字符串乱讲

qwq123

CJ 2020 信息组

2024 年 1 月 29 日

前言

看了一下最近几年的题目，发现字符串这个东西基本没有怎么出现诶

你可以说他不常考，但不能说他不重要。

个人认为，字符串算法的难点在于理解算法过程，只有真正弄懂这一个算法的本质 & 功能，才可以知道这个算法究竟能处理什么样的问题，在打上板子之后应该怎么往下写。

今天的任务

1. Manacher、exKMP
2. KMP
3. AC 自动机
4. SA,SAM
5. PAM
6. 题目

Manacher

我的评价，真不如 PAM。

理论上 PAM 应该是可以解决绝大多数的 Manacher 问题的。

那么我就在后面详细介绍一下 PAM。

exKMP

$z[i]$ 表示 S 和 $S[i, n-1]$ 的最长公共前缀的长度

算法的过程中我们维护右端点最靠右的匹配段。为了方便，记作 $[l, r]$ 。

算法流程：

- 如果 $i \le r$，那么根据 $[l, r]$ 的定义 $S[i, r] = S[i-l.r-l]$，因此 $z[i] \ge \min(z[i-l], r-i+1)$。这时：
  - 若 $z[i-l] < r-i+1$，则 $z[i] = z[i-l]$。
  - 否则 $z[i-l] \ge r-i+1$，这时我们令 $z[i] = r-i+1$，然后暴力枚举下一个字符扩展 $z[i]$ 直到不能扩展为止。
- 如果 $i > r$，那么我们直接暴力求出 $z[i]$ 。
- 在求出 $z[i]$ 后，如果 $z[i] > r$ ，我们就需要更新 $[l, r]$ ，即令 $l = i, r = i + z[i] - 1$ 。

exKMP

题目？本人只知道 [NOIP2020] 字符串匹配了…

想讲同样欢迎幸运观众上来讲题。

KMP

KMP 是最基础的字符串算法，核心是处理出 next 数组，即最长的相等的真前缀与真后缀的长度。

很多人会认为，KMP 只会用于字符串匹配，虽然 KMP 的用途大部分是匹配，但把 KMP 与匹配问题挂钩是不能理解 KMP 本质的。

话说 KMP 模板竟然变绿了，之前还是橙题的...

KMP

next 数组存的是啥？最长后缀等于前缀

但他存下来的不仅仅是一个最长长度，还包含了一些别的信息。关于 next 数组的一些性质：

- next 是帮助我们快速退回前一个状态的链接。
- next 的对应关系会构成一棵树。（字符串算法很多都和树相关）
- next 与 border 直接挂钩...

下面还是以题目来说明

KMP

P3435 [POI2006] OKR-Periods of Words

求给定字符串所有前缀的最大周期长度之和。
周期的定义为：周期 Q 是 a 的真前缀且 a 是无数个 Q 相连的前缀。

KMP

P3435 [POI2006] OKR-Periods of Words

求给定字符串所有前缀的最大周期长度之和。
周期的定义为：周期 Q 是 a 的真前缀且 a 是无数个 Q 相连的前缀。

题解

手动画图可知：字符串 S 的周期就是 |S|-next[|S|]，|S|-next[next[|S|]]...
而题目所要求的最长周期就是要找到 $\min\{$next[|S|],next[next[|S|]]...$\}$

KMP

P2375 [NOI2014] 动物园

子符串 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]，求出所有 num[i]

KMP

P2375 [NOI2014] 动物园

子符串 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]，求出所有 num[i]

题解

经典题了，欢迎幸运观众上来讲题。

KMP

P5829 【模板】失配树

给定一个字符串 s，定义它的 k 前缀 pre_k 为字符串 $s_{1\dots k}$，k 后缀 suf_k 为字符串 $s_{|s|-k+1\dots|s|}$，其中 $1 \le k \le |s|$。

定义 $\textbf{Border}(s)$ 为对于 $i \in [1, |s|)$，满足 $pre_i = suf_i$ 的字符串 pre_i 的集合。$\textbf{Border}(s)$ 中的每个元素都称之为字符串 s 的 border。

有 m 组询问，每组询问给定 p, q，求 s 的 $\boldsymbol{p}$ 前缀和 $\boldsymbol{q}$ 前缀的最长公共 border 的长度。

