题解

BDFZ NOIP 2020 day1

Elegia

2020 年 11 月 20 日

目录

1 小鸣的疑惑 (aminusb)

答案就是 $a_1 - a_2$。

1.1 坑点

1. 虽然 > 2 的数没用，但是要读完。

2. 注意判一下 $n = 1$ 的情况。如果 $n = 1$ 则 $a_2 = 0$。

3. 别当取模要求不存在。

2 红蔷薇白玫瑰 (rose)

给一颗 01-trie 和一个 01 串 S，对于每个节点，如果按照 S 能走到一个节点则输出该节点，否则输出 0。

2.1 暴力 $O(n|S|)$

略。

2.2 链 $O(n)$

我们发现实际上按顺序用 KMP 扫就可以，如果匹配上了就说明这个节点的 $|S|$ 级祖先走到了它。

2.3 正解 $O(n)$

预处理 KMP 转移的自动机就可以消去均摊。或者使用 hash 也可以通过。

3 山遥路远 (distant)

给定 n 个点，m 条边的有向图，其中每条边有个长度 w，且有一个左括号或者右括号，称路径合法当且仅当沿着路径走过的括号序列是合法括号序列。有 q 组询问，每次问一组 s 到 t 的最短合法路径长度。

3.1 observation

事实上路径长度不会超过 n^3w，括号**层数**不会超过 n^2。

证明将会在后面的算法叙述中自然而然地出现。

3.2 dijkstra $O(mn^3 \log n)$

我们从每个起点开始做一个 dijkstra，不妨将左括号看做 $+1$，右括号看做 -1，那么要求就是任何一个前缀和都是非负的，且终点位置是 0，我们记 $f(u,d)$ 表示到达 u 节点且现在和为 $d(d \geq 0)$ 的情况下的最短路。因此对于每个起点都有 n^3 个状态（$0 \leq d \leq n^2$），对每个相同的 d 都会有 m 种转移，所以复杂度是 $O(n \cdot mn^2 \log n)$。如果你不知道层数有这个上界而是在程序里内嵌了一个 $MAX(MAX \geq n^2)$ 的话，复杂度会是 $O(n \cdot m \cdot MAX \log n)$。由于数据将括号层数构造到了 $\Omega(n^2)$，所以 MAX 的大小不对的话会显著影响你的分数。构造方法留作习题。

3.3 类 Floyd 转移 $O((m^2 + n^3) \log n)$

我们记 $f(u,v)$ 是 u 到 v 的最短路径，若 $u \neq v$，则说明这个括号序列非空，我们有两种可能的转移情况：

- 存在 a, b，这条路径的经过顺序是 $u \to a \rightsquigarrow b \to v$，其中 $a \rightsquigarrow b$ 是合法的，$u \to a$ 是左括号，$b \to v$ 是右括号。

- 存在 w，这条路径的经过顺序是 $u \rightsquigarrow w \rightsquigarrow v$，其中 $u \rightsquigarrow w, w \rightsquigarrow v$ 分别是合法的。

因此我们考虑贪心，将所有的 $f(u,v)$ 推到优先队列里，每次最小的 $f(u,v)$ 取出来后，说明该点对的答案已经确定，我们用于更新剩下的答案。那么第一种情况是两边加一个括号，我们枚举 u 的所有入边，v 的所有出边检查是否能够更新，注意到每对边最多被更新一次，这个将引发 $O(m^2)$ 次更新。第二种情况是枚举 w，检查能否更新 $u \rightsquigarrow v \rightsquigarrow w$ 和 $w \rightsquigarrow u \rightsquigarrow v$，总共引发 $O(n^3)$ 次更新。如果使用 `priority_queue` 那么复杂度是 $O((m^2+n^3)\log n)$，如果使用诸如 `__gnu_pbds::priority_queue` 中一些支持 $O(1)$ decrease-key 的堆，那么复杂度是 $O(m^2+n^3)$。

