مشروع بنى معطيات

بحث عن هاش داخل ملف نصي

اشراف :

عبد المهيمن الاسماعيل

اعداد الطلاب :

احمد عمر العباس

احمد خليل عباس

احمد محمد ياسر بدوي

الدالة main هي نقطة البداية للبرنامج.

filelocation يحتوي على مسار الملف النصي الذي يحتوي على النصوص المراد البحث فيها.

searchitem يحتوي على النص الذي نبحث عنه.

try تستخدم لمحاولة تنفيذ جزء من الكود يمكن أن يحدث فيه خطأ، وفي حال حدوث خطأ يتم التقاطه بواسطة catch.

دالة readLines

الدالة readLines تقرأ النصوص من الملف وتخزنها في قائمة.

تستخدم BufferedReader لقراءة الملف سطرًا بسطر.

تضيف كل سطر من الملف إلى قائمة liness.

تعيد القائمة التي تحتوي على جميع النصوص.

دالة searchAndPrintResults

الدالة searchAndPrintResults تنفذ عمليات البحث وتطبع النتائج.

linearSearch يتم استدعاء دالة البحث الخطي للحصول على عدد الخطوات اللازمة.

إذا كانت خطوات البحث الخطي أكثر من ١٥، يتم استدعاء binarySearch (البحث الثنائي).

إذا كانت خطوات البحث الثنائي أكثر من ١٠، يتم استدعاء ternarySearch (البحث الثلاثي).

إذا وجدت الخوارزمية النص، يتم طباعة رسالة توضح عدد الخطوات والخوارزمية المستخدمة.

دالة linearSearch

الدالة linearSearch تنفذ البحث الخطي وتعيد عدد الخطوات.

تبدأ من أول النصوص وتبحث واحدًا تلو الآخر.

تزيد عدد الخطوات في كل مرة حتى تجد النص أو تنتهي النصوص.

دالة binarySearch

الدالة binarySearch تنفذ البحث الثنائي وتعيد عدد الخطوات

يبدأ البحث من المنتصف ويقسم النصوص إلى نصفين.

يقارن النص الهدف مع النص في المنتصف، إذا كان النص الهدف أصغر، يتم البحث في النصف الأيسر، وإذا كان أكبر، يتم البحث في النصف الأيمن.

دالة ternarySearch

الدالة ternarySearch تنفذ البحث الثلاثي وتعيد عدد الخطوات.

يقسم النصوص إلى ثلاثة أجزاء ويقارن النص الهدف مع النقاط الموجودة في الثلث الأول والثلث الثاني.

يحدد الجزء الذي يحتوي على النص الهدف ويستمر في البحث داخل ذلك الجزء.

الغرض من الكود

الكود مصمم للبحث عن نصوص (هاشات) داخل ملف نصي باستخدام ثلاث طرق مختلفة للبحث: البحث الخطي، البحث الثنائي، والبحث الثلاثي. الهدف هو العثور على النص المحدد بأكثر الطرق فعالية.

أجزاء الكود الأساسية

١. **استيراد المكتبات اللازمة:**

- BufferedReaderو FileReaderلقراءة الملف النصي.
- ArrayListو Arraysو Listللعمل مع القوائم والمصفوفات.

٢. **الدالة الرئيسية:(main)**

- تحدد موقع الملف والنص المراد البحث عنه.
- تقرأ النصوص من الملف وتخزنها في قائمة.
- تحول القائمة إلى مصفوفة وتقوم بترتيبها.
- تستخدم دالة searchAndPrintResultsللبحث عن النص.

٣. **دالة قراءة النصوص من الملف:(readLines)**

- تقرأ كل سطر من الملف النصي وتضيفه إلى قائمة.

٤. **دالة البحث وطباعة النتائج:(searchAndPrintResults)**

- تبحث عن النص باستخدام البحث الخطي، وإذا لم يكن البحث الخطي فعالاً تنتقل إلى البحث الثنائي، وإذا لم يكن البحث الثنائي فعالاً تنتقل إلى البحث الثلاثي.
- تطبع النتيجة مع عدد الخطوات التي استغرقتها العملية.