KMP

P5829 【模板】失配树

给定一个字符串 s，定义它的 k 前缀 pre_k 为字符串 $s_{1\dots k}$，k 后缀 suf_k 为字符串 $s_{|s|-k+1\dots|s|}$，其中 $1 \le k \le |s|$。

定义 $\textbf{Border}(s)$ 为对于 $i \in [1, |s|)$，满足 $pre_i = suf_i$ 的字符串 pre_i 的集合。$\textbf{Border}(s)$ 中的每个元素都称之为字符串 s 的 border。

有 m 组询问，每组询问给定 p, q，求 s 的 $\boldsymbol{p}$ 前缀和 $\boldsymbol{q}$ 前缀的最长公共 border 的长度。

题解

也算是经典题？

这个题展示了 next 的对应关系会构成一棵树。

KMP 自动机

现在我们就要初步理解自动机是在干什么。

我们喂给自动机一个字符串 T，他可以帮助我们尽可能的将去 T 的后缀匹配预先设置好的 S 串。

如何构建？就是对于每个 i 和 c，求出 ne[i][c]，表示'S[1,i]+c' 与 S 能匹配多少位。这里我们是通过 next 实现快速转移来求解的。

KMP 自动机

P3193 [HNOI2008] GT 考试

给一个数字串 S，求有多少长度为 n 的数字串 T 不包含 S 作为子串。$n <= 10^9$

KMP 自动机

P3193 [HNOI2008] GT 考试

给一个数字串 S，求有多少长度为 n 的数字串 T 不包含 S 作为子串。$n <= 10^9$

题解

也算是经典题了，要重新讲的话也欢迎幸运观众上来讲题。

AC 自动机

对于 AC 自动机，你也需要理解他在干什么

实际上 AC 自动机就是多个字符串下的 KMP 自动机，能将我手上的一个字符串拆成一个个字符，每次单独加入（转移），它能尽可能的匹配到最长的包含这个字符的一串

只不过，相对于 KMP 自动机，AC 自动机可能会匹配过了头，我们无法判断这最长的一串中有没有包含模板串，也就是带有结束标记的状态，这时候就需要 fail 树上找它的祖先，有没有这种状态。

AC 自动机

P2414 [NOI2011] 阿狸的打字机

给出一棵 n 个节点的 Trie 树，q 次询问 Trie 树上 x 节点代表的字符串在 y 节点代表的字符串中出现了多少次。

AC 自动机

P2414 [NOI2011] 阿狸的打字机

给出一棵 n 个节点的 Trie 树，q 次询问 Trie 树上 x 节点代表的字符串在 y 节点代表的字符串中出现了多少次。

题解

如何处理某一个文本串 S 在另一个模式串 T 中出现的个数？

那么问题就等价于 Trie 树上根节点到 y 路径上的所有点中有多少点在 fail 树上 x 的子树里，树状数组维护即可。

AC 自动机

P3041 [USACO12JAN] Video Game G

给定 n 种特定的字符串 $S_1 ... S_n$（只包含 A，B，C）。你要构造一个长度为 m 的字符串，使包含各个字符串的次数的总和最大

数据范围：$n \leq 20, |S_i| \leq 15, m \leq 10^3$。

AC 自动机

P3041 [USACO12JAN] Video Game G

给定 n 种特定的字符串 $S_1 ... S_n$（只包含 A，B，C）。你要构造一个长度为 m 的字符串，使包含各个字符串的次数的总和最大

数据范围：$n \le 20, |S_i| \le 15, m \le 10^3$。

题解

设 $f[i][j]$ 表示长度为 i 且当前在 AC 自动机的匹配状态是 j 时最大出现次数，转移就是 $f[i][tr[i][k]] \leftarrow f[i-1][j] + cnt[tr[i][k]]$

