Remember the Word

LA3942, Uva1401

Time: 3 seconds

Problem Descriptions(1/2)

- Neal is very curious about combinatorial problems, and now here comes a problem about words. Knowing that Ray has a photographic memory and this may not trouble him, Neal gives it to Jiejie. Since Jiejie can’t remember numbers clearly, he just uses sticks to help himself.

Problem Descriptions(2/2)

- Allowing for Jiejie’s only 20071027 sticks, he can only record the remainders of the numbers divided by total amount of sticks.
- The problem is as follows: a word needs to be divided into small pieces in such a way that each piece is from some given set of words.
- Given a word and the set of words, Jiejie should calculate the number of ways the given word can be divided, using the words in the set.

Input

- The input file contains multiple test cases.
- For each test case: the first line contains the given word whose length is no more than 300000.
- The second line contains an integer S, 1 ≤ S ≤ 4000.
- Each of the following S lines contains one word from the set. Each word will be at most 100 characters long. There will be no two identical words and all letters in the words will be lowercase.
- There is a blank line between consecutive test cases. You should proceed to the end of file.

Output

- For each test case, output the number, as described above, from the task description modulo 20071027.

Sample Input / Output

```
abcd
4
a
b
cd
ab
```

A given word

Number of word in word set

S	a	b	c	d	
i	0	1	2	3	4
dp[i]	8	4	2	1	1

```
Case 1: 2
```

abcd=a+b+cd
=ab+cd

State Definition

S[i..N-1]

Word Set:
{a, b, cd, ab}

S	a	b	c	d	
i	0	1	2	3	4
dp[i]	2	1	1	0	1

{c d}

State Definition

S[i..N-1]

Word Set:
{a, b, cd, ab}

S	a	b	c	d	
i	0	1	2	3	4
dp[i]	2	1	1	0	1

{b}+{c d}

State Definition

S[i..N-1]

Word Set:
{a, b, cd, ab}

S	a	b	c	d	
i	0	1	2	3	4
dp[i]	2	1	1	0	1

{a}+{b}+{c d}

{a b}+{c d}

State Definition

S[i..N-1]

Word Set:
{a, b, cd, ab}

S	a	b	c	d	
i	0	1	2	3	4
dp[i]	2	1	1	0	1

{a}+{b}+{c d}

{a b}+{c d}

Calculation

Word Set:
{a, b, cd, ab}

從 N-1 到 0 計算 ←

S	a	b	c	d	
i	0	1	2	3	4
dp[i]	①	1 ②	1	0	1

① Prefix {a} dp[0]=dp[0]+dp[1] {a}+{b c d}

② Prefix {a b} dp[0]=dp[0]+dp[2] {a b}+{c d}

,

Solution 1

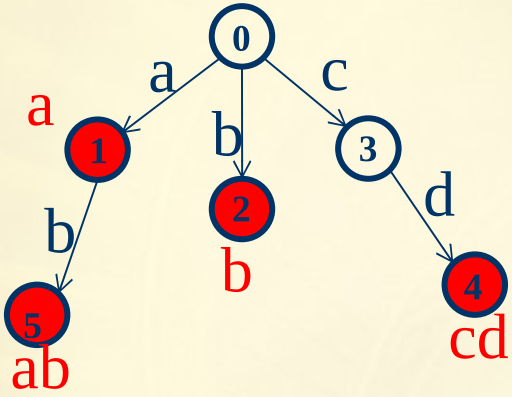
- Check whether each word in the word set is the prefix of S?
- There maybe 4000 words in the word set.
- Beyond the time limit.

Solution 2

- Check each prefix of S and check whether this prefix is in the word set.
- How to construct an efficient data structure to fast check?

Trie (Prefix Tree)

- **Word set {a, b, cd, ab}**



How to implement Trie?

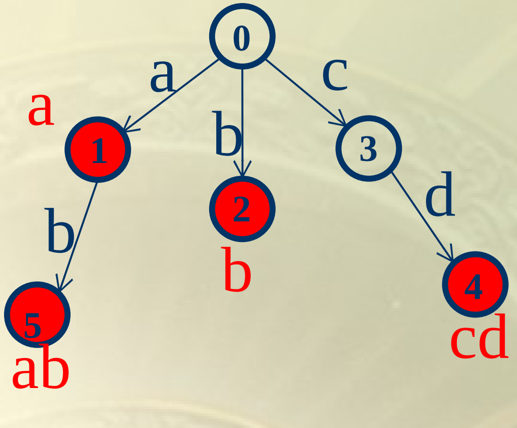
val

0	**root**
1	1
2	1
3	0
4	2
5	2

ch 26 letters

	a	b	c	d	e	f	g	
	0	**1**	**2**	**3**	**4**	**5**	**6**	**…**
0	1	2	3					
1		5						
2								
3				4				
..								

