

## **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 .....	5
2.4 Matcher.java .....	6
2.5 Matrix.java .....	12
2.6 SearchWord.java .....	13
2.7 ReadFile.java.....	16
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
IV. <i>REPOSITORY</i> .....	28

## 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.

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, program akan memanggil fungsi-fungsi pencocokan (terletak dalam file `Matcher.java`). 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.

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.

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.

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. Proses ini berulang hingga semua kata kunci telah ditelusuri. 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.

*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);
            }

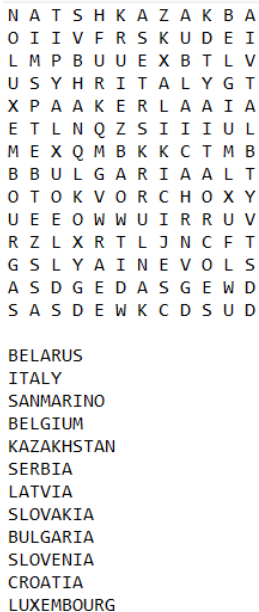
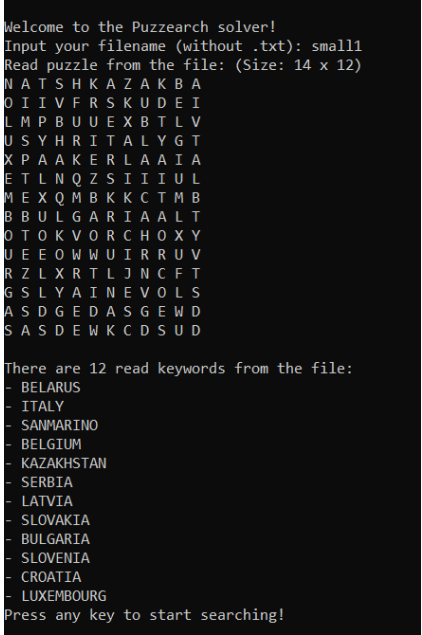
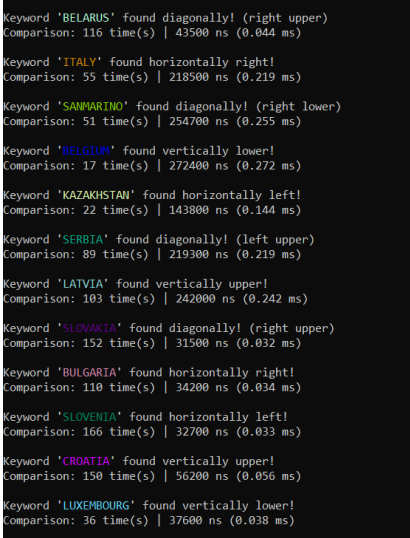
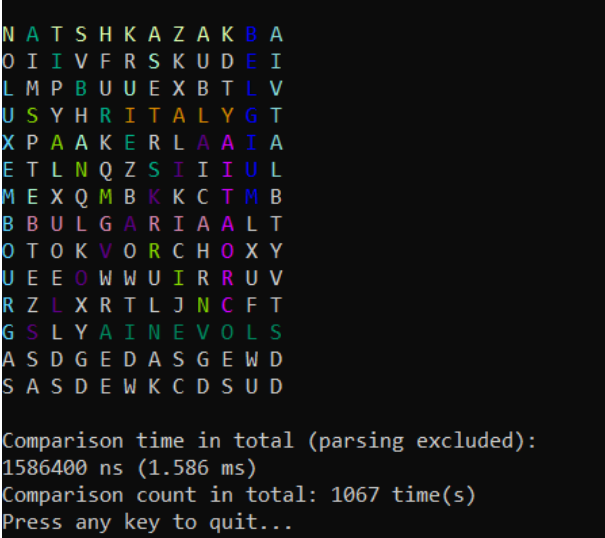
        } finally {
            lineReader.close();
        }

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

### BAB III. PENGUJIAN (INPUT/OUTPUT)

Pengujian dilakukan di *operating system* Windows 10, dijalankan di terminal *Command Prompt* menggunakan *batch* file dengan nilai *Registry Editor* yang telah diubah (selengkapnya di berkas README.md)

