

TUGAS KECIL

Implementasi Algoritma Brute Force dalam Penyelesaian Word Search Puzzle

LAPORAN

**Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma
pada**

Semester II

Tahun Akademik 2021-2022

oleh

Owen Christian Wijaya

13520124



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG**

2022

DAFTAR ISI

BAB I. ALGORITMA <i>BRUTE FORCE</i>	3
BAB II. <i>SOURCE PROGRAM</i> DALAM BAHASA JAVA	4
2.1 CharObj.java	4
2.2 Exec.java.....	4
2.3 MainProgram.java.....	4
2.4 Matcher.java.....	6
2.5 Matrix.java.....	12
2.6 SearchWord.java.....	13
2.7 ReadFile.java	13
BAB III. PENGUJIAN (INPUT/OUTPUT)	18
3.1 Pengujian terhadap file small1.txt (ukuran 14 x 12).....	18
3.2 Pengujian terhadap file small2.txt (ukuran 16 x 14).....	19
3.3 Pengujian terhadap file small3.txt (ukuran 14 x 12).....	20
3.4 Pengujian terhadap file medium1.txt (ukuran 20 x 18).....	21
3.5 Pengujian terhadap file medium2.txt (ukuran 22 x 20).....	22
3.6 Pengujian terhadap file medium3.txt (ukuran 24 x 22).....	23
3.7 Pengujian terhadap file large1.txt (ukuran 32 x 30).....	24
3.8 Pengujian terhadap file large2.txt (ukuran 34 x 32).....	25
3.9 Pengujian terhadap file large3.txt (ukuran 36 x 34).....	26

BAB I. ALGORITMA *BRUTE FORCE*

Dalam tugas kecil ini, algoritma *brute force* diimplementasikan dalam pencarian kata-kata yang ada dalam permainan *word search puzzle*. *Puzzle* disimpan dalam bentuk matriks karakter, sementara kata kunci disimpan ke dalam sebuah *list*. Algoritma *brute force* akan mencari keberadaan sebuah kata dalam *puzzle* dengan pencocokan karakter pertama dari kata kunci yang hendak dicari dengan karakter yang ada di dalam *puzzle* secara sekuensial. Apabila ditemukan karakter yang sesuai dengan karakter pertama kata kunci, maka pencarian dilanjutkan dengan mencocokkan karakter-karakter lain dengan karakter di kata kunci.

Pencocokan dilakukan terhadap delapan arah: ke arah kanan, ke arah kiri, ke arah atas, ke arah bawah, ke arah diagonal atas kiri, ke arah diagonal atas kanan, ke arah diagonal bawah kiri, dan ke arah diagonal bawah kanan. Sebelum melakukan pencocokan, program akan melakukan pengecekan apakah panjang kata kunci melebihi dimensi matriks *puzzle*.

Apabila panjang kata kunci yang hendak dicek sesuai arah melebihi dimensi matriks *puzzle* (akan menyebabkan *out of bounds*), maka pengecekan ke arah tersebut tidak akan dilakukan dan program akan melakukan pengecekan ke arah lainnya. Sebaliknya, apabila panjang kata kunci tidak menyebabkan *out of bounds*, maka pengecekan akan dilakukan. Pengecekan akan dihentikan apabila ditemukan satu karakter di arah tersebut yang tidak cocok. Jika pengecekan yang dilakukan berhasil, maka pencarian akan dihentikan dan program akan memberikan keluaran berupa arah yang ditemukan, jumlah perbandingan yang dilakukan, dan waktu yang dibutuhkan untuk melakukan perbandingan. Setelah semua kata kunci dicocokkan, program akan memberikan keluaran berupa matriks awal dengan kata kunci yang di-*highlight* untuk menandai posisi kata kunci di dalam *puzzle*.

Algoritma pencocokan dilakukan dengan mencocokkan karakter pertama di kata kunci dengan tiap karakter di matriks secara sekuensial. Apabila ditemukan karakter di matriks yang sama dengan karakter pertama kata kunci yang hendak dicari, fungsi-fungsi tersebut akan dipanggil. Fungsi menerima argumen berupa matriks *puzzle*, kata kunci yang telah dipisah dalam bentuk *array*, dan koordinat dimana karakter pertama ditemukan. Setiap fungsi akan melakukan pengecekan sesuai arah. Dengan menggunakan skema pencarian menggunakan *boolean*, fungsi akan melakukan pencocokan per karakter sampai *out of bounds*. Apabila ada satu karakter yang tidak cocok, pencarian akan berhenti dan fungsi akan mengembalikan Exec

dengan flag bernilai `false`. Sebaliknya, apabila pencocokan berhasil secara keseluruhan, maka program akan melakukan perubahan terhadap elemen matriks yang dicek. Nilai variabel `colorChar` dalam elemen `charObj` dalam matriks akan diubah menjadi karakter berwarna sesuai dengan nilai koordinat dan panjang kata yang dicek menggunakan kode ANSI. Pada akhir program, matriks yang akan ditampilkan adalah matriks dengan variabel `colorChar` yang telah berwarna.

Compiler Java bersifat *just-in-time*, dan karena itu, pemanggilan instansi pertama akan mengambil waktu lebih lama. Oleh karena itu, untuk membuat hasil pengukuran yang lebih akurat, program akan memanggil instansi `search` terlebih dahulu sebelum melakukan iterasi. Pemanggilan dilakukan dengan mengecek kata pertama di daftar kata kunci dengan indeks -1. Indeks -1 menandakan bahwa *runtime* hanya untuk melakukan pemanggilan instansi, sehingga pada saat melakukan iterasi, waktu yang diperoleh lebih akurat dan tidak menghitung waktu pemanggilan instansi.

BAB II. SOURCE PROGRAM DALAM BAHASA JAVA

2.1 CharObj.java

```
public class CharObj{  
    String oriChar;  
    String colorChar;  
  
    public CharObj(String character){  
        this.oriChar = character;  
        this.colorChar = character;  
    }  
}
```

2.2 Exec.java

```
public class Exec {  
    int count;  
    boolean flag;  
    public Exec(int compCount, boolean flag, String[] keyword, int i, int j){  
        this.count = compCount;  
        this.flag = flag;  
    }  
}
```

```
import java.util.Scanner;

import java.util.ArrayList;

public class MainProgram {
    public static void main(String[] args){

        long duration = 0;
        long count = 0;
        long[] results;
        Matrix m = new Matrix(100, 100);
        ArrayList<String> keywords = new ArrayList<String>();
        Scanner sc = new Scanner(System.in);
        System.out.println(" _____
_____ 
");
        System.out.println("/ _\\ \\ /\\/_ _\\_ _\\ __// _
\\/_ _\\/_ _\\/_ \\ /|");
        System.out.println("| \\|| | || / / / | \\ | /
\\|| \\|| / | |_|");
        System.out.println("| _/| \\_||/ _/ _/ _| |_||| _/| \\_
| | ||");
        System.out.println("\\\\/_ _\\_/_\\_/_\\_/_\\_\\\\\\/_
\\\\\\_/_\\_\\\\\\_/_\\_/ \\|");
        System.out.println();

        System.out.println("Welcome to the Puzzeearch solver!");

        ReadFile.readText(m, keywords, sc);

        System.out.println("Read puzzle from the file: (Size: " + m.rows + " x
" + m.cols + ")");
        m.printOriMatrix();

        System.out.println("\nThere are " + keywords.size() +" read keywords
from the file: ");
        for(int i = 0; i < keywords.size(); i++){
            System.out.println("- " + keywords.get(i));
        }

        System.out.println("Press any key to start searching!");
        sc.nextLine();

        SearchWord.search(m, keywords.get(0), -1);

        for(int i = 0; i < keywords.size(); i++){
```

```
        results = SearchWord.search(m, keywords.get(i), i);
        duration += results[0];
        count += results[1];
    }

    m.printColorMatrix();
    System.out.println();
    System.out.println("Comparison time in total (parsing excluded): " );
    System.out.println(duration + " ns (" + String.format("%.3f",
(double)(duration / 10e5)) + " ms)");
    System.out.println("Comparison count in total: " + count + "
time(s)");
    System.out.println("Press any key to quit...");
    sc.nextLine();
}
}
```

