实验四 文本索引（Text Indexing）

一、 实验目的

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。

二、 实验内容

例如，有两个文本文件，分别为 corpus.txt 和 pattern.txt。corpus.txt 文件中存有大量文本，pattern.txt 存有一些字符串，以换行分隔。我们的任务是找到 patterns.txt 中的每个字符串在 corpus.txt 文本中首次出现的位置并输出。

例如 corpus.txt 文本如下：

```
it was the best of times it was the worst of times it was the age of wisdom it was the
age of foolishness it was the epoch of belief it was the epoch of incredulity it was the
season of light it was the season of darkness it was the spring of hope it was the
winter of despair
```

patterns.txt 文本如下：

```
wisdom
season
age of foolishness
age of fools
```

最终查询输出如下：

```
18 wisdom
40 season
22  ge of foolishness
23  age of fools
```

Wisdom 出现在位置 18，season 出现在位置 40，age of foolishness 出现在位置 22，age of fools 未出现。

三、 实验方法

有很多种算法可以用来解决这个问题，这些方法都有着不同的时间复杂度和空间复杂度。

首先我们可以考虑**暴力搜索**（Brute-Force）算法。这种方法不需要建立指针，只用在 corpus 语料库中搜索 patterns 中的每一个字符串即可。如果 corpus 语料库很小或者待查询的字符串不多，这种方法是很好的。一旦 corpus 很大或待查询的字符串很多，这种方法就十分低效。

一种快速搜索的方法是在语料库（每个字符位置一个指针）上进行指针排序，然后使用**折半搜索**（binary search）。这种方法需要构建指针索引，按照排序顺序访问关键字，并找出调用 bsearch 的必要接口使用索引执行查询。

本次实验，我使用**博耶-穆尔字符串（BoyerMoore）搜索算法**进行查询。在用于查找子字符串的算法当中，Boyer-Moore 算法被认为最高效的字符串搜索算法，教材（算法 第四版）的 5.3.4 节对该算法进行了详细的讲解并提供了源代码，我的实验以此为基础进行。

Boyer-Moore 算法实现如下：

```java
class BoyerMoore {
    private final int R;
    private int[] right;
    private String pat;
    public BoyerMoore(String pat) {
        this.R = 256;
        this.pat = pat;
        right = new int[R];
        for (int i = 0; i < R; i++) {
            right[i] = -1;
        }
        for (int j = 0; j < pat.length(); j++)
            right[pat.charAt(j)] = j;
    }
    public int search(String corpus) {
        int m = pat.length();
        int n = corpus.length();
        int skip;
        for (int i = 0; i < n - m; i += skip) {
            skip = 0;
            for (int j = m - 1; j >= 0; j--) {
                if (pat.charAt(j) != corpus.charAt(i + j)) {
                    skip = Math.max(1, j - right[corpus.charAt(i + j)]);
                }
            }
            if (skip == 0) return i;//找到
        }
        return -1;//没找到
    }
    public int disPos(String corpus, int n) {
        int pos = 1;
        char[] toCh = corpus.toCharArray();
        for (int i = 0; i < n; i++) {
            if (toCh[i] == ' ') pos++;
        }
        return pos;
    }
}
```

