**字符串**

2020.09.05

目录

[字符串 2](#_Toc50246559)

[KMP 2](#_Toc50246560)

[二维KMP 2](#_Toc50246561)

[扩展KMP(Z函数) 3](#_Toc50246562)

[最小表示法 3](#_Toc50246563)

[AC自动机 4](#_Toc50246564)

[后缀数组 6](#_Toc50246565)

[后缀自动机与广义后缀自动机 8](#_Toc50246566)

[后缀自动机 8](#_Toc50246567)

[广义后缀自动机 9](#_Toc50246568)

[后缀树 13](#_Toc50246569)

[Manacher算法 13](#_Toc50246570)

[回文自动机 14](#_Toc50246571)

[字符串哈希 15](#_Toc50246572)

# 字符串

## KMP

void getfail()

{

int i,j;

j=0;

for(i=2;i<=n;i++)

{

while(j&&s[i]!=s[j+1]) j=nxt[x][j];

if(s[i]==s[j+1]) j++;

nxt[i]=j;

}

}

#### 最小循环节

如果len%(len-nxt[x])=0，则存在循环节，最小循环节长度为len-nxt[x]

#### 性质

## 二维KMP

//nxt[i][j]表示s[i……n]中j的失配位置

void getfail(int x)

{

int i,j;

j=x-1;

nxt[x][x]=x-1;

for(i=x+1;i<=n;i++)

{

while(j>=x&&s[i]!=s[j+1]) j=nxt[x][j];

if(s[i]==s[j+1]) j++;

nxt[x][i]=j;

}

}

## 扩展KMP(Z函数)

**a[1……m], b[1……n]  
s = a + ‘\*’ + b  
nxt[i]表示S[1……n+m]与s[i……n+m+1]的最长公共前缀  
f[i]表示b[i……n]与a[1……m]的最长公共前缀**

void get\_next(int len) // len = n + m + 1

{

int l,r,i;

nxt[1]=len;

l=1;

r=2;

for(i=2;i<=len;i++)

{

nxt[i]=0;

if(i<=r) nxt[i]=min(r-i,nxt[i-l+1]);

while(s[i+nxt[i]]==s[1+nxt[i]]) nxt[i]++;

if(i+nxt[i]>r)

{

l=i;

r=i+nxt[i];

}

}

}

void get\_f(int n,int m)

{

int i;

for(i=m+2;i<=n+m+1;i++)

f[i-m-1]=min(nxt[i],m-1);

}

## 最小表示法

void mins(int n)

{

for(int i=1;i<=n;i++) a[i+n]=a[i];

int p1=1, p2=2;

while(p1<=n&&p2<=n)

{

int j;

for(j=0;j<n;j++) if(a[p1+j]!=a[p2+j]) break;

if(a[p1+j]>a[p2+j]) p1=p1+j+1;

else p2=p2+j+1;

if(p1==p2) p1++;

}

if(p2<=n) swap(p2,p1);

for(int i=0;i<n;i++) printf("%d ",a[p1+i]);

}

## AC自动机

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<cmath>

const int N=2e5+10;

const int M=2e6+10;

struct EDGE{

int to,nxt;

EDGE(){}

EDGE(int x,int y){to=x;nxt=y;}

}edge[N\*2];

int t[N],cnt[N],pos[N],fail[N],ch[N][26],que[N];

char s[M];

int K,tot;

void addedge(int x,int y)

{

edge[++K]=EDGE(y,t[x]);

t[x]=K;

}

void ins(int id)

{

scanf("%s",s+1);

int now=0;

for(int i=1;s[i];i++)

{

if(!ch[now][s[i]-'a']) ch[now][s[i]-'a']=++tot;

now=ch[now][s[i]-'a'];

}

pos[id]=now;

}

void get\_fail()

{

int head=0,tail=0;

for(int i=0;i<26;i++)

if(ch[0][i])

que[++tail]=ch[0][i];

while(head<tail)

{

head++;

int x=que[head];

for(int i=0;i<26;i++)

if(ch[x][i])