2.4 Matcher.java

```
public class Matcher {
    public static Exec checkHL(Matrix m, String[] keyword, int i, int j){
//horizontal left
        boolean flag = true;
        int compCount = 0;
        Exec tempExec;
        if (j + 1 - keyword.length < 0){
            flag = false;
        } else {
            int count = 1;
            while (count < keyword.length && flag){
                compCount++;
                if (!m.buffer[i][j - count].oriChar.equals(keyword[count])){
                    flag = false;
                } else {
                    count++;
                }
            }
        }
        if(flag){
            int a = 0;
            while (a < keyword.length){
                m.buffer[i][j - a].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16 * i) + (16 * j) + 19) % 185) + 20) + "m" +
m.buffer[i][j - a].oriChar + "\u001B[0m";
                a++;
            }
        }
    }
}
```

```
    }
    tempExec = new Exec(compCount, flag);
    return tempExec;
}

public static Exec checkHR(Matrix m, String[] keyword, int i, int j){
//horizontal right
    boolean flag = true;
    int compCount = 0;
    Exec tempExec;

    if (j + keyword.length > m.cols){
        flag = false;
    } else {
        int count = 1;
        while (count < keyword.length && flag){
            compCount++;
            if (!m.buffer[i][j +
count].oriChar.equals(keyword[count])){
                flag = false;
            } else {
                count++;
            }
        }
    }

    if(flag){
        int a = 0;
        while (a < keyword.length){
            m.buffer[i][j + a].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16 * i) + (16 * j) + 19) % 185) + 20) + "m" +
m.buffer[i][j + a].colorChar + "\u001B[0m";
            a++;
        }
    }
    tempExec = new Exec(compCount, flag);
    return tempExec;
}

public static Exec checkVU(Matrix m, String[] keyword, int i, int j){
//vertical upper
    boolean flag = true;
    int compCount = 0;
    Exec tempExec;

    if (i - keyword.length + 1 < 0){
```

```

        flag = false;
    } else {
        int count = 1;
        while (count < keyword.length && flag){
            compCount++;
            if (!m.buffer[i - count][j].oriChar.equals(keyword[count])){
                flag = false;
            } else {
                count++;
            }
        }
    }
    if(flag){

        int a = 0;
        while (a < keyword.length){
            m.buffer[i - a][j].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16* i) + (16*j) + 19) % 185) + 20) + "m" + m.buffer[i -
a][j].oriChar + "\u001B[0m";
            a++;
        }
    }
    tempExec = new Exec(compCount, flag);
    return tempExec;
}

public static Exec checkVL(Matrix m, String[] keyword, int i, int j){
    boolean flag = true;
    int compCount = 0;
    Exec tempExec;
    if (i + keyword.length > m.rows){
        flag = false;
    } else {
        int count = 1;
        while (count < keyword.length && flag){
            compCount++;
            if (!m.buffer[i + count][j].oriChar.equals(keyword[count])){
                flag = false;
            } else {
                count++;
            }
        }
    }
    if(flag){
        int a = 0;
        while (a < keyword.length){

```



```

        m.buffer[i + a][j].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16 * i) + (16 * j) + 19) % 185) + 20) + "m" +
m.buffer[i + a][j].oriChar + "\u001B[0m";
        a++;
    }
}
tempExec = new Exec(compCount, flag);
return tempExec;
}

public static Exec checkDLU(Matrix m, String[] keyword, int i, int j){
//diagonal left upper
    boolean flag = true;
    int compCount = 0;
    Exec tempExec;

    if (j + 1 - keyword.length < 0 || i - keyword.length + 1 < 0){
        flag = false;
    } else {
        int count = 1;
        while (count < keyword.length && flag){
            compCount++;
            if (!m.buffer[i - count][j -
count].oriChar.equals(keyword[count])){
                flag = false;
            } else {
                count++;
            }
        }
    }
    if(flag){
        int a = 0;
        while (a < keyword.length){
            m.buffer[i - a][j - a].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16 * i) + (16 * j) + 19) % 185) + 20) + "m" +
m.buffer[i - a][j - a].oriChar + "\u001B[0m";
            a++;
        }
    }
    tempExec = new Exec(compCount, flag);
    return tempExec;
}

public static Exec checkDLL(Matrix m, String[] keyword, int i, int j){
//diagonal left lower
    boolean flag = true;

```

```

        int compCount = 0;
        Exec tempExec;

        if (j + 1 - keyword.length < 0 || i + keyword.length > m.rows){
            flag = false;
        } else {
            int count = 1;
            while (count < keyword.length && flag){
                compCount++;
                if (!m.buffer[i + count][j -
count].oriChar.equals(keyword[count])){
                    flag = false;
                } else {
                    count++;
                }
            }
        }
        if(flag){
            int a = 0;
            while (a < keyword.length){
                m.buffer[i + a][j - a].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16 * i) + (16 * j) + 19) % 185) + 20) + "m" +
m.buffer[i + a][j - a].oriChar + "\u001B[0m";
                a++;
            }
        }
        tempExec = new Exec(compCount, flag);
        return tempExec;
    }

    public static Exec checkDRU(Matrix m, String[] keyword, int i, int j){
//diagonal right upper
        boolean flag = true;
        int compCount = 0;
        Exec tempExec;

        if (j + keyword.length > m.cols || i - keyword.length + 1 < 0){
            flag = false;
        } else {
            int count = 1;
            while (count < keyword.length && flag){
                compCount++;
                if (!m.buffer[i - count][j +
count].oriChar.equals(keyword[count])){
                    flag = false;
                } else {

```

```

        count++;
    }
}
}
if(flag){
    int a = 0;
    while (a < keyword.length){
        m.buffer[i - a][j + a].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16 * i) + (16 * j) + 19) % 185) + 20) + "m" +
m.buffer[i - a][j + a].oriChar + "\u001B[0m";
        a++;
    }
}
tempExec = new Exec(compCount, flag);
return tempExec;
}

public static Exec checkDRL(Matrix m, String[] keyword, int i, int j){
//diagonal right lower
    boolean flag = true;
    int compCount = 0;
    Exec tempExec;

    if (j + keyword.length > m.cols || i + keyword.length > m.rows){
        flag = false;
    } else {
        int count = 1;
        while (count < keyword.length && flag){
            compCount++;
            if (!m.buffer[i + count][j +
count].oriChar.equals(keyword[count])){
                flag = false;
            } else {
                count++;
            }
        }
    }
    if(flag){
        int a = 0;
        while (a < keyword.length){
            m.buffer[i + a][j + a].colorChar = "\u001B[1m\u001B[38;5;" +
(((keyword.length + (16 * i) + (16 * j) + 19) % 185) + 20) + "m" +
m.buffer[i + a][j + a].oriChar + "\u001B[0m";
            a++;
        }
    }
}

```

```
    }  
    }  
    tempExec = new Exec(compCount, flag);  
    return tempExec;  
}  
}
```

2.5 Matrix.java

```
public class Matrix {  
    CharObj buffer[][];  
    int rows;  
    int cols;  
  
    public Matrix(int row, int col){  
        this.rows = row;  
        this.cols = col;  
        this.buffer = new CharObj[row][col];  
        for(int i = 0; i < this.rows; i++){  
            for(int j = 0; j < this.cols; j++){  
                this.buffer[i][j] = new CharObj(" ");  
            }  
        }  
    }  
  
    public void printOriMatrix(){  
        for(int i = 0; i < this.rows; i++){  
            for(int j = 0; j < this.cols; j++){  
                System.out.print(this.buffer[i][j].oriChar + " ");  
            }  
            System.out.println();  
        }  
    }  
  
    public void printColorMatrix(){  
        for(int i = 0; i < this.rows; i++){  
            for(int j = 0; j < this.cols; j++){  
                System.out.print(this.buffer[i][j].colorChar + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```

2.6 SearchWord.java

```

public class SearchWord {
    public static long[] search(Matrix m, String keyword, int index){
        String[] keyArr = keyword.split("");
        String first = keyArr[0];
        boolean found = false;
        int i = 0;
        int j = 0;
        int count = 0;
        long time = 0;
        long tempStart = System.nanoTime();

        while (i < m.rows && !found){
            while (j < m.cols && !found){
                count++;
                if (first.equals(m.buffer[i][j].oriChar)){
                    Exec temp = Matcher.checkHR(m, keyArr, i, j);

                    if (!found){
                        count += temp.count;
                        if (temp.flag && index != -1){
                            time = System.nanoTime() - tempStart;
                            System.out.println("Keyword \" +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\" found horizontally right!");
                            System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

                            found = true;
                        }
                    }

                    temp = Matcher.checkDRL(m, keyArr, i, j);
                    if (!found){
                        count += temp.count;
                        if (temp.flag && index != -1){
                            time = System.nanoTime() - tempStart;
                            System.out.println("Keyword \" +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\" found diagonally! (right lower)");
                            System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

                            found = true;
                        }
                    }
                }
            }
        }
    }
}

```

```

    }

    temp = Matcher.checkVL(m, keyArr, i, j);
    if (!found){
        count += temp.count;
        if (temp.flag && index != -1){
            time = System.nanoTime() - tempStart;
            System.out.println("Keyword \'' +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\' found vertically lower!");
            System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

            found = true;
        }
    }

    temp = Matcher.checkDLL(m, keyArr, i, j);
    if (!found){
        count += temp.count;
        if (temp.flag && index != -1){
            time = System.nanoTime() - tempStart;
            System.out.println("Keyword \'' +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\' found diagonally! (left lower)");
            System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

            found = true;
        }
    }

    temp = Matcher.checkHL(m, keyArr, i, j);
    if (!found){
        count += temp.count;
        if (temp.flag && index != -1){
            time = System.nanoTime() - tempStart;
            System.out.println("Keyword \'' +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\' found horizontally left!");
            System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

            found = true;
        }
    }
}

```

```

        temp = Matcher.checkDLU(m, keyArr, i, j);
        if (!found){
            count += temp.count;
            if (temp.flag && index != -1){
                time = System.nanoTime() - tempStart;
                System.out.println("Keyword \'' +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\' found diagonally! (left upper)");
                System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

                found = true;
            }
        }

        temp = Matcher.checkVU(m, keyArr, i, j);
        if (!found){
            count += temp.count;
            if (temp.flag && index != -1 ){
                time = System.nanoTime() - tempStart;
                System.out.println("Keyword \'' +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\' found vertically upper!");
                System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

                found = true;
            }
        }

        temp = Matcher.checkDRU(m, keyArr, i, j);
        if (!found){
            count += temp.count;
            if (temp.flag && index != -1 && !found){
                time = System.nanoTime() - tempStart;
                System.out.println("Keyword \'' +
"\u001B[1m\u001B[38;5;" + (((keyArr.length + (16 * i) + (16 * j) + 19) %
185) + 20) + "m" + keyword + "\u001B[0m\' found diagonally! (right upper)");
                System.out.println("Comparison: " + count + "
time(s) | " + time + " ns (" + String.format("%.3f", (double)(time / 10e5)) +
" ms)");

