浅谈一类基于概率的约瑟夫问题

叶卓睿

杭州第二中学

August 1, 2020

前言

本文将对“给一个 1 到 n 的环，有个指针从 1 开始移动，每次指针所在位置有 p 的概率消失掉，然后指针向右移动”这样一个模型下的一些有趣的概率期望问题进行研究，介绍处理这类题目的各种技巧和思路。

希望能让大家能有所启发。

为方便起见，下文约定 $q = 1 - p$。

一些观察

引理 1： 一个被指针经过 i 次的点，存活概率为 q^i

证明： ~~这几乎是显然的~~。转化问题，认为一个点消失后不会真的从环上消失，而是打上一个“删除”标记。那么存活当且仅当每次经过都没有打“删除”标记，故存活概率为 q^i。

一些观察

引理 2：当指针落在位置 c 时，$1..c-1$ 中每个点已经消失的概率相同，$c+1..n$ 中每个点已经消失的概率也相同。

证明：注意到此时 $1..c-1$ 中每个点经过次数相同，由**引理 1**，它们的存活概率相同。$c+1..n$ 也同理。

迫真大游戏

给一个 1 到 n 的环，有个指针从 1 开始移动，每次指针所在位置有 p 的概率消失掉，然后指针向右移动。求每个点是最后一个消失的概率。

$n \le 2 \cdot 10^5$

迫真大游戏

我们关心的位置未必是起点，考虑先让指针转几步使得它落在我们关心的位置上。

令 f_n 为环大小为 n 时，1 号点最后消失的概率。

则有 $ans_k = \sum_{i=0}^{k-1} \binom{k-1}{i} p^i q^{k-1-i} f_{n-i}$

发现将组合数用阶乘展开后这是个卷积的形式，可以 FFT。问题就变成如何求 f_n 了。

迫真大游戏

枚举第一个分身在第 $i+1$ 轮死亡，可以得到

$$\begin{aligned} f_n &= \sum_{i=0}^{\infty} pq^i(1-q^i)^{n-1} \\ &= \sum_{i=0}^{\infty} pq^i \sum_{j=0}^{n-1} (-1)^j (1-p)^{ij} \binom{n-1}{j} \\ &= p\sum_{j=0}^{n-1} (-1)^j \binom{n-1}{j} \sum_{i=0}^{\infty} q^{(j+1)i} \\ &= p\sum_{j=0}^{n-1} (-1)^j \binom{n-1}{j} \frac{1}{1-q^{j+1}} \end{aligned}$$

这也可以化为卷积的形式。综上，原问题可以通过 2 次 FFT 解决，时间复杂度 $\mathcal{O}(n\log n)$。

小结

仔细分析题目，发现过程可以拆成独立的 2 步，逐个解决即可。

一个原创的改编题

给一个 1 到 n 的环，有个指针从 1 开始移动，每次指针所在位置有 p 的概率消失掉，然后指针向右移动。对于每个 i，求 c 号点是第 i 个消失的概率。

$n \le 10^6$

一个原创的改编题

例题 1 的技巧这里看起来不适用了。我们试试直接用生成函数推式子。

令 a_i 为 c 号点是第 $i+1$ 个消失的概率，我们希望求出 $A(x)=\sum a_i x^i$。则有

$$
\begin{aligned}
A(x) &= \sum_{t=0}^{\infty} q^t p\left(q^{t+1}+(1-q^{t+1})x\right)^{c-1}\left(q^t+(1-q^t)x\right)^{n-c} \\
&= \sum_{t=0}^{\infty} q^t p\left(q^{t+1}(1-x)+x\right)^{c-1}\left(q^t(1-x)+x\right)^{n-c} \\
&= p\sum_i\sum_j \binom{c-1}{i}\binom{n-c}{j}\sum_{t=0}^{\infty} q^i(1-x)^{i+j}x^{n-1-i-j}q^{t(1+i+j)} \\
&= p\sum_i\sum_j \binom{c-1}{i}\binom{n-c}{j} q^i(1-x)^{i+j}x^{n-1-i-j}\frac{1}{1-q^{1+i+j}}
\end{aligned}
$$