其中 $cnt[p]$ 表示状态 p 的祖先出现次数之和（可以提前通过祖先向子树算贡献）

AC 自动机

P2444 [POI2000] 病毒

给定 n 个 01 串，问存不存在无限长的 01 串，使其不包含任意一个 01 串

AC 自动机

P2444 [POI2000] 病毒

给定 n 个 01 串，问存不存在无限长的 01 串，使其不包含任意一个 01 串

题解

如果存在无限长不能匹配的字符串，我们把它放在 AC 自动机里匹配，会出现什么情况？

所以我们只需要判断 trie 图中是否存在未被标记的环。

后缀数组

对于后缀数组，其实是比较重要的，但尴尬的地方在于后缀数组的很多操作可以被后缀自动机完美实现。

看看要不要讲解一下算法流程。Blog。

通过后缀排序，第 i 后缀和第 j 后缀的 LCP 就是 $\min_{id_i\le k\le id_j}\{ht_k\}$，sa 的难点在于对 ht 数组的运用。

后缀自动机

来到了字符串部分难点中的难点。虽然是 10 级，但是如果你不会写 sa 那么就还是老老实实学一学吧

SAM 真的是一个很神奇的东西，可以用 $O(n\sum)$ 的时间空间复杂度维护一个字符串的所有子串。

同样看看要不要讲解一下算法流程。Blog。

题目

P5341 [TJOI2019] 甲苯先生和大中锋的字符串

在字符串中恰好出现了 k 次的子串中，按照字串的长度分类，求出现数量最多的那一类的长度。

题目

P5341 [TJOI2019] 甲苯先生和大中锋的字符串

在字符串中恰好出现了 k 次的子串中，按照字串的长度分类，求出现数量最多的那一类的长度。

题解

SA 怎么处理恰好出现 k 次？

题目

P5341 [TJOI2019] 甲苯先生和大中锋的字符串

在字符串中恰好出现了 k 次的子串中，按照字串的长度分类，求出现数量最多的那一类的长度。

题解

SA 怎么处理恰好出现 k 次？

长度在 $\max(ht[i+1], ht[i-k+1]) + 1 \sim \min_{i-k+2 \le j \le i}\{ht[j]\}$ 的都是恰好出现 k 次的。

SAM 怎么做？

题目

P5341 [TJOI2019] 甲苯先生和大中锋的字符串

在字符串中恰好出现了 k 次的子串中，按照字串的长度分类，求出现数量最多的那一类的长度。

题解

SA 怎么处理恰好出现 k 次？

长度在 $\max(ht[i+1], ht[i-k+1]) + 1 \sim \min_{i-k+2 \le j \le i}\{ht[j]\}$ 的都是恰好出现 k 次的。

SAM 怎么做？

直接统计每个节点出现了多少次，然后直接用差分统计答案。

题目

P4248 [AHOI2013] 差异

给定一个字符串 S，令 T_i 表示它从第 i 个字符开始的后缀。求

$$\sum_{1\leqslant i<j\leqslant n} \operatorname{len}(T_i)+\operatorname{len}(T_j)-2\times \operatorname{lcp}(T_i,T_j)$$

$\mathsf{lcp}(a,b)$ 表示字符串 a 和字符串 b 的最长公共前缀的长度。

题目

P4248 [AHOI2013] 差异

给定一个字符串 S，令 T_i 表示它从第 i 个字符开始的后缀。求

$$\sum_{1\leqslant i<j\leqslant n} \operatorname{len}(T_i)+\operatorname{len}(T_j)-2\times\operatorname{lcp}(T_i,T_j)$$

lcp(a,b) 表示字符串 a 和字符串 b 的最长公共前缀的长度。

题解

难点在于后面的 $2\times\operatorname{lcp}(T_i,T_j)$。

在 SA 中，答案就是 ht 数组的每个区间的区间最小值之和，这个可以用单调栈维护。

题目

P4248 [AHOI2013] 差异

给定一个字符串 S，令 T_i 表示它从第 i 个字符开始的后缀。求

$$\sum_{1 \leqslant i < j \leqslant n} \mathrm{len}(T_i) + \mathrm{len}(T_j) - 2 \times \mathrm{lcp}(T_i, T_j)$$

