

数据结构训练报告

第 1 次



姓名 凌晨

班级 软件 2104 班

学号 2214414320

电话 18025402131

Email lingchen47@outlook.com

日期 2023-1-28

目录

任务 1：实现 BST 数据结构.....	3
任务 2：使用 BST 为文稿建立单词索引表.....	7
任务 3：实现 Trie.....	8
任务 4：使用 Trie 实现 T9 键盘.....	13
附录.....	15
任务 1and 任务 2	15
任务 3.....	22
任务 4.....	26

任务 1：实现 BST 数据结构

1.1 BST 数据结构类设计

1.1.1 BST 设计如下：

<code>BST<K extends Comparable<K>,V></code>
<code>private class Node; private Node root; private int numNode; private int height;</code>
<code>public void insert(K key, V value) public V remove(K key) public V search(K key) public boolean update(K key,V value) public boolean isEmpty() public void clear() public void showStructure(PrintWriter pw) public void printInorder(PrintWriter pw) public Node insertHelp(Node root,K key,V value) public Node searchHelp(Node root, K key) public Node removeHelp(Node root,K key) public void printInorderHelp(PrintWriter pw,Node rt) public Node minNode(Node root) public Node delMinNode(Node root) public int getHeight(Node root,int height)</code>

下面进行具体的说明。

首先是关于变量。该类一共有 3 个变量，而 Node 为内部类，后续会介绍。root 为 BST 的根节点，这个在创建 BST 时初始化，原则上后续不继续修改；numNode 记录该树中一共有多少个节点，numNode 在各个方法中刷新；height 记录该树的高度，height 依靠 getHeight 方法刷新。

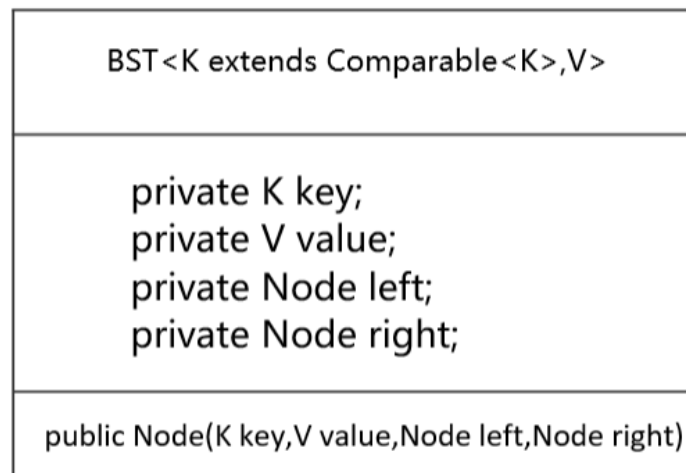
然后，开始介绍方法。题目中要求的方法不再赘述。着重介绍题目中未要求的方法，各种***Help 方法为辅助方法，帮助对应的***方法，比如 insertHelp 方法是为了实现 insert 方法的辅助方法，这么做的原因如下：

- 1.由于题目要求的方法的参数固定，有时候仅仅依靠这些不足以完成；
- 2.对于树的结构，常常利用递归思想设计方法，而返回值有时候不满足条件。
minNode, delMinNode 方法如名字一样，一个是寻找最小节点并且返回该节点，另一

个是在不破坏树的性质删除最小节点。

getHeight 方法前面有提及，不赘述。

1.1.2 关于内部类 Node 设计如下：



下面进行说明：

Node 类的作用如名字意思一致，作为 BST 的节点存在，因此需要存储信息，所以变量中含有键值对，关键词 key 和值 value，同时为了满足 BST 树的性质需要有左子节点 left 和右子节点 right。对于节点无需过多的方法，我只给出了一个构造方法。

1.2 BST 接口实现

1.2.1 insert 方法

实现 insert 操作的主要代码是在 insertHelp 中。下面讲解其方法的核心思想：

插入情况主要分为两种。第一种为向空的根节点插入，第二种为向非空的根节点插入。由于该方法采用了递归，根节点的位置不断向下推进改变，因此遇到第一种情况时候说明找到了相应位置，所以创建新的节点，向节点存储键值对即可。而遇到第二种情况，为了保障 BST 性质则需要对关键字的比对，我设计的 BST 为左子节点小于父节点，右子节点大于父节点，因此对应情况操作即可。

代码附录可见，不再给出，与课上所讲基本一致。

1.2.2 remove 方法

remove 方法可以说是情况最多最复杂的方法了，为了搞定 remove 我们先来了解 minNode 和 delMinNode 方法。

minNode 方法就是寻找键值为最小的节点，根据 BST 的性质，只要递归到最左节点即可，实现较简单，不再赘述。

delMinNode 方法是不破坏 BST 性质，删除最小节点。首先，先来分析有几种情况：1. 该节点无子节点；2. 该节点仅有一个子节点；3. 该节点有两个子节点。其中情况 1 的解决方法直接删除即可。由于删除的是最小节点，情况 2 若有左子节点进行递归，若为右子节点则让该节点的父节点的左子节点设置为该节点的右子节点，从而实现删除。情况 3 综合情况 1 和情况 2 运用递归即可。注意，由于我们不能提前知道最小节点，我们只有 BST 的根节点，因此需要自上而下进行递归，核心代码如下：

```
1. if (root.left==null) return root.right;  
2. else {
```

```

3.     root.left = delMinNode(root.left);
4.     return root;
5. }

```

了解了如上两个辅助方法，现在着手解决 remove 方法。

遇到的情况也可以分为如下三种：1.该节点无子节点；2.该节点仅有一个子节点；3.该节点有两个子节点。1 解决方法为直接删除；2 解决方法与 delMinNode 方法相似，不再赘述；而 3 解决方法比较巧妙，我们可以让该节点的右子树的最小节点代替该节点，然后删除右子树的最小节点。这样子就完美解决了 remove 方法。具体代码不再给出，可见附录，核心思想一致。值得注意的是，我在上文说的是代替，实际上，在编码的时候，我只是替换了键值对信息，没有改变地址值。