一个原创的改编题

观察式子，可以记 $f_k=\sum_{i+j=k}\binom{c-1}{i}\binom{n-c}{j}q^i$，则有

$$A(x)=p\sum_i\frac{f_i}{1-q^{i+1}}(1-x)^i x^{n-1-i}$$

由于 c 为常数，f_n 是一个卷积的形式，可以 FFT。现在考虑求解 $A(x)$，继续推导可知

$$[x^{n-i+j-1}]A(x)=p\sum_i\sum_j\binom{i}{j}(-1)^j\frac{f_i}{1-q^{i+1}}$$

这也是卷积的形式，问题即可使用 2 次 FFT 得到解决。

一个原创的改编题

事实上我们可以做得更快。记 $F(x)$ 为 f_i 对应的普通型生成函数，我们发现 $F(x)=(1+qx)^{c-1}(1+x)^{n-c}$

通过对生成函数求导，可以得到

$$F'(x)=(c-1)q\frac{F(x)}{1+qx}+(n-c)\frac{F(x)}{1+x}$$

对比系数，得到 f_i 的递推式：

$$f_{i+1}=\frac{((c-1)q+n-c-(q+1)i)f_i+q(n-i)f_{i-1}}{i+1}$$

综上所述，我们只需要 1 次 FFT 就解决了这个问题, 时间复杂度 $\mathcal{O}(n\log n)$。

小结

在**引理 1** 的帮助下，只使用数学推导手段就可以在一些问题上取到不错的效果，优势是思维难度不高。不过缺点是推导、计算较为繁琐，在一些情况下可能无法优化。

事实上，使用例题 1 的拆解过程的办法，可以做到 $\mathcal{O}(n\log^2 n)$，因为复杂度较劣这里不进行展示。

Eat Cards, Have Fun

给一个 1 到 n 的环，第 i 个点上有数字 A_i，有个指针从 1 开始移动，每次指针所在位置有 p 的概率消失掉然后加入序列 b 的末尾，然后指针向右移动。求排列 b 在所有 $1..n$ 的排列中，按字典序排序后的序号期望。

$n \le 300$

官方题解做法

和例题 2 一样，大力推式子！

这题变成期望了，我们需要拆贡献。记排列为 P_n，则这个排列的序号为 $Ans=\sum_i (n-i)!\sum_{j>i}[P_j<P_i]+1$。

贡献还可以通过枚举位置 i，以及在 i 消失之前 $1..i-1$ 共消失了 a 个，$i+1..n$ 共消失了 b 个，$1..i-1$ 共消失了 c 个小于 P_i 的，$i+1..n$ 共消失了 d 个小于 P_i 的来计算。

记 $dp_{i,a,b}$ 为在 i 消失之前 $1..i-1$ 共消失了 a 个，$i+1..n$ 中共消失了 b 个的概率，l_i 为 $1..i-1$ 中小于 P_i 的个数，r_i 为 $i+1..n$ 中小于 P_i 的个数，则有

$$Ans=\sum_i\sum_a\sum_b\sum_c\sum_d (n-a-b-1)!(A_i-c-d-1)\binom{l_i}{c}\binom{i-1-l_i}{a-c}\binom{r_i}{d}\binom{n-i-r_i}{b-d}dp_{i,a,b}+1$$

官方题解做法

首先考虑如何计算 $dp_{i,a,b}$。

$$
\begin{aligned}
dp_{i,a,b} &= \sum_{t=0}^{\infty} q^t p(q^{t+1})^{i-1-a}(1-q^{t+1})^a (q^t)^{n-i-b}(1-q^t)^b \\
&= \sum_{j=0}^{a}\sum_{k=0}^{b}\binom{a}{j}\binom{b}{k}(-1)^{j+k} q^{i-1-a+j}\sum_{t=0}^{\infty} q^{t(n+j+k-a-b)} \\
&= \sum_{j=0}^{a}\sum_{k=0}^{b}\binom{a}{j}\binom{b}{k}(-1)^{j+k}\frac{q^{i-1-a+j}}{1-q^{n+j+k-a-b}}
\end{aligned}
$$

暴力计算时间复杂度为 $\mathcal{O}(n^5)$，需要优化。

官方题解做法

定义

$$h_{1,a,b}=\sum_{j=0}^{a}\sum_{k=0}^{b}\binom{a}{j}\binom{b}{k}(-1)^{j+k}\frac{q^{a+j}}{1-q^{n+j+k-a-b}}$$