lcp(a, b) 表示字符串 a 和字符串 b 的最长公共前缀的长度。

题解

难点在于后面的 $2 \times \mathrm{lcp}(T_i, T_j)$。

在 SA 中，答案就是 ht 数组的每个区间的区间最小值之和，这个可以用单调栈维护。

在 SAM 中，我们需要计算出每一个节点贡献是多少，就是 $cnt \times (cnt - 1) \times (r - l + 1)$。

题目

P4094 [HEOI2016/TJOI2016] 字符串

给定一个字符串 S，有 Q 次询问。对于每次询问：
求 S[a..b] 的**所有子串**和 S[c..d] 的最长公共前缀的长度的最大值

题目

P4094 [HEOI2016/TJOI2016] 字符串

给定一个字符串 S，有 Q 次询问。对于每次询问：
求 S[a..b] 的**所有子串**和 S[c..d] 的最长公共前缀的长度的最大值

题解

发现 S[a..b] 的**所有子串**特别不好处理，还有边界问题，所以考虑转化。

先看 SA 怎么做，二分长度 len 之后，我们只需要判断 $a \sim b - len + 1$ 这些后缀中是否存在 i，使 $rk[i] \sim rk[c]$ 间最小的 ht 大于等于 len。

题目

P4094 [HEOI2016/TJOI2016] 字符串

给定一个字符串 S，有 Q 次询问。对于每次询问：
求 S[a..b] 的**所有子串**和 S[c..d] 的最长公共前缀的长度的最大值

题解

发现 S[a..b] 的**所有子串**特别不好处理，还有边界问题，所以考虑转化。

先看 SA 怎么做，二分长度 len 之后，我们只需要判断 $a \sim b - len + 1$ 这些后缀中是否存在 i，使 $rk[i] \sim rk[c]$ 间最小的 ht 大于等于 len。

看起来还是不太可做，但是我们可以对于一个 len，去看能取的边界 l, r 是什么，然后就只需要判断 $a \sim b - len + 1$ 内是否存在 i，使 $rk_i \in [l, r]$，这个就可以用主席树解决了。

P4094 [HEOI2016/TJOI2016] 字符串

再看看 SAM 怎么做，同样二分之后，我们先要找到 $S[c, c+len-1]$ 对应的节点 x，然后需要判断的就是 $endpos(x) \cap [a+len-1, b]$ 是否为空。

如何在 SAM 的 link 树上维护 endpos ? 这个可以通过线段树合并解决。

线段树合并维护 endpos 是很重要的。

回文自动机

其实如果你理解了 SAM 的逻辑，学习 PAM 也是很快的。

回文自动机维护的是字符串所有的回文子串，节点表示一个回文串，节点 fail 树上的父亲是该回文串所有后缀中最长的回文串。

构造方式和 SAM 很类似，详细过程见cmd 的 Blog

回文自动机

P5496 【模板】回文自动机（PAM）

给定一个字符串 s。对于 s 的每个位置，请求出以该位置结尾的回文子串个数。

回文自动机

P5496 【模板】回文自动机（PAM）

给定一个字符串 s。对于 s 的每个位置，请求出以该位置结尾的回文子串个数。

题解

本题相当于在线地询问终止链长度，记录 dep 并递推

回文自动机

P4287 [SHOI2011] 双倍回文

如果 x 能够写成 ww^Rww^R 形式，则称它是一个“双倍回文”。换句话说，若要 x 是双倍回文，它的长度必须是 4 的倍数，而且 x 的前半部分，后半部分都要是回文。

回文自动机

P4287 [SHOI2011] 双倍回文

如果 x 能够写成 ww^Rww^R 形式，则称它是一个“双倍回文”。换句话说，若要 x 是双倍回文，它的长度必须是 4 的倍数，而且 x 的前半部分，后半部分都要是回文。

题解

注意到双倍回文串也是一个回文串，此题可以转化成对于每个回文串，判定是否能拆成两个相同的偶回文串。

可以从该点跳 fail 找到自己的所有回文后缀。若有长度恰为 $\frac{len}{2}$ 的，则符合要求。

直接在 fail 树上操作一下就可以了。

回文自动机

Gym104857 C.Cyclic Substrings

定义循环子串 $S[i,j]$ 为:

- 若 $i \le j$，则循环子串为 $S[i,j]$。
- 若 $i > j$，则循环子串为 $S[i,n] + S[1,j]$。

记 P 为 S 所有循环子串 $1 \le i,j \le n$ 的集合，$f(t),(t \in P)$ 为子串 t 出现的次数，$g(t)$ 为 t 的长度。求:

$$\sum_{t \in P} f(t)^2 \times g(t)$$

其中 $|S| \le 3 \times 10^6$，全由数字构成。

Gym104857 C.Cyclic Substrings

如何处理 $i > j$ 的情况？倍长 S 得 $T = S + S$

这样的话，对于 $n+1 \sim n+n$ 的所有前缀的子串，就包含了所有的循环子串，但也多了一些，就是长度 $> n$ 的，那么在统计答案的时候只考虑 $len \leq n$ 的就可以了。

如何统计出现次数？仿照 SAM 处理就可以了。

题目选讲

题目

ICPC2023 网络预选赛第二场 L Super-palindrome

定义一个字符串 S 是好的，当且仅当存在一种分割方式 $S = S_1 + S_2 + ... + S_{2k}$ ，使得 $\forall\ i \in [1, k], S_i = S_{2k-i+1}$ 。求 S 的所有子串中好的个数。$|S| \leq 5 \times 10^3$ 。

题目

ICPC2023 网络预选赛第二场 L Super-palindrome

定义一个字符串 S 是好的，当且仅当存在一种分割方式 $S = S_1 + S_2 + ... + S_{2k}$ ，使得 $\forall\ i \in [1, k], S_i = S_{2k-i+1}$ 。求 S 的所有子串中好的个数。$|S| \leq 5 \times 10^3$ 。

题解

n^3 的 dp 做法是比较显然的，考虑优化。

题目

ICPC2023 网络预选赛第二场 L Super-palindrome

定义一个字符串 S 是好的，当且仅当存在一种分割方式 $S = S_1 + S_2 + ... + S_{2k}$ ，使得 $\forall\ i \in [1, k], S_i = S_{2k-i+1}$ 。求 S 的所有子串中好的个数。$|S| \leq 5 \times 10^3$ 。

题解

n^3 的 dp 做法是比较显然的，考虑优化。

发现我们无论用什么字符串算法去优化匹配过程，dp $O(n)$ 的转移始终制约着复杂度。所以只能通过找性质来优化转移的过程。

注意如果 S 存在分割方式 $S = A + T + A$，其中 T 是好的。那么一定存在 A 是最小的 $\rightarrow$ T 是最大的。

题目

ICPC2023 网络预选赛第二场 L Super-palindrome

定义一个字符串 S 是好的，当且仅当存在一种分割方式 $S = S_1 + S_2 + ... + S_{2k}$ ，使得 $\forall\ i \in [1, k], S_i = S_{2k-i+1}$ 。求 S 的所有子串中好的个数。$|S| \leq 5 \times 10^3$ 。

题解

n^3 的 dp 做法是比较显然的，考虑优化。

发现我们无论用什么字符串算法去优化匹配过程，dp $O(n)$ 的转移始终制约着复杂度。所以只能通过找性质来优化转移的过程。

注意如果 S 存在分割方式 $S = A + T + A$，其中 T 是好的。那么一定存在 A 是最小的 $\rightarrow$ T 是最大的。

那么只要从每一个位置 i 向两边扩展，左右相等则合并，即可求出所有好的子串。

题目

P3426 [POI2005] SZA-Template

你打算在纸上印一串字母。为了完成这项工作，你决定刻一个印章。印章每使用一次，就会将印章上的所有字母印到纸上。

同一个位置的相同字符可以印多次。例如：用 aba 这个印章可以完成印制 ababa 的工作（中间的 a 被印了两次）。

求最小的印章长度。$|S| \leq 5 \times 10^5$。

P3426 [POI2005] SZA-Template

设 $f[i]$ 为打印 i 前缀的最短印章长度，考虑求解 f

需要你推一推 border 的性质，有以下结论：

- $f[i] = f[nxt[i]]$ 或 i
  可以通过画图证明其充分性和必要性。
- $f[i] = f[nxt[i]]$ 当且仅当 $\exists i \in [i - nxt[i], i]\ f[j] = f[nxt[i]]$
  也可以通过画图解决
  如果 $\exists j \in [i - nxt[i], i]\ f[j] = f[nxt[i]]$，那么一定可以拼一个 $nxt[i]$ 构成 i。