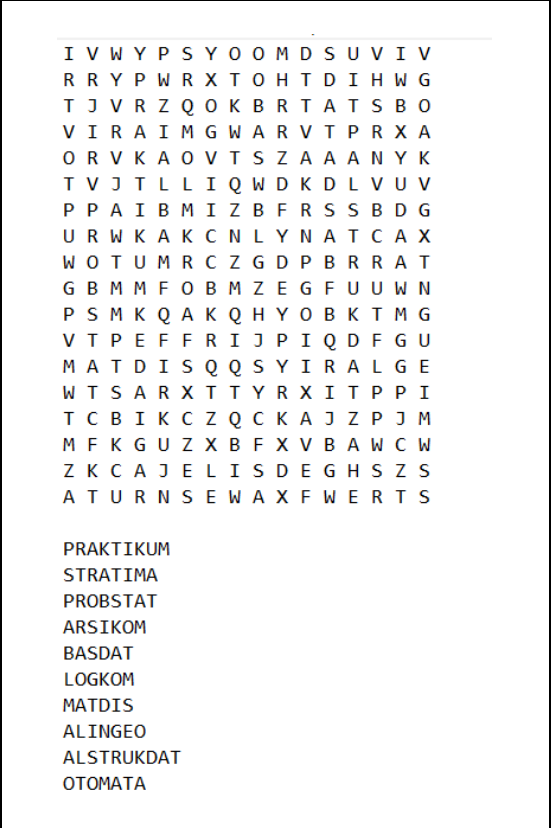
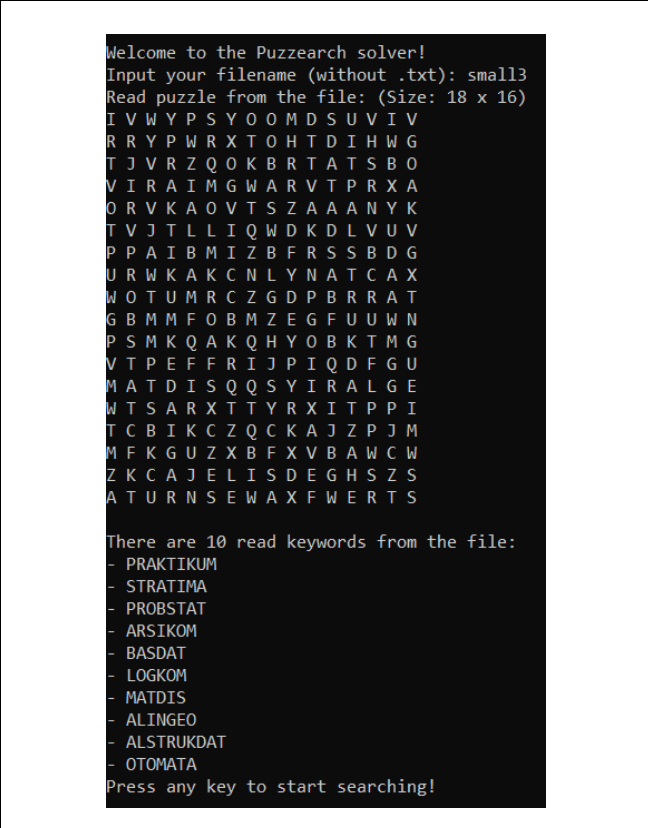
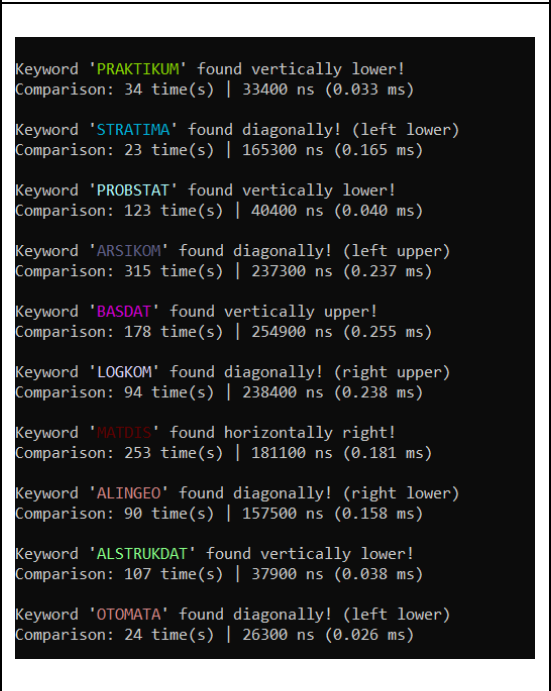
#### 3.1 Pengujian terhadap file small1.txt (ukuran 14 x 12)

	
<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 &amp; banyak perbandingan</p>