                found = true;
            }
        }
    }
}

```

```
        }
        j = j + 1;
    }
    j = 0;
    i = i + 1;
}
if (!found && index != -1){
    time = System.nanoTime() - tempStart;
    System.out.println("Keyword \" + keyword + "\" not found...");
    System.out.println("Comparison: " + count + " time(s) | " + time +
" ns (" + String.format("%.3f", (double)(time / 10e5)) + " ms)");
}
System.out.println();
long[] arr = {time, Long.valueOf(count)};
return arr;
}
}
```

2.7 ReadFile.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.ArrayList;

public class ReadFile{
    public static void readText(Matrix m, ArrayList<String> keywords, Scanner
sc){

        String filename = "";

        System.out.print("Input your filename (without .txt): ");
        filename = sc.nextLine();
        while (filename == "" || !(new File("../test/" + filename +
".txt").exists())){
            System.out.println("Sorry, file name doesn't exist or it has not
been put in the /test folder!");
            System.out.print("Input your filename (without .txt): ");
            filename = sc.nextLine();
        }
        try{
            File text = new File("../test/" + filename + ".txt");
            Scanner sizeReader = new Scanner(text);
            int rowSize = 1;
```



```
        while (sizeReader.hasNextLine() &&
!sizeReader.nextLine().equals("")){
            rowSize++;
        }

        sizeReader.close();
        m.rows = rowSize - 1;

        Scanner lineReader = new Scanner(text);
        m.cols = 0;
        try{
            for(int i = 0; i < m.rows; i++){
                String line = lineReader.nextLine();
                String rows[] = line.split(" ");
                if (m.cols < rows.length){
                    m.cols = rows.length;
                }
                for(int j = 0; j < rows.length; j++){
                    m.buffer[i][j] = new CharObj(rows[j]);
                }

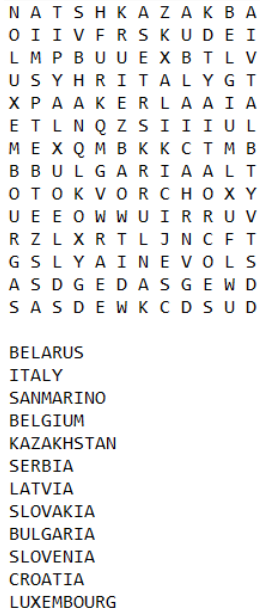
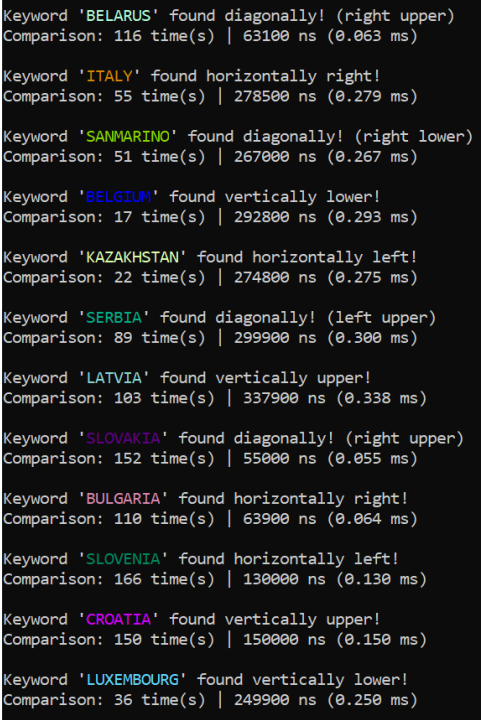
                if (rows.length < m.cols){
                    for(int j = rows.length + 1; j < m.cols; j++){
                        m.buffer[i][j] = new CharObj(" ");
                    }
                }
            }
            lineReader.nextLine();
            while(lineReader.hasNextLine()){
                String kw = lineReader.nextLine();
                keywords.add(kw);
            }

        } finally {
            lineReader.close();
        }

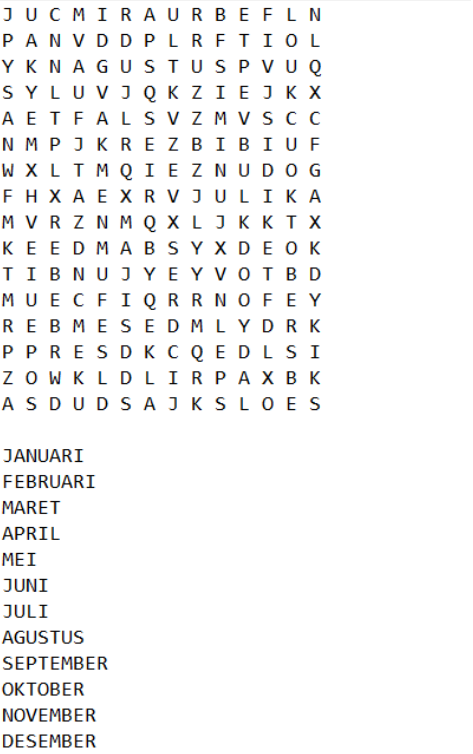
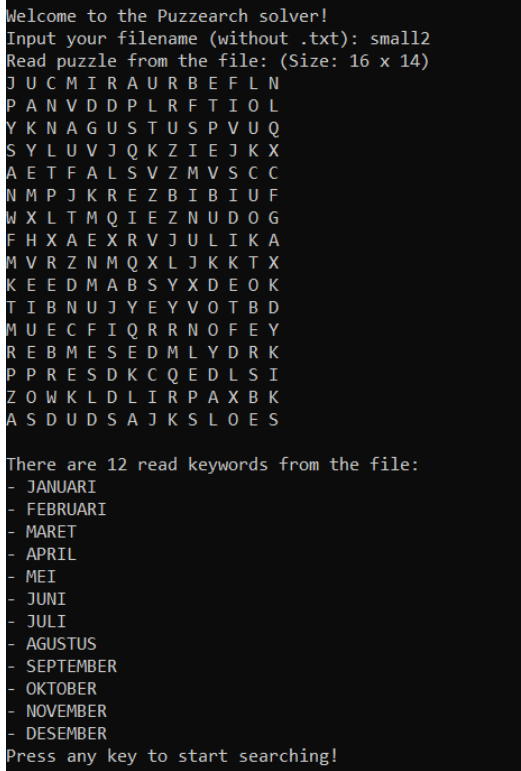
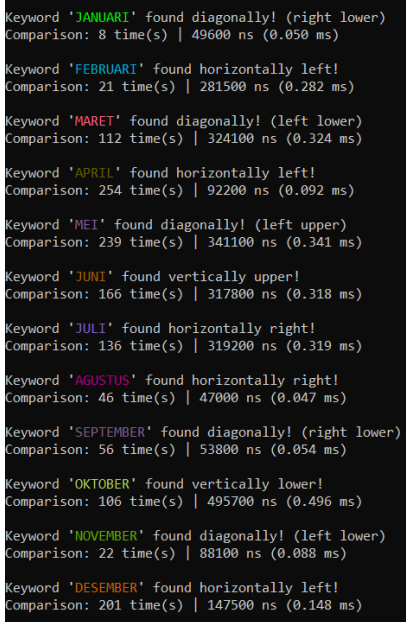
    } catch (FileNotFoundException e){
        System.out.println("File not found!");
        e.printStackTrace();
    }
}
```

BAB III. PENGUJIAN (INPUT/OUTPUT)

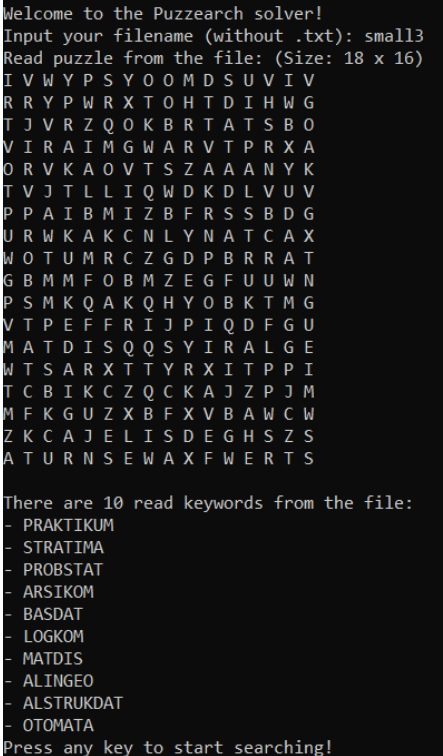
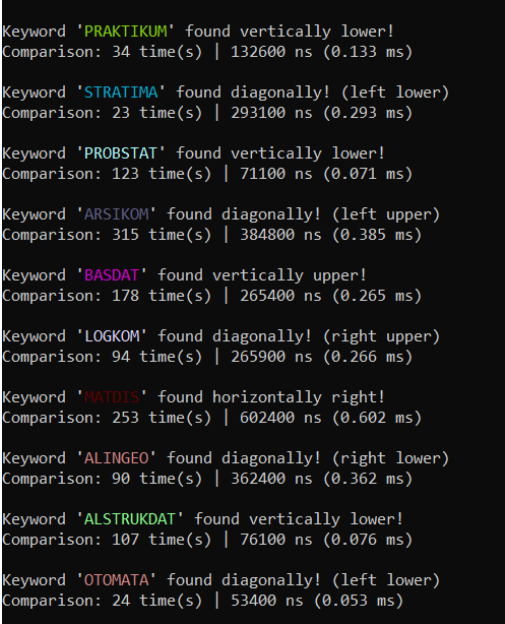
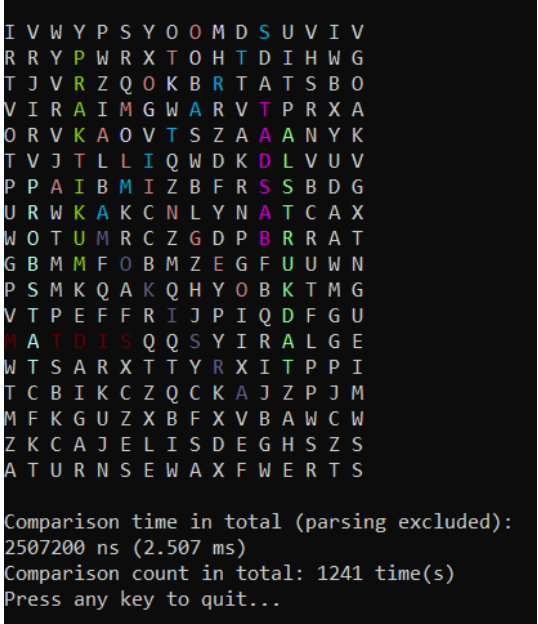
3.1 Pengujian terhadap file small1.txt (ukuran 14 x 12)

