考虑树链剖分，把树链剖分 hash 下来的坐标，我们按照 deep%k 值分类，相同的值放在一起。那么我们相当于把这棵树拆成了 k 棵树，每棵树里面储存的节点 deep 值%k 的值都相同。
然后题目的询问，其实就变成了很普通的查询链上最大值了。
这个复杂度是 nsqrt(n)+qlognlogn
当然这道题还有很多种做法的样子，我只是抛砖引玉的讲了其中一种方法。

来自 <https://post.icpc.camp/d/518-ccpc-2016-online-1009-this-world-need-more-zhu/2>

”

比较关键的就是利用树链剖分后链上点的新下标连续。就实现了快速向上移动 k 步，以及方便于用线段树维护区间最大值

21.

Cf 620F

Trie 上莫队（网上找不到这种标程做法的中文题解），提交代码见 notexist 帐号的提交。

题意：

定义 f(u,v) (u<=v)= u^(u+1)^(u+2)^……^v　给定一个序列和若干询问，每次询问[l,r]区间内最大的一对 a[i] a[j] 使得 f(a[i],a[j]) (取 a[i]<=a[j])最大，其中 i 可以等于 j，输出结果。

有一种可以水过去的暴力方法，网上的这种题解到处都是，唯独缺了标程 Trie+莫队的题解。

标程的处理方法实在是非常巧妙。首先显然可以转化为两个前缀异或和。这样，每次在已经加了一些数的时候，求解当前最大值，就相当于是在一个 01 字典树中尽可能“反向”走这一经典问题。而在莫队中，为了达到 O（n^3/2）的复杂度，增删必须在常数时间内完成，如果像之前做的莫队题目中一样，每次都是完全处理完一个询问再去处理下一个，由于在这个问题中对字典树的更新必须是连续的增后连续的删完，就会导致复杂度爆炸。

为此，采用以下策略：

依然是分块，同一块里的一起处理，并且同一块的按其询问区间的右值由小到大排列。

注意到 a[i]、a[j]这样的（i，j）对按块的右端点划分只有 3 种情况：

1、全在块内

2、一个在块内、一个在块外

3、全在块外

注意到反正每次对某次询问结果的计算只是进行若干次取 max 操作，所以我们只需要保证最后每个区间结果值都被检测过，而不需要连续的处理完一次询问。

通过按询问区间由小到大进行处理，将块区间外的点依次加入字典树（这里再提一下，注意要两个字典树，在字典树上的路径一样，但值不同，因为做异或和区间的头和尾带进去的值是不一样的，并且两棵字典树也要分别维护到达某个结点的最大值、最小值），每次加入后就查询以新加的点为端点的值，更新“最大值”。通过这样就可以 nsqrt(n)的处理完 3 这种情况（一个块其块外右端点最多移动 O（n），更新字典树和答案是 O（1）的，共 sqrt（n）块）。并且可以在遍历块内询问时，对于某次询问，当右端点移动到此时的右侧尽头时，遍历询问区间在块内的部分，尝试更新答

案，这里不加入字典树，每次直接单次查询。这样在块内移动为 O（n^1/2），总共有 O（n）的询问，复杂度也是 O（n^3/2），也就完成了 2 号情况。
之后清空字典树，在此遍历块的询问，但这次[l,r]只取在块内的部分，很容易的 o（n^1/2）就可以把所有点加入字典树，并更新答案。总共 o（n）个查询，故也是 O(n^3/2)
这样就完成了这个问题。

22.
CF633H
利用斐波那契数列的 F[a+b]=F[a+1][b]+F[a][b-1]=F[a][b+1]+F[a-1][b] 这一性质，这样用线段树维护两个数列，对于莫队每次单点更新复杂度 o（log（n））
故整体复杂度 O（n^3/2 log(n)）
（自己交的代码维护的是 a[i]F[i] 和 a[i]F[i-1]序列 实际上维护 a[i]F[i] 和 a[i]F[i+1] 会更好一些 因为使用到的 fib 数列下标全都>=0 注意扩展 Fib 序列的话 1 1 2 之前的序列应为 1 0 即必须保证加上扩展的序列后仍然有 F[i]=F[i-1]+F[i-2]的递推）