٥. **دوال البحث:**

- **البحث الخطي:(linearSearch)**
  - يبحث عن النص بفحص كل عنصر في المصفوفة واحدة تلو الأخرى.
- **البحث الثنائي:(binarySearch)**
  - يبحث عن النص بتقسيم المصفوفة إلى نصفين بشكل متكرر حتى يتم العثور على النص أو يتم استبعاد كل العناصر.
- **البحث الثلاثي:(ternarySearch)**
  - مشابه للبحث الثنائي ولكنه يقسم المصفوفة إلى ثلاثة أجزاء بدلاً من جزأين.

تحليل الكود

- **الكفاءة:**
  - البحث الخطي هو أبسط شكل من أشكال البحث ويعتبر غير فعال للمجموعات الكبيرة من البيانات (زمن البحث.((O(n)O(n)O(n) :
  - البحث الثنائي أكثر كفاءة (زمن البحث ((O(logn)O(\log n)O(logn :لكنه يتطلب أن تكون المصفوفة مرتبة مسبقاً.
  - البحث الثلاثي أيضاً يتطلب ترتيب المصفوفة ويمكن أن يكون أكثر كفاءة من البحث الثنائي في بعض الحالات (زمن البحث.((O(log3n)O(\log_3 n)O(log3n) :
- **التحقق من الأخطاء:**
  - الكود يعالج الأخطاء المحتملة عند قراءة الملف باستخدام جملة .try-catch
- **تحسينات ممكنة:**

- يمكن استخدام بنية بيانات أكثر كفاءة لتخزين النصوص، مثل هاش تيبل (Hash Table) لتحسين سرعة البحث إلى O(1)O(1)O(1)في المتوسط.
- يمكن تحسين دوال البحث للطباعة بشكل أوضح وأكثر تفصيلاً للنتائج.

استنتاج

الكود الحالي يعمل بشكل جيد للبحث عن النصوص باستخدام ثلاث طرق مختلفة، لكنه يعتمد بشكل كبير على ترتيب المصفوفة ويستخدم الطرق التقليدية للبحث. لتحسين الأداء، يمكن استبدال القوائم والمصفوفات بـ HashMapلتسريع عمليات البحث.

هذا الكود:

```
;package datastructures


;import java.io.BufferedReader
;import java.io.FileReader
;import java.io.IOException
;import java.util.ArrayList
;import java.util.Arrays
;import java.util.List


} public class DataStructures


    } public static void main(String[] args)

String filelocation = ;""C:\\Users\\ALWAFER\\Desktop\\vvv.txt
String searchitem = "e10ed94a1ce505fe0de3d4a628f96f70945b3e78f87c7f8;"e3bf55a10240a9900
```

```
                                                    } try

    ;List<String> textsList = readLines(filelocation)
String[] sortedTexts = textsList.toArray(new
                                                    ;String[0])
                                    ;Arrays.sort(sortedTexts)
                                    // ترمي خطاء عند ادخال موقع الملف خطاء
;searchAndPrintResults(sortedTexts, searchitem)


                                    } (catch (IOException e {
System.out.println("Error reading file: " +
                                                    ;e.getMessage())
                                                        {


                                                            {

                                    //دالة قراءة الهاش من الملف النصي
private static List<String> readLines(String filelocation)
                                                    } throws IOException
                                    ;()<>List<String> liness = new ArrayList
try (BufferedReader br = new BufferedReader(new
                                                    } FileReader(filelocation)))
```

```
                                        ;String line
                } while ((line = br.readLine()) != null)
                                    ;liness.add(line)
                                                {
                                                    {
                                        ;return liness
                                                        {

                                        //دالة اختبار البحث
private static void searchAndPrintResults(String[]
                                        } texts, String searchitem)
    ;int linearSteps = linearSearch(texts, searchitem)


                                        } if (linearSteps > 8)
int binarySteps = binarySearch(texts,
                                                    ;searchitem)
                                    } if (binarySteps > 15)
int ternarySteps = ternarySearch(texts,
                                                    ;searchitem)
                                } if (ternarySteps != -1)
System.out.println("The text has been
found" + "||" + ternarySteps + "||" + " Step using ternary
                                                    ;search.")
```

```
                                        } else {
System.out.println("The text is not in the
                                                ;list.")
                                            {
                                        } else {
                                } if (binarySteps != -1)
System.out.println("The text has been
found" + "||" + binarySteps + "||" + " Step using binary
                                                ;search.")
                                        } else {
System.out.println("The text is not in the
                                                ;list.")
                                            {
                                                {
                                        } else {
                                } if (linearSteps != -1)
System.out.println("The text has been found"
;+ "||" + linearSteps + "||" + " Step using linear search.")
                                        } else {
System.out.println("The text is not in the
                                                ;list.")
                                            {
                                                {
                                                    {
```

```
// دالة البحث الخطي
private static int linearSearch(String[] texts, String
                                                } target)
                                                ;int steps = 0
                                            } for (String text : texts)
                                                ;++steps
                                        } if (text.equals(target))
                                                ;return steps
                                                        {
                                                            {
                                                ;return -1
                                                                {

// دالة بحث الثنائي
private static int binarySearch(String[] texts, String
                                                } target)
                                                ;int left = 0
                                            ;int right = texts.length - 1
                                                ;int steps = 0

                                            } while (left <= right)
                                                ;++steps
```