	
<p>Gambar 3.1.1 Isian berkas small1.txt</p>	<p>Gambar 3.1.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.1.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.1.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

3.2 Pengujian terhadap file small2.txt (ukuran 16 x 14)

	
<p>Gambar 3.2.1 Isian berkas small2.txt</p>	<p>Gambar 3.2.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.2.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.2.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

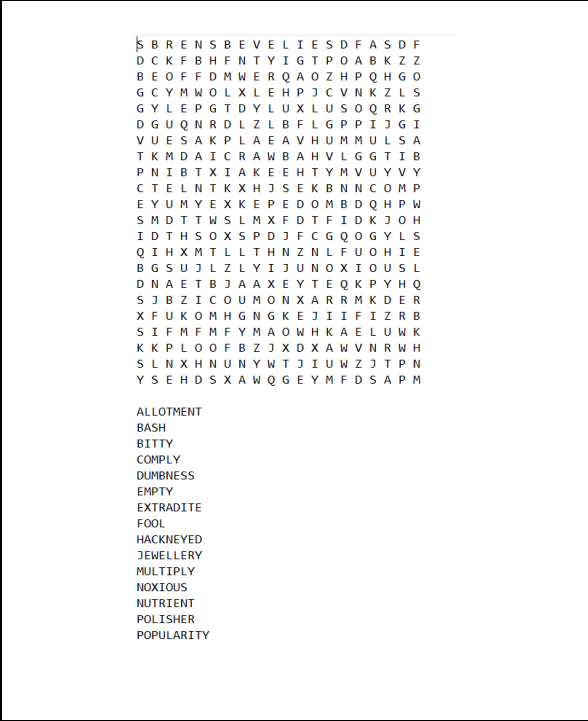
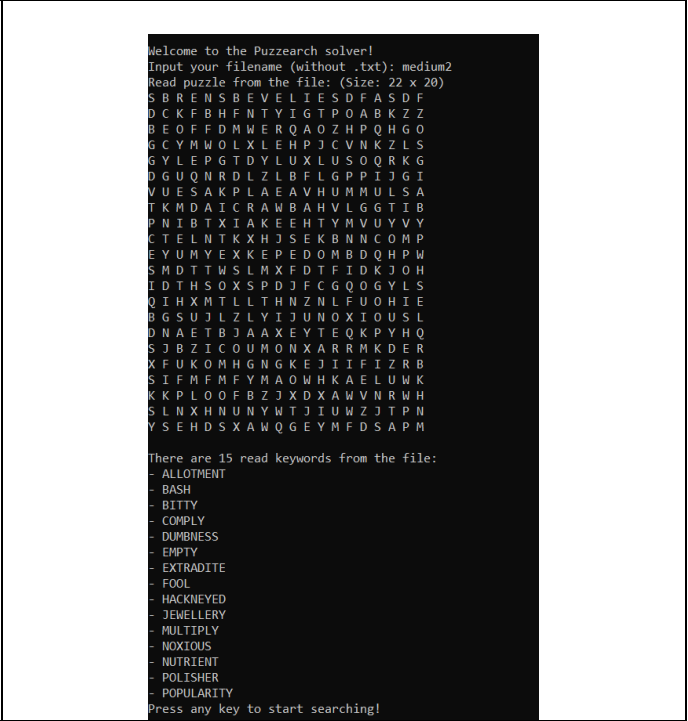
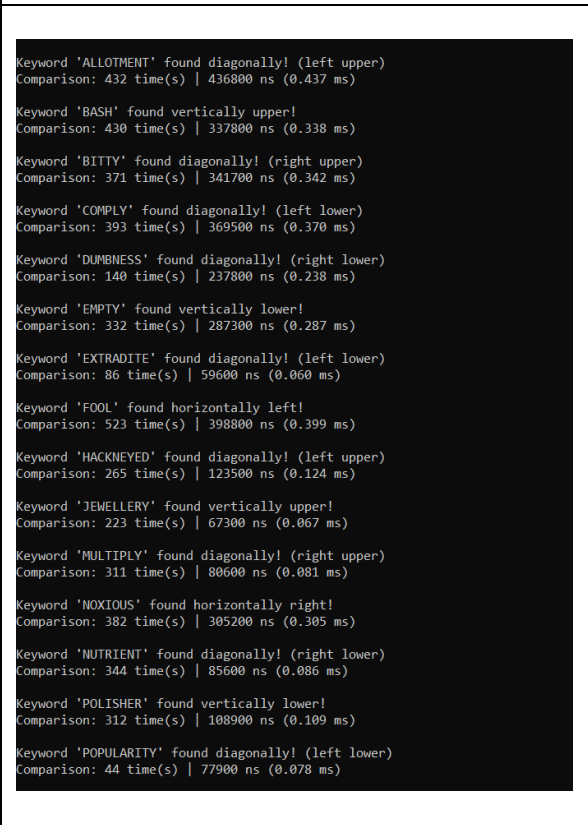
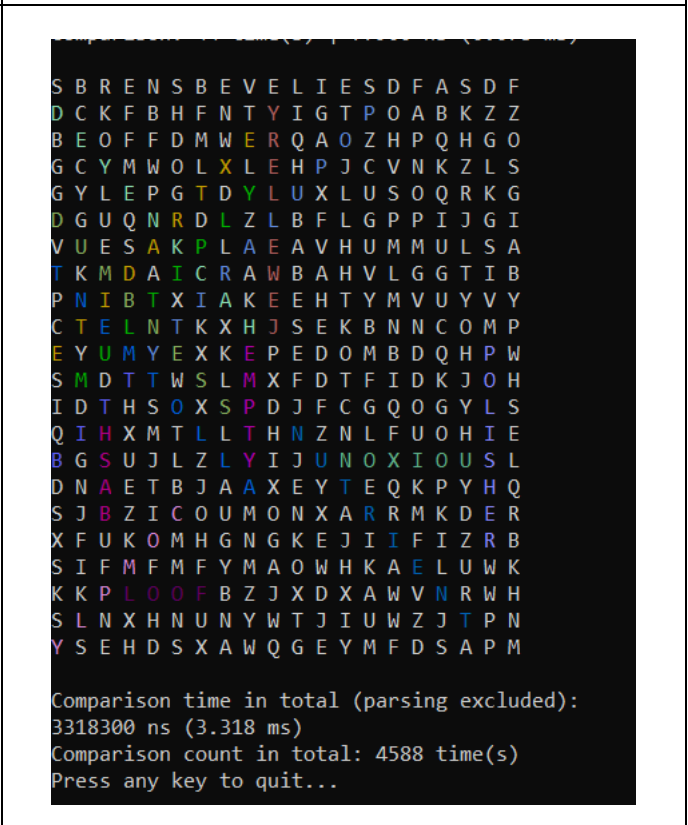
3.3 Pengujian terhadap file small3.txt (ukuran 14 x 12)

	
<p>Gambar 3.3.1 Isian berkas small3.txt</p>	<p>Gambar 3.3.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.3.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.3.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

3.4 Pengujian terhadap file medium1.txt (ukuran 20 x 18)

	
<p>Gambar 3.4.1 Isian berkas medium1.txt</p>	<p>Gambar 3.4.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.4.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.4.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