23.
CF 785E
题意：初始 n 个数，a[i]=i，给定若干次交换操作，每次交换下标为 x、y 的两个数。对于每次操作之后的结果输出操作完的逆序对数。
做法：对于每次输出结果，只需要边更新，边维护答案的变化即可求得。注意到实际上造成的影响，只可能在两个数之间的区间产生。并且需要快速求解区间内比某数小的数的个数。考虑分块，维护每个块内数从小到大排的序列。这样每次查询完整的一个块只需要 log 就可以完成。而非完整查询的一个块也只需要 sqrt(n)即可完成。更新维护每个块的从小到大序列也只需要一个 log。综上，整体复杂度为 qsqrt(n)log

24.
ZOJ 2112
题意：动态求第 k 小。
做法：对于初始的状态，静态建主席树。对于之后的修改使用树状数组维护，树状数组的每个结点记录新建的主席树的根结点。这样在树上二分时，只需要把树状数组的区间和也考虑进去即可。注意两次更新树状数组使用的结点的过程，可能会重复使某点向左子走，或向右子走。而这实际上是没有问题的，因为走两次说明求前缀和时都分别加了进去，那么二者做差的时候就被减去了。
代码

25.
HDU 3507

斜率优化第一题。

代码　需注意与博客 2 中代码的区别为加入了两点 x 坐标相同时返回斜率无穷大

参考博客：博客 1（数学分析法，要注意其斜率分析全反了）博客 2（数形结合法，没系统看完）

26.

HDU 4417

题意：询问区间<=h 的数的个数

做法：建立主席树，只需要查找两个版本的主席树之差 在<=x 范围内的和 这里 x 都是离散化后的，很容易用 upper_bound 求出。需要注意的是，有可能 x=0，即比原范围内最小的还小，遇到这种情况如果还 l==r 就返回当前节点的和，就容易出错

27.

BZOJ 2286

题意：给定一棵树，每条边都有一定权值。若干次询问，每次给出 m 个点，要求去掉权值和最小的一些边，使得 1 与这些点不连通

做法：总点数有 250000，单次树形 DP 复杂度 O（n），但是询问次数较多，无法每次树形 DP。不过注意到询问的所有点总共只有 5e5，考虑每次建立虚树。

个人通过这道题理解的虚树就是所有求解的点以及它们的 LCA 组成的树。这棵树满足其上的点数不多，同时却又能大致描述出整棵树的样子。建立虚树的过程是按 DFS 序从小到大往里加点，按从跟结点出发的各条路径依次建立。参考博客 1：对于这道题这个博客讲的较好。

对于这道题，每个点维护从根节点（1）到这个点路径上的最小边权。代码中采用的递推方式只有在如博客中提到的那种情况才会出现问题，采用博客中提到的方法进行预处理即可。（代码中也是如此处理的，也就是当 u、v 作为询问的点对时，若 lca(u,v) =u 则去掉 u，只保留 v 稍微考虑一下就会知道，在这种情况下 v 就是多余的 代码链接

28.

2018CODEM 复赛 E

题意：给定一棵树，任意不同两点 u、v 都有一个人从 u 往 v 走，一个从 v 往 u 走（树上路径唯一）。给出 m 个点对(x,y)，（对于从 u 走到 v）如果 u 到 v 的路径同时经过点对中的两点，就会无法走到 v，问有多少个人能走到目的地

做法：首先计算 dfn。实际上一对点对只有两种情况：（设两者 lca 为 c）

1、其中一个就是 c：这是较为复杂的情况。画图可得，为使路径必同时经过两点，则路径起点、终点的范围分别只能是[1,in[u]-1]∪[out[a]+1,n] 和 [in[b],out[b]]

2、两者的 lca 与两者均不相同：这时显然起点终点必须一个是[in[u],out[u]] 一个是[in[v],out[v]]
而计算的方法是：按 dfn 大小由小到大遍历，上述的限制等价于在遍历到某点时更新一定区间的可行性，可以使用线段树进行维护（这个地方的处理比较巧妙，多用一个数组记录某个区间是否一定全可行，也就是更新时整块更新。如果整个可行直接就是 r-l+1,不然的话就是 s[lson]+s[rson]（因为只可以置一个区间为可行，不可以置一个区间为不可行，实际上-1 时置的是一个区间并不是全可行，因为有许多个点对，我们要考虑的是他们的并）
代码链接

