Lyndon 串

介绍

Lyndon 串是满足本身为所有循环移位中字典序严格最小的串。

性质：$u<v$ 均为 Lyndon 串，则 uv 也为 Lyndon 串。

又由于单个字符是一个 Lyndon 串，可依靠这种思想进行合并。

Lyndon 分解：将字符串 s 分成 $s=s_1s_2s_3..s_m$，使得每个 s_i 都是 Lyndon 串，且 $\forall 1\le i<n:s_i\ge s_{i+1}$。

算法

求 Lyndon 分解有常见的两种做法：

Duval 算法

Duval 算法过程：

维护三个变量 i,j,k，$[0..i-1]$ 为一些已经确定的 Lyndon 串，$[i..k-1]$ 为 Lyndon 串 t 进行多次重复后加一个 t 的真前缀 v，当前处理的字符位置为 k，$j=k-len(t)$，分类讨论：

- $s[k]=s[j]$，直接 $k=k+1,j=j+1$。
- $s[k] > s[j]$，此时 $s[i..k]$ 为 Lyndon 串，合并一下，$k=k+1,j=i$。
- $s[k]<s[j]$，此时每一个 t 都为 Lyndon 串，将 i 前移到 v 的开头。

```
//输出 s_1 到s_m 这些串长度的右端点的位置。位置编号为1到 n 。
char s[2000010];
int main(){
    int i,j,k,l;
    scanf("%s",s+1);
    l=strlen(s+1);
    for(i=1;i<=l;){
        j=i,k=i+1;
        while(k<=l&&s[k]>=s[j]){
            if(s[k]>s[j])j=i;
            else j++;
            k++;
        }
        while(i<=j){
            i+=k-j;
            printf("%d ",i-1);
        }
```

```
    }
    return 0;
}
```

后缀数组+单调栈

Duval 算法虽然简便易写，但无法快速求出一个字符串中以每个字符开始的 Lyndon 分解。

利用 Lyndon 串的性质（本身为最小的后缀），构建后缀数组，沿字符串从后往前枚举，构建一个单调增的单调栈即可维护表示以 i 开始、最长 Lyndon 串的右端点位置 $nxt[i]$ 数组。

```
sta[top++]=n+1;
for(i=n;i;i--){
    while(rnk[sta[top-1]]>rnk[i])top--;
    nxt[i]=sta[top-1];
    sta[top++]=i;
}
```

例题

模板题

练习题2020 CCPC-Wannafly Winter Camp Day3 J 简单字符串

- 题意：见链接
- 题解：对于每次询问 (l,k) 首先算出 $r=nxt[l]$ 进行分类讨论：
    - $lcp(suf(l),suf(r))=0$ 答案必为 $[l,r-1]$
    - $lcp(suf(l),suf(r))>0$ 设 $len = r-l,tot=len\cdot\left\lfloor\dfrac{lcp+len}{len}\right\rfloor$ 即 $s[l..r-1]$ 重复出现的长度。
    - $tot\bmod k=0$ 设 $remain=n-(l+tot\cdot len)$ - $remain=0$ 此时可均分，答案为 $[l,l+\left\lfloor\dfrac {tot}{k}\right\rfloor-1]$ - $remain \ne0$ 此时将最后一段连接到结尾，答案为 $[l+tot-\left\lfloor\dfrac{tot}{k}\right\rfloor,n]$
    - $tot\bmod k\ne 0$ 此时 $remain$ 段比循环段小，故答案为从 l 开始的一段 $[l,l+len\cdot\left\lfloor\dfrac{tot/len}{k}\right\rfloor-1]$
-

```
#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
#include<cstring>
#include<cmath>
```

```cpp
#include<cstdlib>
#include<set>
#include<unordered_map>
#include<vector>
typedef long long ll;
using namespace std;
#define N 100010
char s[N];
int n,sa[N],c[N],x[N],y[N];
void getsa(int m){
    int i,k;
    for(i=0;i<=m;i++)c[i]=0;
    for(i=1;i<=n;i++)c[x[i]=s[i]]++;//also x[i]=s[i]-'a'
    for(i=1;i<=m;i++)c[i]+=c[i-1];
    for(i=n;i;i--)sa[c[x[i]]--]=i;
    for(k=1;k<=n;k<<=1){
        int p=0;
        for(i=n-k+1;i<=n;i++)y[++p]=i;
        for(i=1;i<=n;i++)if(sa[i]>k)y[++p]=sa[i]-k;
        for(i=0;i<=m;i++)c[i]=0;
        for(i=1;i<=n;i++)c[x[y[i]]]++;
        for(i=1;i<=m;i++)c[i]+=c[i-1];
        for(i=n;i;i--)sa[c[x[y[i]]]--]=y[i];
        swap(x,y);
        p=x[sa[1]]=1;
        for(i=2;i<=n;i++)
            x[sa[i]]=y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k]?p:++p;
        if(p>=n)break;
        m=p;
    }
}
int h[N],rnk[N];
void geth(){
    int i,j,k=0;
    for(i=1;i<=n;i++)rnk[sa[i]]=i;
    for(i=1;i<=n;i++){
        if(k)k--;
        j=sa[rnk[i]-1];
        while(s[i+k]==s[j+k])k++;
        h[rnk[i]]=k;
    }
}
int rmq[N][22],lg[N];
void initlcp(){
    int i,j;
    for(i=2;i<N;i++)lg[i]=lg[i-1]+(1<<(lg[i-1]+1)==i);
    getsa(260);
    geth();
    for(i=1;i<=n;i++)rmq[i][0]=h[i];
    for(i=1;i<=22;i++)
        for(j=1;j+(1<<i-1)<=n;j++)
```

```
            rmq[j][i]=min(rmq[j][i-1],rmq[j+(1<<i-1)][i-1]);
}
int getlcp(int x,int y){
    if(x==y)return n-x+1;
    if(rnk[x]>rnk[y])swap(x,y);
    int l=rnk[x]+1,r=rnk[y];
    int p=lg[r-l+1];
    return min(rmq[l][p],rmq[r-(1<<p)+1][p]);
}
int sta[N],top,nxt[N];
int main(){
    int i,q;
    scanf("%s%d",s+1,&q);
    n=strlen(s+1);
    initlcp();
    sta[top++]=n+1;
    for(i=n;i;i--){
        while(rnk[sta[top-1]]>rnk[i])top--;
        nxt[i]=sta[top-1];
        sta[top++]=i;
    }
    while(q--){
        int r,l,k;
        scanf("%d%d",&l,&k);
        r=nxt[l];
        if(k==1||r==n+1)printf("%d %d\n",l,n);
        else{
            int lcp=getlcp(l,r);
            if(lcp==0)printf("%d %d\n",l,r-1);
            else{
                int len=r-l,tot=lcp/len*len+len;
                if((tot/len)%k==0){
                    if(l+tot==n+1)printf("%d %d\n",l,l+tot/k-1);
                    else printf("%d %d\n",l+tot-tot/k,n);
                }else printf("%d %d\n",l,l+((tot/len-1)/k+1)*len-1);
            }
        }
    }
    return 0;
}
```