链式反应 解题报告

湖北省武汉市第二中学 吕凯风

1 试题来源

原创。可以在这里找到：http://uoj.ac/problem/50。

2 试题大意

著名核物理专家 Picks 最近发现了一种非常适合核裂变的元素叫囧元素。囧元素原子序数为 1024，囧-2333 如果被一个中子撞击后会分裂成蒟-1234 和蒻-1098 同时释放出恰好 2 个中子，并且以破坏死光的方式释放光子。

核物理专家 Picks 做实验从来不用实验仪器。他用手指从宇宙中挑选出了 n 个囧-2333 原子并编号为 1 到 n，并用帅气的眼神发射中子撞击了编号为 1 的囧原子，启动了链式反应。

当一个囧-2333原子被中子撞击时，有两种情况。要么这个囧-2333原子变为了囧-2334 并不再参与后续反应，要么囧-2333会进行核裂变，一方面，裂变产生的破坏死光会照射到 c 个不同的囧-2333原子并且这些原子会极为神奇地分解为氢和氦的混合物并不再参与后续反应。另一方面，裂变后产生的 2 个中子会分别撞上两个不同的囧-2333原子。显然被破坏死光照射后就不是囧-2333了，所以不可能又被中子撞上又被破坏死光照射到。

经过反复实验，核物理专家 Picks 终于确定了 c 的范围 A，其中 A 是一个非负整数集合。每次破坏死光会照射并影响到的囧-2333原子数量 c 满足 $c \in A$ 。

链式反应时 Picks 会用肉眼观察实验中出现的事件（仅包括中子的撞击和破坏死光的信息），结束后 Picks 会写下实验记录。

但是很不幸 Picks 的实验记录丢失了。他只记得链式反应后没有剩余的囧-2333原子，而且每次一个囧-2333原子核裂变时，中子总是撞击编号比它大的囧-2333原子，破坏死光也总是照射编号比它大的囧-2333原子。

给你 A 和 $n_{\max}$，求可能会有多少种不同的实验记录。你需要对于 $n = 1, \ldots, n_{\max}$ 输出答案。你只用输出答案对 998244353（$7 \times 17 \times 2^{23} + 1$，一个质数）取模后的值。

两个实验记录 T_1，T_2 被认为是相同的当且仅当：

1. “编号为 v 的囧-2333分裂产生的中子撞上了编号为 u 的囧-2333”这个事件在 T_1 中当且仅当这个事件在 T_2 中。
2. “编号为 v 的囧-2333分裂产生的破坏死光照射了编号为 u 的囧-2333”这个事件在 T_1 中当且仅当这个事件在 T_2 中。

编号	n
1	≤ 8
2	≤ 100
3	≤ 100
4	≤ 200
5	≤ 5000
6	≤ 5000
7	≤ 50000
8	≤ 50000
9	≤ 70000
10	≤ 200000

时间限制：4s

空间限制：256MB

3 算法介绍

3.1 算法一

按照题意进行爆搜，可以通过第一个测试点获得 10 分。

3.2 算法二

我们得思考一下这题题意到底是啥意思。其实说，一棵有标号的树，编号满足堆的性质，对于儿子方向每个结点有两条红色边，c 条黄色边，c 属于集合 A，统计方案数。

所以我们用 $f[n]$ 表示 n 个结点的这个样子的树的方案数。然后要么 1 号点没有进行核裂变，此时必有 $n=1$ 且 $f[n]=1$。要么 1 号结点进行了核裂变，此时我们枚举两个中子打中的结点的子树的大小 j,k，剩下的就是被破坏死光打中的，于是要满足 $i-1-j-k\in A$。然后考虑两个中子打中的结点的子树，我们先选出它们的编号，方案数是 $\binom{i-1}{j}\cdot\binom{i-1-j}{k}$，然后我们就可以安全地把这两棵子树的结点都重标号，方案数自然是 $f[j]$ 和 $f[k]$。所以我们把这些杂八事综合起来就得到了一个简单的DP：$f[i]=\sum_j\sum_k\binom{i-1}{j}\cdot\binom{i-1-j}{k}\cdot f[j]\cdot f[k]$。这里的 j,k 要满足 $i-1-j-k\in A$。直接暴力递推就得到了一个 $O(n^3)$ 的算法。

什么，过不了样例？当然过不了样例了，因为有重复计数。那两个中子是等价的，一个方案自然会被统计了两遍，只要把 DP 方程除以 2 就好了。

可以通过前 4 个测试点获得 40 分。

3.3 算法三

其实只要在算法二基础上优化就行了。我们仔细观察式子：$\binom{i-1}{j}\cdot\binom{i-1-j}{k}=\frac{(i-1)!}{j!k!(i-j-k-1)!}$。有四个部分，第一个只跟 i 有关，第二个只跟 j 有关，第三个只跟 k 有关，第四个只跟 $i-j-k-1$ 有关。所以我们可以递推一个 $g[i]=\sum_k\frac{f[k]}{k!}\frac{f[i-k]}{(i-k)!}$。然后递推 $f[i]$ 时我们枚举 $j+k$ 的和 s，那么直接读取 $g[s]$ 的值，然后剩下的部分就是一些跟 i 和 s 有关的了。

好像说得挺意识流的，具体就是：$f[i]=\frac{1}{2}\sum_s g[s]\cdot\frac{(i-1)!}{(i-s-1)!}$。$s$ 要满足 $i-s-1\in A$。