29、

2018CODEM 复赛 D
题意很明确，最终的解为使用整个图(I,a[i])的下凸包时的答案。
原因尚不清楚，待补。

30、

2018CODEM 复赛 C
题意：n 个点，m 条边，每条边 0、1 或-1（待填），求有多少种方案使得给-1 赋值后，所有的环边的异或和为 0
首先有结论：一个联通图，其中有 n 个待填的边，则方案数为 2^(n-1)种。（稍稍想一想即知道了）
这样就

31、

BZOJ 1468（POJ 1741）
借用博客：
例题：POJ-1741

题意：一棵有 n 个节点的树，每条边有个权值代表相邻 2 个点的距离，要求求出所有距离不超过 k 的点对(u,v)

点分治算法的过程：

1)找出树的重心

①计算以 u 为根的树中每棵子树的大小

②根据子树大小找出树的重心 root(以树的重心为根的树，可以使其根的子树中节点最多的子树的节点最少)

2)将树的重心作为根节点 root,计算树中每个点到 root 的距离 dir

3)计算树中所有满足 dir[u]+dir[v]<=k 的点对数 cnt1

4)计算以 root 的子节点为根的子树中，满足 dir[u]+dir[v]<=k 的点对数 cnt2

5)ans+=cnt1-cnt2

6)删掉节点 root，分别遍历 root 的子树，回到第 1)步

题解链接：POJ-1741

来自 <https://blog.csdn.net/qq_31759205/article/details/75579558>

32、

HDU 4812　代码链接

题意：给定一棵树，每个点有权值，求字典序最小的点对(I,j)满足 i、j 路径上点的积为 k (模 MOD 意义下)

做法：点分治。对于一棵树，先找到其重心(O(n))。标记 vi 为 true，封锁其儿子结点的路径。使用 map（这里数据范围小，就数组就行）记录 mp[i]表示 从根节点到某点满足路径点积为 i 的字典序最小的点的编号（如果没有这样的点，mp[i]=0）。更新顺序如下：逐个 dfs 其儿子，对于某个儿子，对从其开始的所有链进行检查（map 有无与其积恰为 k 的），之后把这个根节点加进去，再遍历该儿子对应的子树一遍，这时每一条链都是从根开始的，用这些链更新 map。这样就保证了经过该点的都被统计到。不经过该点的路径只能全部在其子树中。再分别分治其儿子对应的子树即可。

33、 (2-SAT 水题口头 AC)

首先 2-sat 这样的检验似乎都可以转换为 tarjan（反正本来也都差不多 orz）以下是各个水题

①HDU 3622

题意：每对炸弹必须选一个引爆，引爆构成一个圆，使所有的圆不相交，求圆的最大半径。

做法：最大化最小值的问题，采取二分答案。每个答案分别建图进行 2-SAT。建图方法为：2*i 表示第 i 对选择前一个圆，2*i+1 表示第 i 对选择后一个圆。每次 O(n^2)建图，e.g. 2i 和 2j 如果这两个圆相交了，则连边(2i,2j+1) (2j,2i+1) （意义为：选择了 2i，则只能选择 2j+1 选择了 2j 则只能选择 2i+1）如果不相交就可以任选，那么就不用连边 两个圆必须且只能选 1 个，无需体现在建图过程中

②HDU 3715

题意：给定长度为 m 的数组 a\b\c,其范围为[0,n-1),自己分配 x[i]的值，使满足 x[a[i]]+x[b[i]]!=c[i] 的 i 的最大值（要求从 0 开始连续到 i 的都不可以）最大。求解是几。

做法：二分答案，对前缀建 sat 方法为：c[i]=0，2*a[i]指向 2*b[i]+1，2*b[i]指向 2*a[i]+1 若 c[i]=1 2*a[i]指向 2*b[i],2*b[i]指向 2*a[i] 2*a[i]+1 指向 2*b[i]+1 2*b[i]+1 指向 2*a[i]+1 若 c[i]=2 2*a[i]+1 指向 2*b[i] 2*b[i]+1 指向 2*a[i] 如此建图检验即可

