

后缀数组

基本定义与概念

后缀 $suf(i)$ 代表字符串 s 从 i 位置开始的后缀 (由 $s[i] \sim s[n-1]$ 组成的字符串)

$sa[i]$ 是一个一维数组, 保存了对字符串 s 所有后缀排序后的结果 $sa[i]$ 代表第 i 小的串在原串中的位置。

$rnk[i]$ 是一个一维数组, 按起始位置保留了每个后缀的排名 $rnk[i]$ 则为 $suf(i)$ 在所有后缀中的排名 (ps: $rnk[sa[i]] = i$)

高度数组 $hgt[i]$ 是一个数组, 保存了相邻两个后缀的最长公共前缀 (LCP) 的长度。

构造和优化

朴素的构造这样一个数组, 最显然的方式显然是直接快速排序。时间复杂度 $O(n^2 \log n)$ 显然很难满足我们大部分使用的需要。

因此, 我们采取倍增的思想来对这些后缀排序。

假设我们对 $shehehda$ 这样的字符串的后缀进行排序。

从每个位置开始, 长度为 2^0 的字串的排序为:

$s[i]$	h	e	h	e	d	a
rank[i]	3	2	3	2	1	0

为了求出长为 2^1 的字符串的排名, 我们以每个位置 i 开始, 长度为 2^0 的排名为第一关键字 $i+2^0$ 位置的排名为第二关键字来进行排序 $i+2^0 \geq n$ 的部分我们就值为 -1

$s[i]$	h	e	h	e	d	a
first[i]	3	2	3	2	1	0
second[i]	2	3	2	1	0	-1
rank[i]	4		4	2	1	0

重复以上过程, 我们可以求出长度为 2^2 的排序结果:

$s[i]$	h	e	h	e	d	a
first[i]	4	3	4	2	1	0
second[i]		2	1	0	-1	
rank[i]	5	3	4	2	1	0

不难看出，这个时候，我们已经完成了排序，而最坏的情况下，这种算法也可以在 $\log n$ 次完成排序。

用快排进行的话，时间复杂度为 $O(n \log n)$

考虑到所有排序数的范围在 $[-1, n)$ 之间，采取基数排序，能够将复杂度优化到 $\log n$

模板

下面我们给出一道模板题 [P3809](#) [【模板】后缀排序](#)

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: <https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:hotpot:%E5%90%8E%E7%BC%80%E6%95%B0%E7%BB%84&rev=1599132083>

Last update: 2020/09/03 19:21