# KMP、TRIE与 AC-automaton水题选讲

Yali-lizongru

# KMP

- 一种高效的字符串匹配算法。
- 精髓在于其next数组的实现。
- 时间复杂度为O(n+m)。
- 具体实现大家应该都知道，就不浪费时间了。

# P2353 背单词

## Description

老师给了小明一篇长度为N的英语文章，然后让小明背$M$个单词。为了确保小明不会在背单词时睡着，老师会向他提$Q$个问题，每次老师随机选择一个区间$L..R$，小明要回答在这段文字中他背过的单词总共出现过多少次。

## Constraints

$1 <= N <= 10^6, 1 <= M <= 10, 1 <= Q <= 10^6$

$1 <= length\ of\ each\ word <= N, 1 <= L <= R <= N$

# Solution

由于$m$很小，可以将每一个单词做一个模式串与文章（文本串）做$KMP$。

匹配得出的位置可以记一个前缀和，每次询问即可在$O(m)$的时间内得出答案。

# P3193 [HNOI2008] GT考试

## Decription

阿申准备报名参加 GT 考试，准考证号为$N$位数 $X_1, X_2 \dots X_n (0 \le X_i \le 9)$，他不希望准考证号上出现不吉利的数字。 他的不吉利数字$A_1, A_2 \dots A_m (0 \le A_i \le 9)$有$M$位，不出现是指 $X_1, X_2 \dots X_n$中没有恰好一段等于 $A_1, A_2 \dots A_m$， $A1$和 $X_1$可以为$0$。

## Constraints

$N \le 10^9, M \le 20, K \le 1000$

# Solution

题意即准考证号和不吉利数字做一遍$KMP$，匹配不到结果。

设 $f[i][j]$ 表示长串匹配到第 $i$ 位，短串最多可以匹配到第 $j$ 位的方案数。那么为了让它不能找到完全的匹配，答案就是$\sum_{i=0}^{m-1} f[n][i]$。

想一想怎么转移，考虑对于已经匹配了的$f[i][j]$转移到 $f[i+1][k]$新加一个新的字符造成的影响。

- 如果下一位继续与不吉利数字匹配，则转移到$f[i+1][j+1]$。
- 否则转移的式子是$f[i][j] = \sum_{k=0}^{m-1} f[i-1][k] \times g[k][j]$，$g[i][j]$表示从短串的第$i$位转换到第$j$位的方案数。

这里由于短串已知，所以这个$g$数组是可以使用$KMP$预处理出来的，复杂度为$O(10 \times M)$。

所以这个$DP$的复杂度是$O(NM^2)$的，无法得到满分。

继续观察可以发现这个转移的模式固定，考虑矩阵快速幂优化，即可获得满分。

# P2375 [NOI2014]动物园

## Description

$KMP$算法只能求出$next$数组。我现在希望求出一个更强大$num$数组——对于字符串$S$的前$i$个字符构成的子串，既是它的后缀同时又是它的前缀，并且该后缀与该前缀不重叠，将这种字符串的数量记作$num[i]$。

例如$S$为$aaaaa$，则$num[4] = 2$。这是因为 $S$ 的前 4 个字符为$aaaa$，其中$a$和$aa$都满足性质'既是后缀又是前缀'，同时保证这个后缀与这个前缀不重叠。而$aaa$虽然满足性质'既是后缀又是前缀'，但遗憾的是这个后缀与这个前缀重叠了，所以不能计算在内。同理， $num[1] = 0, num[2] = num[3] = 1, num[5] = 2$ 。

求一个串的$num$数组。

## Constraints

$1 \leq length \leq 10^6$

# Solution

$num$不同于$next$记录的是一个最大值，它记录的是一个和值。考虑一个前缀 $i$ 的 $next[i]$ ， $next[next[i]]$，$next[next[next[i]]]$ …… …都是这个前缀串i的公共前后缀，而且只有它们是公共前后缀。

我们其实只要在求 $next$ 的过程中，顺便把这个公共前后缀的数量递推一下，就得到了一个弱化版的 $num$ 数组：可以重叠的公共前后缀数量，我们称之为 $ans$ 。

考虑去重：首先 $next$ 数组有一个性质： $next[i] < i$ 也就是说，一旦有一个递归了$n$层的$next$，比原前缀$i$的长度的一半要小，那么这个$next$的递推出的答案 $ans$ 就是$i$的 $num$ 了。