3.4 优化 $O((mn+n^3)\log n)$

我们考虑优化边枚举的部分。定义一个中间状态 $g(u,v)$ 表示 g 引出了一个左括号还没有被消掉，那么 f 更新的时候枚举一边先用于更新 g，g 更新的时候枚举一边再用于更新 f 就可以了。

复杂度是 $O((mn+n^3)\log n)$，或者 $O(mn+n^3)$（高效堆）。

3.5 答案上界的证明

根据上述构造，我们记 $d(u,v)$ 是达到 $f(u,v)$ 的最短路径，此时的括号深度。则根据转移形式，

- 存在 a,b，$d(u,v)=d(a,b)+1$
- 存在 w，$d(u,v)=\max(d(u,w),d(w,v))$

我们考虑最短路径的依赖关系，一个极重要的性质是：$f(u,v)$ 依赖的路径中显然不会有它自己。

因此我们考虑 $d(u,v)$，这说明必然在路径中存在一个长为 $d(u,v)$ 的依赖链。但是因为依赖链中不能出现两个相同元素，根据鸽笼原理，$d(u,v) < n^2$。

反观最初的 bfs 算法必然能算出答案，而最短路径必然经过每个点最多一次，故路径长度 $f(u,v) < n \cdot (d+1) \leq n^3$。

3.6 阳间的去 log 方法

我们假设你不知道如何使用 `__gnu_pbds::priority_queue`，设正整数 k，我们如果用一个 $V^{1/k}$ 叉树来用于维护堆的话，那么当一个元素的值减小时只需要 $O(k)$ 就能进行更新，弹出最小值只需要 $O(kV^{1/k})$。

我们的最短路中这两种操作次数其实不太平衡，所以复杂度就是 $O(k(nm + n^3) + kn^{2/k}n^2)$。当 k 取任何 ≥ 2 的常数的时候复杂度都是 $O(nm + n^3)$。

4 命运歧途 (fate)

对于每个 k，询问有多少 n 阶排列满足 $|p_i - p_{i+1}| \neq k$。

4.1 每次 $O(n^2)$ DP

我们先考虑 $k=1$ 怎么做。$f(i,j,0/1)$ 表示前 i 个数总共有 j 个相邻的数位置现在还是相同的，$0/1$ 表示 i 和 $i-1$ 是不是相邻的，接下来我们考虑 $i+1$ 插在这个排列的哪个位置，只有 $O(1)$ 种转移。

注意到对于 $\mathrm{mod} k$ 同余的类互不干扰，我们可以按照 $1, k+1, \ldots$，然后 $2, k+2, \ldots$ 的顺序插入排列，可以做任意 k 的 DP。

4.2 容斥原理

我们先考虑 $k=1$ 怎么做。我们现在有 $n-1$ 个条件：对于第 i 个条件，我们记命题 P_i 是“i 的位置和 $i+1$ 的位置相邻”。我们要计数 $P_1, \ldots, P_{n-1}$ 均不成立的情况。容斥原理就是展开式子 $(1-P_1)(1-P_2)\cdots(1-P_{n-1})$，那么我们看看一些 P 强制成立的时候会发生什么：

如果 $P_L, P_{L+1}, \ldots, P_R$ 强制成立，那么说明 $|p_L - p_{L+1}| = 1, |p_{L+1} - p_{L+2}| = 1, \ldots, |p_R - p_{R+1}| = 1$，也就是说 $p_L, \ldots, p_{R+1}$ 必然是连续上升或者连续下降的。

因此我们可以考虑这样一个状态：$f_{n,k}$ 表示将一个序列分割为 k 段，设 u 为方案中长度 >1 的段数，全体 2^u 求和。分割为了 k 段，就说明有 $n-k$ 个要求被强制成立了，因此答案就是

