LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2021/2022

Adelline Kania Setiyawan / 13520084 / K-3

I. ALGORITMA *BRUTE FORCE*

1. Untuk menyelesaikan permasalahan *Word Search Puzzle*, input file dari pengguna berupa huruf-huruf dalam puzzle akan dimasukkan kedalam sebuah matriks statis wordsMatrix dengan jumlah baris dan kolom menyesuaikan dengan ukuran puzzle yang diterima serta input berupa kata yang dicari akan dimasukkan dalam satu array statis words dengan ukuran sesuai jumlah kata yang akan dicari dalam puzzle.
2. Setelah sudah terbentuk matriks dan array, program akan mulai mencari kata-kata dalam array words pada puzzle yang sudah berbentuk matriks, dimulai dari kata pertama dalam array. Program akan mencari apakah huruf pertama dalam kata yang sedang dicari sesuai dengan huruf pada sel matriks. Apabila tidak sesuai, program akan mencari terus huruf pada matriks yang sesuai dengan mengiterasi setiap sel matriks satu per satu.
3. Apabila pada suatu sel matriks hurufnya sesuai dengan huruf pertama pada kata yang sedang dicari. Program akan mengecek apakah ukuran panjang kata yang sedang dicari tidak melebihi ukuran matriks dari titik sel yang sedang diiterasi ke arah atas, bawah, kanan, kiri, serong kanan atas, serong kanan bawah, serong kiri atas, atau serong kiri bawah. Apabila ukuran panjang kata melebihi, maka program akan kembali mengiterasi sel matriks selanjutnya.
4. Apabila panjang kata tidak melebihi, program akan mulai mengiterasi huruf-huruf pada kata yang sedang dicari dan membandingkannya dengan huruf-huruf pada sel matriks yang bersangkutan. Apabila terdapat huruf yang tidak sesuai, program akan kembali mengiterasi sel matriks selanjutnya (mengulang pencarian ke sel selanjutnya, dimulai dari huruf pertama kata pertama). Sedangkan apabila semua huruf sesuai dengan kata yang sedang dicari, program akan memberikan output berupa lokasi kata-kata tersebut ditemukan.
5. Program akan terus melakukan iterasi sampai semua huruf pada puzzle di iterasi dan menemukan semua posisi kata yang sedang dicari pada puzzle.
6. Setelah kata pertama sudah selesai, program akan lanjut mengiterasi dan mencari kata selanjutnya pada puzzle sampai semua kata pada array words selesai diiterasi/ditemukan.

II. *SOURCE PROGRAM*

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

public class Main {
    public static void main(String[] args) throws FileNotFoundException {
        // Meminta input file path kepada user
        Scanner scInput = new Scanner(System.in);
        System.out.print("Enter File Path Location: ");
        String filePath = scInput.nextLine();
        scInput.close();
        // Membaca file dan menyimpan data pada matrix wordsMatrix dan array words
        File myFile = new File(filePath);
        Scanner sc1 = new Scanner(myFile);
        Integer n = 0;
        String data = "P";
        // Menghitung jumlah huruf dan kolom pada Word Search Puzzle
        while (sc1.hasNext() && data.length() == 1) {
            data = sc1.next();
            n++;
        }
        sc1.close();
        n--;
        Integer rows = 1;
        Scanner sc2 = new Scanner(myFile);
        while (!sc2.nextLine().isEmpty()) {
            rows++;
        }
        rows--;
        Integer nWord = 0;
        while (sc2.hasNextLine()) {
            nWord++;
            sc2.nextLine();
        }
        sc2.close();
        Integer cols = n / rows;
        String wordsMatrix[][] = new String[rows][cols];
        String words[] = new String[nWord];
        Scanner sc3 = new Scanner(myFile);
        for (int i = 0; i < wordsMatrix.length; i++) {
            for (int j = 0; j < wordsMatrix[i].length; j++) {
                wordsMatrix[i][j] = sc3.next();
            }
        }
        Integer p = 0;
```

```
        while (sc3.hasNextLine()) {
            String newWord = sc3.nextLine();
            if (!newWord.isEmpty()) {
                words[p] = newWord;
                p++;
            }
        }
        sc3.close();
        Boolean isEqual;
        Integer nComparison;
        long startTime = System.nanoTime();
        for (int i = 0; i < words.length; i++) {
            nComparison = 0;
            System.out.println(words[i]);
            isEqual = false;
            String currentChar = String.valueOf(words[i].charAt(0));
            Integer currentCharLen = words[i].length();
            SearchWord(wordsMatrix, rows, cols, currentChar, currentCharLen, isEqual, words, i, nComparison);
            System.out.println(" ");
        }
        long stopTime = System.nanoTime();
        double exTime = (double) (stopTime - startTime) / 1_000_000_000;
        System.out.println(
                "Waktu eksekusi: " + exTime + " sekon");
    }
    public static void SearchWord(String[][] wordsMatrix, Integer rows, Integer cols, String currentChar,
            Integer currentCharLen, boolean isEqual, String[] words, Integer q, Integer nComparison) {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                nComparison++;
                if (wordsMatrix[i][j].equals(currentChar)) {
                    // Mengecek ke arah kanan
                    if ((j + currentCharLen) <= cols) {
                        isEqual = true;
                        int k = 1;
                        int j2 = j + 1;
                        while (k < currentCharLen && isEqual) {
                            if (!wordsMatrix[i][j2].equals(String.valueOf(words[q].charAt(k)))) {
                                isEqual = false;
                            }
                            nComparison++;
                            k++;
                            j2++;
                        }
                        if (isEqual) {
                            printFoundMatrix(wordsMatrix, i, i, j, j + currentCharLen - 1, 0);
                            System.out.println("Jumlah operasi perbandingan : " + nComparison);
```