Maximum number of node



Example

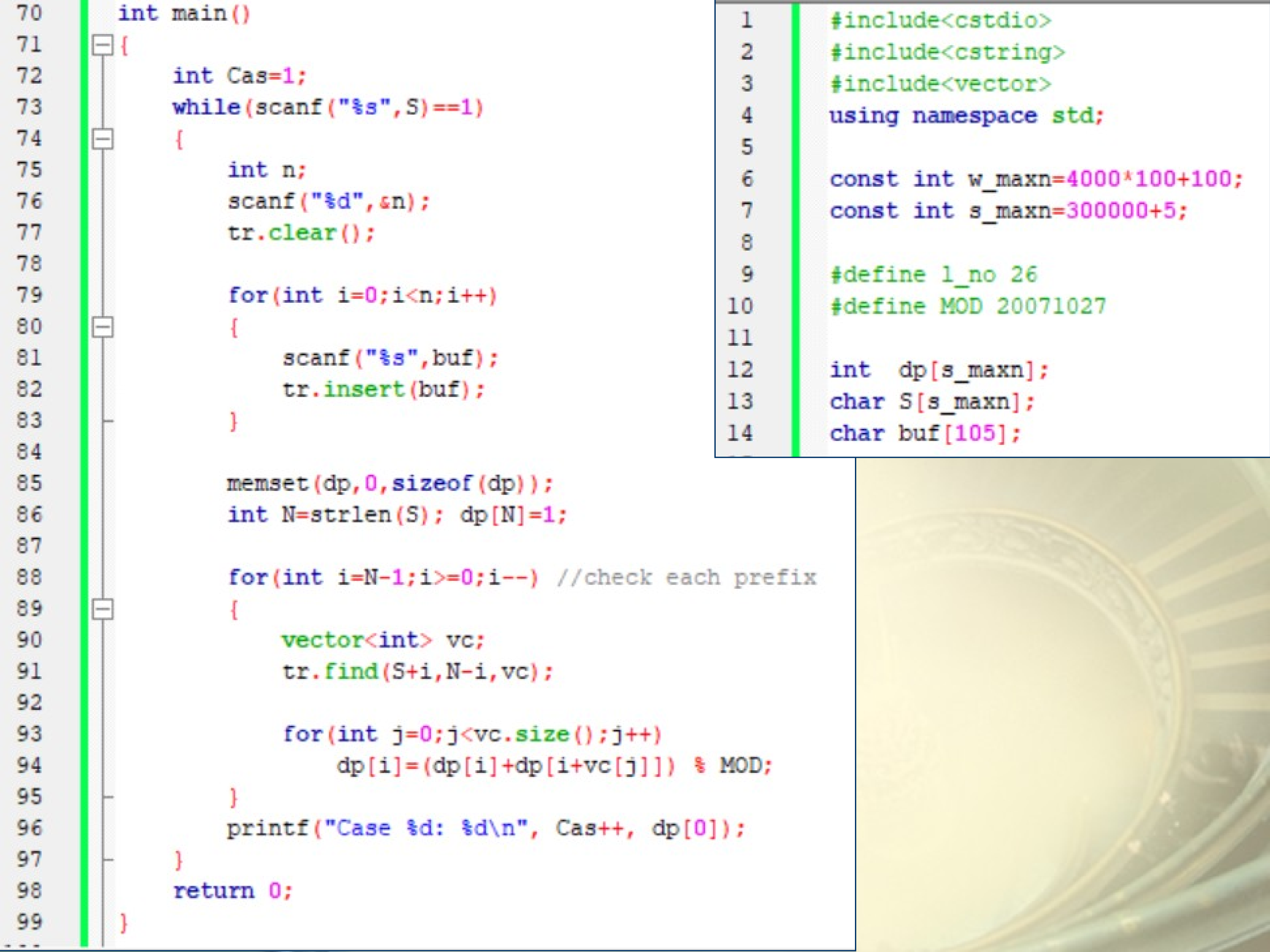
Word Set:
{a, b, cd, ab}

dp	a	b	c	d	
i	0	1	2	3	4
d[i]				0	1

dp	a	b	c	d	
i	0	1	2	3	4
d[i]		1	1	0	1

dp	a	b	c	d	
i	0	1	2	3	4
d[i]			1	0	1

dp	a	b	c	d	
i	0	1	2	3	4
d[i]	2	1	1	0	1



```cpp
#include<cstdio>
#include<cstring>
#include<vector>
using namespace std;

const int w_maxn=4000*100+100;
const int s_maxn=300000+5;

#define l_no 26
#define MOD 20071027

int  dp[s_maxn];
char S[s_maxn];
char buf[105];
```

