由于基础知识不扎实而“众里寻他千百度”的bug

1:对负数的取余：-19%10=-9；负数取余后为负数

2：ceil,floor函数只对浮点数有效；对整数无效。

3：for(auto tmp:dp)中的tmp相当于一个暂存变量，改变它的值并不会改变数组中的值。

4: 树结点的指针需要初始化为null，不然会导致指针错误指引引起错误。指针数组的初始化可以用memset(名称，初值，sizeof（名称);

```
struct TreeNode{
        char ch;
        bool isEnd;
        TreeNode*next[26];
        TreeNode():ch('0'),isEnd(false){}
        TreeNode(char c,bool p):ch(c),isEnd(p){}
    };
    TreeNode*root;

WordDictionary() {
        root=new TreeNode();
        memset(root->next, NULL, sizeof(root->next));//指针数组的初始化
    }
```

5：回溯算法中的 return与终止

```
1: -->for(int i=0;i<26;i++){
                    if(backtracking(word,0,root->next[i])){//正确写法，这样写可以维持for循环的运行，并在第一次出现true的时候终止运行;这样写适用只需要用回溯寻找一种答案的情况;这样在找到一个答案之后可终止后面的不必要的运行;
                        return true;
                        break;
                    }
                    //return true;
                }
2-->for(int i=0;i<26;i++){
                   retrun backtracking(word,0,root->next[i]);//错误写法，直接return的话，这里永远只会在i=0的时候运行，使得for循环失去了作用
                }
```

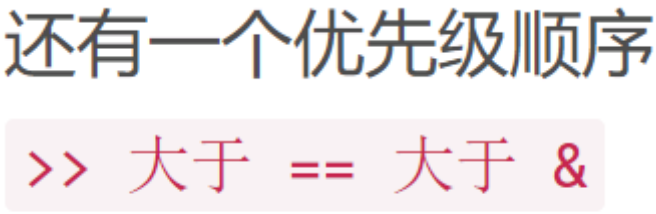
for循环终止后的值：

```
int i;
for(i=0;i<prefix.size();i++){
            if(temp->next[prefix[i]-'a']){
                temp=temp->next[prefix[i]-'a'];
            }else break;
        }
//这个循环终止后i的值为prefix.size();而不是prefix.size()-1;
//注意
```

vector中的无符号与负数进行比较时可能出现bug

```
//由于有符号和无符号的比较产生超出时间限制
        int n=num.size();
        for(int i=0;i<n-2;i++){
            //此处如果n-2<0；由于无符号时没有负数，负数反而表示大整数；
        }
```

位运算与关系运算的优先级



运算符的优先级（从高到低）

优先级	描述	运算符
1	括号	()、[]
2	正负号	+、-
3	自增自减，非	++、--、！
4	乘除，取余	*、/、%
5	加减	+、-
6	移位运算	<<、>>、>>>
7	大小关系	>、>=、<、<=
8	相等关系	==、!=
9	按位与	&
10	按位异或	^
11	按位或	|
12	逻辑与	&&
13	逻辑或	||
14	条件运算	?:
15	赋值运算	=、+=、-=、*=、/=、%=
16	位赋值运算	&=、|=、<<=、>>=、>>>=


```
if (((j >> i) & 1) == 0) {//此处若写为(j>>i)&1==0则会先运算1==0结果位0,即false;
                    flag = 0;
                    break;
                }
```

for循环中的越界

```
for (int j = 5; j <= 2147483648; j++)
    //对于这个循环，再j=2147483648时若再加1会使得有符号整数变为-1;
    //所以在存在j=2147483648的情况下应该将有符号设置位无符号
```