$$\sum_k k!(-1)^{n-k} f_{n,k}$$

注意我们可以通过前缀和优化在 $\Theta(n^2)$ 时间内算出 f 的所有。

4.3 扫动 DP $\Theta(n^{2.5})$

对于一般的 k 的情况，我们是要对 $\bmod k$ 相同的分为一组进行容斥。有 a 个大小为 $\lfloor n/k \rfloor$ 的组和 b 个大小为 $\lceil n/k \rceil$ 的组。我们令 $F_m(x) = \sum_k (-1)^{m-k} f_{m,k} x^k$，那么令 $G(x) = F_{\lfloor n/k \rfloor}(x)^a F_{\lceil n/k \rceil}(x)^b$，答案就是

$$\sum_k k![x^k]G(x)$$

然而本题采用了任意模数，也是为了提示标算并非 FFT。我们考虑这样一个计算过程：

令 k 从 1 至 n 枚举，我们发现 $G(x)$ 的幂的表示会有重叠的部分，比如 $k \ge n/2$ 的时候有 $G(x) = F_1(x)^{n-2k} F_2(x)^k$。因此我们只需维护一下 $G(x)$ 的改变即可。注意添加/去掉一项 m 次多项式只需要 $\Theta(nm)$ 的时间。总共出现的多项式次数均为 n/k 的上下取整情况，每个相同次数的多项式出现次数乘以多项式本身的次数显然不超过 n，所以

$$\sum \deg F = \Theta(n^{1.5})$$

因此这一做法的复杂度是 $\Theta(n^{2.5})$。

4.4 $\Theta(n^2 \log n)$ 做法

事实上存在一个 $\Theta(n^2 \log n)$ 的无 FFT 做法。

现在我们需要计算 $H(x) = F(x)^a G(x)^b$，考虑求导，有

$$\begin{aligned} H'(x) &= aF(x)^{a-1}G(x)^b F'(x) + bF(x)^a G(x)^{b-1} G'(x) \\ &= H(x)\left(a\frac{F'(x)}{F(x)} + b\frac{G'(x)}{G(x)}\right) \end{aligned}$$

我们按照 $[x^n]H(x) \to [x^n]H(x)F'(x) \to [x^n]\frac{H(x)F'(x)}{F(x)}$ 的顺序，每一步都是 $\Theta(\deg F)$ 的计算，因此这样就可以在 $\Theta(n(\deg F + \deg G))$ 的时间完成快速幂。

这样根据调和级数可知总复杂度就是 $\Theta(n^2\log n)$ 了，但是由于是任意模数，我们并不能直接根据这个式子做，因为有除法。但所幸我们最终只需要知道所有 $k![x^k]G(x)$。

接下来我们需要将模数拆分成两部分，令 $M=2^tM'$，其中 M' 是奇数。

4.4.1 模 2^t

设 k 是最小的满足 $2^t\mid k!$，因此我们只需要暴力做快速幂算前 k 项就可以了。注意到 $k=\Theta(t)=O(\log M)$，所以复杂度 $\Theta(n\log^2 M\log n)$。

4.4.2 模 M'

我们接下来考虑一个重要性质：

$F_m(x)$ 不存在常数项，且 $|[x^1]F_m(x)|\le 2$。(特别地，$F_1(x)=x$，而对于 $m\ge 2$ 来说有 $[x^1]F_m(x)=2(-1)^{m-1}$)。

因为 M' 是奇数，所以就算是 2 我们也可以把它除掉了。我们考虑补充阶乘 $\widehat{F}(x)=\sum_k k!x^k[x^k]F(x)$ 和二项乘法 $\widehat{F}\star\widehat{G}=\widehat{FG}$，我们发现这样 $\widehat{F'}$ 与 $\widehat{F}$ 的关系就消去了除法，等价于一个位移了。

通过这样的转化，我们可以无障碍地计算出 $\widehat{F^aG^b}$，而这足够求出答案。

最后我们再用 CRT 合并一下就得到了最终的答案。时间复杂度 $\Theta(n^2\log n)$。

由于评测机是 32 位的，所以没有 `__int128` 可以用，Barret Reduction 可能不太方便，但可以使用 Montgomery Reduction 来做取模优化，由于 M' 是奇数，在这里使用 Montgomery Reduction 是很方便的。可参看 Min25 Blog 以及 std 的实现。