1.2.3 search 方法

利用 BST 性质进行递归即可，若关键字一致的即返回值，若大于则转向右子树，小于则转向左子树。代码不给出，可见附录。

1.2.4 update 方法

步骤为找到对应节点，若存在更改值并返回 true，不存在返回 false 即可。代码不给出，可见附录。

1.2.5 isEmpty 方法

return root==null;即可

1.2.6 clear 方法

将几个变量初始化为新建该类时候的状态即可。

1.2.7 showStructure 方法

略；

1.2.8 printInorder 方法

为了保障输出是按照一定顺序的，采用了中序遍历。剩余较简单，不再赘述。

1.3 BST 测试

1.3.1 测试类的编写

测试类的编写难度主要在于对于文件中的字符串进行处理，处理完的字符串操作简单，使用 switch 块即可完成，因此我将在下文着重讲解字符串的处理。

需要处理的有五种情况，分别如下：

```

1. +( mutton , "n. 羊肉" )
2. =( laggard , "laggard" )
3. ?( penalize )
4. -( miscreant )
5. #

```

归类后只有两种情况，1, 2 为一种，3, 4, 5 为一种。使用 Scanner 逐行扫描字符串。

使用 `char syn = str.charAt(0);` 保存操作符号。

对于情况 1 采用如下代码获得 key 和 value

```
1. int pivot = 2;
2. while(str.charAt(pivot)!=';') pivot++;
3. key = value = "";
4. for(int i=3;i<pivot-1;i++) key+=str.charAt(i);
5. for(int i=pivot+3;i<str.length()-3;i++) value+=str.charAt(i);
```

对于情况 2 只需要获得 key 即可，仿照上面的代码即可。

因此，我们处理完了字符串，剩下的只需要按照操作符号操作即可，不再赘述，具体代码见附录。

1.3.2 验证

编写了验证代码，核心思想就是逐行比对字符串，比对结果为全部准确，下面给出部分截图。

<pre>remove success ---deflect v. 偏离, 转向 update success ---mattress mattress search success ---acquaintance n. 熟知, 熟人 update success ---penury penury remove success ---rampant adj. 蔓生的, 猖獗的 search success ---circumvent v. 用计谋战胜, 规避 remove unsuccessful ---drenched search success ---clientele n. (医生、律师) 顾客, (----- There are 206 nodes in this BST. The height of this BST is 13. ----- search success ---contumacy n. 抗命, 不服从 search success ---dispatch dispatch</pre>	<pre>remove success ---deflect v. 偏离, 转向 update success ---mattress mattress search success ---acquaintance n. 熟知, 熟人 update success ---penury penury remove success ---rampant adj. 蔓生的, 猖獗的 search success ---circumvent v. 用计谋战胜, 规避 remove unsuccessful ---drenched search success ---clientele n. (医生、律师) 顾客, (----- There are 206 nodes in this BST. The height of this BST is 13. ----- search success ---contumacy n. 抗命, 不服从 search success ---dispatch dispatch</pre>
--	--

图片 1

左半部分为测试类生成文本，右半部分为老师提供的文本，行数为 335-347。

<pre>update success ---sophism sophism search unsuccessful ---redeem search success ---gauge n. 标准规格, 测量仪 remove success ---prevail v. 战胜, 盛行 remove success ---primp v. (妇女) 刻意打扮 ----- There are 300 nodes in this BST. The height of this BST is 15. ----- remove success ---stipulate v. 要求以...为条件, 约定 update success ---piercing piercing search success ---inaugurate v. 举行就职典礼, 开创 remove success ---flush n. v. 脸红, 奔流 search unsuccessful ---veal remove success ---baroque n. adj. (艺术、建筑等) 高度装饰 search success ---omission omission search success ---traduce v. 中伤, 诽谤 remove success ---legislature n. 立法机关 update success ---temerity temerity</pre>	<pre>update success ---sophism sophism search unsuccessful ---redeem search success ---gauge n. 标准规格, 测量仪 remove success ---prevail v. 战胜, 盛行 remove success ---primp v. (妇女) 刻意打扮 ----- There are 300 nodes in this BST. The height of this BST is 15. ----- remove success ---stipulate v. 要求以...为条件, 约定 update success ---piercing piercing search success ---inaugurate v. 举行就职典礼, 开创 remove success ---flush n. v. 脸红, 奔流 search unsuccessful ---veal remove success ---baroque n. adj. (艺术、建筑等) 高度装饰 search success ---omission omission search success ---traduce v. 中伤, 诽谤 remove success ---legislature n. 立法机关 update success ---temerity temerity</pre>
--	--

图片 2

左半部分为测试类生成文本，右半部分为老师提供的文本，行数为 487-506。

不再给出截屏，上面验证程序和截屏都可以说明测试通过，编写的 BST 准确无误

任务 2：使用 BST 为文稿建立单词索引表

2.1 数据清洗

2.1.1 规则制定

数据清洗规则如下：

1. 单词不区分大小写，存储一律按照全部小写进行存储。

2. 对于含有特殊符号的单词处理如下：

如 I'll 只保留特殊符号前的单词 I

如 Mr.** 只保留特殊符号前的单词 Mr

如 (well) 删去 ()，保留 well

3. 单词长度不做限制，均保留

2.1.2 清洗数据实现

编写 public static String clean(String s) 方法作为清洗数据的方法

具体步骤如下：

1. 全部转化为小写；

```
1. s = s.toLowerCase();
```

2. 过滤前面的特殊符号

```
1. for (int i = 0; i < s.length(); i++) {  
2.     tem = s.charAt(i);  
3.     if(tem<='z'&&tem>='a'){  
4.         pivot=i;  
5.         break;  
6.     }  
7. }
```

3. 取字符直到特殊符号（代码与步骤 2 的代码类似，不再给出）

2.2 数据录入

将处理完的单词依次录入即可，情况简单，代码如下：

```
1. s = clean(s);  
2. if (s.equals("")) break;  
3. String preKey = myBST.search(s);  
4. if (preKey == null) {  
5.     myBST.insert(s, row + "");  
6. } else {  
7.     myBST.update(s, preKey + " "+row);  
8. }
```

