/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tire

O(len)

Test Code:https://paste.ubuntu.com/p/DnPMP7FZgK/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct node{

int cnt, ex; // cnt表示存在此前缀的单词数， ex为末尾标记

int next[27]; // 此模板为只有小写字母的状态， 非递归， 无指针操作

node(){ cnt = 0, ex = 0, memset(next, 0, sizeof next); }

};

int cnt;

node tr[10000];

void init()

{

cnt = 0;

tr[0] = node();

}

void insert(char v[], int len)

{

int now = 0;

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

{

if (tr[now].next[v[i] - 'a'] == 0) // 无相关路径， 需新建

{

tr[now].next[v[i] - 'a'] = ++cnt;

tr[cnt] = node();

}

tr[now].cnt++; // 计数

now = tr[now].next[v[i] - 'a'];

}

tr[now].cnt++;

tr[now].ex = 1; // 末尾标记

}

int check(char v[], int len) // 返回以单词v为前缀的单词个数

{

int now = 0;

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

{

if (tr[now].next[v[i] - 'a'] == 0) // 未找到单词v

return 0;

now = tr[now].next[v[i] - 'a'];

}

return tr[now].ex ? tr[now].cnt : 0;

}