{

int y=ch[x][i];

int j=fail[x];

while(j&&!ch[j][i]) j=fail[j];

if(ch[j][i]) j=ch[j][i];

fail[y]=j;

que[++tail]=y;

}

}

}

void match()

{

scanf("%s",s+1);

int now=0;

for(int i=1;s[i];i++)

{

while(now&&!ch[now][s[i]-'a']) now=fail[now];

if(ch[now][s[i]-'a'])

{

now=ch[now][s[i]-'a'];

cnt[now]++;

}

}

}

void dfs(int x)

{

for(int p=t[x];p;p=edge[p].nxt)

{

int y=edge[p].to;

dfs(y);

cnt[x]+=cnt[y];

}

}

int main()

{

int n;

scanf("%d",&n);

for(int i=1;i<=n;i++) ins(i);

get\_fail();

match();

for(int i=1;i<=tot;i++) addedge(fail[i],i);

dfs(0);

for(int i=1;i<=n;i++) printf("%d\n",cnt[pos[i]]);

return 0;

}

另一种不用建fail树的写法

const int N = 1e5+10 , A = 26;

int ch[N][A] , fail[N];

queue <int> q;

void getfail(){

fail[1] = 1;

for(int i=0;i<A;++i)

if(ch[1][i]){

fail[ch[1][i]]=1;

q.push(ch[1][i]);

}

else

ch[1][i]=1;

while(q.size()>0){

int c=q.front(); q.pop();

for(int i=0;i<A;++i)

if(ch[c][i]){

fail[ch[c][i]]=ch[fail[c]][i];

q.push(ch[c][i]);

}

else

ch[c][i]=ch[fail[c]][i];

}

}

## 后缀数组

void get\_sa(int n,int m)

{

for(int i = 1; i <= n; i++) x[i] = ss[i];

for(int i = 0; i <= m; i++) c[i] = 0;

for(int i = 1; i <= n; i++) c[x[i]]++;

for(int i = 1; i <= m; i++) c[i] +=c[i - 1];

for(int i = n; i > 0; i--) sa[c[x[i]]--] = i;

for(int i = 1; i <= n; i++) y[i] = x[i];

m = 1;

for(int i = 1; i <= n; i++)

{

if(y[sa[i]] != y[sa[i - 1]]) m++;

x[sa[i]] = m;

}

for(int k = 1; k < n; k <<= 1)

{

int num = 0;

for(int i = n - k + 1; i <= n; i++) y[++num] = i;

for(int i = 1; i <= n; i++)

if(sa[i] > k)

y[++num] = sa[i] - k;

for(int i = 0; i <= m; i++) c[i] = 0;

for(int i = 1; i <= n; i++) c[x[y[i]]]++;

for(int i = 1; i <= m; i++) c[i] += c[i - 1];

for(int i = n; i > 0; i--) sa[c[x[y[i]]]--] = y[i];

for(int i = 1; i <= n; i++) y[i] = x[i];

m = 0;

for(int i = 1; i <= n; i++)

{

if(sa[i] + k > n || sa[i - 1] + k > n || y[sa[i]] != y[sa[i - 1]] || y[sa[i] + k] != y[sa[i - 1] + k])

m++;

x[sa[i]] = m;

}

}

}

void get\_height()

{

j = 0;

for(int i = 1; i <= n; i++)

{

if(j) j--;

while(s[i + j] == s[sa[x[i] - 1] + j]) j++;

hi[x[i]] = j;

}

}

## 后缀自动机与广义后缀自动机

### 后缀自动机

#### 插入（见后缀自动机）

char s[N];

int ch[N\*2][26],fa[N\*2],len[N\*2];

int last=1,tot=1;

void insert(int x)

{

int p=last;

int np=last=++tot;

len[np]=len[p]+1;

while(p&&!ch[p][x]) { ch[p][x]=np; p=fa[p]; }

if(!p) fa[np]=1;

else

{

int q=ch[p][x];

if(len[q]==len[p]+1) fa[np]=q;

else

{

int nq=++tot;

for(int i=0;i<26;i++) ch[nq][i]=ch[q][i];

fa[nq]=fa[q]; fa[np]=fa[q]=nq;

len[nq]=len[p]+1;

while(p&&ch[p][x]==q)