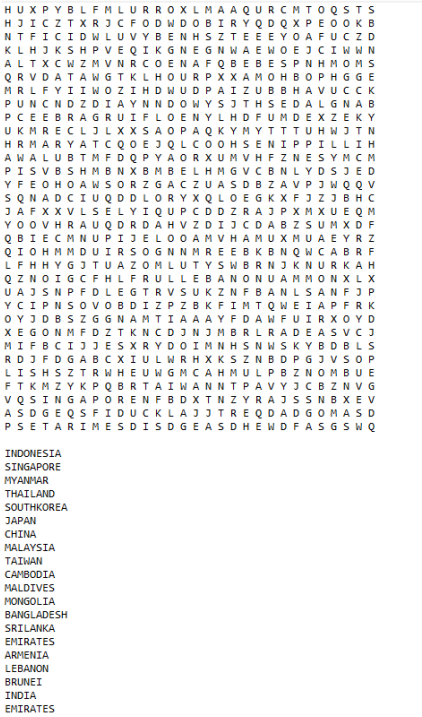
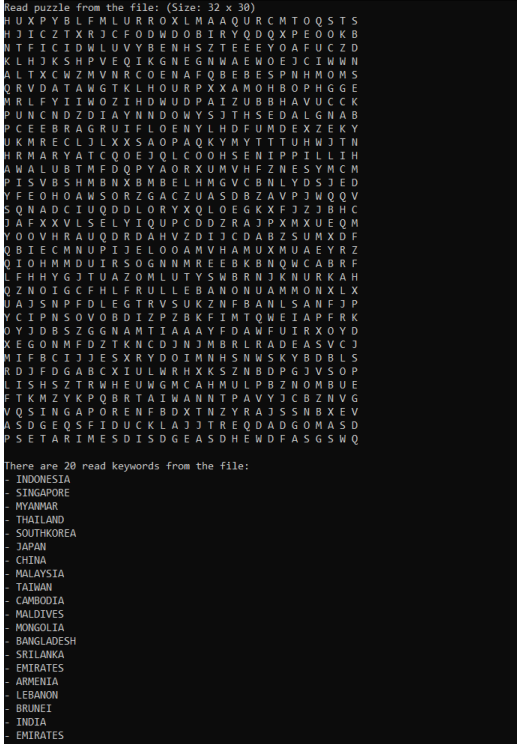
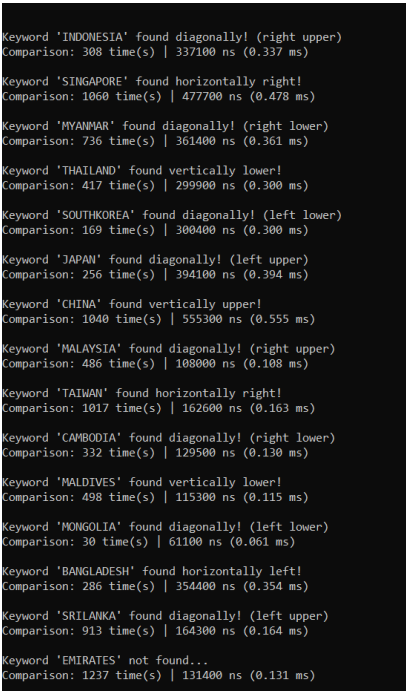

3.5 Pengujian terhadap file medium2.txt (ukuran 22 x 20)

 <p>§ B R E N S B E V E L I E S D F A S D F D C K F B H F N T Y I G T P O A B K Z Z B E O F F D M W E R Q A O Z H P Q H G O G C Y M W O L X L E H P J C V N K Z L S G Y L E P G T D Y L U X L U S O Q R K G D G U Q N R D L Z L B F L G P P I J G I V U E S A K P L A E A V H U M M U L S A T K M D A I C R A W B A H V L G G T I B P N I B T X I A K E E H T Y M V U Y V Y C T E L N T K X H J S E K B N N C O M P E Y U M Y E X K E P E D O M B D Q H P W S M D T T W S L M X F D T F I D K J O H I D T H S O X S P D J F C G Q O G Y L S Q I H X M T L L T H N Z N L F U O H I E B G S U J L Z L Y I J U N O X I O U S L D N A E T B J A A X E Y T E Q K P Y H Q S J B Z I C O U M O N X A R R M K D E R X F U K O M H G N G K E J I I F I Z R B S I F M F M F Y M A O W H K A E L U W K K K P L O O F B Z J X D X A W V N R W H S L N X H N U N Y W T J I U W Z J T P N Y S E H D S X A W Q G E Y M F D S A P M</p> <p>ALLOTMENT BASH BITTY COMPLY DUMBNESS EMPTY EXTRADITE FOOL HACKNEYED JEWELLERY MULTIPLY NOXIOUS NUTRIENT POLISHER POPULARITY</p>	 <pre>Welcome to the Puzsearch solver! Input your filename (without .txt): medium2 Read puzzle from the file: (Size: 22 x 20) § B R E N S B E V E L I E S D F A S D F D C K F B H F N T Y I G T P O A B K Z Z B E O F F D M W E R Q A O Z H P Q H G O G C Y M W O L X L E H P J C V N K Z L S G Y L E P G T D Y L U X L U S O Q R K G D G U Q N R D L Z L B F L G P P I J G I V U E S A K P L A E A V H U M M U L S A T K M D A I C R A W B A H V L G G T I B P N I B T X I A K E E H T Y M V U Y V Y C T E L N T K X H J S E K B N N C O M P E Y U M Y E X K E P E D O M B D Q H P W S M D T T W S L M X F D T F I D K J O H I D T H S O X S P D J F C G Q O G Y L S Q I H X M T L L T H N Z N L F U O H I E B G S U J L Z L Y I J U N O X I O U S L D N A E T B J A A X E Y T E Q K P Y H Q S J B Z I C O U M O N X A R R M K D E R X F U K O M H G N G K E J I I F I Z R B S I F M F M F Y M A O W H K A E L U W K K K P L O O F B Z J X D X A W V N R W H S L N X H N U N Y W T J I U W Z J T P N Y S E H D S X A W Q G E Y M F D S A P M There are 15 read keywords from the file: - ALLOTMENT - BASH - BITTY - COMPLY - DUMBNESS - EMPTY - EXTRADITE - FOOL - HACKNEYED - JEWELLERY - MULTIPLY - NOXIOUS - NUTRIENT - POLISHER - POPULARITY Press any key to start searching!</pre>
<p>Gambar 3.5.1 Isian berkas medium1.txt</p>	<p>Gambar 3.5.2 Hasil pembacaan file masukan</p>
 <pre>Keyword 'ALLOTMENT' found diagonally! (left upper) Comparison: 432 time(s) 436800 ns (0.437 ms) Keyword 'BASH' found vertically upper! Comparison: 430 time(s) 337800 ns (0.338 ms) Keyword 'BITTY' found diagonally! (right upper) Comparison: 371 time(s) 341700 ns (0.342 ms) Keyword 'COMPLY' found diagonally! (left lower) Comparison: 393 time(s) 369500 ns (0.370 ms) Keyword 'DUMBNESS' found diagonally! (right lower) Comparison: 140 time(s) 237800 ns (0.238 ms) Keyword 'EMPTY' found vertically lower! Comparison: 332 time(s) 287300 ns (0.287 ms) Keyword 'EXTRADITE' found diagonally! (left lower) Comparison: 86 time(s) 59600 ns (0.060 ms) Keyword 'FOOL' found horizontally left! Comparison: 523 time(s) 398800 ns (0.399 ms) Keyword 'HACKNEYED' found diagonally! (left upper) Comparison: 265 time(s) 123500 ns (0.124 ms) Keyword 'JEWELLERY' found vertically upper! Comparison: 223 time(s) 67300 ns (0.067 ms) Keyword 'MULTIPLY' found diagonally! (right upper) Comparison: 311 time(s) 80600 ns (0.081 ms) Keyword 'NOXIOUS' found horizontally right! Comparison: 382 time(s) 385200 ns (0.385 ms) Keyword 'NUTRIENT' found diagonally! (right lower) Comparison: 344 time(s) 85600 ns (0.086 ms) Keyword 'POLISHER' found vertically lower! Comparison: 312 time(s) 108900 ns (0.109 ms) Keyword 'POPULARITY' found diagonally! (left lower) Comparison: 44 time(s) 77900 ns (0.078 ms)</pre>	 <pre>§ B R E N S B E V E L I E S D F A S D F D C K F B H F N T Y I G T P O A B K Z Z B E O F F D M W E R Q A O Z H P Q H G O G C Y M W O L X L E H P J C V N K Z L S G Y L E P G T D Y L U X L U S O Q R K G D G U Q N R D L Z L B F L G P P I J G I V U E S A K P L A E A V H U M M U L S A T K M D A I C R A W B A H V L G G T I B P N I B T X I A K E E H T Y M V U Y V Y C T E L N T K X H J S E K B N N C O M P E Y U M Y E X K E P E D O M B D Q H P W S M D T T W S L M X F D T F I D K J O H I D T H S O X S P D J F C G Q O G Y L S Q I H X M T L L T H N Z N L F U O H I E B G S U J L Z L Y I J U N O X I O U S L D N A E T B J A A X E Y T E Q K P Y H Q S J B Z I C O U M O N X A R R M K D E R X F U K O M H G N G K E J I I F I Z R B S I F M F M F Y M A O W H K A E L U W K K K P L O O F B Z J X D X A W V N R W H S L N X H N U N Y W T J I U W Z J T P N Y S E H D S X A W Q G E Y M F D S A P M Comparison time in total (parsing excluded): 3318300 ns (3.318 ms) Comparison count in total: 4588 time(s) Press any key to quit...</pre>
<p>Gambar 3.5.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.5.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