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

 <pre> J U C M I R A U R B E F L N P A N V D D P L R F T I O L Y K N A G U S T U S P V U Q S Y L U V J Q K Z I E J K X A E T F A L S V Z M V S C C N M P J K R E Z B I B I U F W X L T M Q I E Z N U D O G F H X A E X R V J U L I K A M V R Z N M Q X L J K K T X K E E D M A B S Y X D E O K T I B N U J Y E Y V O T B D M U E C F I Q R R N O F E Y R E B M E S E D M L Y D R K P P R E S D K C Q E D L S I Z O W K L D L I R P A X B K A S D U D S A J K S L O E S  JANUARI FEBRUARI MARET APRIL MEI JUNI JULI AGUSTUS SEPTEMBER OKTOBER NOVEMBER DESEMBER         </pre>	 <pre> Welcome to the Puzzeearch solver! Input your filename (without .txt): small2 Read puzzle from the file: (Size: 16 x 14) J U C M I R A U R B E F L N P A N V D D P L R F T I O L Y K N A G U S T U S P V U Q S Y L U V J Q K Z I E J K X A E T F A L S V Z M V S C C N M P J K R E Z B I B I U F W X L T M Q I E Z N U D O G F H X A E X R V J U L I K A M V R Z N M Q X L J K K T X K E E D M A B S Y X D E O K T I B N U J Y E Y V O T B D M U E C F I Q R R N O F E Y R E B M E S E D M L Y D R K P P R E S D K C Q E D L S I Z O W K L D L I R P A X B K A S D U D S A J K S L O E S  There are 12 read keywords from the file: - JANUARI - FEBRUARI - MARET - APRIL - MEI - JUNI - JULI - AGUSTUS - SEPTEMBER - OKTOBER - NOVEMBER - DESEMBER Press any key to start searching!         </pre>
<p>Gambar 3.2.1 Isian berkas small2.txt</p>	<p>Gambar 3.2.2 Hasil pembacaan file masukan</p>
 <pre> Keyword 'JANUARI' found diagonally! (right lower) Comparison: 8 time(s)   24100 ns (0.024 ms)  Keyword 'FEBRUARI' found horizontally left! Comparison: 21 time(s)   163800 ns (0.164 ms)  Keyword 'MARET' found diagonally! (left lower) Comparison: 112 time(s)   167600 ns (0.168 ms)  Keyword 'APRIL' found horizontally left! Comparison: 254 time(s)   46300 ns (0.046 ms)  Keyword 'MEI' found diagonally! (left upper) Comparison: 239 time(s)   160900 ns (0.161 ms)  Keyword 'JUNI' found vertically upper! Comparison: 166 time(s)   308800 ns (0.309 ms)  Keyword 'JULI' found horizontally right! Comparison: 136 time(s)   265300 ns (0.265 ms)  Keyword 'AGUSTUS' found horizontally right! Comparison: 46 time(s)   31100 ns (0.031 ms)  Keyword 'SEPTEMBER' found diagonally! (right lower) Comparison: 56 time(s)   31400 ns (0.031 ms)  Keyword 'OKTOBER' found vertically lower! Comparison: 106 time(s)   156600 ns (0.157 ms)  Keyword 'NOVEMBER' found diagonally! (left lower) Comparison: 22 time(s)   25800 ns (0.026 ms)  Keyword 'DESEMBER' found horizontally left! Comparison: 201 time(s)   32600 ns (0.033 ms)         </pre>	 <pre> J U C M I R A U R B E F L N P A N V D D P L R F T I O L Y K N A G U S T U S P V U Q S Y L U V J Q K Z I E J K X A E T F A L S V Z M V S C C N M P J K R E Z B I B I U F W X L T M Q I E Z N U D O G F H X A E X R V J U L I K A M V R Z N M Q X L J K K T X K E E D M A B S Y X D E O K T I B N U J Y E Y V O T B D M U E C F I Q R R N O F E Y R E B M E S E D M L Y D R K P P R E S D K C Q E D L S I Z O W K L D L I R P A X B K A S D U D S A J K S L O E S  Comparison time in total (parsing excluded): 1414300 ns (1.414 ms) Comparison count in total: 1367 time(s) Press any key to quit...         </pre>
<p>Gambar 3.2.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.2.4 Matriks akhir dan informasi waktu eksekusi &amp; banyak perbandingan</p>


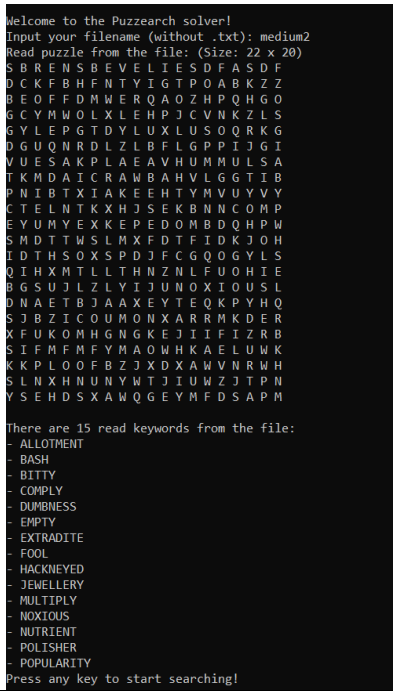
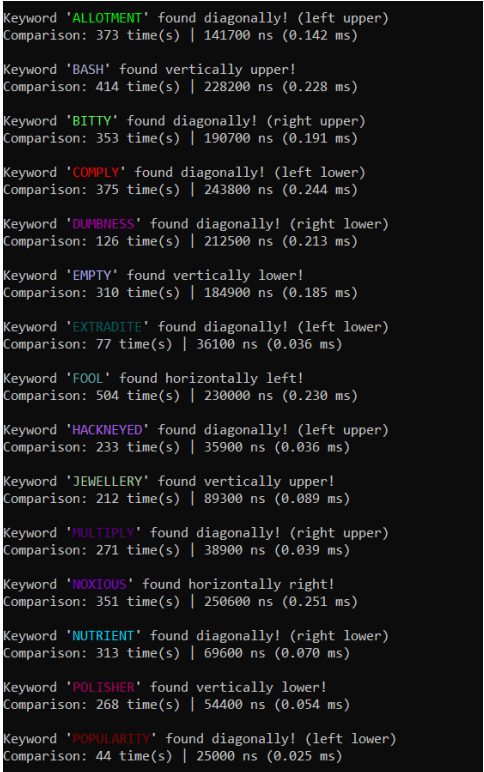
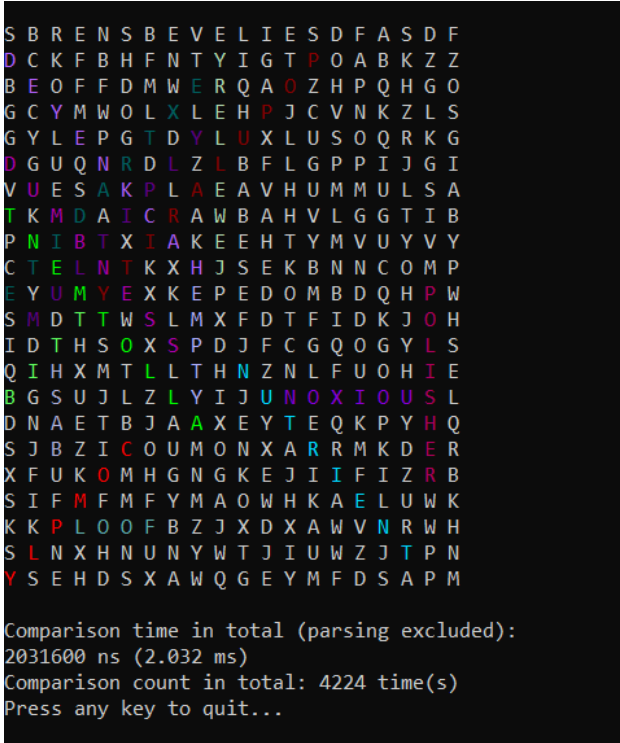
### 3.3 Pengujian terhadap file small3.txt (ukuran 14 x 12)

