牛客网 ACM 多校训练营（第九场）题解

wwwwodddd

1 A. Circulant Matrix

1.1 题目大意

数组 a 和数组 x 的 FWT 结果是 b。
已知 a 和 b，求 x。

1.2 题目分析

对 a 和 b 做 FWT。
计算 b/a 的结果 x。
对 x 做 FWT。
参考代码

1.3 参考资料

PKU Campus 2014 D
2017 年 NOI 冬令营讲课课件

2 B. Enumeration not optimization

2.1 题目大意

NOIP2017 宝藏 改成了计数题。
新的题目时间复杂度是 $O(n^2 3^n)$。

2.2 题目分析

d 个 $O(n^2 3^n)$ 的做法，思路大概是 $f[S][i]$。
大概是点集 S 构成一棵子树，根节点是 i。
同样需要记录方案数和深度之和。

最后每条边乘以深度之和即可。

2.3 参考资料

一份错误的代码
数据在评论的第二页。

3 C. Gambling

3.1 题目大意

一个比较有趣的面试题。
理解样例是关键。

3.2 题目分析

首先考虑如何计算当前的胜率。
这个有两种算法，动态规划和组合数。

3.3 动态规划

假设当前甲赢了 i 场，乙赢了 j 场。
如果 $i = n$，那么 $f_{i,j} = 1$。
如果 $j = n$，那么 $f_{i,j} = 0$。
对于一般情况，有 $f_{i,j} = \frac{1}{2}(f_{i+1,j} + f_{i,j+1})$。
可以换一种方法表示，设当前胜率为 p。
如果下一场比赛甲赢了，胜率会变成 $p + q$。
如果下一场比赛甲输了，胜率会变成 $p - q$。
即胜率的增加量和减少量一样。
这场比赛应该下注 $2q \times 2^{2n-1}$。
这样如果胜率增加就赢钱，胜率减少就输钱。
如果胜率从 $1/2$ 增加到 1，就会赢 2^{2n-1}。
如果胜率从 $1/2$ 减少到 1，就会输 2^{2n-1}。
换句话说，当前剩余钱数，是由胜率决定的。
胜率为 $1/2$，钱数为 0。
胜率为 1，钱数为 2^{2n-1}。
胜率为 0，钱数为 -2^{2n-1}。
对于一般情况胜率 $1/2 + q$，钱数为 $2q \times 2^{2n-1}$。
这样你便可以写出一个 $O(n^2)$ 的做法。

3.4 组合数

核心问题就是，当前甲赢了 i 场，乙赢了 j 场，甲的胜率是多少。

不妨假设没有提前结束的情况，剩下的 $t = 2n - 1 - i - j$ 场比赛一定打完。

如果甲在其中赢了 $n - i$ 场或更多，甲就获胜了，所以甲的胜率是一个组合数的求和

$$\frac{\sum_{n-i\le k\le t}\binom{t}{k}}{2^t}$$

这个不是很容易解决。

如果下一场甲赢了，胜率变为

$$\frac{\sum_{n-i-1\le k\le t-1}\binom{t-1}{k}}{2^{t-1}}$$

如果下一场甲输了，胜率变为

$$\frac{\sum_{n-i\le k\le t-1}\binom{t-1}{k}}{2^{t-1}}$$

两者求和中只差一项 $\binom{t-1}{n-i-1} = \binom{2n-2-i-j}{n-i-1}$。

带入动态规划的结论，答案是

$$\frac{\binom{2n-2-i-j}{n-i-1}}{2^{2n-2-i-j}} \times 2^{2n-1} = 2^{1+i+j}\binom{2n-2-i-j}{n-i-1}$$

所以统计前缀中，甲赢得次数 i，乙赢的次数 j。

然后计算组合数和 2 的次幂即可。

组合数的计算可以预处理，或者是利用相邻两项只需要乘一项和除一项来计算。

可以了解一下 $O(n)$ 求 1 到 n 的逆元。

参考代码

3.5 参考资料

150 Most Frequently Asked Questions on Quant Interviews

Chapter 2.7 Brainteasers, Question 20.

(Answer: Chapter 3.7 Brainteasers, Question 20.)

4 D. The number of circuits

4.1 题目大意

求欧拉回路的个数。

4.2 题目分析

首先思考如何暴力?