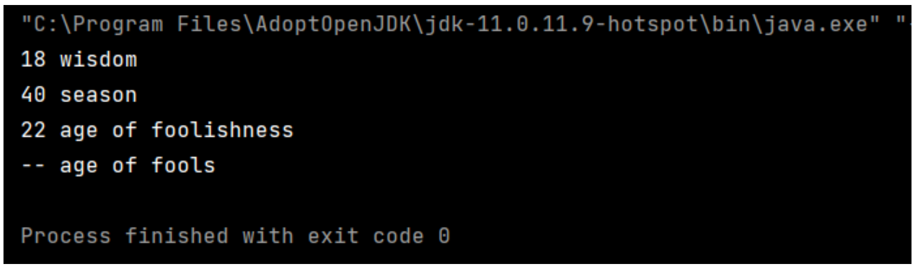
四、　源代码

源代码如下：

```java
import edu.princeton.cs.algs4.StdOut;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
class BoyerMoore {
    private final int R;
    private int[] right;
    private String pat;
    public BoyerMoore(String pat) {
        this.R = 256;
        this.pat = pat;
        right = new int[R];
        for (int i = 0; i < R; i++) {
            right[i] = -1;
        }
        for (int j = 0; j < pat.length(); j++) {
            right[pat.charAt(j)] = j;
        }
    }
    public int search(String corpus) {
        int m = pat.length();
        int n = corpus.length();
        int skip;
        for (int i = 0; i < n - m; i += skip) {
            skip = 0;
            for (int j = m - 1; j >= 0; j--) {
                if (pat.charAt(j) != corpus.charAt(i + j)) {
                    skip = Math.max(1, j - right[corpus.charAt(i + j)]);
                }
            }
            if (skip == 0) return i;//找到
        }
        return -1;//没找到
    }

    public int disPos(String corpus, int n) {
        int pos = 1;
        char[] toCh = corpus.toCharArray();
        for (int i = 0; i < n; i++) {
            if (toCh[i] == ' ') pos++;
        }
        return pos;
    }
}

public class text_indexing {
    public static void main(String[] args) {
        String corpus1 = readCorpus();
        File file = new File("D:\\Mdesktop\\pattern.txt");
        FileInputStream in = null;
        try {
            if (file.length() == 0)
                StdOut.println("文件为空！");
            else {
                in = new FileInputStream(file);
                InputStreamReader streamReader = new InputStreamReader(in);
                BufferedReader reader = new BufferedReader(streamReader);
                String line;
                while ((line = reader.readLine()) != null) {
                    BoyerMoore boyermoore = new BoyerMoore(line);
                    int pos = boyermoore.search(corpus1);
                    if (pos == -1) StdOut.println("-- " + line);
                    else {
                        int dispos = boyermoore.disPos(corpus1, pos);
                        StdOut.println(dispos + " " + line);
                    }
                }
                reader.close();
                in.close();
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static String readCorpus() {
        File file = new File("D:\\Mdesktop\\corpus.txt");
        FileInputStream in = null;
        StringBuilder stringBuilder = null;
        try {
            if (file.length() == 0)
                StdOut.println("文件为空！");
            else {
                in = new FileInputStream(file);
                InputStreamReader streamReader = new InputStreamReader(in);
                BufferedReader reader = new BufferedReader(streamReader);
                String line;
                stringBuilder = new StringBuilder();
                while ((line = reader.readLine()) != null) {
                    stringBuilder.append(line);
                    stringBuilder.append(' ');
                }
                reader.close();
                in.close();
            }
        }
        catch (Exception e) {
              e.printStackTrace();
          }
          return String.valueOf(stringBuilder);
       }
   }
```

五、　实验结果

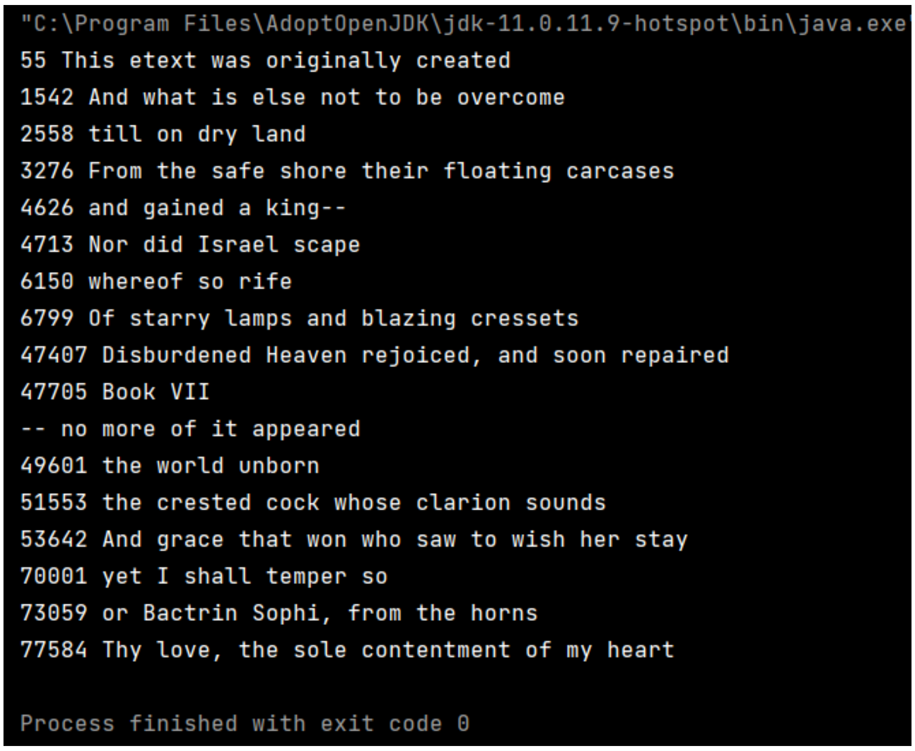
使用实验内容中的样例，运行结果如下：



```
"C:\Program Files\AdoptOpenJDK\jdk-11.0.11.9-hotspot\bin\java.exe" "
18 wisdom
40 season
22 age of foolishness
-- age of fools

Process finished with exit code 0
```

查询成功。

我们不妨使用较大的语料库（从 https://corpus.canterbury.ac.nz/descriptions/中下载）查询较多的字符串，结果如下：



```
"C:\Program Files\AdoptOpenJDK\jdk-11.0.11.9-hotspot\bin\java.exe
55 This etext was originally created
1542 And what is else not to be overcome
2558 till on dry land
3276 From the safe shore their floating carcases
4626 and gained a king--
4713 Nor did Israel scape
6150 whereof so rife
6799 Of starry lamps and blazing cressets
47407 Disburdened Heaven rejoiced, and soon repaired
47705 Book VII
-- no more of it appeared
49601 the world unborn
51553 the crested cock whose clarion sounds
53642 And grace that won who saw to wish her stay
70001 yet I shall temper so
73059 or Bactrin Sophi, from the horns
77584 Thy love, the sole contentment of my heart

Process finished with exit code 0
```

查询成功。

六、 实验体会

本次实验较为灵活，可采用的算法较多。除了本次采用的 Boyer-Moore 算法，诸如 KMP 算法、Rabin-Karp 算法、Sunday 算法等都是很高效的解决方法。

在实验过程中，读文件时要加入异常报错机制，否则无法通过编译。其他的部分只要算法不出错就不太会有问题。