2.3 数据验证

数据录入完成后，调用 `printlnorder` 方法即可。
详情可见附件

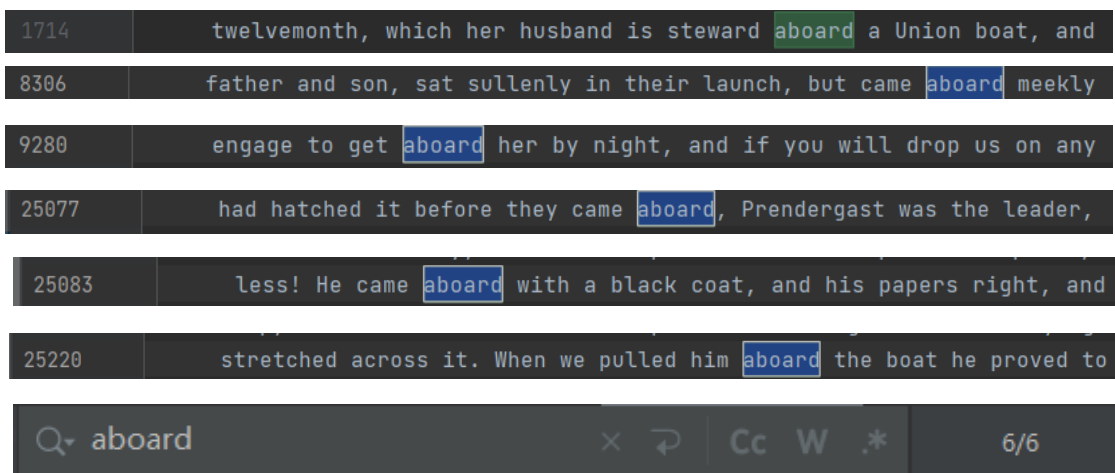


index_result.txt

随机选择一个单词进行验证：

[aboard ---- < 1714 8306 9280 25077 25083 25220 >]

在 article.txt 中进行查找，截屏如下：



可以看到查找一共有六个结果，而且行数都准确，这说明数据准确无误，数据清洗没有造成过多的单词损失。

任务 3：实现 Trie

3.1 Trie 类设计

3.1.1 Trie

Tire 类设计如下：

<h1>Trie</h1>
<pre>public static final int RED = 1; public static final int BLUE = 0; public Node root = new Node();</pre>
<pre>public void insert(String s) public void insert(Node rt, String s) public boolean search(String s) public Node search(Node rt,String s) public void delete(String s) public void delete(Node rt, String s) public void showStr(String s) public void showStr(Node rt,String s) public void showStr(Node rt)</pre>

下面进行说明：

两个常量代表着是否存储有用值——是否存储关键字。当存储关键字标记为 RED，反之标记为 BLUE。

变量 root 表示根节点，一般在创建 Trie 实例初始化，初始化后一般不改变。

可以看到，在这个类中使用了大量方法重载，出于以下考虑：

- 1.减小代码编写难度，方法重载可以使用一样的方法名，代码编写更加方便。
- 2.由于是自己设计的类，希望插入，查找，删除，展示结构的方法更加多元化，更加灵活，为后面的验证也起到了辅助作用。

3.1.2 Node 内部类

Node 内部类的设计如下：

<h1>Node</h1>
<pre>private int color = BLUE; private String res = ""; public Node[] subNode = new Node[26];</pre>
<pre>public Node() public Node(Node rt,char s) public void setBlue() public void setRed()</pre>

下面进行具体说明：

color 作为标记是否存储关键词;
res 为关键词或者前缀;
subNode 则为子树, 26 个代表 a-z;
有两个构造方法, 第一种为创建根节点, 第二种为创建非根节点
有两个实例方法, 改变标记。不赘述。

3.2 方法实现

3.2.1 insert 方法

insert 方法根据不同的参数输入有不同的操作, 所以只讲解核心操作, 下面方法一样。

根据题目要求可以知道, 假设插入的单词长度为 n , 那么这个单词形成的前缀树的深度就为 n 。可以通过遍历该单词, 即 `char c = s.charAt(i);` 找到 c 对应的 subNode 的位置, 若存在, 向下继续; 若不存在, 则新建类, 向下继续。当遍历完单词后, 把该节点设置为 RED 即可。

核心代码如下:

```
1.     if (rt.subNode[pivot] == null) {  
2.         rt.subNode[pivot] = new Node(rt, c);  
3.     }  
4.     rt = rt.subNode[pivot];
```

3.2.2 search 方法

可以借鉴 insert 方法, insert 是开辟并找到单词的对应位置, 而 search 无需开辟, 当遇到 null 即可返回 false, 当找到单词并且标记为 RED 时即可返回 true, 综上所述, 核心代码与 insert 一致。

3.2.3 delete 方法

删除有两种方式删除, 一种是显性删除, 相当于为 Trie 剪枝; 另一种是隐形删除, 即标记为 BLUE 即可。两种方式各有好处。第一种耗时长但节省了空间开支, 第二种耗时短但存在大量空间浪费的情况。

综合考虑, 我选择了第二种删除方式, 理由如下:

1. 第二种删除方式编码简单, 耗时短, 易操作。
2. 当有大量单词录入时, 删除插入操作是很频繁的, 因此, 隐形删除更好。
3. 由于前缀和的性质, 许多单词共享多种前缀, 当显性删除时, 必将涉及极其复杂的情况, 想要准确编码较难, 造成不必要的麻烦。
4. 本次实验只有 26 个字符, 假设单词长度为 n , 删除该单词最坏的空间浪费仅为 $26 \cdot n$, 而英语单词长度普遍较短, 空间浪费完全可以接受。

核心代码如下:

```
1. Node res = search(rt, s);  
2. res.setBlue();
```

3.2.4 showStr 方法

该方法采用了中序遍历, 输出结果为有序的。该方法核心思想为递归, 与之前的 BST 中 printInorder 方法编码基本一致, 不过是 2 个方向换为 26 个方向, 不再赘述。

3.3 测试

3.3.1 测试类的编写

测试类需要把 dictionary.txt 文件中所有的单词全部插入 Trie 中，这处理方式与 BST 测试类基本一致，甚至简略了数据清洗的过程。因此不再赘述。