{ ch[p][x]=nq; p=fa[p]; }

}

}

}

int main()

{

scanf("%s",s+1);

int ls=strlen(s+1);

for(int i=1;i<=ls;i++)

insert(s[i]-'a');

return 0;

}

#### 匹配

int p=1,now=0;

for(int i=1;i<=lt;i++)

{

int x=t[i]-'a';

if(ch[p][x])

{ now++; p=ch[p][x];}

else

{

while(p&&!ch[p][x]) p=fa[p];

if(!p){p=1;now=0;}

else

{ now=len[p]+1; p=ch[p][x];}

}

}

### 广义后缀自动机

#### 在线版

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<cmath>

#define LL long long

using namespace std;

const int N=2e6+10;

int ch[N][26],len[N],fa[N];

char s[N];

int last,tot=1;

int insert(int c,int last)

{

int p=last;

if(ch[p][c])

{

int np=ch[p][c];

if(len[p]+1==len[np]) return np;

else

{

int nq=++tot;

len[nq]=len[p]+1;

for(int i=0;i<26;i++) ch[nq][i]=ch[np][i];

while(p&&ch[p][c]==np) ch[p][c]=nq,p=fa[p];

fa[nq]=fa[np],fa[np]=nq;

return nq;

}

}

int q=++tot;

len[q]=len[p]+1;

while(p&&!ch[p][c]) ch[p][c]=q,p=fa[p];

if(!p) fa[q]=1;

else

{

int np=ch[p][c];

if(len[p]+1==len[np]) fa[q]=np;

else

{

int nq=++tot;

len[nq]=len[p]+1;

for(int i=0;i<26;i++) ch[nq][i]=ch[np][i];

while(p&&ch[p][c]==np) ch[p][c]=nq,p=fa[p];

fa[nq]=fa[np],fa[np]=fa[q]=nq;

}

}

return q;

}

int main()

{

int n;

LL ans=0;

scanf("%d",&n);

for(int i=1;i<=n;i++)

{

scanf("%s",s+1);

last=1;

int l=strlen(s+1);

for(int j=1;j<=l;j++) last=insert(s[j]-'a',last);

}

for(int i=2;i<=tot;i++) ans+=len[i]-len[fa[i]];

printf("%lld ",ans);

return 0;

}

#### 离线版

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<cmath>

#include<queue>

#define LL long long

using namespace std;

const int N=2e6+10;

const int M=1e6+10;

char ss[M];

struct Trie{

int tot,col[M],fa[M],ch[M][26];

Trie(){tot=1;}

void insert(char s[])

{

int now=1;

for(int i=1;s[i];i++)

{

int c=s[i]-'a';

if(!ch[now][c])

ch[now][c]=++tot,fa[tot]=now,col[tot]=c;

now=ch[now][c];

}

}

}T1;

struct Suffix\_Auto{

int tot,pos[N],fa[N],len[N],ch[N][26];

queue<int>Q;

Suffix\_Auto(){tot=1;}

int insert(int c,int last)

{

int p=last;

int q=++tot;

len[q]=len[p]+1;

while(p&&!ch[p][c]) ch[p][c]=q,p=fa[p];

if(!p)fa[q]=1;

else

{

int np=ch[p][c];

if(len[p]+1==len[np])fa[q]=np;

else

{

int nq=++tot;

len[nq]=len[p]+1;

for(int i=0;i<26;i++)

ch[nq][i]=ch[np][i];

while(p&&ch[p][c]==np)

ch[p][c]=nq,p=fa[p];

fa[nq]=fa[np],fa[q]=fa[np]=nq;

}

}

return q;

}

void build()

{

for(int i=0;i<26;i++)

if(T1.ch[1][i])

Q.push(T1.ch[1][i]);

pos[1]=1;

while(Q.empty()==0)

{

int x=Q.front();