附（网上对 SAT 建图的一个总结）：

AND 结果为1：建边 ~x->x,~y->y （两个数必须全为1）

AND 结果为0：建边 y->~x,x->~y （两个数至少有一个为0）

OR 结果为1：建边 ~x->y,~y->x （两个数至少有一个为1）

OR 结果为0：建边 x->~x,y->~y （两个数必须全为0）

XOR 结果为1：建边 x->~y,y->~x,~y->x,~x->y （两个数必须不同）

XOR 结果为0：建边 x->y,y->x,~x->~y,~y->~x （两个数必须相同）

③HDU 1815

题意：n 个点，每个点可以跟 S1 或者 S2 相连，两类约束条件，①a，b 不能与同一个 S 相连 ② a,b 必须与同一个相连，问满足条件的每两个点之间的距离的最大值 最小是多少

其中，给出的是每个点坐标，先算距离，距离是曼哈顿距离

分析：二分枚举答案，每次重新建图，边分三类

A 表示点 a 连 S1，A’表示点 a 连 S2

① a,b 不能与同一个 S 相连,加边 A->B',A'->B,B->A',B'->A;

② a,b 必须与同一个 S 相连,加边 A->B,A'->B',B->A,B'->A';

③ a 连 S1，b 连 S1 距离和大于当前枚举的最大距离 ，加边 A->B',B->A';

a 连 S2，b 连 S2 距离和大于当前枚举的最大距离 ，加边 A'->B,B'->A;

a 连 S1，b 连 S2 距离和大于当前枚举的最大距离 ，加边 A->B,B'->A';

a 连 S2，b 连 S1 距离和大于当前枚举的最大距离 ，加边 A'->B',B->A;

*特别注意一下给的约束条件下标从 1 开始

④HDU 1816

题意：n 对钥匙,m 扇门,每扇门有 2 个钥匙可以打开，钥匙对(a,b)表示 ab 不能共存,门上的钥匙(a,b)表示 a，b 必须存在一个,求最多能打开多少扇门，必须打开以前扇门才能开下一扇

分析：二分枚举能打开的门数,每次根据钥匙对和当前能打开的门建图

A 表示选钥匙 a,A'表示不选钥匙 a

①钥匙对 a,b,建边 A->B',B->A';

②同一扇门上的钥匙 a,b,建边 A'->B,B'->A;

⑤HDU 1814

题意：共和国要组建委员会，现在有 N 个党派，每个党派中有两人，而且有且只有一人能参加委员会，但是现在不同党派间有些人有矛盾，因此这些人是不能同时属于委员会的。现在要你求出每个党派的派出人选，如果有多组解，则输出字典序最小的那组解。

做法：

2–SAT 问题。有 N 个党派相当于 N 个变量，每个变量的取值只能 2 选 1。如果两个人相互矛盾，如果第一个党派的第一个人和第二个党派的第一个人有矛盾，那么就从第一个党派的第一个人向第二个党派的第二个人连边，表示，选了这边的一个人就一定要选那边的一个人。然后就跑 2-SAT 算法，这里我们采用暴力染色的方式，因为这样可以保证字典序最小。 （即默认使用的模版就已经保证是字典序最小的了）

⑥HDU 4115

【题目大意】

Bob 和 Alice 玩剪刀石头布，一个玩 n 轮，Alice 已经知道了 Bob 每次要出什么，1 代表剪刀，2 代表石头，3 代表布，然后 Bob 对 Alice 作出了一些限制：

给 m 行，每行是 a b k，如果 k 是 0，表示 Alice 第 a 次和 b 次出的拳必须相同，如果 k 是 1，表示 Alice 第 a 次和 b 次出的拳必须不相同。

一但 Alice 破坏了这个限制规则，或者输了一局，那么 Alice 就彻底输了。

问 Alice 可不可能赢？

来自 <https://blog.csdn.net/shuangde800/article/details/8875505>

【分析】

因为 Alice 一次都不能输，所以根据 Bob 出的拳，Alice 只可以赢或者平局，即每次有两种选择，是 2-SAT 模型

然后会有一些矛盾对，假设第 a 次可以出 a1，a2， 第 b 次可以出 b1 和 b2

如果第 a 次和 b 次要求相同， 但是 a1 和 b1 不同，说明这个矛盾，建立连接 a1—>b2, b1—>a2

同理，第 a 次和 b 次要求不相同，但是 a1 和 b2 相同，说明这个矛盾，建立链接 a1—>b1, b2—>a2

……

然后用 2-SAT 判断即可 (这里的比较是 a1、a2 与 b1、b2 的 2*2 的组合都进行比较 所以不要以为这种连边有问题 这样的话 e.g. 一个是石头 一个是剪刀 要求相同 连边时就已经有环了！ 详见代码)

