字符串 3

后缀数组

算法简介

后缀树的一种替代品，可以用于解决各种字符串问题。

算法例题

例题一

洛谷p4051

题意

给定字符串 $S=s_1s_2\cdots s_n$ 将其视为一个环，任意选择环的起点，可以得到 n 个新字符串 $T_k=s_ks_{k+1}\cdots s_{k-1}$

询问将所有 T_i 按字典序从小到大排序后依次取每个 T_i 的最后一个字母构成的字符串。

题解

考虑将 S 倍长为 SS 求 SS 每个后缀的排名，即可得到每个字符串 T_i 的排名。

关于正确性，考虑字符串 abc 于是 T_2 代表的字符串 bca 变为 $bcabc$ 实际上这相当于 T_2 再与 T_2 的前缀拼接而成，不影响排序结果。

时间复杂度 $O(n\log n)$

```
const int MAXN=2e5+5;
namespace SA{
    int sa[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(char *s,int n,int m){//s下标从1开始
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<<=1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
```

```
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
_rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?pos:++
pos;
            if(pos==n)break;
            m=pos;
        }
    }
}
char buf[MAXN];
int main()
{
    scanf("%s",buf+1);
    int n=strlen(buf+1);
    _rep(i,1,n)buf[i+n]=buf[i];
    buf[2*n+1]='\0';
    SA::get_sa(buf,n<<1,'z');
    _rep(i,1,2*n){
        if(SA::sa[i]<=n)
        putchar(buf[SA::sa[i]+n-1]);
    }
    return 0;
}
```

例题二

洛谷p2870

题意

给定一个字符串 S 和一个空串 T。每个可以选择 S 的首字符或末字符，将其删去后加入到 T 末尾。

问所有可能的 T 中字典序最小的。

题解

考虑贪心，假设现在字符串为 $s_{L}s_{L+1}\cdots s_{R-1}s_R$。显然选取 S_L,S_R 中字典序最小的最优。

如果 $S_L=S_R$。接下来考虑选择 S_{L+1},S_{R-1} 中字典序最小的，直到比较到端点为止。

上述操作等价于比较 $S[L,n]$ 和 $S[1,R]$ 的字典序。考虑构造字符串 $s_1s_2\cdots s_n+\text{\0}+s_n\cdots s_2s_1$。

于是可以通过后缀数组得到 $S[L,n]$ 和 $S[1,R]$ 的排名，时间复杂度 $O(n\log n)$。

```
const int MAXN=1e6+5;
namespace SA{
    int sa[MAXN],rk[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(char *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<<=1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
_rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?pos:++
pos;
            if(pos==n)break;
            m=pos;
        }
        _rep(i,1,n)rk[sa[i]]=i;
    }
}
char buf[MAXN];
int main()
{
    int n=read_int(),len=2*n+1;
    _rep(i,1,n)buf[i]=buf[len+1-i]=get_char();
    buf[n+1]=0;
    SA::get_sa(buf,len,'Z');
    int L=1,R=n,cnt=0;
    while(L<=R){
        if(SA::rk[L]<SA::rk[len+1-R])
        putchar(buf[L++]);
        else
        putchar(buf[R--]);
        if((++cnt)%80==0)putchar('\n');
    }
    return 0;
}
```

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%AD%97%E7%AC%A6%E4%B8%B2_3&rev=1598774291

Last update: **2020/08/30 15:58**