

# 后缀数组(SA)

## 一些约定

字符串相关的定义请参考字符串基础

字符串下标从  $s[1]$  开始。

“后缀  $s[i]$ ”代指以第  $i$  个字符开头的后缀。

## 后缀数组是什么？

后缀数组(Suffix Array)主要是两个数组  $sa$  和  $rk$

其中  $sa[i]$  表示将所有后缀排序后第  $i$  小的后缀的编号  $rk[i]$  表示后缀  $s[i]$  的排名。

这两个数组满足性质  $sa[rk[i]] = rk[sa[i]] = i$

后缀数组示例：



## 后缀数组怎么求？

## $O(n^2 \log n)$ 做法

我相信这个做法大家还是能自己想到的，用 `string + sort` 就可以了。由于比较两个字符串是  $O(n)$  的，所以排序是  $O(n^2 \log n)$  的。

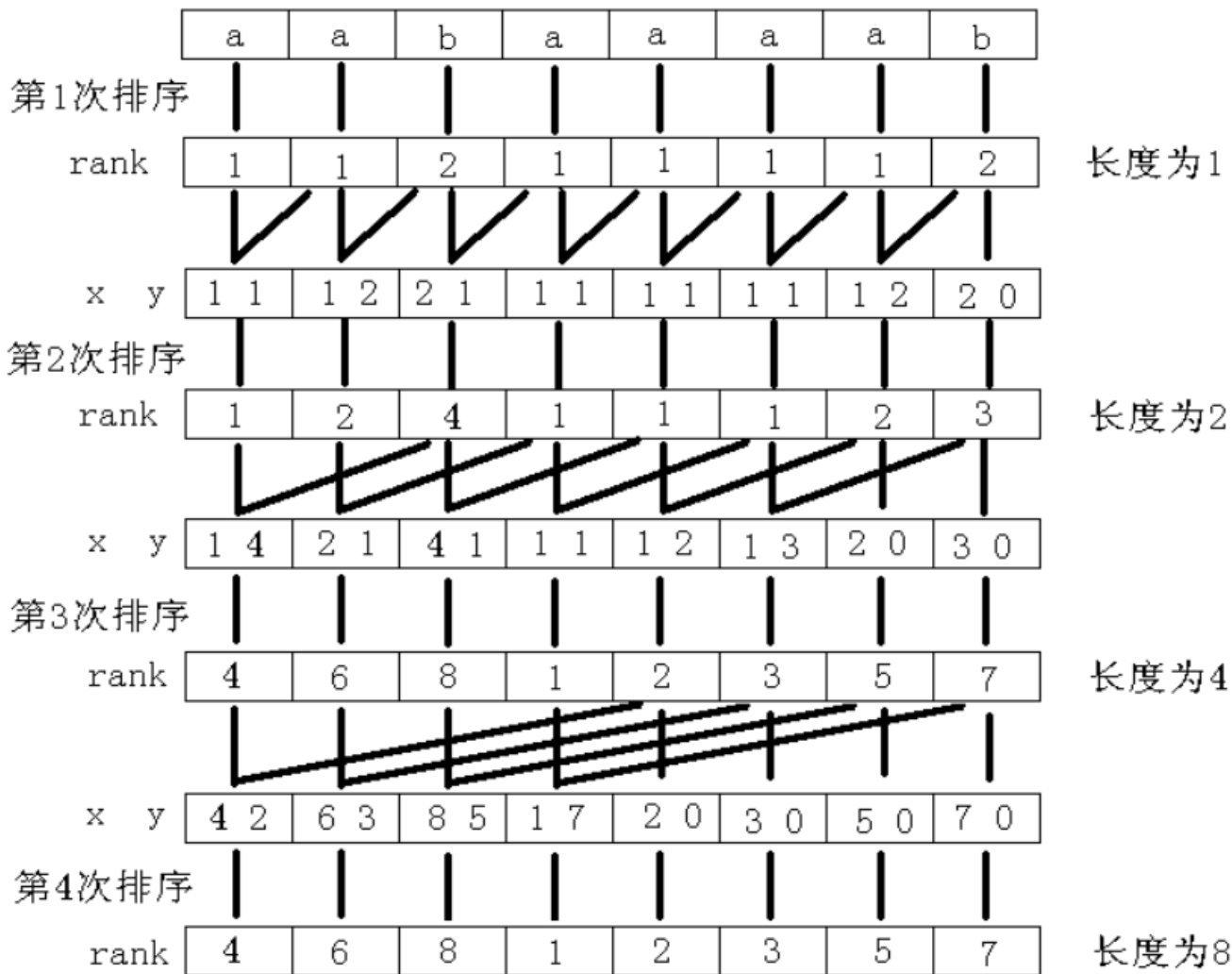
## $O(n \log^2 n)$ 做法

这个做法要用到倍增的思想。

先对每个长度为  $1$  的子串（即每个字符）进行排序。

假设我们已经知道了长度为  $w$  的子串的排名  $rk_w[1..n]$ （即， $rk_w[i]$  表示  $s[i..i+w-1]$  在  $\{s[x..x+w-1] | x \in [1, n]\}$  中的排名），那么，以  $rk_w[i]$  为第一关键字、 $rk_w[i+w]$  为第二关键字（若  $i+w > n$  则令  $rk_w[i+w]$  为无穷小）进行排序，就可以求出  $rk_{2w}[1..n]$ 。

倍增排序示意图：



如果用 `sort` 进行排序，复杂度就是  $O(n \log^2 n)$  的。

参考代码：

```

#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>

using namespace std;

const int N = 1000010;

char s[N];
int n, w, sa[N], rk[N << 1], oldrk[N << 1];
// 为了防止访问 rk[i+w] 导致数组越界，开两倍数组。
// 当然也可以在访问前判断是否越界，但直接开两倍数组方便一些。

int main() {
    int i, p;

    scanf("%s", s + 1);
    n = strlen(s + 1);
    for (i = 1; i <= n; ++i) sa[i] = i, rk[i] = s[i];

    for (w = 1; w < n; w <= 1) {
        sort(sa + 1, sa + n + 1, [](int x, int y) {
            return rk[x] == rk[y] ? rk[x + w] < rk[y + w] : rk[x] < rk[y];
        }); // 这里用到了 lambda
        memcpy(oldrk, rk, sizeof(rk));
        // 由于计算 rk 的时候原来的 rk 会被覆盖，要先复制一份
        for (p = 0, i = 1; i <= n; ++i) {
            if (oldrk[sa[i]] == oldrk[sa[i - 1]] &&
                oldrk[sa[i] + w] == oldrk[sa[i - 1] + w]) {
                rk[sa[i]] = p;
            } else {
                rk[sa[i]] = ++p;
            } // 若两个子串相同，它们对应的 rk 也需要相同，所以要去重
        }
    }

    for (i = 1; i <= n; ++i) printf("%d ", sa[i]);

    return 0;
}

```

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

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:%E5%90%8E%E7%BC%80%E6%95%B0%E7%BB%84\\_lgwza&rev=1595577950](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E5%90%8E%E7%BC%80%E6%95%B0%E7%BB%84_lgwza&rev=1595577950)

Last update: 2020/07/24 16:05