inline void insert(char\* word, int value) {

int len = strlen(word), j = 0;

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

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

if(!ch[j][c]) ch[j][c] = ++size;

j = ch[j][c];

}

val[j]+=value;

}

inline void GetFail() {

queue<int> q;

fail[0] = 0;

for (int c = 0; c < Sigma\_Size; ++c) {

int p = ch[0][c];

if(p) {

fail[p] = last[p] = 0;

q.push(p);

}

}

while(!q.empty()) {

int head = q.front();

q.pop();

for (int c = 0; c < Sigma\_Size; ++c) {

int u = ch[head][c];

if(!u) continue;

q.push(u);

int v = fail[head];

while(v && !ch[v][c]) v = fail[v];

fail[u] = ch[v][c];

last[u] = val[fail[u]] ? fail[u] : last[fail[u]];

//这样保证了沿last数组经过的节点(除了u与root) 都会是单词节点(val>0)

//val[u]有可能大于0

}

} }

inline void Founded(int x) {

for(; x; x=last[x]) cnt[x]++;}

// last[i]=j表j节点表示的单词是i节点单词的后缀，且j节点是单词节点

// 递归打印与结点i后缀相同的前缀节点编号

// 进入此函数前需保证val[x]>0

// cnt[] 记录某个点代表的单词 在文章中出现的次数

inline void Find(char\* text) {

int j = 0, len = strlen(text);

memset(cnt, 0, sizeof(cnt));

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

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

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

j = ch[j][c];

if(val[j]) Founded(j);

else if(last[j]) Founded(last[j]);

}

}

// main(): insert(P, 1); GetFail(); Find(T);