所以只需要维护 $f[j] = i$ 的最大的 j 是多少就可以解决了。

题目

CF1654F Minimal String Xoration

定义长度为 2^n 字符串 s 是字符串 t 的"异或串"，当且仅当存在一个数 j 使 $\forall i \in [0, 2^n-1], s_i = t_{i \oplus j}$ 。给定 t ，求 t 所有"异或串"中字典序最小的。$n \le 18$ 。

题目

CF1654F Minimal String Xoration

定义长度为 2^n 字符串 s 是字符串 t 的" 异或串"，当且仅当存在一个数 j 使 $\forall i \in [0, 2^n-1], s_i = t_{i \oplus j}$ 。给定 t ，求 t 所有" 异或串" 中字典序最小的。$n \le 18$ 。

题解

首先肯定最多只有 2^n 个" 异或串"，分别对应 $j = 0, 1, .., 2^n - 1$ 。

题目

CF1654F Minimal String Xoration

定义长度为 2^n 字符串 s 是字符串 t 的" 异或串"，当且仅当存在一个数 j 使 $\forall i \in [0, 2^n - 1], s_i = t_{i \oplus j}$ 。给定 t ，求 t 所有" 异或串" 中字典序最小的。$n \leq 18$ 。

题解

首先肯定最多只有 2^n 个" 异或串"，分别对应 $j = 0, 1, .., 2^n - 1$ 。

我们考虑倍增，从 $1 \sim n$ 枚举 i ，对于当前的 i ，我们考虑异或只对二进制最后 i 位有效的情况下，维护每个 j 对应的异或串的排名 rk_j 和排名为 j 的异或串对应的标号 id_j 。

这个过程很像求后缀数组的过程，更进一步来说，这对应了一系列倍增对字典序排序的过程。

CF1654F Minimal String Xoration

考虑 i 增加一位会带来什么影响。

参考后缀数组的过程，求 $rj_{i+1,j}$ 的时候会参考 $(rk_{i,j}, rk_{i,j+2^i})$ 这个二元组的大小，并以此作为权值排序。

套用在这里，求 $rj_{i+1,j}$ 的时候会参考 $(rk_{i,j}, rk_{i,j\oplus 2^{i+1}})$，那么之后就是正常的求后缀数组的过程了。

```
bool pd(int s1,int s2){
	if(rk[s1]^rk[s2])return rk[s1]<rk[s2];
	return rk[s1^op]<rk[s2^op];
}
for(int i=0;i<n;++i){
	op=1<<i;
	sort(id,id+m,pd);
	for(int j=1;j<m;++j)
	d[b[j]]=d[b[j-1]]+pd(id[j-1],id[j]);
	memcpy(rk,d,sizeof(int)*(m));
}
```

题目

P6698 [BalticOI 2020 Day2] 病毒

题面

题目

P6698 [BalticOI 2020 Day2] 病毒

题面

题解

首先要注意，只有当序列只含 $0, 1$ 时才停止突变，也就是那些长度无限长的病毒根本没有意义。

题目

P6698 [BalticOI 2020 Day2] 病毒

题面

题解

首先要注意，只有当序列只含 $0, 1$ 时才停止突变，也就是那些长度无限长的病毒根本没有意义。

发现有多串的匹配问题（不能以 c 作为子串），直接考虑 AC 自动机表示状态，对于 fail 树祖先中有结束状态的节点都设为不可访问点。所以有 dp，设 $f_{i,j,k}$ 表示数字 i 经过扩展能让 AC 自动机上状态 $j \to k$ 的最小长度。

对于一个分裂关系 $i \to a_1, a_2, ..., a_k$，如果 $a_1 \sim a_k$ 的 dp 值我们都知道了，那么就可以再来一个 dp 求出 $f_{i,j,k}$。

具体来说，就是设 $g_{i,j,k}$ 表示考虑了 $a_1 \sim a_i$ ，起点是 j ，现在到了 k 的最小长度。当然 i 是可以省略的。

P6698 [BalticOI 2020 Day2] 病毒

但是我们是有假设的，怎么处理 $a_1 \sim a_k$ 的 dp 值我们都知道了这个条件呢？

现在我们把问题抽象一点，把 i 看成点，$f_{i,x,x}$ 看成点的权值，一个分裂关系 $i \to a_1, a_2, ..., a_k$ 看成边，那么这个 dp 就成了一个最短路问题。