	
<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 &amp; 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 &amp; banyak perbandingan</p>

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

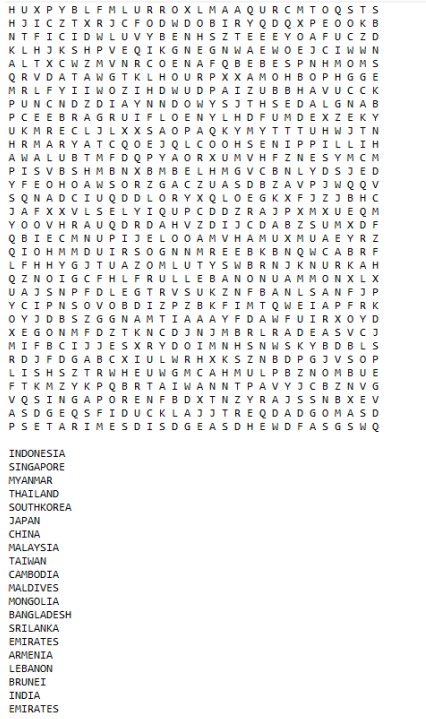
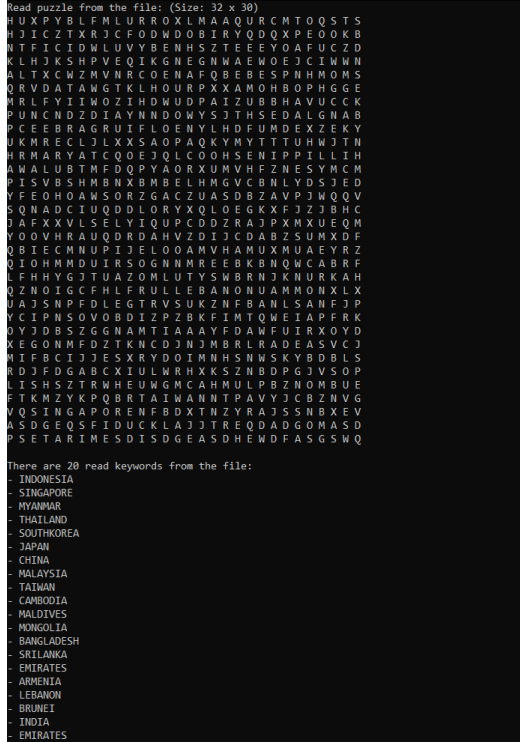
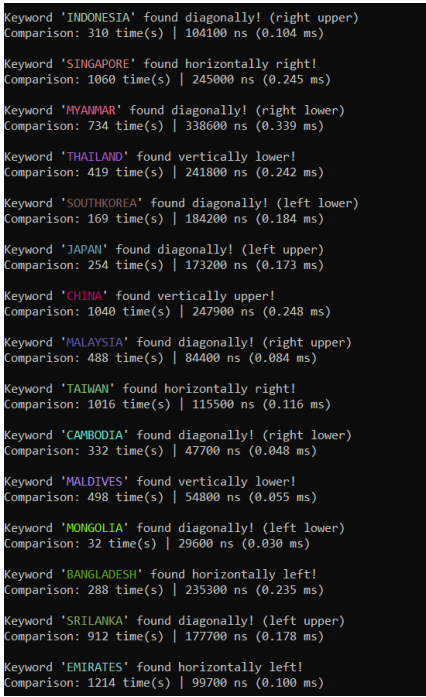

