// 后缀数组+高度数组

// 使用基数排序优化，复杂度O(nlogn)

const int MAXN=1e5;

char s[MAXN+48];bool isfirst;int maxind;

int sa[MAXN+48],rnk[MAXN+48],tmp[MAXN+48],lcp[MAXN+48],k,len;

struct List

{

int head[MAXN+48],nxt[MAXN+48],cur[MAXN+48],tot;Pair val[MAXN+48];

inline void init() {memset(head,0,sizeof(head));tot=0;}

inline void insert(int pos,Pair v)

{

if(!head[pos]){head[pos]=++tot;cur[pos]=head[pos];val[tot]=v;nxt[tot]=0;return;}

nxt[cur[pos]]=++tot;cur[pos]=tot;

val[tot]=v;nxt[tot]=0;

}

}l1,l2;

inline bool compare\_sa(int x,int y)

{

if (rnk[x]!=rnk[y]) return rnk[x]<rnk[y];

int tmp1=x+k<=len?rnk[x+k]:-1,tmp2=y+k<=len?rnk[y+k]:-1;

return tmp1<tmp2;

}

inline int getrnk(int x,bool type)

{

if (!type) return rnk[x];

return x+k<=len?rnk[x+k]:0;

}inline void Sort()

{

int i,j;

l1.init();

for (i=1;i<=len;i++) l1.insert(getrnk(sa[i],true),mp(getrnk(sa[i],false),sa[i]));

l2.init();

if (isfirst)

{

for (i=97;i<=122;i++)

for (j=l1.head[i];j;j=l1.nxt[j])

l2.insert(l1.val[j].x,mp(l1.val[j].y,0));

int tt=0;

for (i=97;i<=122;i++)

for (j=l2.head[i];j;j=l2.nxt[j])

sa[++tt]=l2.val[j].x;

return;

}

for (i=0;i<=maxind;i++)

for (j=l1.head[i];j;j=l1.nxt[j])

l2.insert(l1.val[j].x,mp(l1.val[j].y,0));

int tt=0;

for (i=0;i<=maxind;i++)

for (j=l2.head[i];j;j=l2.nxt[j])

sa[++tt]=l2.val[j].x;

}

inline void construct\_sa()

{

int i;

for (i=1;i<=len;i++) sa[i]=i,rnk[i]=s[i];

for (k=0;k<=len;k==0?k++:k<<=1)

{

if (k==0) isfirst=true; else isfirst=false;

Sort();

tmp[sa[1]]=1;int cnt=0;

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

{

tmp[sa[i]]=tmp[sa[i-1]]+(compare\_sa(sa[i-1],sa[i])?1:0);

cnt+=(tmp[sa[i]]!=tmp[sa[i-1]]);

}

maxind=tmp[sa[len]];

for (i=1;i<=len;i++) rnk[i]=tmp[i];

if (cnt>=len-1) break;

}

}

inline void construct\_lcp()

{

int i,j,h=0;

for (i=1;i<=len;i++) rnk[sa[i]]=i;

for (i=1;i<=len;i++)

{

j=sa[rnk[i]-1];

if (h) h--;

while (i+h<=len && j+h<=len && s[i+h]==s[j+h]) h++;

lcp[rnk[i]-1]=h;

}

}