

Tire树

1. 简介

英文字典树每一个内部结点都有26个子结点，树的高度为最长字符串长度一棵子树代表具有相同前缀的关键码的集合。例如“an”子树代表具有相同前缀an-的关键码集合{and, ant}

2. 实现原理

1. 插入

首先根据插入纪录的关键码找到需要插入的结点位置。如果该结点是叶结点，那么就将其分裂出两个子结点，分别存储这个纪录和以前的那个纪录。如果是内部结点，则在那个分支上应该是空的，所以直接为该分支建立一个新的叶结点即可。

2. 查找

对待查询的的字符与树中的子节点进行比较，第一个字符与树根相比，如果不为空，则查询树根的所有子树并且和第二个字符相比。以此类推，如果进行到某一层，所有字符均已被匹配，且当前层存在标记，则查询成功。若在任意一层对应位置的字符的子树为空或者，最后一层不存在标记，则查询失败。

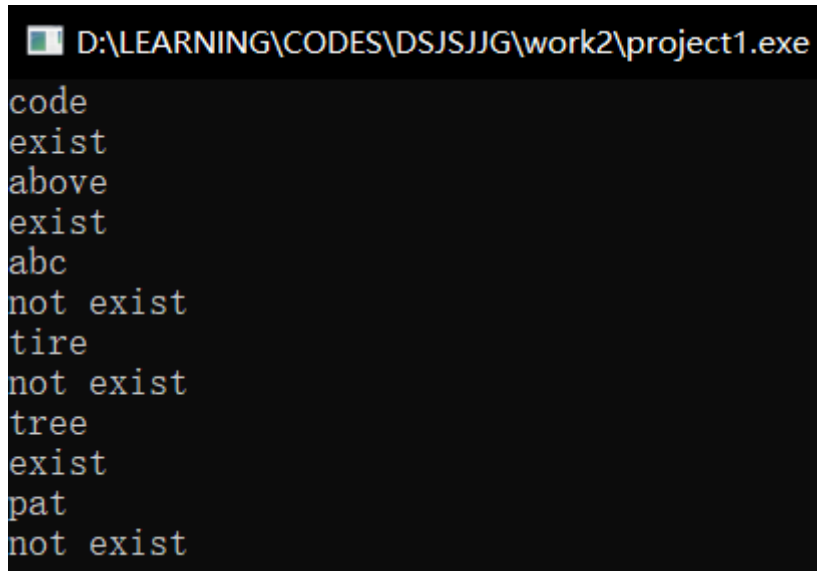
3. 删除

根据插入纪录的关键码找到需要删除的结点位置。如果一个被删除结点的父结点没有其他的儿子，那么就需要合并。否则只需要将此分支设置为空即可。

实验实现

此次实验，使用了个人完成的tire树进行。

1. 实验任务1



```
D:\LEARNING\CODES\DSJSJG\work2\project1.exe
code
exist
above
exist
abc
not exist
tire
not exist
tree
exist
pat
not exist
```

