文本索引（Text Indexing）

1 问题描述

编写一个构建大块文本索引的程序，然后进行快速搜索，来查找某个字符串在该文本中的出现位置。

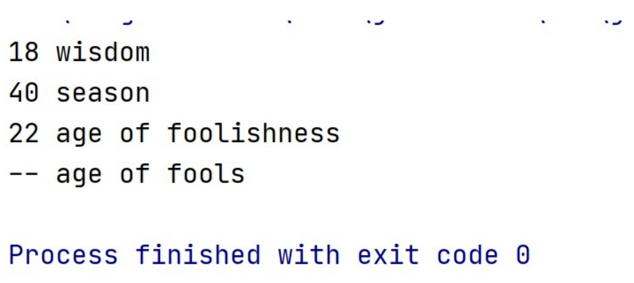
2 解决思路

首先得到文本的后缀数组，然后将后缀数组排序，最后使用二分查找来搜索。

3 程序实现

```
public class SuffixArrayTest {
    public static void main(String[] args) {
        In in = new In("D:\\Java_test\\test1\\exp4\\input.txt");
        String s = in.readAll().replaceAll("\n", " ").trim();
        Suff[] sa = getSa(s);
        In words = new In("D:\\Java_test\\test1\\exp4\\words.txt");
        while (words.hasNextLine()) {
            match(sa, s, words.readLine());
        }
    }

    static void match(Suff[] _sa, String _s, String _p) {
        String s = _s;
        String p = _p;
        Suff[] sa = _sa; // 后缀数组
        int l = 0;
        int r = s.length() - 1;
        boolean flag = false;
        // 二分查找
        while (l < r) {
            int mid = l + ((r - l) >> 1);
            Suff midSuff = sa[mid];
            String suffStr = midSuff.str;
            int compareRes;
            // 将后缀和模式串比较
            if (suffStr.length() >= p.length()) {
                compareRes = suffStr.substring(0, p.length()).compareTo(p);
            } else {
                compareRes = suffStr.compareTo(p);
            }
            if (compareRes == 0) {
                System.out.println(getWordPos(s, midSuff.index) + 1 + " " + p);
                flag = true;
                break;
            } else if (compareRes < 0) {
                l = mid + 1;
            } else {
                r = mid;
            }
        }
        if (flag == false) System.out.println("--" + " " + p);
    }

    // 根据字符位置找到单词位置
    public static int getWordPos(String s, int idx) {
        int cnt = 0;
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == ' ') {
                cnt++;
            }
            if (idx == i) break;
        }
        return cnt;
    }

    // 对所有后缀排序
    public static Suff[] getSa(String src) {
        int strLength = src.length();
        // sa 即SuffixArray,后缀数组
        Suff[] suffixArray = new Suff[strLength];
        for (int i = 0; i < strLength; i++) {
            String suffI = src.substring(i);   //截取后缀
            suffixArray[i] = new Suff(suffI, i);
        }
        Arrays.sort(suffixArray);   //对后缀数组进行排序
        return suffixArray;
    }

    static class Suff implements Comparable<Suff> {
        String str;  //后缀文本
        int index;   //后缀的索引

        public Suff(String str, int index) {
            super();
            this.str = str;
            this.index = index;
        }

        @Override
        public int compareTo(Suff o2) {
            return this.str.compareTo(o2.str);
        }

        @Override
        public String toString() {
            return "Suff{" + "str='" + str + "\'" + ",index=" + index + "}";
        }
    }
}
```

4 实验结果



```
18 wisdom
40 season
22 age of foolishness
-- age of fools

Process finished with exit code 0
```

5 心得体会

通过本次实验，我了解了后缀数组在字符串操作方面的应用，学习到了使用后缀数组匹配文本的方法。