假如我们拿到的串是$10^6$个'$a$'，那么上面那个算法就会被卡成$O(len^2)$。显然是无法接受的，我们应该考虑如何优化这个算法的复杂度。

这个算法的瓶颈就在于一直重复递归，这和$KMP$要解决的问题非常相似，自然可以想到用类似于$KMP$的算法解决：首先在求$next$数组的同时将$ans$数组预处理出来，在用一个辅助变量$j$，在求解时不断更新保证$j$的位置一定在$\frac{i}{2}$左边，这时的$ans[j]$符合要求。

# TRIE

- 字典树，一个用于查找的数据结构
- 本质上是一个DFA(确定有限状态自动机)，对于一个给定的属于该自动机的状态和一个属于该自动机字母表的字符，它都能根据事先给定的转移函数转移到下一个状态。
- 应该也没有人不知道。

# P2292 [HNOI2004]L语言

## Description

给定一个字典$D$， 判断若干段文章在字典$D$下是否能够被理解。并给出其在字典$D$下能够被理解的最长前缀的位置。

## Constraints

$1 \leq n, m \leq 20$

$length\ of\ each\ word \leq 10$

$length\ of\ each\ passage \leq 10^6$

# Solution

对于字典建一棵Trie。

设$dp[i]$记文章在$i$前的部分都可以匹配。

枚举文章的每一个字符，再枚举字典里的串，如果这个位置可以匹配，则将文章放在Trie上跑。

如果文章的这一部分与这个串匹配，设这个串为str,则$dp[i+strlen(str)]=1$

最后一位$dp$值为$1$的位置即答案。

# P3369 【模板】普通平衡树

**这个题可以用Trie来做。**

1. 插入$x$数
2. 删除$x$数(若有多个相同的数，因只删除一个)
3. 查询$x$数的排名(排名定义为比当前数小的数的个数 +1+1 。若有多个相同的数，因输出最小的排名)
4. 查询排名为$x$的数
5. 求$x$的前驱(前驱定义为小于$x$，且最大的数)
6. 求$x$的后继(后继定义为大于$x$，且最小的数)

# Solution

将每个数字二进制拆分（需要加上$10^7$使得都是非负整数）。

1. 插入:从高位到地位插入到$01Trie$中，给最后节点一个的$val$加一。
2. 删除:从高位到地位插入到$01Trie$中，给最后节点一个的$val$减一。
3. 查询$x$的排名:在插入和删除时维护每个节点的$size$，就可以按照普通二叉搜索树那样向下递归查找了。
4. 查询排名为$x$的数:与上一个操作类似。‘
5. 查前驱:先找出这个数的排名$rank$，再找出排名为$rank$的是哪个数。
6. 查后继:与上一个操作类似。

# BZOJ 3689 异或之

## Description

给定$n$个非负整数$A[1], A[2], \ldots\ldots, A[n]$。

对于每对$(i, j)$满足$1 \leq i < j \leq n$，得到一个新的数$A[i] xor A[j]$，这样共有$\frac{n \times (n-1)}{2}$个新的数。求这些数（不包含$A[i]$）中前$k$小的数。

## Constraints

$2 \leq n \leq 100000, 1 \leq k \leq min(250000, \frac{n \times (n-1)}{2})$

$0 \leq A[i] < 2^{31}$

# Solution

首先我们对所有数字建立二进制$Trie$树，可以利用$Trie$树上的$size$域查询出一个数与其它数异或值的第$k$小

然后我们维护一个堆，将所有数与其它异或值的第$2$小加入堆(第一小是自己异或自己，不在题目要求范围内)，当取出一个数异或值的第$k$小后，将第$k+1$小加入堆

一个异或值会被两个数分别取出一次，所以取出奇数次时输出，取$2 \times k$次即可

# P3294 [SCOI2016]背单词

## Description

总共有$n$个单词，对于一个序号为$x$的单词(序号 $1 \dots x-1$都已经被填入):

1. 如果存在一个单词是它的后缀，并且当前没有被填入表内，代价为$n \times n$才能学会;
2. 当它的所有后缀都被填入表内的情况下，如果在$1 \dots x-1$的位置上的单词都不是它的后缀，那么代价为$x$;
3. 当它的所有后缀都被填入表内的情况下，如果$1 \dots x-1$的位置上存在是它后缀的单词，所有是它后缀的单词中，序号最大为$y$，那么代价为$x-y$。