	
<p>Gambar 3.5.1 Isian berkas medium1.txt</p>	<p>Gambar 3.5.2 Hasil pembacaan file masukan</p>
	
<p>Gambar 3.5.3 Informasi hasil eksekusi per kata</p>	<p>Gambar 3.5.4 Matriks akhir dan informasi waktu eksekusi &amp; banyak perbandingan</p>

### 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 &amp; banyak perbandingan</p>

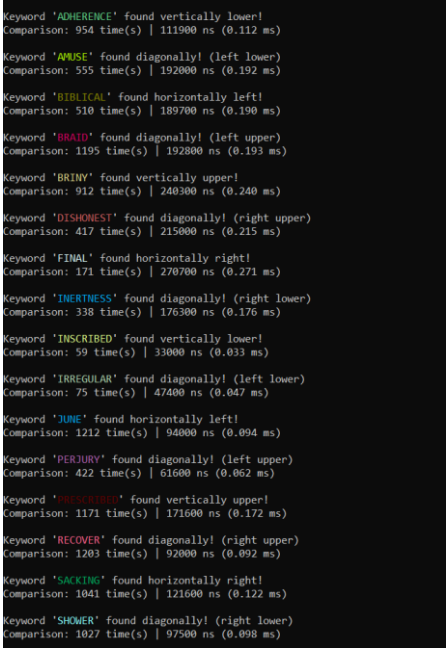
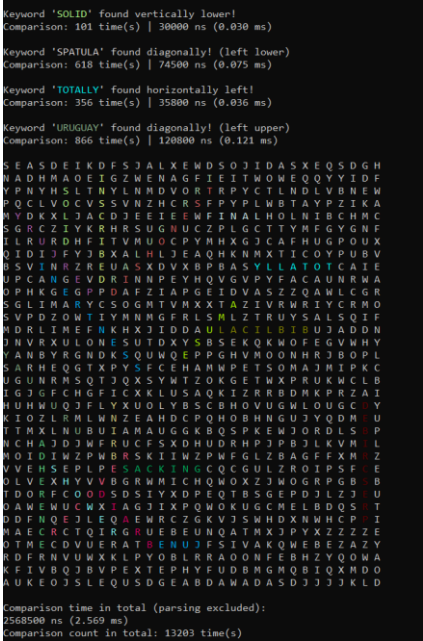


## 3.7 Pengujian terhadap file large1.txt (ukuran 32 x 30)

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



### 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 &amp; banyak perbandingan</p>

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

<pre> NSZNRUCAXWMFMYRMCNFCZCPAYPDPRIA AYRIKAUAKNFUKFYEZQKZSAMTEDNKTMPY GQLASDKLMTJIOVQCSBMPKIPCALLIDACZ IHJSCZUNLEYRCYBAVSDGATOTYOTNAQSEY NRIVXNAGJAMFEMAXJDEXTNWWXIDEBERT ONIDCXEHBUHTUOMYLPGNPOQKFSSHORZJ OTKIBCBRLZLXCRVXZBQLNPHNKKCOGLASX HEYNGXMAATVBWUMIZUWGPEIDHRWMMTSI PLGPEIDXOLBXXFAFDMVVUTUKHXILVYDISP BONNERAMOQCHEKVRAOXNTCQZOZWAWRRL QRTTIEOCIXEMSZGJJUKQBHOJXGMIRGTZ AVKFFNLGVNZLTABTVASRENAULTDQOQEM METNNVTDUVJDIOQNLINTNGMYKIPLJPTM LHBDVILSHBYWVCHMTMQDIVSWNUZXHWYS NCAAWITPUSIQAOFAIDAEOENAPFBZOLKYH EYXHPAVGNWNOLLCLSSUHYRHOBELHWLAA GQFPDDADDWUFLPVMGILRFSEBNQELPSY EHRGKTPAGANIPOSIDIKDRQESOATOIFEOT QXCJTLYCFWRBEPASTONMARTINLWTRRNW LJOIJXBWBCINAIZJXTDQLYAZEEAQRU OQAMKXPBRAWUHGYMKHANAROSCYTYINGS ETMLOKJLRYLUAJYLDSESEZYUPBIMLQGGHI FNFBWTEHAINSLPEMDPLBVENXEHOEPE YCIIOOIITRCCNLHBJBXAADDCEFTZRER JGKPRRUVIKOENIGSEGGRCMSBGDBGROFR WJJALZGEEVOGJTPJRKUGNRBTAVLYSVPA YVILNAAHSPUEGEOTTUVPUIAJWPPVEEUC DBOXHSDTIHNRADIIIRKAGVLYSXQARRCA HIUXKGWENHWELPEZSOBVFPIYKCNHYKJR DLGHBZDAXOIZNEBSEDECREMYLUMAXBWS MKVSQUGLTQNHGNICARGREBSROFYDSICE MITSUBISHISFQEHUWRPDVXUPVTJJRSPD RIPASDETERAUTLINTSIDNEOASDHDSKIA FIASNIHSSKTHEFGHTASDFHEQSFHDSAWN </pre>	<div> <div>ACURA</div> <div>ALPINE</div> <div>APOLLO</div> <div>ARIEL</div> <div>ASTONMARTIN</div> <div>AUSTINHEALEY</div> <div>BENTLEY</div> <div>BUGATTI</div> <div>CADILLAC</div> <div>CATERHAM</div> <div>CHEVROLET</div> <div>DATSUN</div> <div>DEBERTI</div> <div>EXOMOTIVE</div> <div>FERRARI</div> <div>FORMULADRIFT</div> <div>FORSBERGRACING</div> <div>HENNESSEY</div> <div>HOONIGAN</div> <div>INFINITI</div> <div>KOENIGSEGG</div> <div>LAMBORGHINI</div> <div>RANGEROVER</div> <div>MASERATI</div> <div>MERCEDESSENZ</div> <div>MCLARENCARS</div> <div>MITSUBISHI</div> <div>NISSAN</div> <div>PAGANI</div> </div> <div> <div>PEEL</div> <div>PENHALL</div> <div>PEUGEOT</div> <div>PLYMOUTH</div> <div>POLARIS</div> <div>PONTIAC</div> <div>RELIANT</div> <div>RENAULT</div> <div>RJANDERSON</div> <div>SIERRACARSE</div> <div>SUBARU</div> <div>TOYOTA</div> <div>VAUXHALL</div> <div>VOLKSWAGEN</div> <div>WILLYS</div> <div>GYMKHANA</div> <div>FORZATHON</div> <div>FESTIVAL</div> <div>HORIZON</div> <div>MEXICO</div> <div>ARCADE</div> </div>
---	--