来自 <https://blog.csdn.net/shuangde800/article/details/8875505>

⑦HDU 4421

```
void calculate(int a[N], int b[N][N]) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            if (i == j) b[i][j] = 0;
            else if (i % 2 == 1 && j % 2 == 1) b[i][j] = a[i] | a[j];
            else if (i % 2 == 0 && j % 2 == 0) b[i][j] = a[i] & a[j];
            else b[i][j] = a[i] ^ a[j];
        }
    }
}
```

题意：这个函数时求 b 数组的。题目为给出了 b 数组，要求判断是否存在可行的 a 数组。

做法：显然按位来看的话就是一个 2-SAT 问题。根据对应的&or|or^运算进行 2-SAT 的建边。这里出现的 trick 是，通过 2*i 连向 2*i+1 这类的操作就使得 2*i 是不可取的。（详见代码）

参考博客

⑧POJ 2296

题意：给出以下二维坐标点，然后让你往平面上放正方形，点必须落在正方形上面边的中点或者下面边的中点，正方形不能重叠，可以共用边。问最大正方形边的边长。

做法：二分最大边长，2-SAT

⑨POJ 3207

题意：平面上，一个圆，圆的边上按顺时针放着 n 个点。现在要连 m 条边，比如 a，b，那么 a 到 b 可以从圆的内部连接，也可以从圆的外部连接。给你的信息中，每个点最多只会连接的一条边。问能不能连接这 m 条边，使这些边都不相交。

做法：有交错的边必须一个在内，一个在外，对边建图表示在内还是在外做 2-SAT 即可。

⑩POJ 3648

有一对新人结婚，邀请 n 对夫妇去参加婚礼。

有一张很长的桌子，人只能坐在桌子的两边，还要满足下面的要求：1.每对夫妇不能坐在同一侧 2.n 对夫妇之中可能有通奸关系（包括男男，男女，女女），有通奸关系的不能同时坐在新娘的对面，可以分开坐，可以

同时坐在新娘这一侧。如果存在一种可行的方案，输出
与新娘同侧的人。

求解的时候去选择和新郎同一侧的人，输出的时候换一下就是新娘同一侧的人。
如果 i 和 j 有奸情，则增加一条 i 到 j',j 到 i'的边，
同时增加一条新娘到新郎的边，表示必须选新郎。

本题直接选新娘一边的容易错。因为新娘也可能有奸情，需要排除，具体可以见 discuss

11.POJ3678

也是位运算给结果求原值，建图方式与 HDU4421 原理相同

34、

CF668E（见 2-SAT 专题） 代码链接

题意：给出两个 2-sat，如果存在使其中一组成立，另一组不成立的变量取值，输出之，否则输出 SIMILAR

做法：（代码的方法与官方题解基本相同）

首先，对于一个 2-SAT 其有解的**充要条件**是（并不确定正确性）：其闭包中不存在 2i->2i+1 2i+1->2i 这样的点对。

以此为标准，先求出二者的闭包（注意求的时候 i->I 要有连向自己的边 ）这可以直接使用 floyd 完成。并且其中有的点在 2i、2i+1 中已经确定必须选某一个，对于这样的就先一路 dfs 下去，把能确定的都确定出来。

之后开始比较两个 2-SAT，对于两者已确定的点。如果一个是 2i，一个是 2i+1，那么显然直接输出其中任意一个的解（输出过程对还没确定的 dfs）就已经满足了是一个成立，一个不成立。

不然的话，两者已确定的点不存在矛盾，那么就再去找不一样的关系。对于 2-SAT 甲，若其没有 u->v 且 u、v 均是没有确定的，而 2-SAT 乙有这个约束。那么就可以对甲 dfs(u) dfs(v^1)，即使得 u 成立而 v 不成立，这样 dfs 结果为甲是成立的，而乙因为 u->v 就不满足，便不成立。即可得到答案。