求最小代价。

## Constraints

$1 <= n <= 100000$

$\sum length \leq 510000$

# Solution

第一种情况一定不会存在——因为我们总能避免这种情况。而且在避免这种情况的情况下，最大代价不会超过$n \times n$。现在相当于，题目多了一个限制条件，每个字符串的后缀必须先被填入。我们把每个字符串$reverse$，转化为每个字符串的前缀必须先被填入。

现在问题就转化为填入一个字符串后，以它为前缀的字符串填入顺序如何确定。

首先建一棵Trie树，将无用的节点省去，这样每一个节点都表示一个单词。问题即转化为给这棵树上的节点编号。

因为没有第一种情况出现，父亲节点总是小于孩子节点的编号，考虑贪心，即向$size$更小的走，得出的答案总是最小的。

# P2536 [AHOI2005]病毒检测

## Descrirption

每个$RNA$片段都是由$A$、$C$、$T$、$G$组成的序列。科学家们也总结出了$Samuel$星球上的“病毒模版片段”。一个模版片段是由$A$、$C$、$T$、$G$的序列加上通配符$*$ 和 ？来表示。其中$*$的意思是可以匹配上$0$个或任意多个字符，而 ？的意思是匹配上任意一个字母。

如果一个$RNA$片段能够和“病毒模版片段”相匹配，那么这个$RNA$片段就是未知的病毒。

例如，假设“病毒模版片段”为$A*G?C$。$RNA$片段：$AGTC$，$AGTGTC$都是未知的病毒，而$RNA$片段$AGTGC$则不是病毒。

现在请你编写程序统计$N$个$RNA$片段中哪些$RNA$片段不是病毒。

## Constraints

$N \leq 500$
“病毒模版片段”的长度$\leq 1000$
“$RNA$片段长度”$\leq 500$

# Solution

观察到这道题范围很小，我们可以考虑建一棵$Trie$在上面进行$bfs$暴力匹配。

在队列中用$pair$记录两个值，$first$表示走到$Trie$上的节点位置，$second$表示匹配到模板串的位置。

下面考虑各种转移的情况：

- $s[v+1]=' A' or ' T' or ' C' or ' G'$ 时：可扩展节点$make_pair(ch[u][' A' or ' T' or ' C' or ' G'], v+1)$
- $s[v+1]=' ?'$ 时：可扩展节点$make_pair(ch[u][' A' and ' T' and ' C' and ' G'], v+1)$
- $v[v+1]=' *'$ 时，可扩展 $\{make\_pair(ch[u][' A' and ' T' and ' C' and ' G'], v), make\_pair(u, v+1)\}$
- 直到到了一个有值的位置，就统计当前位置上面的答案。

然而这样做会$T$几个点，我们考虑优化这个算法。由于搜索时有很多重复状态，我们可以使用一个$vis$数组记录已经搜过哪些位置。注意这里必须用$bitset$不然会$MLE$。

这道题细节有点多，写的时候要小心一点。

# AC-automaton

- 用于在输入的一串字符串中匹配有限组“字典”中的子串。它与普通字符串匹配的不同点在于同时与所有字典串进行匹配。算法均摊情况下具有近似于线性的时间复杂度，约为字符串的长度加所有匹配的数量。
- 主要依靠构造一个有限状态机（类似于在一个trie树中添加失配指针）来实现。这些额外的失配指针允许在查找字符串失败时进行回退（例如设Trie树的单词cat匹配失败，但是在Trie树中存在另一个单词cart，失配指针就会指向前缀ca），转向某前缀的其他分支，免于重复匹配前缀，提高算法效率。
- 这个东西也不需要介绍了吧。

# P3966 [TJOI2013]单词

## Description

小张最近在忙毕设,所以一直在读论文。一篇论文是由许多单词组成但小张发现一个单词会在论文中出现很多次,他想知道$N$个单词分别在论文中出现了多少次。

## Constraints

$N \leq 200$

$\sum length\ of\ the\ words \leq 10^6$

## Solution

$AC$自动机模板题。$AC-automaton$可以用来多模式串匹配。对所有的单词建$AC-automaton$，在所有单词中间加一个非字母字符隔开组成论文。在$AC-automaton$中匹配论文串，记录每个串的匹配次数即可。