Q.pop();

pos[x]=insert(T1.col[x],pos[T1.fa[x]]);

for(int i=0;i<26;i++)

if(T1.ch[x][i])

Q.push(T1.ch[x][i]);

}

}

void query()

{

LL ans=0;

for(int i=2;i<=tot;i++) ans+=len[i]-len[fa[i]];

printf("%lld",ans);

}

}SAM;

int main()

{

int n;

scanf("%d",&n);

for(int i=1;i<=n;i++)

{

scanf("%s",ss+1);

T1.insert(ss);

}

SAM.build();

SAM.query();

return 0;

}

## 后缀树

## Manacher算法

**//O(n)求每一个位置的回文半径，字符串需要从abc变成\*a\*b\*c\***

#include <cstdio>

using namespace std;

const int N = 25e6 + 10;

char s[N] , c; int f[N];**//f存储的是回文半径-1，即是以i为中心最大回文串**

int n = 0;

int main(){

c = getchar();

s[n ++] = '\*';

while(c >= 'a' && c <= 'z'){

s[n ++] = c;

s[n ++] = '\*';

c = getchar();

}

s[n] = '\0';

int mid = 0 , ans = 1 , r = mid + f[mid];

for(int i = 1 ; i < n ; ++ i){

if(r < i){

mid = i; r = i;

}

int t = min(r - i , f[(mid << 1) - i]);

while(t < i && s[i + t + 1] == s[i - t - 1])

t ++;

f[i] = t; if(t > ans) ans = f[i];

if(i + t >= r){

mid = i; r = i + t;

}

}

return 0;

}

## 回文自动机

两棵Trie树，分别记录长度为奇数/偶数的回文串。fail指针指向对侧的树。先通过跳前一个位置的fail找到当前位置向前的最长回文串，并新建节点表示该串。然后，对跳该串的父节点的fail以确定该串的fail。

**注意**初始状态：fa[0] =1, fa[1] =0, len[0]=0, len[1]=-1;

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<cmath>

using namespace std;

const int N=2e6+10;

int len[N],fail[N],ch[N][26],cnt[N];

char s[N];

int tot=1,last=0;

void init()

{

fail[0]=1; fail[1]=0;

len[0]=0; len[1]=-1;

}

void extend(int x,int l)

{

int u=last;

while(s[l-len[u]-1]!=s[l]) u=fail[u];

if(ch[u][x]==0)

{

int now=++tot;

len[now]=len[u]+2;

int v=fail[u];

while(s[l-len[v]-1]!=s[l]) v=fail[v];

fail[now]=ch[v][x];

ch[u][x]=now;

cnt[now]=cnt[fail[now]]+1;

}

last=ch[u][x];

}

int main()

{

scanf("%s",s+1);

int n=strlen(s+1);

int ans=0;

init();

for(int i=1;i<=n;i++)

{

s[i]=(s[i]-97+ans)%26+97;

extend(s[i]-'a',i);

ans=cnt[last];

printf("%d ",ans);

}

return 0;

}

## 字符串哈希

#include<iostream>

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<queue>

#include<vector>

#include<map>

#define maxn 100005

#define mid ((nl+nr)>>1)

using namespace std;

typedef long long ll;

class hash

{

private:

static const int mod=99989,seed=13;

int head[maxn],nex[maxn];

int tot;

string s[maxn];

public:

int gethash(string t)

{

int now=0,len=t.size();

for(int i=0;i<len;i++)

now=(now\*seed+t[i])%mod;

return now;

}

void insert(string t)

{

int val=gethash(t),flag=1;

for(int i=head[val];i;i=nex[i])

if(s[i]==t)

{

flag=0;

break;

}

if(flag)

{

s[++tot]=t; nex[tot]=head[val]; head[val]=tot;

}

}

int getnum()

{

return tot;

}

hash()

{

tot=0;

for(int i=0;i<mod;i++)

head[i]=0;

}

}h;

int main()

{

int \_;

scanf("%d",&\_);

while(\_--)

{

string t;

cin>>t;

h.insert(t);

}

printf("%d\n",h.getnum());

}