

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 `hehehda` 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 $O(n \log n \log n)$

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 $n \log n$

模板

下面我们给出一道模板题 [P3809](#) [【模板】后缀排序](#)

```
#include<algorithm>
#include<stack>
#include<ctime>
#include<cstring>
#include<cmath>
#include<iostream>
#include<iomanip>
#include<cstdio>
#include<queue>
using namespace std;
inline int read(){
    int num=0,f=1;char x=getchar();
    while(x<'0' || x>'9'){if(x=='-')f=-1;x=getchar();}
    while(x>='0' && x<='9'){num=num*10+x-'0';x=getchar();}
    return num*f;
}
const int maxn=1000005;
char s[maxn];
int n;
int b[maxn],a[maxn];
int fir[maxn],sec[maxn],tmp[maxn],buc[maxn];
int rnk[maxn],sa[maxn];
void suf_sort(){
    int tot=n;
    //对数据离散化，保证数据范围在[0,n]
    for(int i=1;i<=n;++i)b[i]=s[i];
    sort(b+1,b+tot+1);
```

```

tot=unique(b+1,b+tot+1)-b-1;
for(int i=1;i<=n;++i)a[i]=lower_bound(b+1,b+tot+1,s[i])-b;
//得到初始排序:
for(int i=1;i<=n;++i)++buc[a[i]];
for(int i=1;i<=n;++i)buc[i]+=buc[i-1];
for(int i=1;i<=n;++i)rnk[i]=buc[a[i]-1]+1;

for(int t=1;t<=n;t<=1){
    //对第一和第二关键字进行初始化:
    for(int i=1;i<=n;++i)fir[i]=rnk[i];
    for(int i=1;i<=n;++i)sec[i]=i+t>n?0:rnk[i+t];
    //对第二大关键字排序 tmp[i] 保存第i大的关键字
    for(int i=0;i<=n;++i)buc[i]=0;
    for(int i=1;i<=n;++i)++buc[sec[i]];
    for(int i=1;i<=n;++i)buc[i]+=buc[i-1];
    for(int i=1;i<=n;++i)tmp[n--buc[sec[i]]]=i;//--buc[sec[i]]是一个[0,n)的值,减去它得到[1,n]的值
    //对第一关键字排序,按照tmp中的顺序领取排名,大的先领取便被放到了后面,完成了排序。
    for(int i=0;i<=n;++i)buc[i]=0;
    for(int i=1;i<=n;++i)++buc[fir[i]];
    for(int i=1;i<=n;++i)buc[i]+=buc[i-1];
    for(int i=1;i<=n;++i){
        int t=tmp[i];
        sa[buc[fir[t]]--]=t;
    }
    //根据sa[]数组的顺序求出名次数组rnk[],注意根据并列情况讨论
    bool uni=true;
    for(int i=1,lst=0;i<=n;++i){
        int t=sa[i];
        if(!lst)rnk[t]=1;//第一个
        else
            if(fir[t]==fir[lst]&&sec[t]==sec[lst])rnk[t]=rnk[lst],uni=false;//和上一个相同
            else rnk[t]=rnk[lst]+1;//不同
        lst=t;
    }
    if(uni)break;//排序已经完成。
}
for(int i=1;i<=n;++i)printf("%d ",sa[i]);
}

int main(){
    scanf("%s",s+1);
    n=strlen(s+1);
    suf_sort();

    return 0;
}

```

Last
update: 2020-2021:teams:hotpot: <https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:hotpot:%E5%90%8E%E7%BC%80%E6%95%B0%E7%BB%84>
2020/09/03 后缀数组
19:55

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> 

Last update: **2020/09/03 19:55**