2. 实验任务2

```
D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
a 13
abandon 1
about 2
air 1
all 6
always 1
am 5
and 27
anger 1
animals 1
any 1
anymore 1
anything 1
appeared 1
are 1
arms 1
as 1
at 5
ate 1
attention 2
away 2
baby 13
bad 1
beautiful 1
behind 1
better 1
big 1
body 1
bowl 1
but 3

D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
came 1
caught 1
clearly 1
cliff 2
closer 1
come 3
comfortable 1
coming 3
complained 1
could 4
cried 8
day 3
days 1
death 2
delicious 2
did 2
die 3
dived 1
do 3
down 6
dying 2
each 1
eagle 11
eat 1
end 1
enjoying 1
even 1
ever 2
experienced 1
eyes 1
```

```
D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
face 1
fair 1
faster 4
feed 1
feeling 1
fell 1
felt 1
few 1
first 1
flew 1
flying 1
food 5
for 1
from 3
fun 1
get 2
give 1
god 1
going 3
good 1
great 2
grew 2
ground 2
grous 1
had 5
happens 1
have 1
he 23
head 1
hear 1

D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
her 2
here 2
him 3
his 10
how 4
huh 1
hungry 2
i 5
idea 1
in 3
is 8
it 7
its 1
just 1
know 1
known 1
land 1
lano 1
last 2
later 1
laughed 1
like 1
liked 1
little 1
lives 1
living 1
looked 5
lots 1
love 2
magic 1
```

```
D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
many 1
me 4
meal 2
minute 1
more 1
mother 14
mountain 1
moving 1
much 1
nearby 1
nest 6
no 2
not 3
nourishing 1
of 5
on 2
once 1
one 3
only 1
or 1
out 1
over 2
overlooking 1
own 1
perched 1
picked 2
pointt 1
provide 1
pushed 1
rushed 1

D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
said 9
saw 1
screamed 2
scurm 1
sharp 1
she 8
sky 2
snapped 1
so 2
soared 1
some 1
something 1
soon 2
speed 2
stopped 1
strange 1
streaming 1
streams 1
strong 1
sun 1
sure 1
swooped 1
swot 1
tasty 1
tears 1
terrible 1
that 1
the 38
their 1
then 2
```

```
D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
there 3
they 1
this 5
time 6
times 1
to 11
top 1
towards 1
trees 1
unlike 1
until 1
up 6
upon 1
valley 1
very 3
view 1
vision 1
warm 1
was 10
waterfalls 1
were 1
what 2
where 1
whined 2
why 1
with 7
worked 1
world 1
would 2
yes 1

D:\LEARNING\CODES\DSJSJJG\work2\project2.exe
time 6
times 1
to 11
top 1
towards 1
trees 1
unlike 1
until 1
up 6
upon 1
valley 1
very 3
view 1
vision 1
warm 1
was 10
waterfalls 1
were 1
what 2
where 1
whined 2
why 1
with 7
worked 1
world 1
would 2
yes 1
you 4
your 3
```

源代码

```

tire.h
#include <string>
struct node
{
    node *next[26];
    bool isStr;
    int count = 0;
    node()
    {
        for (int i = 0; i < 26; i++)
        {
            next[i] = NULL;
        }
    }
    void insert(std::string s)
    {
        node *p = this;
        for (int i = 0; i < s.size(); i++)
        {
            if (p->next[s[i] - 'a'] == NULL)
            {
                node *temp = new node;
                temp->isStr = false;
                p->next[s[i] - 'a'] = temp;
            }
            p = p->next[s[i] - 'a'];
        }
        p->isStr = true;
        p->count++;
    }
    bool find(std::string s)
    {
        node *p = this;
        for (int i = 0; i < s.size(); i++)
        {
            if (p->next[s[i] - 'a'] == NULL)
            {
                return false;
            }
            else
            {
                p = p->next[s[i] - 'a'];
            }
        }
        if (p->isStr)
            return true;
        else
            return false;
    }
    bool remove(std::string s)
    {

```

```

node *p = this;
for (int i = 0; i < s.size(); i++)
{
    if (p->next[s[i] - 'a'] == NULL)
    {
        return false;
    }
    else
    {
        p = p->next[s[i] - 'a'];
    }
}
if (p->isStr)
{
    p->isStr = false;
    p->count = 0;
    return true;
}
else
{
    return false;
}
}
};

```

project1:

```

#include "tire.h"
#include <iostream>
#include <fstream>
#include <string>
void init(node &root)
{
    std::ifstream wordlist("wordlist1.txt");
    std::string line;
    while(getline(wordlist, line))
    {
        root.insert(line);
    }
}
int main()
{
    node T;
    init(T);
    while(1)
    {
        std::string op;
        std::cin >> op;
        if(op=="q")
        {
            return 0;
        }
    }
}

```

```

    }
    else
    {
        if(T.find(op))
        {
            std::cout << "exist" << std::endl;
        }
        else
        {
            std::cout << "not exist" << std::endl;
        }
    }
}
}
}

```

project2:

```

#include "tire.h"
#include <iostream>
#include <fstream>
#include <string>
void init(node &root)
{
    std::ifstream file("article1.txt");
    std::string str;
    while (file >> str)
    {
        root.insert(str);
    }
}
void show(node *root, std::string now)
{
    for (int i = 0; i < 26; i++)
    {
        if (root->next[i] == NULL)
            continue;
        if (root->next[i]->isStr)
        {
            std::cout << now + (std::string(1, char('a' + i))) << " " << root->next[i]->count <<
        }
        show(root->next[i], now + (std::string(1, char('a' + i))));
    }
}
int main()
{
    node T;
    init(T);
    show(&T, "");
    getchar();
}

```