则有

$$dp_{i,a,b}=pq^{i-1}h_{1,a,b}$$

定义

$$h_{2,a,b}=\sum_{k}\binom{b}{k}(-1)^{k}\frac{1}{1-q^{n-a-b+k}}$$

则有

$$h_{1,a,b}=\sum_{j}\binom{a}{j}(-1)^{j}q^{j-a}h_{2,a-j,b}$$

那么，我们可以分别计算出 $h_{2,a,b},h_{1,a,b},dp_{i,a,b}$，这部分时间复杂度为 $\mathcal{O}(n^3)$。

官方题解做法

回到答案式子

$$Ans=\sum_i\sum_a\sum_b\sum_c\sum_d(n-a-b-1)!(A_i-c-d-1)\binom{l_i}{c}\binom{i-1-l_i}{a-c}\binom{r_i}{d}\binom{n-i-r_i}{b-d}dp_{i,a,b}+1$$

暴力计算仍然是 $\mathcal{O}(n^5)$ 的。

官方题解做法

接下来我们定义一些辅助函数加速计算：

$$f_{i,0,a} = \sum_c \binom{l_i}{c}\binom{i-1-l_i}{a-c}$$

$$f_{i,1,a} = \sum_c \binom{l_i}{c}\binom{i-1-l_i}{a-c}c$$

$$g_{i,0,b} = \sum_d \binom{r_i}{b}\binom{n-i-r_i}{b-d}$$

$$g_{i,1,b} = \sum_d \binom{r_i}{b}\binom{n-i-r_i}{b-d}d$$

官方题解做法

然后将 $A_i - c - d - 1$ 拆为 $(A_i - 1) - c - d$ 三部分分别算贡献，可以得到：

$$Ans = \sum_i \sum_a \sum_b \left((A_i - 1) f_{i,0,a} g_{i,0,b} - f_{i,1,a} g_{i,0,b} - f_{i,0,a} g_{i,1,b}\right)$$
$$(n - a - b - 1)! dp_{i,a,b} + 1$$

时间复杂度 $\mathcal{O}(n^3)$。

新的思考角度

推式子太麻烦了！我们换个新的思考角度。

根据期望的线性性，贡献可以拆成 $\sum_i \sum_j \sum_t [A_i > A_j] Pr(i\text{先于}j \wedge i\text{在时刻}t\text{消失})(n-t)!$。

由**引理 2**，我们发现固定 i, t 后，贡献只需与 j, i 的相对大小有关。

这一性质将对解题有巨大帮助。

接下来我们先讨论 $i > j$ 的情况，可以设计 dp：$f_{n,i}$ 表示长度为 n 的环，i 先于 $i-1$ 消失的所有情况下 $(n-t)!$ 的期望。

新的思考角度

接下来我们先讨论 $i > j$ 的情况，可以设计 dp：$f_{n,i}$ 表示长度为 n 的环，i 先于 $i-1$ 消失的所有情况下 $(n-t)!$ 的期望。

转移如下：

$$f_{n,i} = \begin{cases} qf_{n,n} + p(n-1)! & i = 1 \\ qf_{n,i-1} & i = 2 \\ qf_{n,i-1} + pf_{n-1,i-1} & i > 2 \end{cases}$$

按 n 从小到大进行求解，对于每个 n 直接高斯消元可以 $\mathcal{O}(n^3)$。事实上环上消元是可以做到 $\mathcal{O}(n)$ 的，具体做法是：设 $x = f_{n,n}$，那么对于 $i \in [1, n]$ 可以分别把 $f_{n,i}$ 表示成 $k_i x + b_i$ 的形式，从而得到一个一元一次方程，解出 x 后即可计算出 $f_{n,i}$。

