**字典树**

struct NODE{

int child;

//这里仅仅定义了子节点，可以根据题目的要求选择增加定义字典树节点的变量类型

};

const int maxn = 1e6+10;

struct Trie{

NODE t[maxn][30];

int cnt = 0;

void insert(string s)

{

int pos = 0;

int len = s.length();

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

int x = s[i] - 'a' + 1;

if(t[pos][x].child == 0)t[pos][x] = {++cnt};//构造新的字典树节点

//这里也可以根据具体题目需求来添加相应的语句。

pos = t[pos][x].child;

}

}

bool exist(string s)

{

int pos = 0;

int len = s.length();

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

int x = s[i] - 'a' + 1;

if(t[pos][x].child == 0)return false;

pos = t[pos][x].child;

}

return true;

}

}trie;

**字符串哈希**

#include <bits/stdc++.h>

using namespace std;

const int L = 1e6 + 5;

const int HASH\_CNT = 2;

int hashBase[HASH\_CNT] = {29, 31};

int hashMod[HASH\_CNT] = {int(1e9 + 9), 998244353};

struct StringWithHash {

char s[L];

int ls;

int hsh[HASH\_CNT][L];

int pwMod[HASH\_CNT][L];

void init() { // 初始化

ls = 0;

for (int i = 0; i < HASH\_CNT; ++i) {

hsh[i][0] = 0;

pwMod[i][0] = 1;

}

}

StringWithHash() { init(); }

void extend(char c) {

s[++ls] = c; // 记录字符数和每一个字符

for (int i = 0; i < HASH\_CNT; ++i) { // 双哈希的预处理

pwMod[i][ls] =

1ll \* pwMod[i][ls - 1] \* hashBase[i] % hashMod[i]; // 得到b^ls

hsh[i][ls] = (1ll \* hsh[i][ls - 1] \* hashBase[i] + c) % hashMod[i];

}

}

vector<int> getHash(int l, int r) { // 得到哈希值

vector<int> res(HASH\_CNT, 0);

for (int i = 0; i < HASH\_CNT; ++i) {

int t =

(hsh[i][r] - 1ll \* hsh[i][l - 1] \* pwMod[i][r - l + 1]) % hashMod[i];

t = (t + hashMod[i]) % hashMod[i];

res[i] = t;

}

return res;

}

};

bool equal(const vector<int> &h1, const vector<int> &h2) {

assert(h1.size() == h2.size());

for (unsigned i = 0; i < h1.size(); i++)

if (h1[i] != h2[i]) return false;

return true;

}

int n;

StringWithHash s, t;

char str[L];

void work() {

int len = strlen(str); // 取字符串长度

t.init();

for (int j = 0; j < len; ++j) t.extend(str[j]);

int d = 0;

for (int j = min(len, s.ls); j >= 1; --j) {

if (equal(t.getHash(1, j), s.getHash(s.ls - j + 1, s.ls))) { // 比较哈希值

d = j;

break;

}

}

for (int j = d; j < len; ++j) s.extend(str[j]); // 更新答案数组

}

int main() {

scanf("%d", &n);

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

scanf("%s", str);

work();

}

printf("%s\n", s.s + 1);

return 0;

}