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) NSZNRUCAXWMFMYRMCNFCZCPAYPDPRIA AYRIKAUAKNFUKFYEZQKZSAMTEDNKTMPY GQLASDKLMTJIOVQCSBMPKIPCALLIDACZ IHJSCZUNLEYRCYBAVSDGATOTYOTNAQSEY NRIVXNAGJAMFEMAXJDEXTNWWXIDEBERT ONIDCXEHBUHTUOMYLPGNPOQKFSSHORZJ OTKIBCBRLZLXCRVXZBQLNPHNKKCOGLASX HEYNGXMAATVBWUMIZUWGPEIDHRWMMTSI PLGPEIDXOLBXXFAFDMVVUTUKHXILVYDISP BONNERAMOQCHEKVRAOXNTCQZOZWAWRRL QRTTIEOCIXEMSZGJJUKQBHOJXGMIRGTZ AVKFFNLGVNZLTABTVASRENAULTDQOQEM METNNVTDUVJDIOQNLINTNGMYKIPLJPTM LHBDVILSHBYWVCHMTMQDIVSWNUZXHWYS NCAAWITPUSIQAOFAIDAEOENAPFBZOLKYH EYXHPAVGNWNOLLCLSSUHYRHOBELHWLAA GQFPDDADDWUFLPVMGILRFSEBNQELPSY EHRGKTPAGANIPOSIDIKDRQESOATOIFEOT QXCJTLYCFWRBEPASTONMARTINLWTRRNW LJOIJXBWBCINAIZJXTDQLYAZEEAQRU OQAMKXPBRAWUHGYMKHANAROSCYTYINGS ETMLOKJLRYLUAJYLDSESEZYUPBIMLQGGHI FNFBWTEHAINSLPEMDPLBVENXEHOEPE YCIIOOIITRCCNLHBJBXAADDCEFTZRER JGKPRRUVIKOENIGSEGGRCMSBGDBGROFR WJJALZGEEVOGJTPJRKUGNRBTAVLYSVPA YVILNAAHSPUEGEOTTUVPUIAJWPPVEEUC DBOXHSDTIHNRADIIIRKAGVLYSXQARRCA HIUXKGWENHWELPEZSOBVFPIYKCNHYKJR DLGHBZDAXOIZNEBSEDECREMYLUMAXBWS MKVSQUGLTQNHGNICARGREBSROFYDSICE MITSUBISHISFQEHUWRPDVXUPVTJJRSPD RIPASDETERAUTLINTSIDNEOASDHDSKIA FIASNIHSSKTHEFGHTASDFHEQSFHDSAWN </pre>	<div> <div>ACURA</div> <div>ALPINE</div> <div>APOLLO</div> <div>ARIEL</div> <div>ASTONMARTIN</div> <div>AUSTINHEALEY</div> <div>BENTLEY</div> <div>BUGATTI</div> <div>CADILLAC</div> <div>CATERHAM</div> <div>CHEVROLET</div> <div>DATSUN</div> <div>DEBERTI</div> <div>EXOMOTIVE</div> <div>FERRARI</div> <div>FORMULADRIFT</div> <div>FORSBERGRACING</div> <div>HENNESSEY</div> <div>HOONIGAN</div> <div>INFINITI</div> <div>KOENIGSEGG</div> <div>LAMBORGHINI</div> <div>RANGEROVER</div> <div>MASERATI</div> <div>MERCEDESSENZ</div> <div>MCLARENCARS</div> <div>MITSUBISHI</div> <div>NISSAN</div> <div>PAGANI</div> </div> <div> <div>PEEL</div> <div>PENHALL</div> <div>PEUGEOT</div> <div>PLYMOUTH</div> <div>POLARIS</div> <div>PONTIAC</div> <div>RELIANT</div> <div>RENAULT</div> <div>RJANDERSON</div> <div>SIERRACARS</div> <div>SUBARU</div> <div>TOYOTA</div> <div>VAUXHALL</div> <div>VOLKSWAGEN</div> <div>WILLYS</div> <div>GYMKHANA</div> <div>FORZATHON</div> <div>FESTIVAL</div> <div>HORIZON</div> <div>MEXICO</div> <div>ARCADE</div> </div>
---	---