另一种情况类似，这里不再赘述。

这样时间复杂度为 $\mathcal{O}(n^2)$，实现难度较低。

新的思考角度

Best solutions for Problem 6339

Language : All G++ GCC C++ C Pascal Java

Rank	Author	Exe. Time	Exe. Memory	Code Len.	Language	Date
1	supy	15MS	2076K	2164B	G++	2019-12-05 16:00:26
2	JZdavid	327MS	4568K	3591B	G++	2018-08-01 22:41:27
3	orbitingflea	343MS	2264K	2534B	G++	2018-08-02 10:51:36
4	zhou8888	483MS	3956K	3369B	G++	2019-04-15 23:30:10
5	laofudasuan	483MS	59336K	3524B	G++	2018-08-02 19:44:27
6	Emma Yin	546MS	2112K	2184B	G++	2018-08-02 16:14:50
7	赵子龙	577MS	12608K	4248B	G++	2020-07-02 17:05:33
8	OMTWOCZWEIXVI	592MS	57648K	2670B	G++	2018-09-19 11:26:54
9	et1999	608MS	2560K	3580B	G++	2019-03-22 14:47:28
10	20032019	748MS	6496K	3221B	G++	2018-08-06 19:05:46
11	kut_msm	764MS	120688K	2557B	G++	2018-08-02 11:42:45
12	SYC	951MS	2456K	3244B	G++	2018-08-01 17:53:43
13	FlappyFish	1060MS	6280K	2919B	G++	2018-08-02 21:37:41
14	wxh010910	1294MS	114020K	2642B	G++	2018-08-30 15:22:35
15	Windows 64 bit on Wi	1388MS	226852K	13199B	G++	2018-08-07 19:19:41
16	SCUT_Judge	1435MS	3632K	5071B	G++	2019-06-25 12:38:01
17	大弱逼	1903MS	108108K	2346B	G++	2018-08-01 20:26:20
18	赵丽斌	2090MS	112464K	2502B	G++	2018-08-09 21:03:31
19	jnxxhzz	2199MS	112468K	2500B	G++	2019-09-24 18:31:49

继续优化

事实上，这个算法还可以继续优化……

dp 转移式子最麻烦的一点在于有环，我们尝试先求解第一列。记 $f_n = dp_{n,1}$。枚举有多少个是在 1 之后消失的：

$$\begin{aligned}
f_n &= \sum_{i=0}^{n-2} \binom{n-2}{i} \sum_{t=0}^{\infty} q^t p q^t (q^t)^i (1-q^t)^{n-2-i} (i+1)! \\
&= \sum_{i=0}^{n-2} \binom{n-2}{i} \sum_{j=0}^{n-2-i} \binom{n-2-i}{j} (-1)^j p (i+1)! \sum_{t=0}^{\infty} q^{t(2+i+j)} \\
&= p \sum_{i=0}^{n-2} \binom{n-2}{i} \sum_{j=0}^{n-2-i} \binom{n-2-i}{j} (-1)^j (i+1)! \frac{1}{1-q^{2+i+j}} \\
&= p \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} \binom{n-2}{i+j} \binom{i+j}{i} (-1)^j (i+1)! \frac{1}{1-q^{2+i+j}}
\end{aligned}$$

继续优化

为加速计算，我们令

$$g_k = \sum_{i+j=k} \binom{i+j}{i}(i+1)!(-1)^j$$
$$= \sum_{i+j=k} \frac{(i+j)!(i+1)(-1)^j}{j!}$$

这是一个卷积的形式，可以 FFT。于是

$$f_n = p\sum_k \frac{1}{1-q^{2+k}}\binom{n-2}{k} g_k$$
$$= p\sum_k \left(\frac{1}{1-q^{2+k}} g_k\right)\binom{n-2}{k}$$

将组合数用阶乘展开后，这也是一个卷积的形式，可以 FFT。因此，我们可以 $\mathcal{O}(n\log n)$ 地求出 $dp_{i,1}$。

继续优化

回顾 dp 转移方程

$$f_{n,i}=\begin{cases} qf_{n,n}+p(n-1)! & i=1 \\ qf_{n,i-1} & i=2 \\ qf_{n,i-1}+pf_{n-1,i-1} & i>2 \end{cases}$$

已知 $dp_{i,1}$，可以 $\mathcal{O}(n)$ 地计算出 $dp_{i,2}$。
记 $g_i=dp_{i,2}, h_i=dp_{n,i+2}$。计算每个 g_j 对 h_i 的贡献，有：

$$h_i=\sum_j g_j\binom{i}{n-j}p^{n-j}q^{i+j-n}$$

将组合数用阶乘展开后，这也是一个卷积的形式，可以 FFT 解出 $f_{n,i}$。

继续优化

考虑求解原问题，$i > j$ 的情况贡献为

$$Ans = \sum_{i>j}[A_i > A_j]f_{n,i}$$

，使用树状数组统计 $Ans = \sum_{i>j}[A_i > A_j]$，$i < j$ 的部分使用类似的做法即可做到总时间复杂度 $\mathcal{O}(n \log n)$。

继续优化

考虑求解原问题，$i > j$ 的情况贡献为

$$Ans = \sum_{i>j} [A_i > A_j] f_{n,i}$$

，使用树状数组统计 $Ans = \sum_{i>j} [A_i > A_j]$，$i < j$ 的部分使用类似的做法即可做到总时间复杂度 $\mathcal{O}(n \log n)$。