3.3.2 设计测试及结果

现在已经得到了录入所有单词的 Trie，设计了如下测试。

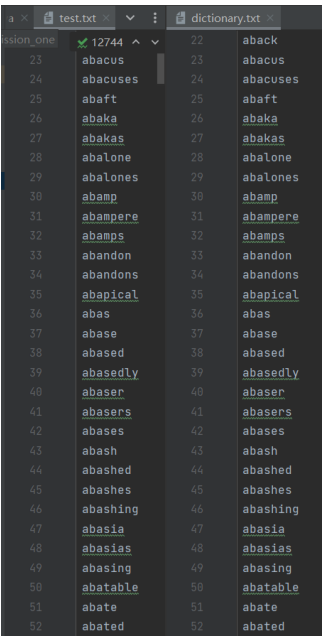
第一个测试：检验是否所有单词均录入，而且输出是否按照中序遍历。

编写程序如下，只需要调用 showStr 方法，录入参数为空字符串即可输出全部单词。

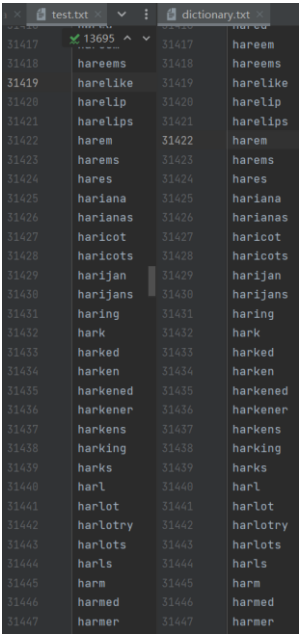
```
1. myTrie.showStr("");
```

接下的验证工作与 BST 的工作相似——利用 Printwriter 输入到 txt 文件中，再利用 BST 中逐行验证的程序即可。结果是全部匹配。

下面给出部分截图：



图片 3



图片 4

随便截取了一段，可以看出截屏内行数，单词全部一一对应，而且准确无误，可以说所有单词均录入而且中序输出准确无误。

第二个测试，测试前缀和。测试代码如下：

```
1. myTrie.showStr("well");
```

结果如下：

welladay wellaway wellborn wellcurb welldoer welled wellhead wellhole wellie wellies
welling wellness wells wellsite welly

人工比对均准确。这里需要提及一点，为什么 well 没有输出，因为按照我的理解，**well** 不算 **well** 的前缀和，特此说明。

第三个测试，该测试实验 delete 方法，代码如下：

```
1. myTrie.delete("welladay");
2. myTrie.showStr("wella");
```

输出为 wellaway

根据测试二的部分结果可以知道，welladay 的的确确地被删除了。

第四个测试，该测试测试 search 方法，判断能否准确查找

注意 search 有两种返回值。

先测试 boolean 返回值的方法，代码如下：

```
1. System.out.println(myTrie.search("well"));
```

输出为 true

```
1. System.out.println(myTrie.search("wellaa"));
```

输出为 false

结果准确无误。

再测试返回值 Node 方法，这个测试可以进行很多有趣的操作，代码如下：

```
1. System.out.println(myTrie.search(myTrie.root, "well"));
```

输出为 Trie\$Node@3feba861，这里需要解释，因为 Node 是 private 标记的内部类，在测试类中无法直接访问值或者调用方法，但是，通过 debugger，可以判断这是正确的。我们还可以进行如下操作：

```
1. System.out.println(myTrie.search(myTrie.search(myTrie.root, "well"), "
    aday"));
```

这个测试非常有趣，先找到 well 的 Node，再根据 well 的 Node 向下查找 aday，实际上就是在查找 welladay，结果为 Trie\$Node@3feba861，通过 debugger 可以知道答案准确。

（注意，Node 的返回地址每次结果都是随机的，答案准确是通过 debugger 判断的）

第五个测试，其实测试到这里就差不多结束了，而且基本能判断单词录入准确无误，各种方法都是完美实现的，因此最后的测试我要测试程序的健壮性，即说明我的删除操作使用隐形删除的正确性。

测试操作如下，我把 dictionary.txt 复制一份，在从中随机删除一些单词，命名为 dic.txt。把 dictionary.txt 所有单词录入后，根据 dic.txt 删除单词，计算程序运行的时间，代码如下：

```
1. System.out.println("开始删除");
2. long start = System.currentTimeMillis();
3. for (int i = 0; i < sum2.length(); i++) {
4.     str = new StringBuilder();
5.     while (i < sum2.length() && sum2.charAt(i) != '\n') {
6.         str.append(sum2.charAt(i));
7.         i++;
8.     }
9.     String strip = str.toString().strip();
10.    myTrie.delete(strip);
11. }
12. long end = System.currentTimeMillis();
```

```
13. System.out.println("用时:"+(end-start)+"ms");
14. System.out.println("删除完毕");
15. myTrie.showStr("");
```

输出如下:

开始删除

用时:56ms

删除完毕

cooeys coof coofs cooing cooingly cook cookable nankins nannie nannies nanny
nanogram nanowatt nans naoi naphtol naphtols napiform napkin napkins napless
napoleon nappe napped zyme zymes zymogen zymogene zymogens zymogram
zymology zymosan zymosans

可以看到录入了接近全部的数据，删除了大部分数据，只留了少部分单词，但是耗时极短。而且这么做有一个好处，当再次录入删除的单词时，无需开辟新空间，时间大大减少。

综上诉说，五个测试从不同方面展示该数据结构。而且均完美通过。

任务 4：使用 Trie 实现 T9 键盘

4.1 T9Trie

该数据结构核心思想与 Trie 基本一致，只是做出了如下调整:

1.Trie 的内部类 Node 是以 a-z 为区分，作子树的判断；T9Trie 的内部类 Node 则是以 2-9 作为区分，作子树的判断。

2.Trie 的内部类 Node 存储的 res 为 String 类型，作为前缀和；T9Trie 的内部类 Node 存储的 data 为 Sting[]类型，存储的是所有满足到该路径的单词。举个例子：

ace bad 的数字编码均为 223，因此 233 节点 Node 的 data 储存为{"ace","bad";...}

3.T9Trie 内部类 data 不是固定空间，可以通过 resize () 方法调整大小，这点避免了空间浪费。

上面说明得很详细了，因此设计图不再给出，详细代码可见附录。

4.2 数据的预处理

dictionary.txt 只给出各个单词，而没有进行 T9 键盘的数字编码，因此，需要对单词进行数字编码，即预处理。遍历单词，采取 switch 语法块可以快速完成该任务，实现简单，不再给出代码。

4.3 测试类

测试类主要是通过死循环和 sc 来获取键盘输入的字符串来达到题目要求的效果。

```
1. Scanner sc = new Scanner(System.in);
```