P6698 [BalticOI 2020 Day2] 病毒

但是我们是有假设的，怎么处理 $a_1 \sim a_k$ 的 dp 值我们都知道了这个条件呢？

现在我们把问题抽象一点，把 i 看成点，$f_{i,x,x}$ 看成点的权值，一个分裂关系 $i \to a_1, a_2, ..., a_k$ 看成边，那么这个 dp 就成了一个最短路问题。

我们仿照 Bellman–Ford 算法，用 $g_{i,j,k}$ 的一次 dp 处理边 $i \to a_1, a_2, ..a_k$ 的松弛操作，那么就可以把整个图的 $f_{i,x,x}$ 都求出来了。复杂度是 $O(GN(\sum l)^3)$

当然也可以仿照 Dijkstra 算法，只不过因为多权值的关系，处理起来会稍微有点麻烦。

题目

hdu7138 String

给定字符串 S 和常数 k，我们定义 F_i 为满足以下条件的 x 的个数。

- $1 \le x \le i$，$S[1, x] = S[x - i + 1, i]$
- 区间 $[1, x]$ 和 $[i - x + 1, i]$ 相交的长度大于 0，且整除于 k。

求 $\prod_{i=1}^{n}(F_i + 1)$。$n \le 10^6, T \le 10$。

题目

hdu7138 String

给定字符串 S 和常数 k，我们定义 F_i 为满足以下条件的 x 的个数。

- $1 \le x \le i$，$S[1,x] = S[x-i+1,i]$
- 区间 $[1,x]$ 和 $[i-x+1,i]$ 相交的长度大于 0，且整除于 k。

求 $\prod_{i=1}^{n}(F_i+1)$。$n \le 10^6, T \le 10$。

题解

发现要对每一个 S_i 的所有 Border 做统计，那么我们可以建立 fail 树，这样从根到 i 节点上的所有点都是 i 的 Border，那么就可以进行树上操作了。

题目

hdu7138 String

给定字符串 S 和常数 k，我们定义 F_i 为满足以下条件的 x 的个数。

- $1 \le x \le i$，$S[1,x] = S[x-i+1,i]$
- 区间 $[1,x]$ 和 $[i-x+1,i]$ 相交的长度大于 0，且整除于 k。

求 $\prod_{i=1}^{n}(F_i+1)$。$n \le 10^6, T \le 10$。

题解

发现要对每一个 S_i 的所有 Border 做统计，那么我们可以建立 fail 树，这样从根到 i 节点上的所有点都是 i 的 Border，那么就可以进行树上操作了。

那么如何处理条件三呢？实际上就是 $i < 2x$ 且 $2x \equiv i \mod k$，那么开一个桶就可以维护，因为 i 是随深度递增的，所以 $i < 2x$ 也是很好维护的。

题目

2022.3.28T1 我的名字

给定字符串 T，由小写字母组成。

多次操作，要么把 $T[x]$ 修改为 c，要么输入字符串 S，求在 $T[l\sim r]$ 中 S 出现了几次，其中 S 由小写字母和 ? 组成，? 和所有字母都是相同的。

数据范围：$|T|\le 10^5,\sum S$ 的非 ? 长度 $\le 10^5$，即 $|S|\le 10^5$ 。

2022.3.28T1 我的名字

我们对每一个小写字母用 bitset 维护所有出现位置，记为 pos_i 。

然后在匹配的时候，维护当前还可以匹配继续参与匹配的位置，记为 ans ，初始全为 1。

- 有长度为 x 的 ? ： $ans = ans << x$。
- 有一小写字母为 c ： $ans = (ans << 1)\&pos_c$ 。

最后 $ans = ans >> |S|$ ，现在 ans 中为 1 的位置就是可以匹配的位置。这题只需要我们求数量，那么用 count() 函数就是结果。

但是题目需要我们求的是 T 的一个子串的答案，其实也很好处理，直接通过 ans 的左移和右移清楚掉两边的情况。

CF1286E Fedya the Potter Strikes Back

给定一个字符串 S 和一个序列 W，初始时它们都为空。

你需要完成 n 次操作。每次操作在 S 后面添加一个字符 c，在序列 W 后面添加一个数字 w_i。

定义一个子区间 $[l, r]$ 的权值为：若子串 $[l, r]$ 和前缀 $[1, r-l+1]$ 相同，则其权值为 $\min_{i=l}^{r} w_i$。否则为 0。