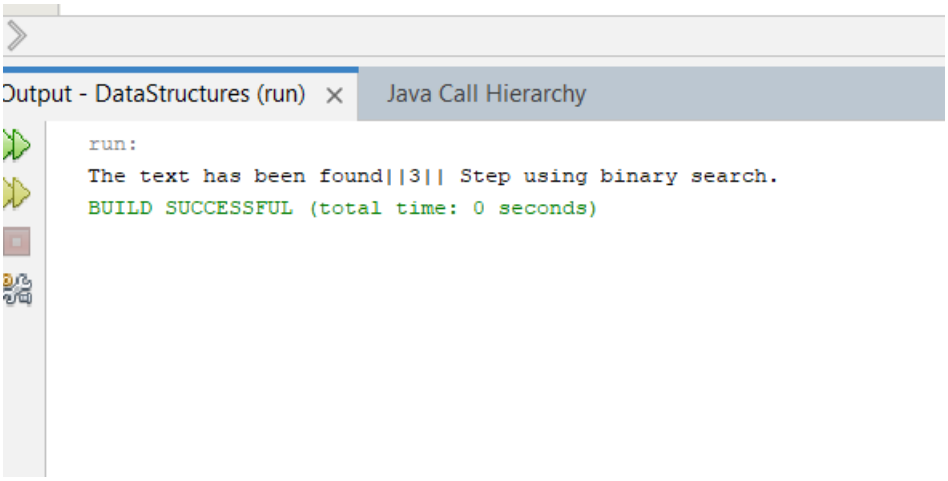
```
                ;int mid = left + (right - left) / 2
            ;int cmp = texts[mid].compareTo(target)


                        } if (cmp < 0)
                    ;left = mid + 1
                } (else if (cmp > 0 {
                    ;right = mid - 1
                            } else {
                    ;return steps
                                {
                                    {
                    ;return -1
                                        {

                // دالة البحث الثلاثي
private static int ternarySearch(String[] texts, String
                                        } target)
                    ;int left = 0
                ;int right = texts.length - 1
                    ;int steps = 0


                } while (left <= right)
```

```
                                        ;++steps
                        ;int m1 = left + (right - left) / 3
                        ;int m2 = right - (right - left) / 3

        ;int c1 = texts[m1].compareTo(target)
        ;int c2 = texts[m2].compareTo(target)

                                    } if (c1 == 0)
                                  ;return steps
                                            {
                                    } if (c2 == 0)
                                  ;return steps
                                            {
                                     } if (c1 > 0)
                                ;right = m1 - 1
                               } (else if (c2 < 0 {
                                 ;left = m2 + 1
                                         } else {
                                 ;left = m1 + 1
                                ;right = m2 - 1
                                            {
                                               {
```

;return -1

{

{


Output - DataStructures (run)
Java Call Hierarchy
run:
The text has been found||3|| Step using binary search.
BUILD SUCCESSFUL (total time: 0 seconds)