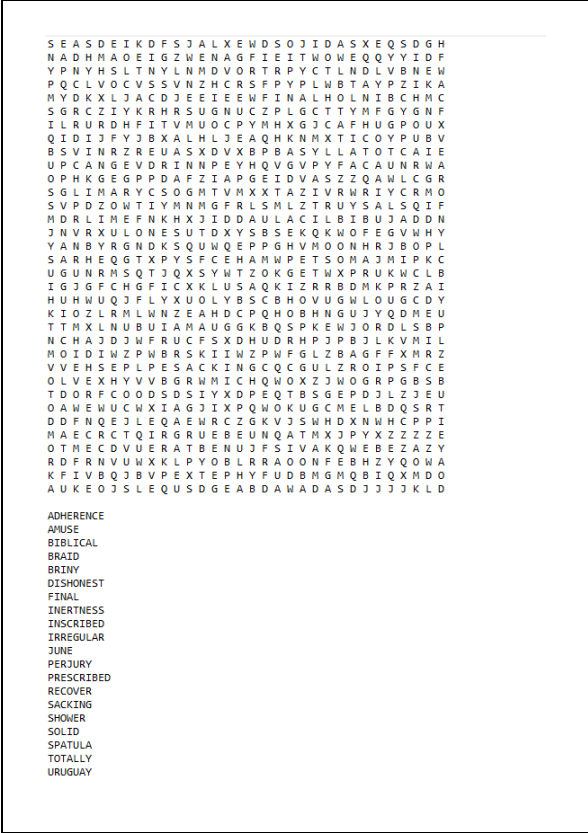
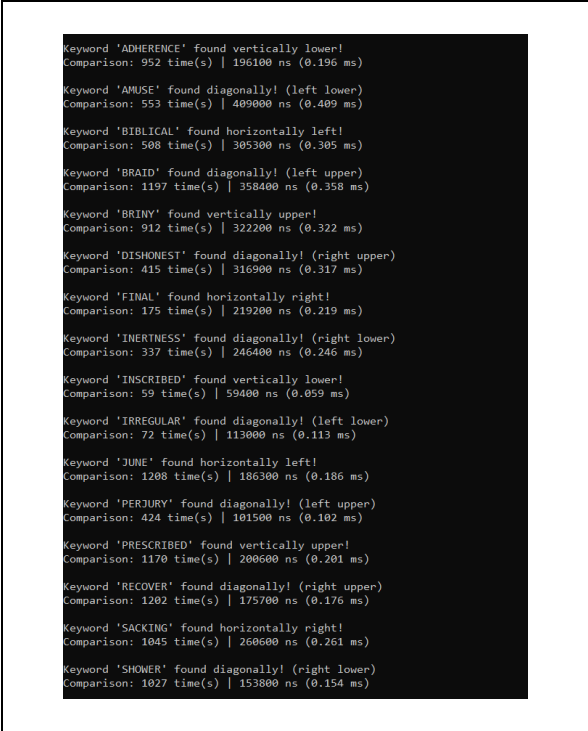
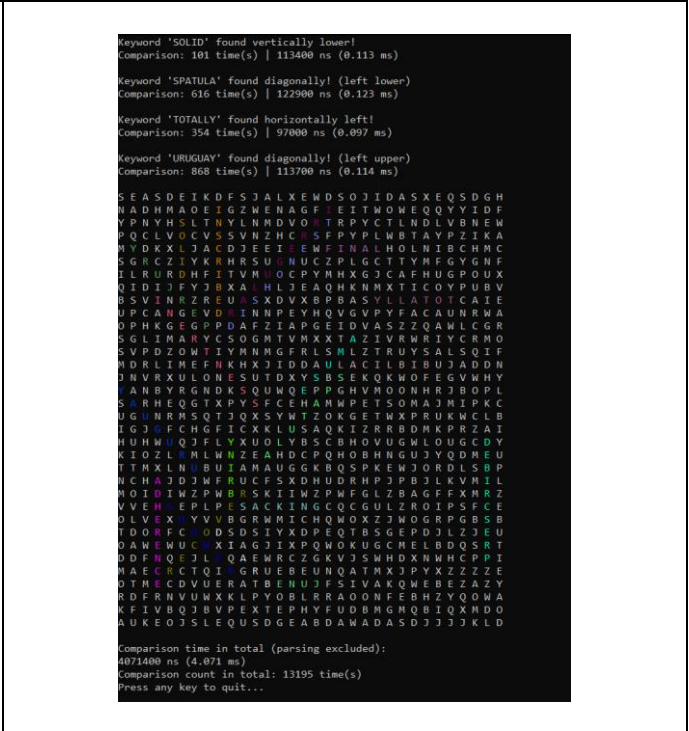
3.6 Pengujian terhadap file medium3.txt (ukuran 24 x 22)

	
<p>Gambar 3.6.1 Isian berkas medium3.txt</p>	<p>Gambar 3.6.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.6.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.6.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

3.7 Pengujian terhadap file large1.txt (ukuran 32 x 30)

	
<p>Gambar 3.7.1 Isian berkas large1.txt</p>	<p>Gambar 3.7.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.7.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.7.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

3.8 Pengujian terhadap file large2.txt (ukuran 34 x 32)

	
<p>Gambar 3.8.1 Isian berkas large2.txt</p>	<p>Gambar 3.8.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.8.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.8.4 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan</p>

3.9 Pengujian terhadap file large3.txt (ukuran 36 x 34)

<pre> NSZNRARUCAXWMFMYRMCNFCZCPAYPDPRIA J AYRIKAUAKNFUKFYEZQKZSAMTEDNKTMPY GQLASD KLMTJIOVQCSBMPKIPCALLIDACZ IHJSCZUNLEYRCYBAVSDGATOTYOTNAQSEY NRIVXNAGJAMFEMAXJDEXTNWXXIDEBERT ONIDCXEHBUHTUOMYLPGNPOQKFSSHORZJ OTKIBCBRLZLXCRVXZBQLNPHNKKCOGLASX HEYNGXMAATVBWUMIZUWGPEIDHRWMMTISI PLGPEIDXOLB XF AFDMVVTUKHXILVYDISP BONNERAMOQCHEKVRAOXNTCQZOWAWRRRL QRTTIEOCIXEMS ZGJJUKQBHOJXGMI RGTZ AVKFFNLGVNZLTABTVASRENAULTD TOQEM METNNVTDUVJDIOQNLINTNGMYKIPLJPTM LHBDVILSHBYWVCHMTMQDIVSWNUZXHWYS NCAAWITPUSIQAOFAIDAOENAPFBZOLKYH EYXHPAVGNWNOLLCLSSUHYRHOBELHWLAA GQFPDDADDFWUFLPVMGILRFSEBNQELPSY EHRGKTPAGANIPOS DIKDRQESOATOIFEOT QXCJTLYCFWRBEPASTONMARTINLWTRRNW LJOIJXBWEBCINAI ZJXTDQLYAZEEAQRU O AQMKPXBRAWUHG YMKHANAROSCYTYINGS ETMLOKJLRYLUAJYLDSESEZYUPBIMLQGGHI FNF BWTEHAINSL ETPEMDPLBVENXEHOEPE YCI OOIITRCCNLHBJBUXAADD CERFTZRER JGKPRRUVIKOENIGSEGGRCMSBGDBGROFR WJJALZGEEVOGJTPJRKUGNRBTAVLYSVPA YVILNAAHSPUEGEOTTUVPUIAJWPPVEEUC D BOXHSDTIHNRADII RKAGVLYSXQARRCA HIUXKG EWHNWELPEZSOBVFPIYKCNHYKJR DLGHBZDAXOIZNEBSEDECREMYLUMAXBWS MKVSQUGLTQNHGNICARGREBSROFYDSICE MITSUBISHISFQEHUWRPDVXUPVTJJRSPD RIPASDETERAUTLINTSIDNEOASDHDSKIA FIASNIHSSKTHEFGHTASDFHEQSFHDSAWN </pre>	<table> <tr><td>ACURA</td><td>PEEL</td></tr> <tr><td>ALPINE</td><td>PENHALL</td></tr> <tr><td>APOLLO</td><td>PEUGEOT</td></tr> <tr><td>ARIEL</td><td>PLYMOUTH</td></tr> <tr><td>ASTONMARTIN</td><td>POLARIS</td></tr> <tr><td>AUSTINHEALEY</td><td>PONTIAC</td></tr> <tr><td>BENTLEY</td><td>RELIANT</td></tr> <tr><td>BUGATTI</td><td>RENAULT</td></tr> <tr><td>CADILLAC</td><td>RJANDERSON</td></tr> <tr><td>CATERHAM</td><td>SIERRACARSE</td></tr> <tr><td>CHEVROLET</td><td>SUBARU</td></tr> <tr><td>DATSUN</td><td>TOYOTA</td></tr> <tr><td>DEBERTI</td><td>VAUXHALL</td></tr> <tr><td>EXOMOTIVE</td><td>VOLKSWAGEN</td></tr> <tr><td>FERRARI</td><td>WILLYS</td></tr> <tr><td>FORMULADRIFT</td><td>GYMKHANA</td></tr> <tr><td>FORSBERGRACING</td><td>FORZATHON</td></tr> <tr><td>HENNESSEY</td><td>FESTIVAL</td></tr> <tr><td>HOONIGAN</td><td>HORIZON</td></tr> <tr><td>INFINITI</td><td>MEXICO</td></tr> <tr><td>KOENIGSEGG</td><td>ARCADE</td></tr> <tr><td>LAMBORGHINI</td><td></td></tr> <tr><td>RANGEROVER</td><td></td></tr> <tr><td>MASERATI</td><td></td></tr> <tr><td>MERCEDES BENZ</td><td></td></tr> <tr><td>MCLARENCARS</td><td></td></tr> <tr><td>MITSUBISHI</td><td></td></tr> <tr><td>NISSAN</td><td></td></tr> <tr><td>PAGANI</td><td></td></tr> </table>	ACURA	PEEL	ALPINE	PENHALL	APOLLO	PEUGEOT	ARIEL	PLYMOUTH	ASTONMARTIN	POLARIS	AUSTINHEALEY	PONTIAC	BENTLEY	RELIANT	BUGATTI	RENAULT	CADILLAC	RJANDERSON	CATERHAM	SIERRACARSE	CHEVROLET	SUBARU	DATSUN	TOYOTA	DEBERTI	VAUXHALL	EXOMOTIVE	VOLKSWAGEN	FERRARI	WILLYS	FORMULADRIFT	GYMKHANA	FORSBERGRACING	FORZATHON	HENNESSEY	FESTIVAL	HOONIGAN	HORIZON	INFINITI	MEXICO	KOENIGSEGG	ARCADE	LAMBORGHINI		RANGEROVER		MASERATI		MERCEDES BENZ		MCLARENCARS		MITSUBISHI		NISSAN		PAGANI	
ACURA	PEEL																																																										
ALPINE	PENHALL																																																										
APOLLO	PEUGEOT																																																										
ARIEL	PLYMOUTH																																																										
ASTONMARTIN	POLARIS																																																										
AUSTINHEALEY	PONTIAC																																																										
BENTLEY	RELIANT																																																										
BUGATTI	RENAULT																																																										
CADILLAC	RJANDERSON																																																										
CATERHAM	SIERRACARSE																																																										
CHEVROLET	SUBARU																																																										
DATSUN	TOYOTA																																																										
DEBERTI	VAUXHALL																																																										
EXOMOTIVE	VOLKSWAGEN																																																										
FERRARI	WILLYS																																																										
FORMULADRIFT	GYMKHANA																																																										
FORSBERGRACING	FORZATHON																																																										
HENNESSEY	FESTIVAL																																																										
HOONIGAN	HORIZON																																																										
INFINITI	MEXICO																																																										
KOENIGSEGG	ARCADE																																																										
LAMBORGHINI																																																											
RANGEROVER																																																											
MASERATI																																																											
MERCEDES BENZ																																																											
MCLARENCARS																																																											
MITSUBISHI																																																											
NISSAN																																																											
PAGANI																																																											