没有解，当且仅当没有上述的情况。

通过这道题进一步加深对 2-SAT 的理解。尤其是上述中使用到的充要条件，以及由此导出的判断有无解方法，某些点取值已确定（没确定的点就表示不管去哪一个都能有解），这些之前不曾了解的东西。

35、

Gym 101190B 代码链接

真的是 **2-SAT** 建图的神题！之前快速刷过的 2-SAT 题目真的都是太基础了，而到了这道题就变了形式，变成了更为灵活的应用。网上的博客对这道题几乎都完全没有详细的介绍，看了博客也看不懂，在这里尽可能的详细写一下。

题意：给定一些字符串（01），有的字符串中有一个？，请你找是否有在？中填 0 或 1 的方法，使得这些字符串集合中不存在一个是另一个的前缀。

做法：显然为了表示状态，需要建立 trie 树。题目中的限制实际上就是一条树链上只能有一个结尾点（即插入字符串在结尾进行的标记）。那么我们就只需要在原本一条链上的结尾点建立对应的 2-SAT 关系即可。而需要注意的是，n 比较大，不可能 n^2 的建立。考虑到每条链都是链状结构，可以通过类似链表的方法把它们串起来。而且由于需要更新整个链，直接在原图上建立显然是不行的，需要额外添加新的点来表示。首先通过这样的方法，把一条链上不同节点的点串了起来，接下来还要再处理以同一个节点结尾的点，类似的方法，我们再抽象一些点出来在 2-SAT 中表示它们之间的关系即可。

（详见代码！）

36、

Hihocoder 1230 代码即为算法模板中的 FWT 例题代码

快速沃尔什变化 （此链接为少有的讲清楚了证明的）

概括的说就是 nlog 的计算位运算的卷积 e.g.求每个数的取值在[l,r]区间时 有多少个给定长度为 len 的序列 其所有数异或值为 0 (基于此描述的题目)

对于 FWT 基本上看了上述两个博客就能回想起来。

对于具体题目来说关键就是怎么转换为 FWT 的形式。

对于本题： 题意为：

题意：有 2*n+1 个人，第一遍给每个人先发 [0,m]中任意值的钱，第二遍再给每个人发 x 元，x 的范围为[L,R]。问有多少种第一遍发钱的方法使得：存在 x 使得这些人的钱的异或值在第二遍发完之后为 0。

做法：分析每一位，每个数加完 x 后必须在该位是偶数个 1 才行。而最初该位的 0\1 个数就已经确定了 因此加的 x 在该位是 0 还是 1 就已经确定了。即给定了一个序列，就可以确定要给的 x，但给了 x 却不能确定原序列。因此要枚举 x 来计数。对于加完 x 的序列，其答案就等价于每个数都在[x,x+m]范围内，要使得所有数异或结果为 0 的个数。这样参照上述例题的解释，实际上就是一个异或的卷积形式，卷积“序列中数的个数”次，即可。

37、

CF 453D

题意：见链接，可以算是 FWT 的稍微变一下的形式 orz

Codeforces 453D - Little Pony and Elements of Harmony

题面翻译

谐律精华由代表着和谐（谐律）的六个主观方面的超自然神器组成。它们可以说是整个小马国最强大的力量。谐律精华的内部能被视为一个n个点的完全图，这些点被标记为0到$n-1$，其中n是2的幂，等于2^m。

谐律精华中的能量是在不断变化着的。据一本古书上的记载，点u在i时刻的能量（$e_i[u]$）等于：

$$e_i[u] = \sum_u e_{i-1}[v] \cdot b[f(u,v)]$$

此处的$b[]$被称作"转换系数"，为一个包含有$m+1$个整数的数组，而$f(u,v)$则是数字$(u\ xor\ v)$的二进制表示中1的个数。

给出转换系数和0时刻的能量分布（$e_0[]$）。帮助暮光闪闪预测t时刻的能量分布（$e_t[]$）。答案可能会很大，所以输出时模除p。

做法：初学 FWT，这道题真的是给自己上了很好的一节课 orz

$$e_i[u] = \sum_v e_{i-1}[v] \cdot b[f(u,v)].$$

其中 f(u,v)表示 u^v 二进制下 1 的个数。注意到只需要稍一变换一些写法 就转变为了 ei[u^v]=Σei-1[u]*b[v] （没错，就是这么无脑……）