引理 3：长度为 n 的环，1 先于 2 消失的所有情况下 $(n-t)!$ 的期望与 1 先于 n 相同。

证明：枚举 1 是在第 $i+1$ 次经过时消失的，那么 $2..n$ 的每个点经过次数相同，由**引理 1** 可知它们消失的概率相同，因此在这个问题中是等价的。

由**引理 3**，在处理 $i < j$ 的情况时可以节省一定的代码量。

效果

Best solutions for Problem 6339

Language : All G++ GCC C++ C Pascal Java

Rank	Author	Exe. Time	Exe. Memory	Code Len.	Language	Date
1	supy	0MS	1388K	3596B	G++	2020-07-06 10:37:33
2	马孟起	312MS	2100K	1066B	G++	2020-07-04 15:24:03
3	JZdavid	327MS	4568K	3591B	G++	2018-08-01 22:41:27
4	orbitingflea	343MS	2264K	2534B	G++	2018-08-02 10:51:36
5	zhou8888	483MS	3956K	3369B	G++	2019-04-15 23:30:10
6	laofudasuan	483MS	59336K	3524B	G++	2018-08-02 19:44:27
7	Emma Yin	546MS	2112K	2184B	G++	2018-08-02 16:14:50
8	赵子龙	577MS	12608K	4248B	G++	2020-07-02 17:05:33
9	OMTWOCZWEIXVI	592MS	57648K	2670B	G++	2018-09-19 11:26:54
10	et1999	608MS	2560K	3580B	G++	2019-03-22 14:47:28
11	20032019	748MS	6496K	3221B	G++	2018-08-06 19:05:46
12	kut_msm	764MS	120688K	2557B	G++	2018-08-02 11:42:45
13	SYC	951MS	2456K	3244B	G++	2018-08-01 17:53:43
14	FlappyFish	1060MS	6280K	2919B	G++	2018-08-02 21:37:41
15	wxh010910	1294MS	114020K	2642B	G++	2018-08-30 15:22:35
16	Windows 64 bit on Wi	1388MS	226852K	13199B	G++	2018-08-07 19:19:41
17	SCUT_Judge	1435MS	3632K	5071B	G++	2019-06-25 12:38:01
18	大弱逼	1903MS	108108K	2346B	G++	2018-08-01 20:26:20
19	赵丽斌	2090MS	112464K	2502B	G++	2018-08-09 21:03:31
20	jnxxhzz	2199MS	112468K	2500B	G++	2019-09-24 18:31:49

小结

此题很好地体现了代数推导的局限性，不仅繁琐而且无法做到很优的复杂度。

而笔者提供的做法，通过对问题模型进行观察，得到一个很强的性质，从而得到一种 $\mathcal{O}(n^2)$ 的简洁的 dp 做法。

类似例题 1，指针初始位于 1 的情况是容易解决的，这之后 dp 的转移无环，可以使用常见的生成函数技巧进行优化，从而做到 $\mathcal{O}(n \log n)$ 的优秀复杂度，这相对于官方题解给出的 $\mathcal{O}(n^3)$ 是一个巨大的飞跃。

试题来源、研究思路与成果

笔者最初在做 2018 Multi-University Training Contest 4,Problem H 时思考出 $\mathcal{O}(n^2)$ 的做法，比官方题解的推式子做法更优秀。这启发我对这一问题模型进行研究。之后我发现这一模型存在一些基本结论和观察，而且不同问题在解法上也有一定的联系。例如 Cometoj Contest 4,Problem E 就是先将指针所在位置移动到 1，从而拆解问题，事实上笔者也正是受这个思路启发，才思考出了例题 3 最后一步的优化。在研究过程中，笔者对例题 1 进行修改并进行了深入的思考，最终命制了例题 2，此题虽和例题 1 题意比较相似，做法却大相径庭，最后几乎只需要一些推导的技巧就可以解决，这说明代数推导在一些情况下也有用武之地，选手在做这类题目时需要灵活选择解法，不能思维定式化。

总结

本文通过三个例题展示了解决“一类基于概率的约瑟夫问题”的各种思路和技巧，对这类问题进行了详细的分析研究，最终都给出了 $\mathcal{O}(n\log n)$ 的解法（我相信这也是理论下界），希望读者遇到这类问题时有迹可循。另外，对一个模型进行深入研究，对相关问题进行对比分析也提供了一种很好的出题方向，例如本文例题 2 就是这样命制的。最后，希望本文起到抛砖引玉的作用，吸引更多读者来研究这一类问题，或是将本文提及的思想应用到更多其它问题上。

Thanks

感谢大家的聆听
祝大家冬令营正常发挥