具体的细节不再给出，当输入为“exit”时 break 死循环，退出程序即可。当输入为“#”时，

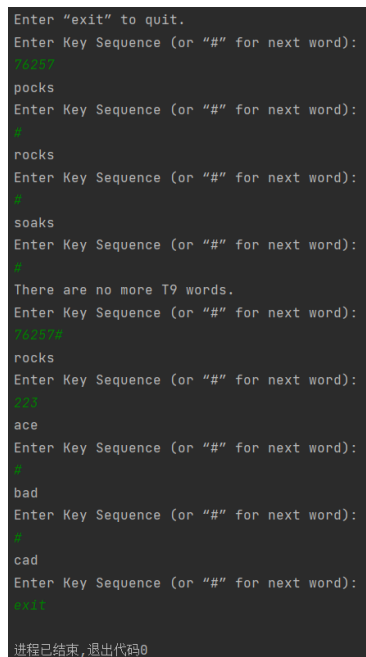
进行 data 取下一个操作即可。剩下情况为截取输入的字符串和后面#的个数，然后根据字符串和#个数输出即可。

代码如下：

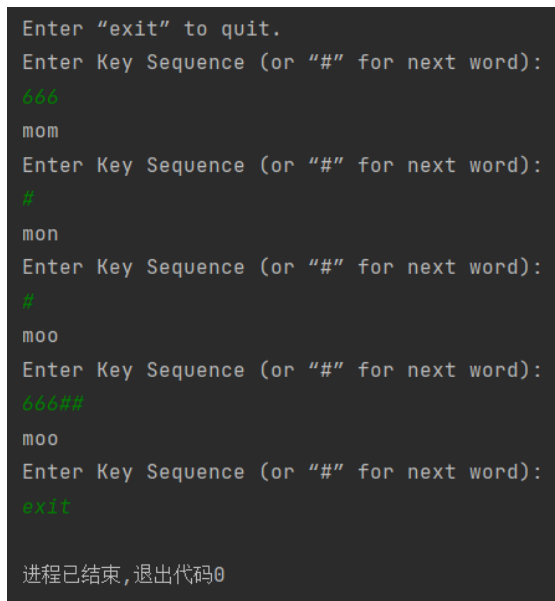
```
1. if (input.equals("exit")) break;
2. else if (input.equals("#")){
3.     resPivot++;
4.     if (resPivot < res.length - 1) {
5.         System.out.println(res[resPivot]);
6.     } else {
7.         System.out.println("There are no more T9 words. ");
8.     }
9.     continue;
10. }
11. ns = getNS(input);
12. num = getNum(input);
13. res = myT9Trie.search(ns);
14. resPivot = num;
15. if (res == null || num > res.length - 1) {
16.     System.out.println("There are no more T9 words. ");
17. } else {
18.     System.out.println(res[num]);
19. }
```

4.4 验证

以下是部分验证截图：



```
Enter "exit" to quit.
Enter Key Sequence (or "#" for next word):
pocks
Enter Key Sequence (or "#" for next word):
rocks
Enter Key Sequence (or "#" for next word):
soaks
Enter Key Sequence (or "#" for next word):
There are no more T9 words.
Enter Key Sequence (or "#" for next word):
rocks
Enter Key Sequence (or "#" for next word):
ace
Enter Key Sequence (or "#" for next word):
bad
Enter Key Sequence (or "#" for next word):
cad
Enter Key Sequence (or "#" for next word):
exit
进程已结束,退出代码0
```



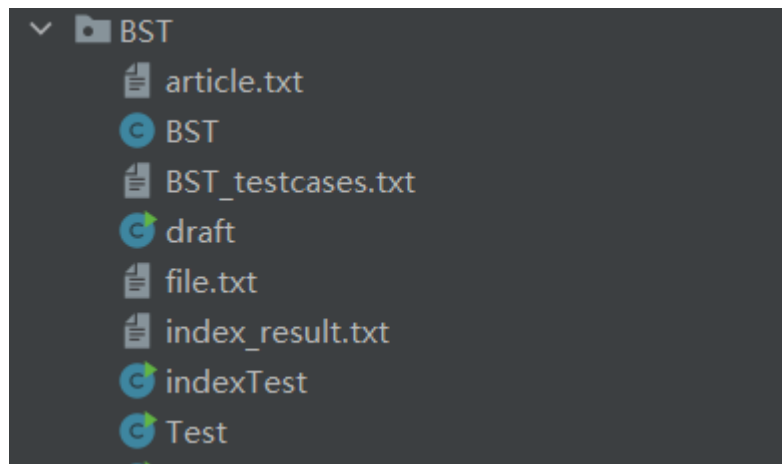
```
Enter "exit" to quit.
Enter Key Sequence (or "#" for next word):
mom
Enter Key Sequence (or "#" for next word):
mon
Enter Key Sequence (or "#" for next word):
moo
Enter Key Sequence (or "#" for next word):
moo
Enter Key Sequence (or "#" for next word):
exit
进程已结束,退出代码0
```

人工检验正确。

附录

任务 1and 任务 2

软件包如下:



BST:

```
1. package BST;
2.
3. import java.io.PrintWriter;
4.
5. public class BST<K extends Comparable<K>,V>{
6.     private class Node{
7.         private K key;
8.         private V value;
9.         private Node left;
10.        private Node right;
11.
12.        public Node(K key,V value,Node left,Node right){
13.            this.key = key;
14.            this.value = value;
15.            this.left = left;
16.            this.right = right;
17.        }
18.    }
19.    private Node root = null;
20.    private int numNode = 0;
21.    private int height = 0;
22.    public void insert(K key, V value){
23.        root = insertHelp(root,key,value);
```

```

24.     }
25.     public V remove(K key){
26.         V res = search(key);
27.         removeHelp(root,key);
28.         return res;
29.     }
30.     public V search(K key){
31.         Node pivot = searchHelp(root,key);
32.         if(pivot==null) return null;
33.         return pivot.value;
34.     }
35.     public boolean update(K key,V value){
36.         Node pivot = searchHelp(root,key);
37.         if(pivot==null) return false;
38.         pivot.value = value;
39.         return true;
40.     }
41.     public boolean isEmpty(){
42.         return root==null;
43.     }
44.     public void clear(){
45.         numNode = 0;
46.         height = 0;
47.         root = null;
48.     }
49.     public void showStructure(PrintWriter pw){
50.         height = getHeight(root,0);
51.         pw.println("-----");
52.         pw.printf("There are %d nodes in this BST.%nThe height of thi
s BST is %d.%n",numNode,height);
53.         pw.println("-----");
54.     }
55.     public void printInorder(PrintWriter pw){
56.         // 中序遍历
57.         printInorderHelp(pw,root);
58.         pw.close();
59.     }
60.     public Node insertHelp(Node root,K key,V value){
61.         if (root==null){
62.             numNode++;
63.             return new Node(key,value,null,null);
64.         }
65.         if (root.key.compareTo(key)>0) root.left = insertHelp(root.le
ft,key,value);