于是就又变成了一个裸的 FWT。

对于这道题，还有两个 trick。（已记录在模板文件中，以下为摘抄）

之前的模板中 "

各个模板中 dwt 中的/2 结合具体情况转化为 2 的逆元

也可以不在过程中除以 2 转为在求完之后直接除以数组长度 (2^k)

并且有一个 trick 为 过程中取模就直接以 (2^k*MOD) 为模

这样最后就可以直接除以 2^k 了

不过这样乘法取模过程中可能会溢出 ll 需要使用常用模板文件夹下的防溢出乘法

"

38、
HDU 6183

题目大意:

有四种操作。

00:清除所有点

1 x y c1 x y c:给点(x,y)(x,y)添加一种颜色 cc

2 x y1 y22 x y1 y2:在(0,y1)(0,y1)与(x,y2)(x,y2)所围成的矩形里有多少种颜色

33:程序结束

来自 <https://blog.csdn.net/oranges_c/article/details/77800825>

做法：可以建立不完整（只更新有用到的路径）的线段树。对每一种颜色建立线段树，按 y 的范围表示每个结点，每个结点记录该范围内 y 的该种颜色点的最小 x 值。对于 1 操作，每次只要在对应的那棵线段树 log(n)的更新那条路径即可。对于 2 操作，就只需逐棵树进行该 y 范围的查询，由于查询是[0,x]范围，只需要看对应的 x 值是否比给定的 X 小与等于就好。对于 0 操作，清空每棵线段树的根节点，并将线段树的节点池清空即可。
总结：这种在结点池中动态建立线段树的方法之前没有见到过，非常值得学习。

39、
2018 牛客多校 1 J
题意：给数组，若干询问(l,j) 表示 a1——ai 和 aj——an 两部分总共有多少个不同的数
做法 1：
记录每个数第一次和最后一次出现的位置。离线所有查询，按查询区间的 r 排序，逐步移 r(O(n))，初始在 0 处，显然答案为总共的不同个数，逐步移动的过程，就会越过某个数最后一次出现的位置，从此刻开始，往后的时候，如果该数最早出现的位置不小于等于 l，那么这个数就不能被统计了。将此刻的 first[a[r]]-1 更新到 BIT 中，这样每次 r 的位置的答案即为 总颜色个数-[l,n]中被加入的点的个数(first[a[x]]出现在这些位置，说明[1,l]不包含某个数），即每次询问是 O(n)的，整体复杂度 O(nlog(n))

做法 2：
复制一份数组放在后面，就转化为了区间不同数的个数问题。莫队可以 n^3/2，但复杂度太大。选择使用主席数。对于询问[l,r] ，使用 r 版本的主席树，其中主席树类似 BIT 的记录，更新的线段树存储对应区间的不同数的个数。建主席树时，每次更新到 i 这个位置时，往线段树的 i 这个位置插 1（因为每次使用 r 进行查询，一定会包含这个点），x[i]上一次出现的位置-1（因为之前版本已经在那个位置加了 1，为避免重复计该数，在跨越这个数上一次出现的位置时，要去掉当时的+1）实际操作顺序为先减后加。这样每次查询[l,r]时，只需要在 r 版本的树上，求[l,r]的和即可。

40、
HDU 2993 / POJ 2018
题意：给定数列，求至少包含 k 个元素的区间的平均值的最大值。
做法：显然转变成（I,sum[i]）就华为了若干个点的斜率最大的问题。做法是维护下凸包（对于上凸包的点是全都不优于下凸包的，可以用 3 个点的上凸包时，中间的点永远至少不优于两边的点中一个来证明）。并且这里为了使用斜率优化，也涉及到维护的队列的前端后移的问题。（后端移动显然就是维护凸包的过程）这里前端移动是当队首与当前点斜率小于等于队列第二个元素与当前点的斜率就移动。这是因为下凸包的延长线将二维空间分成了若干部分，递增的对应每一条线，只有在某条线以上的一个点才优于另一个。而当在其上时，斜率就已经优于了该条线。这道题要求的是斜率的最大值，所以已经有比该条线大的就直接去掉就可以了。（想要更大，如果没跨过更上面的一条线，只保留后面的点也够了。如果跨过了更上面的一条线，不仅前面的点不够优，后面的点也会不够优）
所以，不需要二分就可以维护下凸包 O(n)的斜率优化过去。