需要用到BEST theorem。

学会暴力之后，注意到这一定是个线性递推，可以先生成前几百项（比如 600 项）

然后用高斯消元，或者 Berlekamp Massey 算法，解出线性递推。

直接求第 n 项的结果就可以了。

类似的思路可以用于很多题目，只要是线性递推，阶数不太高，都可以这样做。

几乎可以解决所有状态压缩结合矩阵乘法的题目。

参考代码

大家可以参考一下杜教的 BM 模板

4.3 参考资料

Project Euler 258 经典的 $d^2 \log n$ 解决线性递推数列的题目。

Project Euler 458 先暴力求递推，然后找规律。

Codechef DMCS 矩阵乘法

BZOJ 1494 NOI 老题，也可以用这个方法通过。

BZOJ 1494 的题解

玲珑杯的一个题 感谢 yanQval 教育我不要出数列题，会被暴力 BM 过。

5 E. Music Game

5.1 题目大意

求得分的期望。

5.2 题目分析

首先要学会平方的版本和立方的版本。

然后发现并不需要实际的去维护 m 次方，任意的一个 m 次多项式都可以。

当然是选择最好维护的组合数。

最后再用斯特灵数推回次方。

参考代码

5.3 参考资料

BZOJ 3450 平方的版本。

BZOJ 4318 立方的版本。

hdu 4625 JZPTREE，组合数维护起来比较简单，最后转回次方即可。

可以参考这份题解

学习组合数和次方之间如何相互转化（用第二类斯特灵数）

6 F. Typing practice

6.1 题目大意

TRIE 图上的最短路。

6.2 题目分析

看到过很多人写不是那么正确的 AC 自动机/TRIE 图。

主要问题就是，如果是字符串匹配的话，那么 AC 自动机跳 Fail 指针的次数，至多 $O(L)$ 次。

因为向后跳的次数，一定小于向前进的次数。

但是对于这个题目有退格来说，这个就不一定成立了。

所以对于这个题应该正确建立 TRIE 图，并且在 TRIE 图上做最短路。

一个错误的示范

一个正确的示范

6.3 参考资料

算法合集之《多串匹配算法及其启示》

算法合集之《Trie 图的构建、活用与改进》

7 G. Longest Common Subsequence

7.1 题目大意

4 个字符串，求 LCS。

重复的数字不太多。

7.2 题目分析

如果四个都是排列的话，对于每个数字 x，考虑他在四个数列出现的位置 $pa[x], pb[x], pc[x], pd[x]$。

在四维空间中构造一个点 $(pa[x], pb[x], pc[x], pd[x])$。

对于构造出的 n 个点，求最长上升子序列即可。

高维的最长上升子序列，可以通过 CDQ 分治，或者是 KD 树来解决。

如果数字出现多次，可以类似的转换。

设数字 x 在 a 中出现的位置是 $pa[x][1], pa[x][2], \ldots, pa[x][ca[x]]$。

设数字 x 在 b 中出现的位置是 $pb[x][1], pb[x][2], \ldots, pb[x][cb[x]]$。

设数字 x 在 c 中出现的位置是 $pc[x][1], pc[x][2], \ldots, pc[x][cc[x]]$。

设数字 x 在 d 中出现的位置是 $pd[x][1], pd[x][2], \ldots, pd[x][cd[x]]$。

构造 $ca[x] \times cb[x] \times cc[x] \times cd[x]$ 个点即可。

求最长上升子序列，相邻两项的四个维度必须都严格上升。

所以每个上升子序列就是原题中的一个公共子序列。

可以注意到，只要保证 a, b, c 中出现的次数 ≤ 2，那么最后生成的点数至多 $8n$。

然后再反复 CDQ 分治更新答案就可以了。

然后这个题的数据一如既往的非常麻烦，所以又有一些奇怪的做法水过去了。

继续感慨出题还是要出计数题。

7.3 参考资料

cogs 2479 四维偏序

cogs 2580 五维偏序

但是这 2 个题都是计数题。

BZOJ 2253 三维偏序

这是一个三维偏序，求最优解。

8 H. Prefix Sum

8.1 题目大意

大概是问如何把 $O(m \log nk^2)$ 优化为 $O(m \log nk)$。

数据范围很难决定啊，一个非常简单的暴力做法是 nm 的，所以 k 也不能太大。

不过说起来 $O(nm)$ 的暴力，有两种。

一种是修改 $O(1)$，询问 $O(n)$。

