Trie Tree （字典树）的简单使用 与 模板。 收藏

view plaincopy to clipboardprint?

/\*

Name: Trie Tree

Copyright:

Author: Green Tsai //蔡根同学写的。

Date: 22/03/10 21:56

Description:

\*/

#include <cstdlib>

#include <iostream>

using namespace std;

const int kind=26;

struct node

{

int count; // 记录字典中出现的次数

node \*next[kind];

node()

{

count=0; // 创建结点即代表该单词出现一次

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

{

next[i]=NULL; // 指针初始化为空

}

}

};

typedef node trietree;

void insert(trietree \*root,char \*word)

{

node \*local=root;

if(local==NULL)

{

local=new trietree();

root=local;

}

int i=0; // counter

int branch;

while(word[i]!='\0')

{

branch=word[i]-'a';

if(local->next[branch]!=NULL)

{

//local->next[branch]->count++; // here for solution 2

}

else

{

local->next[branch]=new trietree();

}

i++;

local=local->next[branch];

}

local->count++; // here for solution 1

}

int search(trietree \*root,char \*word)

{

node \*local=root;

if(local==NULL)

{

return 0;

}

int i=0;

int branch;

int ans;

while(word[i]!='\0')

{

branch=word[i]-'a';

if(local->next[branch]==NULL)

{

return 0;

}

i++;

local=local->next[branch];

ans=local->count;

}

// for(i=0;i<kind;i++) // here for solution 2

// {

// if(local->next[i]!=NULL)

// {

// ans-=local->next[i]->count;

// }

// }

return ans;

}

int main()

{

char str[30];

trietree \*root=new trietree();

cout<<"Please input the words:\n";

while(gets(str) && strcmp(str,""))

{

insert(root,str);

}

cout<<"Please input the word you need to search:\n";

cin>>str;

cout<<search(root,str)<<endl;

system("PAUSE");

return 0;

}

// Trie Tree模板。

//

//

////Name: Trie树的基本实现

////Author: MaiK

////Description: Trie树的基本实现 ,包括查找 插入和删除操作\*/

//#include<algorithm>

//#include<iostream>

//using namespace std;

//

//const int sonnum=26,base='a';

//struct Trie

//{

// int num;//to remember how many word can reach here,that is to say,prefix

// bool terminal;//If terminal==true ,the current point has no following point

// struct Trie \*son[sonnum];//the following point

//};

//Trie \*NewTrie()// create a new node

//{

// Trie \*temp=new Trie;

// temp->num=1;temp->terminal=false;

// for(int i=0;i<sonnum;++i)temp->son[i]=NULL;

// return temp;

//}

//void Insert(Trie \*pnt,char \*s,int len)// insert a new word to Trie tree

//{

// Trie \*temp=pnt;

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

// {

// if(temp->son[s[i]-base]==NULL)temp->son[s[i]-base]=NewTrie();

// else temp->son[s[i]-base]->num++;

// temp=temp->son[s[i]-base];

// }

// temp->terminal=true;

//}

//void Delete(Trie \*pnt)// delete the whole tree

//{

// if(pnt!=NULL)

// {

// for(int i=0;i<sonnum;++i)if(pnt->son[i]!=NULL)Delete(pnt->son[i]);

// delete pnt;

// pnt=NULL;

// }

//}

//Trie\* Find(Trie \*pnt,char \*s,int len)//trie to find the current word

//{

// Trie \*temp=pnt;

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

// if(temp->son[s[i]-base]!=NULL)temp=temp->son[s[i]-base];

// else return NULL;

// return temp;

//}