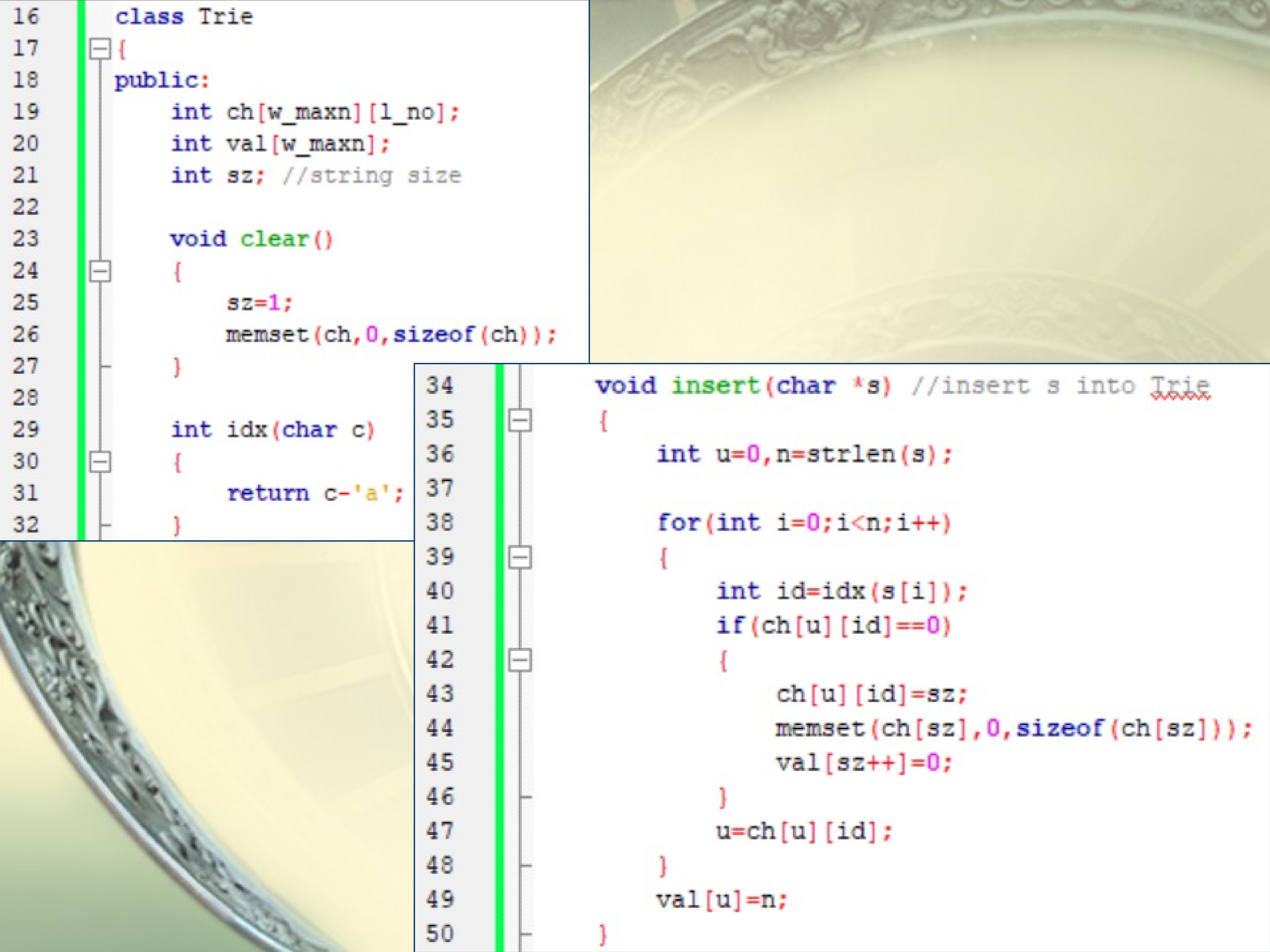
```cpp
int main()
{
    int Cas=1;
    while(scanf("%s",S)==1)
    {
        int n;
        scanf("%d",&n);
        tr.clear();

        for(int i=0;i<n;i++)
        {
            scanf("%s",buf);
            tr.insert(buf);
        }

        memset(dp,0,sizeof(dp));
        int N=strlen(S); dp[N]=1;

        for(int i=N-1;i>=0;i--) //check each prefix
        {
            vector<int> vc;
            tr.find(S+i,N-i,vc);

            for(int j=0;j<vc.size();j++)
                dp[i]=(dp[i]+dp[i+vc[j]]) % MOD;
        }
        printf("Case %d: %d\n", Cas++, dp[0]);
    }
    return 0;
}
```