```



```

66.         else if (root.key.compareTo(key)<0) root.right= insertHelp(ro
           ot.right,key,value);
67.         else root.value = value;
68.         return root;
69.     }
70.     public Node searchHelp(Node root, K key){
71.         if (root==null) return null;
72.         Node cur = root;
73.         while(cur!=null){
74.             if (cur.key.compareTo(key)==0) return cur;
75.             else if (cur.key.compareTo(key)<0) cur = cur.right;
76.             else cur = cur.left;
77.         }
78.         return null;
79.     }
80.     public Node removeHelp(Node root,K key){
81.         if (root==null) return null;
82.         if (root.key.compareTo(key)<0) root.right = removeHelp(root.r
           ight,key);
83.         else if (root.key.compareTo(key)>0) root.left = removeHelp(ro
           ot.left,key);
84.         else{
85.             numNode--;
86.             if (root.left==null){
87.                 if (this.root == root) this.root = root.right;
88.                 return root.right;
89.             }
90.             else if (root.right==null){
91.                 if (this.root == root) this.root =root.left;
92.                 return root.left;
93.             }
94.             else {
95.                 Node rightMinNode = minNode(root.right);
96.                 root.key = rightMinNode.key;
97.                 root.value = rightMinNode.value;
98.                 root.right = delMinNode(root.right);
99.             }
100.        }
101.        return root;
102.    }
103.    public void printInorderHelp(PrintWriter pw,Node rt){
104.        if (rt == null) {
105.            return;
106.        } else {

```

```

107.         printInorderHelp(pw,rt.left);
108.         pw.printf("[ "+rt.key+" ---- < "+rt.value+" >]%n");
109.         printInorderHelp(pw,rt.right);
110.     }
111. }
112. public Node minNode(Node root){
113.     while(root.left != null) root = root.left;
114.     return root;
115. }
116. public Node delMinNode(Node root){
117.     if (root.left==null) return root.right;
118.     else {
119.         root.left = delMinNode(root.left);
120.         return root;
121.     }
122. }
123. public int getHeight(Node root,int height){
124.     if (root.left==null&&root.right==null)
125.         return height + 1;
126.     else if (root.left!=null&&root.right!=null)
127.         return Math.max(getHeight(root.right,height+1),getHeig
ht(root.left,height+1));
128.     else{
129.         if (root.left!=null) return height+getHeight(root.left
,1);
130.         else return height+getHeight(root.right,1);
131.     }
132. }
133. }

```

indexTest:

```

1. package BST;
2.
3. import javax.swing.*;
4. import java.io.*;
5. import java.util.Scanner;
6.
7. public class indexTest {
8.     public static void main(String[] args) throws FileNotFoundExcepti
on {
9.         File file = new File("src/BST/article.txt");
10.        PrintWriter pw = new PrintWriter("src/BST/index_result.txt");
11.
12.        BST<String, String> myBST = new BST<String, String>();

```

```

12.         Scanner sc = new Scanner(file);
13.         int row = -1;
14.         StringBuilder str = new StringBuilder();
15.         String sum = txt2String(file);
16.         for (int i = 0; i < sum.length(); i++) {
17.             str = new StringBuilder();
18.             while (i < sum.length() && sum.charAt(i) != '\n') {
19.                 str.append(sum.charAt(i));
20.                 i++;
21.             }
22.             row++;
23.             str = new StringBuilder(str.toString().strip());
24.             System.out.println(row+ " " + str);
25.             String[] dic = str.toString().split(" ");
26.             for (String s:dic) {
27.                 s = clean(s);
28.                 if (s.equals("")) break;
29.                 String preKey = myBST.search(s);
30.                 if (preKey == null) {
31.                     myBST.insert(s, row + "");
32.                 } else {
33.                     myBST.update(s, preKey+ " "+row);
34.                 }
35.             }
36.         }
37.         myBST.printInorder(pw);
38.     }
39.
40.     public static String txt2String(File file) {
41.         StringBuilder result = new StringBuilder();
42.         try {
43.             BufferedReader br = new BufferedReader(new FileReader(file));
44.             String s = null;
45.             while((s = br.readLine()) != null) { //使用 readLine 方法,
                一次读一行
46.                 result.append(System.lineSeparator() + s);
47.             }
48.             br.close();
49.         } catch (Exception e) {
50.             e.printStackTrace();
51.         }
52.         return result.toString();
53.     }

```

```

54.     public static String clean(String s){
55.         String res = "";
56.         s = s.toLowerCase();
57.         int pivot = 0;
58.         char tem;
59.         for (int i = 0; i < s.length(); i++) {
60.             tem = s.charAt(i);
61.             if(tem<='z'&&tem>='a'){
62.                 pivot=i;
63.                 break;
64.             }
65.         }
66.         for (int i = pivot; i < s.length(); i++) {
67.             tem = s.charAt(i);
68.             if(tem<='z'&&tem>='a') res+=tem;
69.             else return res;
70.         }
71.         return res;
72.     }
73. }

```

Test:

```

1. package BST;
2.
3. import java.io.File;
4. import java.io.FileNotFoundException;
5. import java.io.PrintWriter;
6. import java.util.Scanner;
7.
8. public class Test {
9.     public static void main(String[] args) throws FileNotFoundException {
10.         PrintWriter pw = new PrintWriter("src/BST/file.txt");
11.         BST<String, String> myBST = new BST<String, String>();
12.         Scanner sc = new Scanner(new File("src/BST/BST_testcases.txt"));
13.         while (sc.hasNext()){
14.             String str = sc.nextLine().strip();
15.             char syn = str.charAt(0);
16.             String key = "";
17.             String value = "";
18.             switch (syn){
19.                 case '+':{
20.                     int pivot = 2;

