Suffix Array

 ${\sf WAKing}$

 $11th\ June,\ 2022$

Contents

1 什么是后缀数组?

2 怎么求后缀数组 ?

- O(n log² n) 的垃圾哈希做法
- O(n log n) 的倍增做法
 - 实现

3 求了后缀数组有什么用?

- Height 数组
- 任意两个后缀的 lcp



什么是后缀数组



后缀数组就是把一个字符串所有的后缀按照字典序排序后排名为 第 *i* 的后缀开头的位置形成的数组

什么是后缀数组

后缀数组就是把一个字符串所有的后缀按照字典序排序后排名为第 i 的后缀开头的位置形成的数组例如 "abb" 的后缀数组就是 [1,3,2]

Contents

- 1 什么是后缀数组?
- 2 怎么求后缀数组?
 - O(n log² n) 的垃圾哈希做法
 - O(n log n) 的倍增做法
 - 实现
- 3 求了后缀数组有什么用?
 - Height 数组
 - 任意两个后缀的 lcp





怎么求后缀数组

 $O(n \log^2 n)$ 的垃圾哈希做法

000

怎么求后缀数组?

◆□▶ ◆□▶ ◆■▶ ◆■▶ ■ 少久で

 $O(n \log^2 n)$ 的垃圾哈希做法

哈希

注意到我们可以利用哈希 + 二分在 $O(\log n)$ 的复杂度内求出两个串的最长公共前缀 \log ,然后比较后一位就可以判断出来两个串的大小

 $O(n \log^2 n)$ 的垃圾哈希做法

哈希

注意到我们可以利用哈希 + 二分在 $O(\log n)$ 的复杂度内求出两个串的最长公共前缀 \log ,然后比较后一位就可以判断出来两个串的大小

所以我们考虑用这个方式比较,然后用归并排序什么的就可以做到 $O(n\log^2 n)$ 求出后缀数组了



/立+前

•00

 $O(n \log n)$ 的倍增做法

倍增

注意到如果我们要比较两个字符串 a 和 b,可以先比较 a 的前 k 位和 b 的前 k 位,如果不一样就可以直接确定,一样的话再比较后面的字符串

000

倍增

注意到如果我们要比较两个字符串 a 和 b,可以先比较 a 的前 k 位和 b 的前 k 位,如果不一样就可以直接确定,一样的话再比较后面的字符串

于是我们考虑倍增,首先先按照每一位的字符排个序,然后接下来进行 $\log n$ 轮排序,第 i 轮就是按照每个后缀的前 2^i 位排序,如果长度不到 2^i ,后面用空字符替代

倍增

0.0

 $O(n \log n)$ 的倍增做法

倍增

具体的,我们在第 k+1 轮比较两个开始位置是 a 和 b 的后缀时,可以先比较在上一轮时的排名,如果相等,再去比较 $a+2^k$ 和 $b+2^k$ 的后缀在上一轮的排名

 $O(n \log n)$ 的倍增做法

倍增

具体的,我们在第 k+1 轮比较两个开始位置是 a 和 b 的后缀时,可以先比较在上一轮时的排名,如果相等,再去比较 $a+2^k$ 和 $b+2^k$ 的后缀在上一轮的排名实际上容易发现我们第 k+1 轮排序时就是以后缀 i 在上一轮的排名为第一关键字、后缀 $i+2^k$ 在上一轮的排名(如果 $i+2^k$ 大于 n 的话排名就是 0)为第二关键字排序

000

 $O(n \log n)$ 的倍增做法

倍增

具体的,我们在第 k+1 轮比较两个开始位置是 a 和 b 的后缀时,可以先比较在上一轮时的排名,如果相等,再去比较 $a+2^k$ 和 $b+2^k$ 的后缀在上一轮的排名实际上容易发现我们第 k+1 轮排序时就是以后缀 i 在上一轮的排名为第一关键字、后缀 $i+2^k$ 在上一轮的排名(如果 $i+2^k$ 大于 n 的话排名就是 0)为第二关键字排序可以用基数排序做,时间复杂度 $O(n\log^2 n)$

 什么是后缀数组?
 表文后缀数组?
 求了后缀数组有什么用?

