//AC自动机

namespace AC\_Automaton

{

const int MAXN=1e5;

struct node

{

int nxt[130],fail,ind;

//通常字符集没有这么大

inline void clear() {memset(nxt,0,sizeof(nxt));fail=0;}

}trie[MAXN+48];int tot=1;

inline void init() {tot=1;trie[tot].clear();}

//初始插入字符，建Trie

void Insert(int ind)

{

int i,j,cur=1,len=strlen(s+1);

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

{

if (!trie[cur].nxt[s[i]])

{

trie[cur].nxt[s[i]]=++tot;trie[tot].clear();

trie[tot].fail=trie[tot].ind=0;

}

cur=trie[cur].nxt[s[i]];

if (i==len) trie[cur].ind=ind;

}

}

//构建fail指针(Trie图)

//fail[cur]指的是当前已经匹配到cur这个字符，它的那些儿子们都失配时，该转移到哪里继续匹配

queue<int> q;

void construct\_fail()

{

int i,cur,tmp;q.push(1);

while (!q.empty())

{

cur=q.front();q.pop();

for (i=1;i<=128;i++)

if (trie[cur].nxt[i])

{

tmp=trie[cur].fail;

while (tmp && !trie[tmp].nxt[i]) tmp=trie[tmp].fail;

trie[trie[cur].nxt[i]].fail=(tmp?trie[tmp].nxt[i]:1);

q.push(trie[cur].nxt[i]);

}

else

//构建Trie图（不是很必要），如果没有i这个儿子，就可以通过fail指针提前帮他找好下一个匹配位置

trie[cur].nxt[i]=(cur==1?1:trie[trie[cur].fail].nxt[i]);

}

}

//匹配

void query()

{

int i,cur=1,tmp,len=strlen(s+1);

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

{

cur=trie[cur].nxt[s[i]];

tmp=cur;

while (tmp)

{

if (trie[tmp].ind) cnt[trie[tmp].ind]++;

tmp=trie[tmp].fail;

}

}

}

}