```

```

21.         while(str.charAt(pivot)!='.') pivot++;
22.         key = value = "";
23.         for(int i=3;i<pivot-1;i++) key+=str.charAt(i);
24.         for(int i=pivot+3;i<str.length()-
25.         3;i++) value+=str.charAt(i);
26.         myBST.insert(key,value);
27.         break;
28.     }
29.     case '#':{
30.         myBST.showStructure(pw);
31.         break;
32.     }
33.     case '-':{
34.         key = "";
35.         for(int i=3;i<str.length()-
36.         2;i++) key+=str.charAt(i);
37.         value = myBST.remove(key);
38.         if(value!=null) pw.println("remove success ---
39.         "+key+" "+value);
40.         else pw.println("remove unsuccess ---"+key);
41.         break;
42.     }
43.     case '=':{
44.         int pivot = 2;
45.         while(str.charAt(pivot)!='.') pivot++;
46.         key = value = "";
47.         for(int i=3;i<pivot-1;i++) key+=str.charAt(i);
48.         for(int i=pivot+3;i<str.length()-
49.         3;i++) value+=str.charAt(i);
50.         boolean flag = myBST.update(key,value);
51.         if (flag) pw.println("update success ---
52.         "+key+" "+value);
53.         break;
54.     }
55.     case '?':{
56.         key = "";
57.         for(int i=2;i<str.length()-
58.         2;i++) key+=str.charAt(i);
59.         key = key.strip();
60.         value = myBST.search(key);
61.         if(value==null) pw.println("search unsuccess ---
62.         "+key);
63.         else pw.println("search success ---
64.         "+key+" "+value);

```

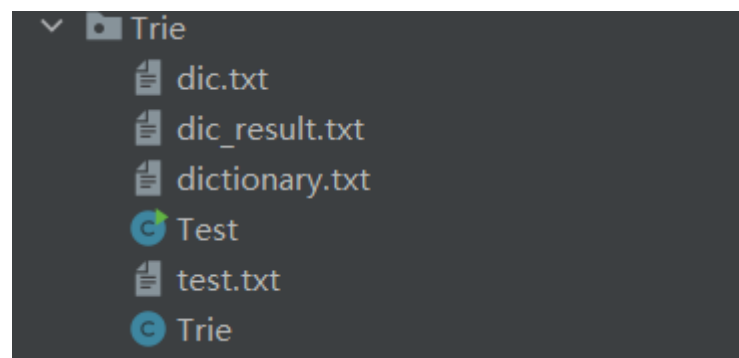
```

57.             break;
58.         }
59.         default:{
60.             break;
61.         }
62.     }
63. }
64. pw.close();
65. }
66. }

```

任务 3

软件包如下：



Trie:

```

1. package Trie;
2.
3. public class Trie {
4.     public static final int RED = 1;
5.     public static final int BLUE = 0;
6.     public Node root = new Node();
7.     private class Node{
8.         private int color = BLUE;
9.         public String res = "";
10.        public Node[] subNode = new Node[26];
11.        public Node(){
12.            public Node(Node rt,char s){
13.                this.res = rt.res+s;
14.            }
15.        public void setBlue(){
16.            color = BLUE;
17.        }

```

```
18.     public void setRed(){
19.         color = RED;
20.     }
21. }
22. public void insert(String s){
23.     insert(root,s);
24. }
25. public void insert(Node rt, String s) {
26.     for (int i = 0; i < s.length(); i++) {
27.         char c = s.charAt(i);
28.         int pivot = c - 'a';
29.         if (rt.subNode[pivot] == null) {
30.             rt.subNode[pivot] = new Node(rt, c);
31.         }
32.         rt = rt.subNode[pivot];
33.         if (i == s.length() - 1) {
34.             rt.setRed();
35.         }
36.     }
37. }
38. public boolean search(String s){
39.     Node res = search(root, s);
40.     return res != null;
41. }
42. public Node search(Node rt,String s){
43.     for (int i = 0; i < s.length(); i++) {
44.         char c = s.charAt(i);
45.         int pivot = c - 'a';
46.         if (rt.subNode[pivot] == null) {
47.             return null;
48.         } else {
49.             rt = rt.subNode[pivot];
50.         }
51.     }
52.     return rt;
53. }
54. public void delete(String s) {
55.     delete(root,s);
56. }
57. public void delete(Node rt, String s) {
58.     Node res = search(rt, s);
59.     if (res == null) {
60.         return;
61.     } else {
```

```

62.         res.setBlue();
63.     }
64. }
65. public void showStr(String s){
66.     showStr(root,s);
67. }
68. public void showStr(Node rt,String s) {
69.     Node cur = search(rt,s);
70.     if (cur == null) {
71.         return;
72.     } else {
73.         showStr(cur);
74.     }
75. }
76. public void showStr(Node rt){
77.     for (int i = 0; i < 26; i++) {
78.         Node subNode = rt.subNode[i];
79.         if (subNode != null) {
80.             if (subNode.color == RED) {
81.                 System.out.print(subNode.res+" ");
82.             }
83.             showStr(subNode);
84.         }
85.     }
86. }
87. }

```

Test:

```

1. package Trie;
2.
3. import java.io.*;
4.
5. public class Test {
6.     public static void main(String[] args) throws FileNotFoundException {
7.         Trie myTrie = new Trie();
8.         File file = new File("src/Trie/dictionary.txt");
9.         File file2 = new File("src/Trie/dic.txt");
10.        PrintWriter pw = new PrintWriter("src/Trie/dic_result.txt");
11.
12.        String sum = txt2String(file);
13.        String sum2 = txt2String(file2);
14.        StringBuilder str;
15.        for (int i = 0; i < sum.length(); i++) {

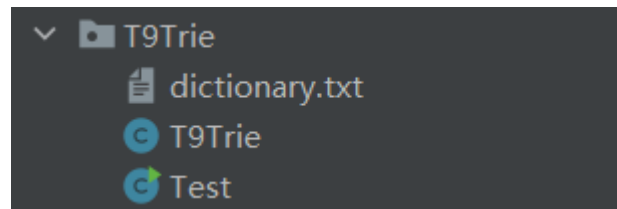
```



```
15.         str = new StringBuilder();
16.         while (i < sum.length() && sum.charAt(i) != '\n') {
17.             str.append(sum.charAt(i));
18.             i++;
19.         }
20.         String strip = str.toString().strip();
21.         myTrie.insert(strip);
22.     }
23.     System.out.println("开始删除");
24.     long start = System.currentTimeMillis();
25.     for (int i = 0; i < sum2.length(); i++) {
26.         str = new StringBuilder();
27.         while (i < sum2.length() && sum2.charAt(i) != '\n') {
28.             str.append(sum2.charAt(i));
29.             i++;
30.         }
31.         String strip = str.toString().strip();
32.         myTrie.delete(strip);
33.     }
34.     long end = System.currentTimeMillis();
35.     System.out.println("用时:"+(end-start)+"ms");
36.     System.out.println("删除完毕");
37.     myTrie.showStr("");
38. }
39. public static String txt2String(File file) {
40.     StringBuilder result = new StringBuilder();
41.     try {
42.         BufferedReader br = new BufferedReader(new FileReader(file)); //构造一个 BufferedReader 类来读取文件
43.         String s = null;
44.         while((s = br.readLine()) != null) { //使用 readLine 方法,
            一次读一行
45.             result.append(System.lineSeparator() + s);
46.         }
47.         br.close();
48.     } catch (Exception e) {
49.         e.printStackTrace();
50.     }
51.     return result.toString();
52. }
53. }
```