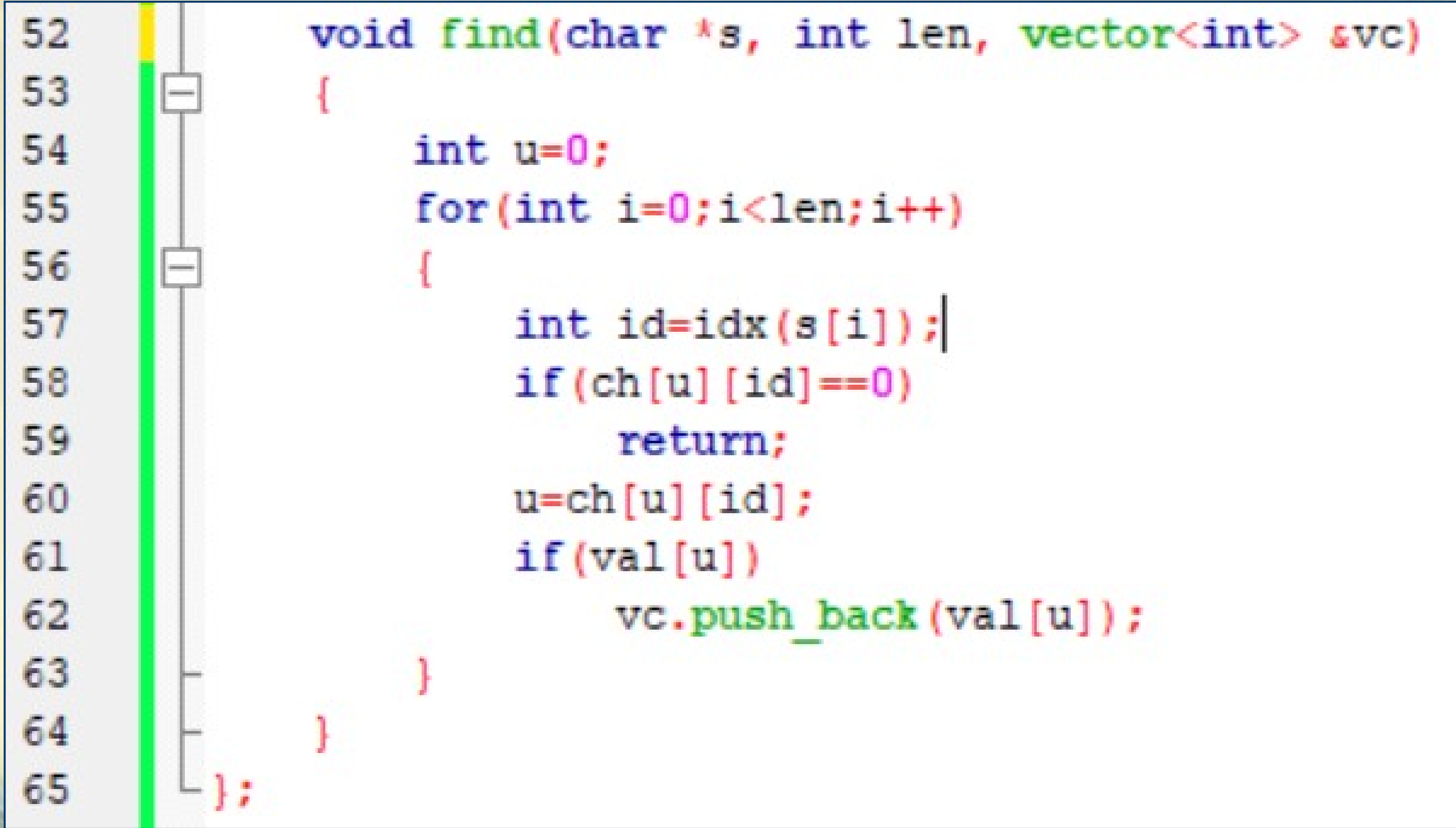
```
class Trie
{
public:
    int ch[w_maxn][l_no];
    int val[w_maxn];
    int sz; //string size

    void clear()
    {
        sz=1;
        memset(ch,0,sizeof(ch));
    }

    int idx(char c)
    {
        return c-'a';
    }
```

```
    void insert(char *s) //insert s into Trie
    {
        int u=0,n=strlen(s);

        for(int i=0;i<n;i++)
        {
            int id=idx(s[i]);
            if(ch[u][id]==0)
            {
                ch[u][id]=sz;
                memset(ch[sz],0,sizeof(ch[sz]));
                val[sz++]=0;
            }
            u=ch[u][id];
        }
        val[u]=n;
    }
```



```
    void find(char *s, int len, vector<int> &vc)
    {
        int u=0;
        for(int i=0;i<len;i++)
        {
            int id=idx(s[i]);
            if(ch[u][id]==0)
                return;
            u=ch[u][id];
            if(val[u])
                vc.push_back(val[u]);
        }
    }
};
```