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: 22 time(s)   23900 ns (0.024 ms)  Keyword 'ALPINE' found diagonally! (left upper) Comparison: 1235 time(s)   341000 ns (0.341 ms)  Keyword 'APOLLO' found vertically upper! Comparison: 903 time(s)   248900 ns (0.249 ms)  Keyword 'ARIEL' found diagonally! (right upper) Comparison: 1205 time(s)   218200 ns (0.218 ms)  Keyword 'ASTONMARTIN' found horizontally right! Comparison: 796 time(s)   388000 ns (0.388 ms)  Keyword 'AUSTINHEALEY' found diagonally! (right lower) Comparison: 393 time(s)   303200 ns (0.303 ms)  Keyword 'BENTLEY' found vertically lower! Comparison: 575 time(s)   295300 ns (0.295 ms)  Keyword 'BUGATTI' found diagonally! (left lower) Comparison: 519 time(s)   157600 ns (0.158 ms)  Keyword 'CADILLAC' found horizontally left! Comparison: 134 time(s)   31400 ns (0.031 ms)  Keyword 'CATERHAM' found diagonally! (left upper) Comparison: 806 time(s)   65600 ns (0.066 ms)  Keyword 'CHEVROLET' found vertically upper! Comparison: 544 time(s)   88400 ns (0.088 ms)  Keyword 'DATSUN' found diagonally! (right upper) Comparison: 647 time(s)   85800 ns (0.086 ms)  Keyword 'DEBERTI' found vertically lower! Comparison: 853 time(s)   77700 ns (0.078 ms)  Keyword 'EXOMOTIVE' found diagonally! (right lower) Comparison: 681 time(s)   57500 ns (0.058 ms)  Keyword 'FERRARI' found vertically lower! Comparison: 706 time(s)   53200 ns (0.053 ms)  Keyword 'FORMULADRIFT' found diagonally! (left lower) Comparison: 73 time(s)   42500 ns (0.043 ms)</pre>	<pre>Keyword 'FORSBERGRACING' found horizontally left! Comparison: 1098 time(s)   135600 ns (0.136 ms)  Keyword 'HENNESSEY' found diagonally! (left upper) Comparison: 319 time(s)   41600 ns (0.042 ms)  Keyword 'HOONIGAN' found vertically upper! Comparison: 257 time(s)   56500 ns (0.057 ms)  Keyword 'INFINITI' found diagonally! (right upper) Comparison: 306 time(s)   62000 ns (0.062 ms)  Keyword 'KOENIGSEGG' found horizontally right! Comparison: 900 time(s)   78500 ns (0.079 ms)  Keyword 'LAMBORGHINI' found diagonally! (right lower) Comparison: 789 time(s)   69900 ns (0.070 ms)  Keyword 'RANGEROVER' found vertically lower! Comparison: 758 time(s)   60400 ns (0.060 ms)  Keyword 'MASERATI' found vertically lower! Comparison: 91 time(s)   44100 ns (0.044 ms)  Keyword 'MERCEDESSENZ' found horizontally left! Comparison: 1130 time(s)   91600 ns (0.092 ms)  Keyword 'MCLAREN CARS' found diagonally! (left upper) Comparison: 415 time(s)   99600 ns (0.100 ms)  Keyword 'MITSUBISHI' found horizontally right! Comparison: 1202 time(s)   110400 ns (0.110 ms)  Keyword 'NISSAN' found diagonally! (left upper) Comparison: 1479 time(s)   100600 ns (0.101 ms)  Keyword 'PAGANI' found horizontally right! Comparison: 687 time(s)   42200 ns (0.042 ms)  Keyword 'PEEL' found diagonally! (right lower) Comparison: 335 time(s)   28800 ns (0.029 ms)  Keyword 'PENHALL' found vertically lower! Comparison: 696 time(s)   48700 ns (0.049 ms)  Keyword 'PEUGEOT' found horizontally right! Comparison: 1056 time(s)   77200 ns (0.077 ms)</pre>
---	---

Gambar 3.9.5 Informasi hasil eksekusi per kata

(1)

Gambar 3.9.6 Informasi hasil eksekusi per kata (2)