任务 4

软件包如下:



T9Trie:

```
1. package T9Trie;
2.
3. public class T9Trie {
4.     public Node root = new Node();
5.     private class Node{
6.         private String[] data = null;
7.         private int dataPivot = -1;
8.         private Node[] subNode = new Node[8];
9.         public Node() {
10.
11.         }
12.         public void add(String s){
13.             if (data == null)
14.                 data = new String[1];
15.             else if (dataPivot == data.length - 1)
16.                 resize();
17.             data[++dataPivot] = s;
18.         }
19.         private void resize(){
20.             String[] tem = new String[data.length*2];
21.             int temPivot = 0;
22.             for (String s:data) {
23.                 tem[temPivot++] = s;
24.             }
25.             data = tem;
26.         }
27.     }
28.     public void insert(String ns,String ss){
29.         insert(root,ns,ss);
30.     }
31.     public void insert(Node rt,String ns,String ss){
32.         for (int i = 0; i < ns.length(); i++) {
33.             char c = ns.charAt(i);
34.             int pivot = c - '2';
```

```

35.         if (rt.subNode[pivot] == null) {
36.             rt.subNode[pivot] = new Node();
37.         }
38.         rt = rt.subNode[pivot];
39.         if (i == ns.length() - 1) {
40.             rt.add(ss);
41.         }
42.     }
43. }
44. public String[] search(String ns){
45.     return search(root,ns);
46. }
47. public String[] search(Node rt,String ns){
48.     for (int i = 0; i < ns.length(); i++) {
49.         char c = ns.charAt(i);
50.         int pivot = c - '2';
51.         if (rt.subNode[pivot] == null) {
52.             return null;
53.         } else {
54.             rt = rt.subNode[pivot];
55.         }
56.     }
57.     return rt.data;
58. }
59. }

```

Test:

```

1. package T9Trie;
2.
3. import java.io.BufferedReader;
4. import java.io.File;
5. import java.io.FileReader;
6. import java.util.Scanner;
7.
8. public class Test {
9.     public static void main(String[] args) {
10.         T9Trie myT9Trie = new T9Trie();
11.         File file = new File("src/T9Trie/dictionary.txt");
12.         String sum = txt2String(file);
13.         StringBuilder str;
14.         for (int i = 0; i < sum.length(); i++) {
15.             str = new StringBuilder();
16.             while (i < sum.length() && sum.charAt(i) != '\n') {
17.                 str.append(sum.charAt(i));

```

```

18.         i++;
19.     }
20.     String strip = str.toString().strip();
21.     String ns = changeNS(strip);
22.     myT9Trie.insert(ns,strip);
23. }
24. System.out.println("Enter “exit” to quit. ");
25. String ns;
26. int num;
27. String[] res = null;
28. int resPivot = 0;
29. while (true) {
30.     System.out.println("Enter Key Sequence (or “#” for next w
ord):");
31.     Scanner sc = new Scanner(System.in);
32.     String input = sc.next().strip();
33.     if (input.equals("exit")) break;
34.     else if (input.equals("#")){
35.         resPivot++;
36.         if (resPivot < res.length - 1) {
37.             System.out.println(res[resPivot]);
38.         } else {
39.             System.out.println("There are no more T9 words. "
);
40.         }
41.         continue;
42.     }
43.     ns = getNS(input);
44.     num = getNum(input);
45.     res = myT9Trie.search(ns);
46.     resPivot = num;
47.     if (res == null || num > res.length - 1) {
48.         System.out.println("There are no more T9 words. ");
49.     } else {
50.         System.out.println(res[num]);
51.     }
52. }
53.
54. }
55. public static String txt2String(File file) {
56.     StringBuilder result = new StringBuilder();
57.     try {
58.         BufferedReader br = new BufferedReader(new FileReader(fil
e)); //构造一个 BufferedReader 类来读取文件

```

```
59.         String s;
60.         while((s = br.readLine()) != null) { //使用 readLine 方法,
            一次读一行
61.             result.append(System.lineSeparator() + s);
62.         }
63.         br.close();
64.     } catch(Exception e) {
65.         e.printStackTrace();
66.     }
67.     return result.toString();
68. }
69. public static String changeNS(String s){
70.     StringBuilder res = new StringBuilder();
71.     for (int i = 0; i < s.length(); i++) {
72.         char tem = s.charAt(i);
73.         if (tem<='c') {
74.             res.append(2);
75.         } else if (tem<='f') {
76.             res.append(3);
77.         } else if (tem<='i') {
78.             res.append(4);
79.         }else if (tem<='l') {
80.             res.append(5);
81.         }else if (tem<='o') {
82.             res.append(6);
83.         }else if (tem<='s') {
84.             res.append(7);
85.         }else if (tem<='v') {
86.             res.append(8);
87.         }else{
88.             res.append(9);
89.         }
90.     }
91.     return res.toString().strip();
92. }
93.
94. public static String getNS(String s) {
95.     StringBuilder res = new StringBuilder();
96.     for (int i = 0; i < s.length(); i++) {
97.         char tem = s.charAt(i);
98.         if (tem<='9'&&tem>='2') {
99.             res.append(tem);
100.        }
101.    }
```

```
102.         return res.toString().strip();
103.     }
104.     public static int getNum(String s) {
105.         int res = 0;
106.         for (int i = 0; i < s.length(); i++) {
107.             char tem = s.charAt(i);
108.             if (tem=='#') {
109.                 res++;
110.             }
111.         }
112.         return res;
113.     }
114.
115. }
```