

## 题目描述

### 题目描述

实现一个可存储若干个单词的字典。用户可以：

- 在字典中加入单词。
- 查找指定单词在字典中的兄弟单词个数。
- 查找指定单词的指定序号的兄弟单词，指定序号指字典中兄弟单词按字典顺序（参见Page 3）排序后的序号（从1开始）
- 清空字典中所有单词。

### 定义，格式说明

#### 单词

由小写英文字母组成，不含其它字符。

#### 兄弟单词

给定一个单词X，如果通过任意交换单词中字母的位置得到不同的单词Y，那么定义Y是X的兄弟单词。

#### 字典顺序

两个单词(字母按照自左向右顺序)，先以第一个字母作为排序的基准，如果第一个字母相同，就用第二个字母为基准，如果第二个字母相同就以第三个字母为基准。依此类推，如果到某个字母不相同，字母顺序在前的那个单词顺序在前。如果短单词是长单词从首字母开始连续的一部分，短单词顺序在前。

举例：bca是abc的兄弟单词；abc与abc是相同单词，不是兄弟单词

## 规格

- $0 \leq \text{字典中所含单词个数} \leq 1000$
- $1 \leq \text{单词所含字母数} \leq 50$

测试用例保证，接口中输入不会超出如上约束。

### 输入描述:

先输入字典中单词的个数，再输入n个单词作为字典单词。  
输入一个单词，查找其在字典中兄弟单词的个数  
再输入数字n

### 输出描述:

根据输入，输出查找到的兄弟单词的个数

示例1

输入

3   abc bca cab abc 1

输出

2   bca

### 解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.TreeSet;

public class Main {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String input = null;
        while((input = br.readLine()) != null){
            String[] s = input.split(" ");
            int n = Integer.valueOf(s[0]);
            int k = Integer.valueOf(s[s.length-1]);
            String str = s[s.length-2];
            int m = 0;
            String[] ss = new String[n];
```

```

        if( n>=0&&n<=1000){
            for(int i = 1; i < s.length-1; i++){
                if(s[i].length()>=1&&s[i].length()<=50){
                    if(isBrother(str, s[i])){
                        ss[m] = s[i];
                        m++;
                        //System.out.println(str+"===="+s[i]+"===="+m+"===="+i);

                    }

                }else{
                    System.out.println("单词长度越界");
                    System.exit(0);
                }
            }
        }else{
            System.out.println("n越界");
            System.exit(0);
        }

        System.out.println(m);
        //System.out.println("k="+k+"---m="+m);
        if(k<=m){
            sort(ss, k);
        }
    }
}

private static void sort(String[] ss, int k) {
    // TODO Auto-generated method stub
    int min;
    for(int i = 0; i < ss.length; i++){
        min = i;
        for(int j = i+1; j < ss.length; j++){
            if(ss[i]!=null && ss[j]!=null){
                if(ss[j].compareTo(ss[min])<0){
                    min = j;
                }
            }
        }

        String temp = ss[min];
        ss[min] = ss[i];
        ss[i] = temp;
    }

    System.out.println(ss[k-1]);

}

private static boolean isBrother(String str, String string) {
    // TODO Auto-generated method stub
    int[] s1 = new int[26];
    for(int i = 0; i < str.length(); i++){
        s1[(int)str.charAt(i) - 97]++;
    }
}

```

```

int[] s2 = new int[26];
for(int i = 0; i < string.length(); i++){
    s2[(int)string.charAt(i) - 97]++;
}

if(str.length() == string.length() && !str.equals(string)){
    for(int i = 0; i < 26; i++){
        if(s1[i] == s2[i]){

        }else{
            return false;
        }
    }
}else{
    return false;
}
return true;
}
}

```

## 总结

- 兄弟单词的处理，那么就比较每个元素的个数
- 字符串比较实用str.compareTo(String)<0