Gambar 3.9.1 Isian berkas large3.txt (Puzzle)

Gambar 3.9.2 Isian berkas large3.txt (Kata kunci)

<pre> Welcome to the Puzzsearch solver! Input your filename (without .txt): large3 Read puzzle from the file: (Size: 34 x 32) NSZNRARUCAXWMFMYRMCNFCZCPAYPDPRIA J AYRIKAUAKNFUKFYEZQKZSAMTEDNKTMPY GQLASD KLMTJIOVQCSBMPKIPCALLIDACZ IHJSCZUNLEYRCYBAVSDGATOTYOTNAQSEY NRIVXNAGJAMFEMAXJDEXTNWXXIDEBERT ONIDCXEHBUHTUOMYLPGNPOQKFSSHORZJ OTKIBCBRLZLXCRVXZBQLNPHNKKCOGLASX HEYNGXMAATVBWUMIZUWGPEIDHRWMMTISI PLGPEIDXOLB XF AFDMVVTUKHXILVYDISP BONNERAMOQCHEKVRAOXNTCQZOWAWRRRL QRTTIEOCIXEMS ZGJJUKQBHOJXGMI RGTZ AVKFFNLGVNZLTABTVASRENAULTD TOQEM METNNVTDUVJDIOQNLINTNGMYKIPLJPTM LHBDVILSHBYWVCHMTMQDIVSWNUZXHWYS NCAAWITPUSIQAOFAIDAOENAPFBZOLKYH EYXHPAVGNWNOLLCLSSUHYRHOBELHWLAA GQFPDDADDFWUFLPVMGILRFSEBNQELPSY EHRGKTPAGANIPOS DIKDRQESOATOIFEOT QXCJTLYCFWRBEPASTONMARTINLWTRRNW LJOIJXBWEBCINAI ZJXTDQLYAZEEAQRU O AQMKPXBRAWUHG YMKHANAROSCYTYINGS ETMLOKJLRYLUAJYLDSESEZYUPBIMLQGGHI FNF BWTEHAINSL ETPEMDPLBVENXEHOEPE YCI OOIITRCCNLHBJBUXAADD CERFTZRER JGKPRRUVIKOENIGSEGGRCMSBGDBGROFR WJJALZGEEVOGJTPJRKUGNRBTAVLYSVPA YVILNAAHSPUEGEOTTUVPUIAJWPPVEEUC D BOXHSDTIHNRADII RKAGVLYSXQARRCA HIUXKG EWHNWELPEZSOBVFPIYKCNHYKJR DLGHBZDAXOIZNEBSEDECREMYLUMAXBWS MKVSQUGLTQNHGNICARGREBSROFYDSICE MITSUBISHISFQEHUWRPDVXUPVTJJRSPD RIPASDETERAUTLINTSIDNEOASDHDSKIA FIASNIHSSKTHEFGHTASDFHEQSFHDSAWN </pre>	<table> <tr><td>- ACURA</td><td>- PEEL</td></tr> <tr><td>- ALPINE</td><td>- PENHALL</td></tr> <tr><td>- APOLLO</td><td>- PEUGEOT</td></tr> <tr><td>- ARIEL</td><td>- PLYMOUTH</td></tr> <tr><td>- ASTONMARTIN</td><td>- POLARIS</td></tr> <tr><td>- AUSTINHEALEY</td><td>- PONTIAC</td></tr> <tr><td>- BENTLEY</td><td>- RELIANT</td></tr> <tr><td>- BUGATTI</td><td>- RENAULT</td></tr> <tr><td>- CADILLAC</td><td>- RJANDERSON</td></tr> <tr><td>- CATERHAM</td><td>- SIERRACARS</td></tr> <tr><td>- CHEVROLET</td><td>- SUBARU</td></tr> <tr><td>- DATSUN</td><td>- TOYOTA</td></tr> <tr><td>- DEBERTI</td><td>- VAUXHALL</td></tr> <tr><td>- EXOMOTIVE</td><td>- VOLKSWAGEN</td></tr> <tr><td>- FERRARI</td><td>- WILLYS</td></tr> <tr><td>- FORMULADRIFT</td><td>- GYMKHANA</td></tr> <tr><td>- FORSBERGRACING</td><td>- FORZATHON</td></tr> <tr><td>- HENNESSEY</td><td>- FESTIVAL</td></tr> <tr><td>- HOONIGAN</td><td>- HORIZON</td></tr> <tr><td>- INFINITI</td><td>- MEXICO</td></tr> <tr><td>- KOENIGSEGG</td><td>- ARCADE</td></tr> <tr><td>- LAMBORGHINI</td><td></td></tr> <tr><td>- RANGEROVER</td><td></td></tr> <tr><td>- MASERATI</td><td></td></tr> <tr><td>- MERCEDES BENZ</td><td></td></tr> <tr><td>- MCLARENCARS</td><td></td></tr> <tr><td>- MITSUBISHI</td><td></td></tr> <tr><td>- NISSAN</td><td></td></tr> <tr><td>- PAGANI</td><td></td></tr> </table>	- ACURA	- PEEL	- ALPINE	- PENHALL	- APOLLO	- PEUGEOT	- ARIEL	- PLYMOUTH	- ASTONMARTIN	- POLARIS	- AUSTINHEALEY	- PONTIAC	- BENTLEY	- RELIANT	- BUGATTI	- RENAULT	- CADILLAC	- RJANDERSON	- CATERHAM	- SIERRACARS	- CHEVROLET	- SUBARU	- DATSUN	- TOYOTA	- DEBERTI	- VAUXHALL	- EXOMOTIVE	- VOLKSWAGEN	- FERRARI	- WILLYS	- FORMULADRIFT	- GYMKHANA	- FORSBERGRACING	- FORZATHON	- HENNESSEY	- FESTIVAL	- HOONIGAN	- HORIZON	- INFINITI	- MEXICO	- KOENIGSEGG	- ARCADE	- LAMBORGHINI		- RANGEROVER		- MASERATI		- MERCEDES BENZ		- MCLARENCARS		- MITSUBISHI		- NISSAN		- PAGANI	
- ACURA	- PEEL																																																										
- ALPINE	- PENHALL																																																										
- APOLLO	- PEUGEOT																																																										
- ARIEL	- PLYMOUTH																																																										
- ASTONMARTIN	- POLARIS																																																										
- AUSTINHEALEY	- PONTIAC																																																										
- BENTLEY	- RELIANT																																																										
- BUGATTI	- RENAULT																																																										
- CADILLAC	- RJANDERSON																																																										
- CATERHAM	- SIERRACARS																																																										
- CHEVROLET	- SUBARU																																																										
- DATSUN	- TOYOTA																																																										
- DEBERTI	- VAUXHALL																																																										
- EXOMOTIVE	- VOLKSWAGEN																																																										
- FERRARI	- WILLYS																																																										
- FORMULADRIFT	- GYMKHANA																																																										
- FORSBERGRACING	- FORZATHON																																																										
- HENNESSEY	- FESTIVAL																																																										
- HOONIGAN	- HORIZON																																																										
- INFINITI	- MEXICO																																																										
- KOENIGSEGG	- ARCADE																																																										
- LAMBORGHINI																																																											
- RANGEROVER																																																											
- MASERATI																																																											
- MERCEDES BENZ																																																											
- MCLARENCARS																																																											
- MITSUBISHI																																																											
- NISSAN																																																											
- PAGANI																																																											