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○
 ○○

 ○○
 ○○</

买现



 $O(n \log n)$ 的倍增做法

实现

https://www.luogu.com.cn/paste/1tdjoxie





Contents

- 1 什么是后缀数组?
- 2 怎么求后缀数组?
 - O(n log² n) 的垃圾哈希做法
 - O(n log n) 的倍增做法
 - 实现
- 3 求了后缀数组有什么用?
 - Height 数组
 - 任意两个后缀的 lcp



求了后缀数组有什么用?

求了后缀数组有什么用?

OO Height 数组

000

求了后缀数组有什么用? ○○ ● ○

Height 数组

Height 数组

Height

数组就是排名相邻的两个后缀的最长公共前缀,也就是 $\operatorname{Height}_i = \operatorname{lcp}\left(s\left[\mathit{rk}_{i-1},\mathit{rk}_{i-1}+1,\mathit{rk}_{i-1}+2,\cdots,n\right],s\left[\mathit{rk}_i,\mathit{rk}_i+1,\mathit{rk}_i+2,\cdots,n\right]\right)$

Height 数组

Height

数组就是排名相邻的两个后缀的最长公共前缀,也就是 $\operatorname{Height}_i = \operatorname{lcp}\left(s\left[rk_{i-1}, rk_{i-1}+1, rk_{i-1}+2, \cdots, n\right], s\left[rk_i, rk_i+1, rk_i+2, \cdots, n\right]\right)$ 特别的 $\operatorname{Height}_1 = 0$

Height

数组就是排名相邻的两个后缀的最长公共前缀,也就是 $\text{Height}_i = \text{lcp}\left(s\left[rk_{i-1}, rk_{i-1}+1, rk_{i-1}+2, \cdots, n\right], s\left[rk_i, rk_i+1, rk_i+2, \cdots, n\right]\right)$

特别的 $Height_1 = 0$

定理: $\operatorname{Height}_{\mathrm{rk}_{i-1}} -1 \leq \operatorname{Height}_{\mathrm{rk}_{i}}$

Height 数组

Height

数组就是排名相邻的两个后缀的最长公共前缀, 也就是 $\text{Height}_i = \text{lcp}(s[rk_{i-1}, rk_{i-1} + 1, rk_{i-1} + 2, \cdots, n], s[rk_i, rk_i + 1, rk_i + 2, \cdots, n])$

特别的 $Height_1 = 0$

定理: $\operatorname{Height}_{\operatorname{rk}_{i-1}} -1 \leq \operatorname{Height}_{\operatorname{rk}_i}$

证明懒得写了,见 oi-wiki

Height 数组

Height

数组就是排名相邻的两个后缀的最长公共前缀, 也就是 $\text{Height}_i = \text{lcp}\left(s\left[rk_{i-1}, rk_{i-1}+1, rk_{i-1}+2, \cdots, n\right], s\left[rk_i, rk_i+1, rk_i+2, \cdots, n\right]\right)$

特别的 $Height_1 = 0$

定理: $\operatorname{Height}_{\operatorname{rk}_{i-1}} -1 \leq \operatorname{Height}_{\operatorname{rk}_i}$

证明懒得写了,见 oi-wiki

利用上面的定理,直接暴力去做就可以做到线性求出 Height 数

组



怎么求后缀数组? oo o ooo 求了后缀数组有什么用? ○○ ○ ●

任意两个后缀的 lcp

任意两个后缀的 lcp

任意两个后缀的 lcp

任意两个后缀的 lcp

容易注意到

$$\begin{split} & \operatorname{lcp}\left(s\left[a, a+1, a+2, \cdots, n\right], s\left[b, b+1, b+2, \cdots, n\right]\right) = \\ & \min_{k = \operatorname{rk}_a + 1} \operatorname{Height}_k \end{split}$$

任意两个后缀的 lcp

任意两个后缀的 lcp

容易注意到

$$lep (s[a, a+1, a+2, \cdots, n], s[b, b+1, b+2, \cdots, n]) = \min_{k=\mathrm{rk}_a+1} \mathrm{Height}_k$$

所以可以转成 RMQ 问题,用四毛子算法解决就行了