SAM – 求本质不同的子串个数

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N=2e6+5,M=2e4+5,inf=0x3f3f3f3f,mod=1e9+7;

#define mst(a) memset(a,0,sizeof a)

#define lx x<<1

#define rx x<<1|1

#define reg register

#define PII pair<int,int>

#define fi first

#define se second

#define pb push\_back

char s[N];//文本串.

int a[N],b[N];//a[i]表示长度第i短的子串对应的结点,b[i]长度为小于等于i的子串的个数,类似于一个桶(bucket)

struct SAM{

//last上一个结点,cnt当前结点个数,ch字典树,fa[i]结点i的fail指针,len[i]结点i的长度.

//sz[i]结点i的子串个数.

int last,cnt;int ch[N<<1][26],fa[N<<1],len[N<<1],sz[N];

void insert(int c){ //插入字符操作.

int p=last,np=++cnt;last=np;len[np]=len[p]+1; //p为上一次的结点,np为新建结点,更新last=np,更新np长度.

for(;p&&!ch[p][c];p=fa[p]) ch[p][c]=np;//如果p不是虚根而且p没有c的转移边,p指向np,然后跳fail

if(!p) fa[np]=1;//如果p已经虚根了,说明当前为根结点，即np的fail指向根(1)就好了.

else {

int q=ch[p][c]; //否则令q为p连向c的结点.

if(len[q]==len[p]+1) fa[np]=q;//如果q的长度是p的长度加1,直接令np指向q就行了.

else { //否则新建立一个结点nq,令nq为p后面一个结点,即使得满足上面一个情况.

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

memcpy(ch[nq],ch[q],sizeof ch[q]);//复制q的所有信息给nq.

fa[nq]=fa[q],fa[q]=fa[np]=nq;//fa[q]也赋给fa[nq],同时q,np都指向nq.

for(;ch[p][c]==q;p=fa[p]) ch[p][c]=nq;//将p沿c转移到q的边全部转移给nq.

}

}

sz[np]=1;//初始化np结点的长度，这里不初始化nq.

}

void build(){

scanf("%s",s+1);int len=strlen(s+1);

last=cnt=1;for(int i=1;i<=len;i++) insert(s[i]-'a');

}

void solve(){

ll ans=0;

for(int i=1;i<=cnt;i++) b[len[i]]++;

for(int i=1;i<=cnt;i++) b[i]+=b[i-1];//求前缀和.

for(int i=1;i<=cnt;i++) a[b[len[i]]--]=i; //类似桶排 将结点长度从小到大排序.

for(int i=cnt;i;i--){ //从长度大的开始遍历，跑拓扑序.

int p=a[i]; //因为长度大的结点的fail结点指向的后缀也是长度大的后缀

sz[fa[p]]+=sz[p];//状态转移.

if(sz[p]>1) ans=max(ans,1LL\*sz[p]\*len[p]);//更新答案.

}

printf("%lld\n",ans);

}

}sam;

int main(){

sam.build();

sam.solve();

return 0;

}

广义SAM Trie+bfs建图

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N=4e5+5,M=2e6+5,inf=0x3f3f3f3f,mod=1e9+7;

#define mst(a) memset(a,0,sizeof a)

#define lx x<<1

#define rx x<<1|1

#define reg register

#define PII pair<int,int>

#define fi first

#define se second

#define pb push\_back

char s[M];

struct Trie{

int id,tr[M][26],fa[M],col[M];

Trie(){id=1;}

void Insert(char \*s){

int rt=1;

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

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

if(!tr[rt][c]) tr[rt][c]=++id,fa[id]=rt,col[id]=c;

rt=tr[rt][c];

}

}

}T;

struct SAM{

int last,cnt;int ch[M<<1][26],fa[M<<1],len[M<<1],pos[M];

queue<int>Q;

SAM(){cnt=1;}

int insert(int c,int last){

int p=last,np=++cnt;len[np]=len[p]+1;

for(;p&&!ch[p][c];p=fa[p]) ch[p][c]=np;

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

else {

int q=ch[p][c];

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

else {

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

memcpy(ch[nq],ch[q],sizeof ch[q]);

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

for(;ch[p][c]==q;p=fa[p]) ch[p][c]=nq;

}

}

return np;

}

void build(){

for(int i=0;i<26;i++) if(T.tr[1][i]) Q.push(T.tr[1][i]);

pos[1]=1;

while(!Q.empty()){

int u=Q.front();Q.pop();

pos[u]=insert(T.col[u],pos[T.fa[u]]);//注意是pos[Trie->fa[u]]

for(int i=0;i<26;++i) if(T.tr[u][i]) Q.push(T.tr[u][i]);

}

}

void solve(){

ll ans=0;

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

printf("%lld\n",ans);

}

}sam;

int main(){

int n;

scanf("%d",&n);

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

scanf("%s",s+1),T.Insert(s);

sam.build();sam.solve();

return 0;

}