还有一种~~比较头铁~~的做法。由$AC-automaton$的性质可以知道，我们要求的答案即为该节点在$fail$树上子树的和，然后直接暴力把$fail$树建出来，树形$DP$统计答案即可。

# P4052 [JSOI2007]文本生成器

## Description

「文本生成器」可以随机生成一些文章 —— 总是生成一篇长度固定且完全随机的文章 —— 也就是说，生成的文章中每个字节都是完全随机的。

如果一篇文章中至少包含使用者们了解的一个单词，那么我们说这篇文章是可读的（我们称文章 $a$ 包含单词 $b$ ，当且仅当单词 $b$ 是文章 $a$ 的子串）。

求生成的所有文本中可读文本的数量。

## Constraints

单词总数$n \leq 60$

任意单词及文本长度$length \leq 100$

# Solution

首先考虑只有一个单词的情况，即这个单词出现了，当且仅当之前连续若干个位置匹配了单词的前面若干个字符，并且当前字符是单词的最后一个字符。

我们可以设计 DP 状态 —— 还要生成$i$个字符，在这之前生成的最后若干个字符匹配了单词中的前$j$个字符，最终生成串不包含单词串方案数。枚举第一个字符，尝试继续匹配，如果不能匹配则跳转到$KMP$的失配位置。

再考虑有多个单词的情况，则将$KMP$转换为$AC-automaton$即可，将$KMP$跳失配转换为$AC-automaton$跳$fail$。

# P3311 [SDOI2014]数数

## Description

我们称一个正整数$N$是幸运数，当且仅当它的十进制表示中不包含数字串集合$S$中任意一个元素作为其子串。例如当$S = (22 , 333 , 0233)$时，$233$是幸运数，$2333$、$20233$、$3223$ 不是幸运数。 给定 $N$和 $S$，计算不大于$N$的幸运数个数。

## Constraints

- $l$ 表示 $N$ 的长度，$L$ 表示 $S$ 中所有串长度之和。
- $1 <= l <= 1200, 1 <= M <= 100, 1 <= L <= 1500$

# Solution

首先将所给的非幸运数字建一个$AC-automaton$。

$f[i][j]$表示当前计算到第$i$位，在$trie$树上跑到了第$j$位的幸运数字的个数。

由于这道题多了一个不能大于$N$的限制 我们可以把$DP$数组开成这样。

$f[0][i][j]$表示当前这个位置的数需要受到限制（也就是不能超过所给整数的这一位）。

$f[1][i][j]$表示当前这个位置的数不需要受到限制（之前已经有一位小于所给整数的对应位置）。

在枚举$f[0][i][j]$的时候如果这一位的数字比所给数对应位置上的数字小就可以转移到$f[1][i][j]$。

然后在$AC-automaton$上跑一遍就行了。

# P2414 [NOI2011]阿狸的打字机

## Description

打字机上只有28个按键，分别印有26个小写英文字母和$'B'$、$'P'$两个字母。经阿狸研究发现，这个打字机是这样工作的：

- 输入小写字母，打字机的一个凹槽中会加入这个字母(这个字母加在凹槽的最后)。
- 按一下印有$'B'$的按键，打字机凹槽中最后一个字母会消失。
- 按一下印有$'P'$的按键，打字机会在纸上打印出凹槽中现有的所有字母并换行，但凹槽中的字母不会消失。

例如，阿狸输入$aPaPBbP$，纸上被打印的字符如下：$a\ aa\ ab$。我们把纸上打印出来的字符串从1开始顺序编号，一直到$n$。打字机有一个非常有趣的功能，在打字机中暗藏一个带数字的小键盘，在小键盘上输入两个数$(x,y)$(其中$1 \leq x,y \leq n$)，打字机会显示第$x$个打印的字符串在第$y$个打印的字符串中出现了多少次。

阿狸发现了这个功能以后很兴奋，他想写个程序完成同样的功能，你能帮助他么？

## Constraints

$n \leq 100000, m \leq 100000$,第一行总长度$\leq 100000$。

# Solution

每次打印出字符串后，就插入到$Trie$树中。搞完后直接搭 $AC-automaton$。看一看匹配是怎么样的：每次沿着$AC-automaton$走，在每一个节点都跳$fail$指针如果有$x$串的末节点，就给答案$+1$这样的话没有必要存下每个串只要给$AC$自动机存一个父亲节点记录一下每个串的结束位置倒着往上跳就可以了。
但是这样显然是会T的，考虑优化这个算法。