每次操作后，你都要求出当前的串的所有子区间的权值之和，**强制在线**。

数据范围：$n \le 6 \times 10^5, w_i \le 2^{30} - 1$ 。

CF1286E Fedya the Potter Strikes Back

首先可以注意到一个子区间 $[l,r]$ 合法（不为 0），当且仅当 $[l,r]$ 是字符串 $[1,r]$ 的一个 border 。

所以我们可以考虑在每一次加入之后，求所有是 border 的后缀的权值和。

CF1286E Fedya the Potter Strikes Back

首先可以注意到一个子区间 $[l,r]$ 合法（不为 0），当且仅当 $[l,r]$ 是字符串 $[1,r]$ 的一个 border 。

所以我们可以考虑在每一次加入之后，求所有是 border 的后缀的权值和。

一种想法是：既然答案和所有的 border 相关，那么就可以在失配树（就是 KMP 中 next[i] $\rightarrow$ i 连成的树）上统计当前节点到根的权值和。

CF1286E Fedya the Potter Strikes Back

首先可以注意到一个子区间 $[l, r]$ 合法（不为 0），当且仅当 $[l, r]$ 是字符串 $[1, r]$ 的一个 border 。

所以我们可以考虑在每一次加入之后，求所有是 border 的后缀的权值和。

一种想法是：既然答案和所有的 border 相关，那么就可以在失配树（就是 KMP 中 next[i] $\rightarrow$ i 连成的树）上统计当前节点到根的权值和。

但是，这里 border 的权值是关于后缀的，所以我们无法更新这棵树上每一个节点的权值，所以我们只能考虑动态的维护 border 的集合。

CF1286E Fedya the Potter Strikes Back

怎么维护 border 的集合呢？

CF1286E Fedya the Potter Strikes Back

怎么维护 border 的集合呢？

考虑加入一个字符 c 带来的改变：

- 若 $c = S_1$ ，那么就把 1 加入 border 集合中。
- 对于集合中的元素 x，若 $S_{x+1} \neq c$ ，那么就把 x 删掉。

因为整个 border 的集合最多加入 n 次，那么如果我们能快速的找到需要删掉的元素，那么就可以在均摊的复杂度内解决。

CF1286E Fedya the Potter Strikes Back

怎么快速找呢？

CF1286E Fedya the Potter Strikes Back

怎么快速找呢？

这时候就要使用一个技巧了：

我们对失配树上每一个节点 i 记录其第一个后继字符不一样的祖先 f_i ，这是可以每次 $O(1)$ 求的。

那么我们在删除的时候，如果发现当前节点 x 满足 $S_{x+1} = c$ ，那么我们直接跳到 f_i 就可以实现 $O(1)$ 找到下一个需要删掉的节点。

所以我们就实现了均摊 $O(n)$ 维护 border 的集合。

CF1286E Fedya the Potter Strikes Back

剩下的就是维护权值了。考虑我们的操作：

- 往集合中加入元素 a_i
- 把集合中每一个元素 $a_i \leftarrow \min(a_i, c)$ 。
- 删除集合中的元素
- 查询集合中权值和。

CF1286E Fedya the Potter Strikes Back

剩下的就是维护权值了。考虑我们的操作：

- 往集合中加入元素 a_i
- 把集合中每一个元素 $a_i \leftarrow \min(a_i, c)$ 。
- 删除集合中的元素
- 查询集合中权值和。

可以用 map、二分 + 单调栈，也可以用并查集 + 单调栈，这里就不做详细说明。

CF1286E Fedya the Potter Strikes Back

剩下的就是维护权值了。考虑我们的操作：

- 往集合中加入元素 a_i
- 把集合中每一个元素 $a_i \leftarrow \min(a_i, c)$ 。
- 删除集合中的元素
- 查询集合中权值和。

可以用 map、二分 + 单调栈，也可以用并查集 + 单调栈，这里就不做详细说明。

小细节：注意到答案可能会爆 `long long` ，所以要开 `__int128` 。

后缀自动机题目选讲

看还有没有时间，如果有是可以讲讲的，Blog

The End