<pre>Keyword 'PLYMOUTH' found horizontally left! Comparison: 213 time(s)   62600 ns (0.063 ms)  Keyword 'POLARIS' found diagonally! (left upper) Comparison: 862 time(s)   63500 ns (0.064 ms)  Keyword 'PONTIAC' found vertically upper! Comparison: 264 time(s)   30600 ns (0.031 ms)  Keyword 'RELIANT' found diagonally! (right upper) Comparison: 972 time(s)   60900 ns (0.061 ms)  Keyword 'RENAULT' found horizontally right! Comparison: 470 time(s)   35000 ns (0.035 ms)  Keyword 'RJANDERSON' found diagonally! (right lower) Comparison: 358 time(s)   129300 ns (0.129 ms)  Keyword 'SERRACARSE' found vertically lower! Comparison: 822 time(s)   65500 ns (0.066 ms)  Keyword 'SUBARU' found diagonally! (left lower) Comparison: 833 time(s)   55800 ns (0.056 ms)  Keyword 'TOYOTA' found horizontally left! Comparison: 153 time(s)   18900 ns (0.019 ms)  Keyword 'VAUXHALL' found diagonally! (left upper) Comparison: 364 time(s)   35800 ns (0.036 ms)  Keyword 'VOLKSWAGEN' found vertically upper! Comparison: 1198 time(s)   65800 ns (0.066 ms)  Keyword 'WILLYS' found diagonally! (right upper) Comparison: 730 time(s)   38300 ns (0.038 ms)  Keyword 'YONKAMA' found horizontally right! Comparison: 770 time(s)   54800 ns (0.055 ms)  Keyword 'FORZATHON' found diagonally! (right lower) Comparison: 836 time(s)   48100 ns (0.048 ms)  Keyword 'FESTIVAL' found vertically lower! Comparison: 309 time(s)   21400 ns (0.021 ms)  Keyword 'HORIZON' found diagonally! (left lower) Comparison: 208 time(s)   41600 ns (0.042 ms)</pre>	<pre>Keyword 'ARCADE' found diagonally! (left upper) Comparison: 1268 time(s)   89400 ns (0.089 ms)  N S Z N A R U C A X W M F Y M R M C N F Z C P A Y P D P R I A J A Y R I K A U A K N F U K Y E Z Q K Z S A M T E D N K T M P Y G Q L A S D K L M T J I O V Q C S B M P K I P C A L L I D A C Z I H J S C Z U N L E Y R C Y B A V S D G A T O Y O T N A Q S E Y N R I V X N A G J A H F E M A X J D E X T N W W X I D E B E R T O N I D C X E H B U H T U O M Y L P G N P O Q K F S S H O R Z J O T K I B C B R L Z L X C R V X Z B Q L N P H N K C O G L A S X H E Y N G X M A A T V B U W M I Z U W G P E I D H R W M M T S I P L G F E I D X O L B X F A F D M V V T U K H X I L Y D I S P B O N N E R A M O Q C H E K V R A O X N T C Q Z O Z W A W R R L Q R T T I E O C I X E M S Z G J J U K Q B H O J X G M I R G T Z A V K F F N L G V N Z L T A B T V A S R E N A U L T D T O Q E M H E N N V T D U V J D I O Q N L I N T N G M V K I P L J P T M L H B D V I L S H Y W V C H M T O D I V S W N U Z X H Y S N C A A W I T P U S I Q A O F A I D O E N A P F B Z O L K Y H E Y X H P A V G N W H O L L C L S S U Y Y R H O B E L H W L A A G Q F P D D A D D F W U L P V M G I L F S E B N O E L P S Y E H R G K T P A G A N I P O S D I K D R Q S O A T O T F E O T Q X C J L Y C F W R B E P A S T O N M M A R I N L W T R R N W L J O J J X B W E B C T N A I Z J X T D Q L Y A Z E E A Q A R U O A Q M K P X B R A W U H G Y M K H A N A R O S C Y T V I N G S E T M L O K J L R Y L U A J Y L D E S E Z Y U P B I M L Q G H I F N F B W T E H A I N S L E T P E M D P L B V E N X E H O E P E Y C I O O I T R C C N L H B J B U X A A D D C E R F T Z R E R J G K P R R U V I K D E H I G S E G I R C M S B G D B G R O F R W J J A L Z G E E V O G J T P J R K U G N R B T A V L Y S V P A Y V I L N A A H S P E U G E O T T U V P U I A J W P P V E U C D B O X H S D T I H N O R A D I R K A G V L Y S X Q A R R C A H I U X K G E W H N W E L P E Z S O B V F I Y K C N H Y K J R D L G H B Z D A X O I Z N E B S E D E C R E M Y L U M A X B W S M K V S Q U G L T Q N H G N I C A R G R E B S R O F Y D S I C E M I T S U B I S H I S F Q E H U W R P D V X U P V T J J R S P D R I P A S D E T E R A U T L I N T S I D N E O A S D H D S K I A F I A S N I H S S K T H E F G H T A S D F H E Q S F H D S A W N  Comparison time in total (parsing excluded): 4586600 ns (4.587 ms) Comparison count in total: 32677 time(s)</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 keluaran	√	
4. Program berhasil menemukan semua kata di dalam puzzle	√	

#### **IV. REPOSITORY**

*Repository* dapat diakses via <https://github.com/clumsyyyy/TucilStima1> (branch main)