由$AC-automaton$的性质可以知道每个节点有且仅有一个$fail$指针，所以$fail$指针会成为一棵树。原来是$y$的某个节点往上跳能不能到达$x$而在$fail$树中反过来：$x$往下跳能够到达几个$y$的节点。所以就把$y$的节点值设为$1$，再求子树和即可。由于子树中$dfs$序相邻，所以可以用一个树状数组来维护子树和。

然而这样还是会$T$，因为我们反复用树状数组插了很多串进去，我们$dfs$这棵$Trie$，访问到的时候$tag+1$结束的时候$tag-1$，类似于差分的优化，就可以过了。

# P3715 [BJOI2017]魔法咒语

## Description

给定$n$个基本词汇，$m$个禁忌词汇，长度$L$。求使用基本词汇组成长度为$L$的咒语的方案数，咒语中不能出现禁忌词汇。组成方式不同即被认为是不同的禁咒法术。

## Constraints

$1 \leq n,\ m \leq 50, 1 \leq l \leq 10^8$
基本词汇长度之和$\leq 100$，禁忌词汇长度之和$\leq 100$。

# Solution

经典的$AC-automaton$上计数$dp$的套路。设$dp[i][j]$为到了咒语的第$i$个字符，$AC-automaton$上第$j$个节点的方案数。

由于我们每次必须选择一个整单词，所以我们刷表枚举每一个可行的$dp$状态进行拓展，转移的时候枚举选哪一个单词，然后如果这个单词在跑的过程中都没有经过一个结束节点，那么我们就可以转移，否则不行，假设我们到达了$p$节点，那么$dp[i+len][p]+=dp[i][j]$就可以了。设单词长$len$，$AC-automaton$节点数为$t$，则这个算法的复杂度为$O(l\times n\times t\times len)$的，显然会$T$。

显然矩阵快速幂优化一下就好了。

# BZOJ1444 [JSOI2009]有趣的游戏

## Description

小阳阳发明了一个有趣的游戏：有 $n$ 个玩家，每一个玩家均有一个长度为 $l$ 的字母序列，任何两个玩家的字母序列不同。共有 $m$ 种不同的字母，所有的字母序列都由这 $m$ 种字母构成，为了方便，我们取大写字母的前 $m$ 个字母。例如 $m=3, l=4$, ABAA, CBCA 为两个合法的字母序列。现在由小阳阳来操控一台神奇的机器，每个时刻机器会随机产生一个字母，其中第 $i$ 种字母随机出来的概率为 $\frac{p_i}{q_i}(0 \le p_i \le q_i)$，显然 $\sum_{k=1}^{m} \frac{p_i}{q_i} = 1$。这样 $T$ 个时刻后机器会产生一个长度为 $T$ 的字母序列。 如果某个时刻某个玩家发现自己的字母序列在机器产生的字母序列中出现了，“出现”的定义是玩家的字母序列是机器产生的字母序列中连续的一段，那么我们称这个玩家获胜，游戏结束。现在小阳阳感兴趣的一个问题是，每个玩家分别有多大的概率能获得这场游戏的胜利呢?

$0 \le p, n, l, m \le 10$

# Solution

- Sol1:
  我们要求的答案是每个人获胜的概率，实际上，如果我们把所有单词建立出 AC 自动机，我们要求的就是到达每个人对应的单词末尾节点的概率。我们定义$x_i$为一局比赛**经过** $i$ 点的概率，则那么 $x_i$就等于所有（$i$ 的前驱节点的概率 $\times i$ 上的字符出现的概率）的和，根据这个可以列出方程。
  然而这样解出来的答案都是$0$，原因是因为常数项都是 $0$，这是个 $n$ 元齐次方程组，存在全 $0$ 解。所以我们考虑把$x_0$定为$1$(因为根节点的概率本来就是$1$)。然而这样还是不对，因为从根节点跑出去后，又会回到根节点，所以会导致根节点的概率大于$1$。
  这个解法无药可救了？不是的，我们可以考虑用时间换精度，多次转移，矩幂优化，精度就能卡进两位小数之内（和我的模拟赛的T2谜之相似）。转移$K$次复杂度$O((nl)^3 \times log(k))$。

- Sol2:

  刚刚的瓶颈在于无法处理根节点的多次转移，不过有一种巧妙的方法可以解决。

  考虑定义$x_i$为一局比赛经过$i$点的**期望次数**。这样的好处是什么呢？显而易见的是由于单词末尾节点走到了，意味着游戏结束，所以经过单词末尾节点的概率就是经过单词末尾节点的期望次数。第二个好处是对于根节点的，由于没有了概率上限为 1 的限制，我们可以自由的转移，则我们可以得到初始值：$x_0 = 1$

  设$P_{i,j}$为$j$转移到$i$的概率，根据全期望公式（$i$ 的期望$=$ $i$的前驱节点的期望$\times i$上的字符出现的概率），根节点的方程是：$x_0 = 1 + P_{0,0}x_0 + P_{0,1}x_1 + \ldots + P_{0,n}x_n$

  其余的节点的方程是：考虑求$P_{i,j}$。设 $p[i]$ 为生成字符 $i$ 的概率，$\text{char}(i)$为节点 $i$ 上的字符。如果 $x_u$能转移到 $x_v$ ，那么 $P_{v,u}$加上 $P[\text{char(v)}]$。如果 $u$ 是单词结尾节点或者是能通过 $\text{fail}$ 到达单词结尾节点的点，表示游戏结束，不再转移，所以直接略去其转移即可。

  由于我们一局比赛走到末尾就停止，所以所有末尾节点经过次数的的期望和一定是1。使用高斯消元来解方程，复杂度$O((nl)^3)$。

  还有如果一个人赢不了需要特判，否则就会输出的都是$nan$。

# P3706[SDOI2017]硬币游戏

## Description

用$H$表示正面朝上，用$T$表示反面朝上，扔很多次硬币后，会得到一个硬币序列。比如$HTT$表示第一次正面朝上，后两次反面朝上。 但扔到什么时候停止呢？大家提议，选出$n$个同学，每个同学猜一个长度为$m$的序列，当某一个同学猜的序列在硬币序列中出现时，就不再扔硬币了，并且这个同学胜利，为了保证只有一个同学胜利，同学们猜的$n$个序列两两不同。 你想知道，如果硬币正反面朝上的概率相同，每个同学胜利的概率是多少。

## Constraints

$$1 \leq n, m \leq 300$$

# Solution

这道题和上一道题题意几乎一模一样，不过从生成$m$个字符变为了生成2个，然后数据范围大了很多。

所以直接建$AC-automaton$肯定会$T$,考虑优化算法。

我们需要的只是$n$个目标状态的答案，而不需要$AC-automaton$上其他点的答案，那么这些点是否可以合并起来，统一考虑呢?

设期望经过非单词末尾节点的次数为 $x_0$，经过序列 $i$的末尾节点的期望次数为 $x_i$ ，根据上一题中的讨论，期望经过次数 $x_i$ 就是第 $i$ 个同学获胜的概率。可以列出第一个方程$x_1+x_2+\ldots+x_n=1$

假设有3个同学，设他们给出的字符串分别为$THT$、$TTH$、$HTT$，则设$S_1=THT,S_2=TTH,S_3=HTT$。考虑一个状态$X$，表示所有没有达到目标状态的字符串。所以在$X$后面接上一个$S$中的字符串就会达到，目标状态，游戏结束，但是还有可能我们没有填完选定的这一个字符串就产生了其他的$S$中的字符串 。

例如设$A=\mathrm{TTH},B=\mathrm{THT}$，如果非单词的末尾状态的最后两个字符是$TH$，在后面加上$A$就会走到$B$而提前结束游戏。因为$A$的长度为1的前缀等于$B$长度为1的后缀。由于非单词末尾状态最后两个字符是出现的概率是均等的，我们固定了非单词末尾状态最后两个字符，所以这种情况出现的概率为$\frac{1}{2^2}$。

不难推广到一般情况，对于$i$，枚举另外一个序列$j$，使用 KMP计算出所有$i$的长度为$k$的前缀是$j$的一段长度为$k$的后缀的情况，并分别累加概率得到$P_{i,j}=\sum_k \frac{1}{2^{m-k}}$。

并可以得到$n$个方程：$x_i+\sum_{i=1}^{n} P_{i,j}x_i=\frac{1}{2^m}x_0$。现有$n+1$个方程与$n+1$个未知数，高斯消元解方程即可。

# THANKS.