Gambar 3.9.3 Hasil pembacaan file masukan (Puzzle)

Gambar 3.9.4 Hasil pembacaan file masukan (Kata kunci)

<pre> Keyword 'ACURA' found horizontally left! Comparison: 18 time(s) 42700 ns (0.043 ms) Keyword 'ALPINE' found diagonally! (left upper) Comparison: 1321 time(s) 462600 ns (0.463 ms) Keyword 'APOLLO' found vertically upper! Comparison: 971 time(s) 387900 ns (0.388 ms) Keyword 'ARIEL' found diagonally! (right upper) Comparison: 1264 time(s) 356900 ns (0.357 ms) Keyword 'ASTONMARTIN' found horizontally right! Comparison: 930 time(s) 457100 ns (0.457 ms) Keyword 'AUSTINHEALEY' found diagonally! (right lower) Comparison: 485 time(s) 309900 ns (0.310 ms) Keyword 'BENTLEY' found vertically lower! Comparison: 597 time(s) 287300 ns (0.287 ms) Keyword 'BUGATTI' found diagonally! (left lower) Comparison: 537 time(s) 281600 ns (0.282 ms) Keyword 'CADILLAC' found horizontally left! Comparison: 138 time(s) 55400 ns (0.055 ms) Keyword 'CATERHAM' found diagonally! (left upper) Comparison: 852 time(s) 109500 ns (0.110 ms) Keyword 'CHEVROLET' found vertically upper! Comparison: 589 time(s) 85300 ns (0.085 ms) Keyword 'DATSUN' found diagonally! (right upper) Comparison: 681 time(s) 111600 ns (0.112 ms) Keyword 'DEBERTI' found vertically lower! Comparison: 903 time(s) 132100 ns (0.132 ms) Keyword 'EXOMOTIVE' found diagonally! (right lower) Comparison: 748 time(s) 146600 ns (0.147 ms) Keyword 'FERRARI' found vertically lower! Comparison: 731 time(s) 181600 ns (0.182 ms) Keyword 'FORMULADRIFT' found diagonally! (left lower) Comparison: 76 time(s) 58700 ns (0.059 ms) Keyword 'FORSBERGRACING' found horizontally left! Comparison: 1184 time(s) 145400 ns (0.145 ms) </pre>	<pre> Keyword 'HENNESSEY' found diagonally! (left upper) Comparison: 351 time(s) 88400 ns (0.088 ms) Keyword 'HOONIGAN' found vertically upper! Comparison: 279 time(s) 67000 ns (0.067 ms) Keyword 'INFINITI' found diagonally! (right upper) Comparison: 348 time(s) 89900 ns (0.090 ms) Keyword 'KOENIGSEGG' found horizontally right! Comparison: 974 time(s) 158100 ns (0.158 ms) Keyword 'LAMBORGHINI' found diagonally! (right lower) Comparison: 889 time(s) 128800 ns (0.129 ms) Keyword 'RANGEROVER' found vertically lower! Comparison: 824 time(s) 100600 ns (0.101 ms) Keyword 'MASERATI' found vertically lower! Comparison: 96 time(s) 63800 ns (0.064 ms) Keyword 'MERCEDES BENZ' found horizontally left! Comparison: 1218 time(s) 233800 ns (0.234 ms) Keyword 'MCLAREN CARS' found diagonally! (left upper) Comparison: 474 time(s) 131800 ns (0.132 ms) Keyword 'MITSUBISHI' found horizontally right! Comparison: 1287 time(s) 162500 ns (0.163 ms) Keyword 'NISSAN' found diagonally! (left upper) Comparison: 1560 time(s) 172700 ns (0.173 ms) Keyword 'PAGANI' found horizontally right! Comparison: 725 time(s) 105400 ns (0.105 ms) Keyword 'PEEL' found diagonally! (right lower) Comparison: 347 time(s) 113300 ns (0.113 ms) Keyword 'PENHALL' found vertically lower! Comparison: 738 time(s) 157000 ns (0.157 ms) Keyword 'PEUGEOT' found horizontally right! Comparison: 1115 time(s) 170000 ns (0.170 ms) Keyword 'PLYMOUTH' found horizontally left! Comparison: 224 time(s) 85200 ns (0.085 ms) Keyword 'POLARIS' found diagonally! (left upper) Comparison: 909 time(s) 132300 ns (0.132 ms) </pre>
Gambar 3.9.5 Informasi hasil eksekusi per kata (1)	Gambar 3.9.6 Informasi hasil eksekusi per kata (2)
<pre> Keyword 'PONTIAC' found vertically upper! Comparison: 283 time(s) 76500 ns (0.077 ms) Keyword 'RELIANT' found diagonally! (right upper) Comparison: 1029 time(s) 175100 ns (0.175 ms) Keyword 'RENAULT' found horizontally right! Comparison: 522 time(s) 71500 ns (0.072 ms) Keyword 'RJANDERSON' found diagonally! (right lower) Comparison: 488 time(s) 136900 ns (0.137 ms) Keyword 'STERRACARSE' found vertically lower! Comparison: 907 time(s) 115800 ns (0.116 ms) Keyword 'SUBARU' found diagonally! (left lower) Comparison: 872 time(s) 127900 ns (0.128 ms) Keyword 'TOYOTA' found horizontally left! Comparison: 163 time(s) 65100 ns (0.065 ms) Keyword 'VAUXHALL' found diagonally! (left upper) Comparison: 384 time(s) 130400 ns (0.130 ms) Keyword 'VOLKSWAGEN' found vertically upper! Comparison: 1290 time(s) 115400 ns (0.115 ms) Keyword 'WILLYS' found diagonally! (right upper) Comparison: 747 time(s) 92100 ns (0.092 ms) Keyword 'GYMKHANA' found horizontally right! Comparison: 807 time(s) 160200 ns (0.160 ms) Keyword 'FORZATHON' found diagonally! (right lower) Comparison: 867 time(s) 132200 ns (0.132 ms) Keyword 'FESTIVAL' found vertically lower! Comparison: 321 time(s) 53300 ns (0.053 ms) Keyword 'HORIZON' found diagonally! (left lower) Comparison: 219 time(s) 55900 ns (0.056 ms) Keyword 'MEXICO' found horizontally left! Comparison: 460 time(s) 92100 ns (0.092 ms) Keyword 'ARCADE' found diagonally! (left upper) Comparison: 1355 time(s) 145900 ns (0.145 ms) </pre>	<pre> NSZNAARUCAXMMFMRYMCMNFZCPAYPDPRIA AVRIKAUAKNFUKFYEZQKZSAMTEDNKTM GQLASDKLMTJIOVQCSBMPKIPCALLIDAC IHJSCZUNLEYRCYBAVSDGATOYOTNAQSEY NRIVXNAGJAMFEMAXJDEXTNWXXIDEBERT ONIDCXHBHTUOMYLPFGNPOQKFSSHORZJ OTKICBBLZLXCRVXZBQLNPHNKCGLASX HEYNGXMAATVBUWMIZUGPEIDHRMMMTSI PLGPEIDXOLBXFAFDMVVTUKHXILVYDIS BONNERAMOCHEKVRAXONTCCQZOWAWRRRL RRTTLEOCXEHMSZGJUKQBHOJXGHIRGTZ AVKFFNLGVNLTABTVASRENAULTDTTOQEH METNMVTDUVJDIQNLINTNGMYKIPLPJPTN LHBDVILSHYVWCHMTMQDIVSNNUZXHWYS NCAAWITPSTIAQOFAIDAEENAPFBZOLKYH EYXHPAVGNWNOLLCLSSUHYRHOBELHMLAA GQFPDDADDFWUFLPVMGILRFSEBNQELPSY EHRGKTPAGANIPOSIDKDRESOATOIFEOT QXCJTLYCWFWRBEPASTONMARTINLWTRRW LJOIJXBWEBCINAIIZXTDQLYAZEEAARU OAMKMPXBRAMUHGYKXHANAROCYTYING ETMLOKJLRYLUAJYLDSESEZYUPBIMLQGH FNFBMWTEHAINSLETPEMDPLBVENXHEOEPE YCIOOIITRCCLHBJBUAXADDDCERFTZRE JGKPRRUUVIEMERCMSSBGDBGRORR UJIALZGEEVOGJTPJRKUGNMBTAVLVSVP VVLNAAHSPUEGEOTTUVPUIAJNPPVEEUC DBOXHSDTIHNRADIRKAGVLYSXQARRCA HIUXKGEWNNMELPEZSOBVFPIYKCNHYKJR DLGHBZDAXOIZNEBSEDECEMFLUMAXBWS MKVSUGLTONHGNICARGREBSROFYDSICE MITSUBISHISFQEHUWRPDXUPVJTJRSPO RIPASDETERAUTLINTSIDNEOASDHDASKIA FIASNIHSSKTHEFGHTASDFHEQSFHDSAWN Comparison time in total (parsing excluded): 7518200 ns (7.518 ms) Comparison count in total: 35017 time(s) Press any key to quit... </pre>
Gambar 3.9.7 Informasi hasil eksekusi per kata (3)	Gambar 3.9.8 Matriks akhir dan informasi waktu eksekusi & banyak perbandingan

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	√	
2. Program berhasil running	√	
3. Program dapat membaca file masukan dan menuliskan luaran	√	
4. Program berhasil menemukan semua kata di dalam puzzle	√	