一种是修改 $O(n)$，询问 $O(1)$。

可以把两种暴力结合起来，变成 $O(m\sqrt{n})$

8.2 题目分析

首先你应该知道，询问 x 回答需要的是

$$\sum_{1 \le i \le x} \binom{x-i+k}{k} a[0][i]$$

其中 $\binom{x-i+k}{k}$ 是没法维护的。

注意到组合恒等式，然后我们把组合数的定义推广到负数。

$$\binom{x-i+k}{k} = \sum_{0 \le j \le k} \binom{x}{j} \binom{k-i}{k-j}$$

$$\sum_{1 \le i \le x} \binom{x-i+k}{k} a[0][i] = \sum_{1 \le i \le x} \sum_{0 \le j \le k} \binom{x}{j} \binom{k-i}{k-j} a[0][i] = \sum_{0 \le j \le k} \binom{x}{j} \left(\sum_{1 \le i \le x} \binom{k-i}{k-j} a[0][i] \right)$$

注意到最后是一个 $\binom{k-i}{k-j} a[0][i]$ 的前缀和，很容易想到用树状数组维护。

一个正常题目，比如 POJ 3468 一般维护 $i^j a[0][i]$，但是 i^j 不利于计算，可以改为同样是 j 次多项式的。

因为 j 有 $k+1$ 个取值，所以我们需要 $k+1$ 个树状数组来维护。

预处理逆元。

$\binom{x}{j} (0 \le j \le k)$ 可以用 $O(k)$ 的时间推出来。

$\binom{k-i}{k-j} (0 \le j \le k)$ 可以用 $O(k)$ 的时间推出来。

每次修改需要修改 $k+1$ 个树状数组，时间复杂度为 $(k+1) \log n$。

每次询问需要求 $k+1$ 个树状数组的前缀和，时间复杂度为 $(k+1) \log n$。

这个题就解决了。

8.3 参考资料

POJ 3468 $k=2$ 的情况就是区间修改，区间求和。

Luogu P4514 二维的情况。

其实还有 $k=3$ 和 $k=4$ 的情况，但是找不到了。

9 I. Juggernaut

9.1 题目大意

大概就是去年 NOI 冬令营讲的题目再推广一次。

9.2 题目分析

首先要学会没有 Burnside 的版本怎么做。

做法很简单，不填最后一行最后一列，前面任意有 $2^{(n-1)(m-1)}$ 种方案，最后一行一列计算出来。

然后回想没有异或为 0 的限制怎么做。

$$(\sum_{a|n}\sum_{b|m}\varphi(a)\varphi(b)2^{nm/\mathrm{lcm}(a,b)})/(nm)$$

如果有异或限制为 0，$nm/\mathrm{lcm}(a,b)$ 需要作出一些改变，具体如下。

设 $l=\mathrm{lcm}(a,b)$。

如果 l/a 是奇数，l/b 是奇数，指数部分应为 $nm/l-n/a-m/b+1$。

如果 l/a 是奇数，l/b 是偶数，指数部分应为 $nm/l-n/a$。

如果 l/a 是偶数，l/b 是奇数，指数部分应为 $nm/l-m/b$。

如果 l/a 是偶数，l/b 是偶数，如果出现了就是程序出错了。

最后考虑对合数取模怎么做。

全程对 nmp 取模。最后直接除以 nm 即可。

为了效率可以当 n,m 和 p 不互质时，再把模数换成 nmp。

Python 也可以通过这个题目。

类似的技巧（最后一次除以 n，没有逆元，那么就全程对 np 取模）也可以用在 FWT 上。

9.3 参考资料

感谢曾耀辉 quailty 的帮助。

2017 年 NOI 冬令营讲课课件

10 J. Maze

10.1 题目大意

大概就是一个连通性相关的状态压缩 DP。

10.2 题目分析

首先把原题做一个转换，原题相当于是随机放红蓝两个钥匙，问两个钥匙联通的方案。

那么就是一个很简单的逐格的连通性状态压缩 DP。

对于每个格子，你需要记录连通性和是否包含红蓝两个钥匙。（最早能连到哪个格子）

对于全局来说，需要额外记录是否出现过红钥匙，是否出现过蓝钥匙，红钥匙和蓝钥匙是否联通。

然后每次增加一格子，讨论新增加的格子和上方，和左方有没有墙，讨论当前格是否放置钥匙。

10.3 参考资料

感谢唐飞虎 xiaodao 的帮助。

插头 DP 大字典

51nod 1633 赫拉迪克之杖