可以通过前 6 个测试点获得 60 分。

3.4 算法四

再观察一下式子，搞个数组 a，其中如果 $i\in A$ 则 $a_i=\frac{1}{i!}$，否则为 0。然后原递推式的右边的第 i 项其实就是 g 跟 a 的卷积的第 $i-1$ 项然后再乘以 $(i-1)!$。然后其实 g 本身也是个卷积，就是 $\frac{f[i]}{i!}$ 这个数列的平方。

我们可以使用分治。每次分治一个区间 $[l,r]$，我们找出中点 m，然后递归 $[l,m]$，然后求出 $[l,m]$ 对 $[m+1,r]$ 的贡献，再递归右边。

考虑左边对右边的贡献，“$\binom{i-1}{j}\cdot\binom{i-1-j}{k}\cdot f[j]\cdot f[k]$”中，左边可能作为 $f[j]$ 也可能作为 $f[k]$ 出现，也可能都出现。我们只要考虑这几种情况然后分别进行 FFT 就行了。看起来要和整个 a 或者 f 进行卷积？其实不然，只用取出一个长度为分治时的区间的长度的前缀就可以了。

时间复杂度 $O(n\log^2 n)$。可以通过前 9 个测试点获得 90 分。常数小的话应该能过。

3.5 算法五

以下内容需要一点微积分知识。

嗯，既然都发现是卷积了，那么肯定少不了生成函数。我们记 $\frac{f[i]}{i!}$ 的生成函数为 $x(t)$，a_i 的生成函数为 $a(t)$，那么生成函数就要满足：$x'(t)=\frac{1}{2}a(t)x^2(t)+1$。

拨一下 mathmatica 就会发现它解不出来这个微分方程，所以只有自力更生了。

首先科普下一般来说应该怎么解一个多项式方程（更严谨地说这里应该是形式幂级数）。假设 $x(t)$ 满足 $f(x(t))=0$，那么我们先求出 0 次项的系数，然后每次倍增。也就是说每次我们有一个多项式 x_n 满足 x_n 和 x 的前 n 项系数是一样的，我们记为 $x\equiv x_n \pmod{t^n}$，然后我们要求出 x_{2n} 满足 $x\equiv x_{2n}\pmod{t^{2n}}$。使用泰勒展开可以得到:

$$0=f(x_{2n})=f(x_n)+f'(x_n)(x_{2n}-x_n)+\frac{1}{2}f''(x_n)(x_{2n}-x_n)^2+\dots \tag{1}$$

注意到如果只考虑前 $2n$ 项，那么就可以去掉二次项及之后的项然后解出：

$$x_{2n}\equiv x_n-\frac{f(x_n)}{f'(x_n)} \pmod{t^{2n}} \tag{2}$$

由于 $T(n)=T(n/2)+O(n\log n)$ 解出来是 $T(n)=O(n\log n)$，所以只要能足够快地把 x 带入 $f(x)$，那么就能 $O(n\log n)$ 解方程。（当然需要 FFT 做多项式乘法）

什么这里有个除法？我们可以利用牛顿迭代求出一个形式幂级数的乘法逆元，于是就能做除法了。

于是微分方程也可以如法炮制，假设有个这样的一阶微分方程：

$$\frac{d}{dt}x = f(x) \tag{3}$$

我们还是老样子：

$$\frac{d}{dt}x_{2n} \equiv f(x_n) + f'(x_n)\cdot(x_{2n} - x_n) \pmod{t^{2n}} \tag{4}$$

$$\tag{5}$$

$$\equiv f(x_n) + f'(x_n)\cdot x_{2n} - f'(x_n)\cdot x_n \pmod{t^{2n}} \tag{6}$$

所以问题变成了这玩意儿怎么解……而这玩意儿其实很好解……我们两边同时乘以 $e^{-\int f'(x_n)dt}$，记作 r：

$$\frac{d}{dt}x_{2n}\cdot r \equiv (f(x_n) - f'(x_n)\cdot x_n)\cdot r + f'(x_n)\cdot x_{2n}\cdot r \pmod{t^{2n}} \tag{7}$$

$$\tag{8}$$

$$\frac{d}{dt}(x_{2n}\cdot r) \equiv (f(x_n) - f'(x_n)\cdot x_n)\cdot r \pmod{t^{2n}} \tag{9}$$

然后只要把右边积分再除以 r 就行了。

注意到虽然人类无法方便地给任意函数积分，但是给多项式求导和求积分简直易如反掌。所以唯一的难点在于 r 怎么求。

好吧其实我几个月前 YY 到这里直接就弃疗了，后来翻了翻论文，知道了怎么求 e^x。由于 e^x 的反函数是 $\ln(x)$ 而 $\ln(x)$ 对 t 的导数是 $\frac{x'}{x}$，所以 $\ln(x)$ 可以快速求，然后我们就可以进行牛顿迭代求 $\ln(x)$ 的反函数得到 e^x。

对于本题，$f(x) = \frac{1}{2}ax^2 + 1$，然后无脑使用上述方法就行了。

本来还想当一个原创算法呢，结果后来发现国外论文上早就有了。

时间复杂度 $O(n\log n)$。可以通过所有测试点获得 100 分。

形式幂级数真是个优美的东西啊，很多在实数域内有条件收敛的算法，到形式幂级数上就一定收敛了，我不得不表示赞叹。

4 数据生成方式

直接随机生成即可。

5 参考程序

- `reaction.cpp` 是参考程序，即算法五。
- `reaction_fast.cpp` 是加了一些常数优化的算法五。
- `reaction_nlog2n_orz.cpp` 是算法四。
- `reaction_nlog2n_fast` 是加了一些常数优化的算法四。
- `reaction_n2_orz.cpp` 是算法三。
- `reaction_orz.cpp` 是算法二。