```
            }
        }
        // Mengecek ke arah kiri
        if ((j - currentCharLen >= -1)) {
            isEqual = true;
            int k = 1;
            int j2 = j - 1;
            while (k < currentCharLen && isEqual) {
                if (!wordsMatrix[i][j2].equals(String.valueOf(words[q].charAt(k)))) {
                    isEqual = false;
                }
                nComparison++;
                k++;
                j2--;
            }
            if (isEqual) {
                printFoundMatrix(wordsMatrix, i, i, j - currentCharLen + 1, j, 0);
                System.out.println("Jumlah operasi perbandingan : " + nComparison);
            }
        }
        // Mengecek ke arah bawah
        if ((i + currentCharLen) <= rows) {
            isEqual = true;
            int k = 1;
            int i2 = i + 1;
            while (k < currentCharLen && isEqual) {
                if (!wordsMatrix[i2][j].equals(String.valueOf(words[q].charAt(k)))) {
                    isEqual = false;
                }
                nComparison++;
                k++;
                i2++;
            }
            if (isEqual) {
                printFoundMatrix(wordsMatrix, i, i + currentCharLen - 1, j, j, 0);
                System.out.println("Jumlah operasi perbandingan : " + nComparison);
            }
        }
        // Mengecek ke arah atas
        if ((i - currentCharLen) >= -1) {
            isEqual = true;
            int k = 1;
            int i2 = i - 1;
            while (k < currentCharLen && isEqual) {
                if (!wordsMatrix[i2][j].equals(String.valueOf(words[q].charAt(k)))) {
                    isEqual = false;
                }
```

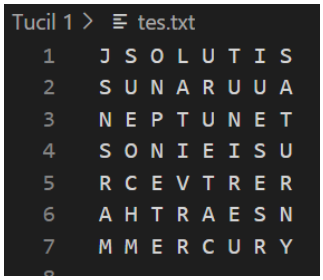
```
                nComparison++;
                k++;
                i2--;
            }
            if (isEqual) {
                printFoundMatrix(wordsMatrix, i - currentCharLen + 1, i, j, j, 0);
                System.out.println("Jumlah operasi perbandingan : " + nComparison);
            }
        }
        // Mengecek ke arah serong atas kiri
        if ((j - currentCharLen >= -1) && (i - currentCharLen >= -1)) {
            isEqual = true;
            int k = 1;
            int j2 = j - 1;
            int i2 = i - 1;
            while (k < currentCharLen && isEqual) {
                if (!wordsMatrix[i2][j2].equals(String.valueOf(words[q].charAt(k)))) {
                    isEqual = false;
                }
                nComparison++;
                k++;
                i2--;
                j2--;
            }
            if (isEqual) {
                printFoundMatrix(wordsMatrix, i - currentCharLen + 1, i, j - currentCharLen + 1, j, 1);
                System.out.println("Jumlah operasi perbandingan : " + nComparison);
            }
        }
        // Mengecek ke arah serong bawah kiri
        if ((j - currentCharLen >= -1) && (i + currentCharLen <= rows)) {
            isEqual = true;
            int k = 1;
            int j2 = j - 1;
            int i2 = i + 1;
            while (k < currentCharLen && isEqual) {
                if (!wordsMatrix[i2][j2].equals(String.valueOf(words[q].charAt(k)))) {
                    isEqual = false;
                }
                nComparison++;
                k++;
                i2++;
                j2--;
            }
            if (isEqual) {
                printFoundMatrix(wordsMatrix, i + currentCharLen - 1, i, j - currentCharLen + 1, j, 2);
                System.out.println("Jumlah operasi perbandingan : " + nComparison);
```

```
                    }
                }
                // Mengecek ke arah serong atas kanan
                if ((j + currentCharLen <= cols) && (i - currentCharLen >= -1)) {
                    isEqual = true;
                    int k = 1;
                    int j2 = j + 1;
                    int i2 = i - 1;
                    while (k < currentCharLen && isEqual) {
                        if (!wordsMatrix[i2][j2].equals(String.valueOf(words[q].charAt(k)))) {
                            isEqual = false;
                        }
                        nComparison++;
                        k++;
                        i2--;
                        j2++;
                    }
                    if (isEqual) {
                        printFoundMatrix(wordsMatrix, i, i - currentCharLen + 1, j, j + currentCharLen - 1, 2);
                        System.out.println("Jumlah operasi perbandingan : " + nComparison);
                    }
                }
                // Mengecek ke arah serong bawah kanan
                if ((j + currentCharLen <= cols) && (i + currentCharLen <= rows)) {
                    isEqual = true;
                    int k = 1;
                    int j2 = j + 1;
                    int i2 = i + 1;
                    while (k < currentCharLen && isEqual) {
                        if (!wordsMatrix[i2][j2].equals(String.valueOf(words[q].charAt(k)))) {
                            isEqual = false;
                        }
                        nComparison++;
                        k++;
                        i2++;
                        j2++;
                    }
                    if (isEqual) {
                        printFoundMatrix(wordsMatrix, i, i + currentCharLen - 1, j, j + currentCharLen - 1, 1);
                        System.out.println("Jumlah operasi perbandingan : " + nComparison);
                    }
                }
            }
        }
    }
}
```

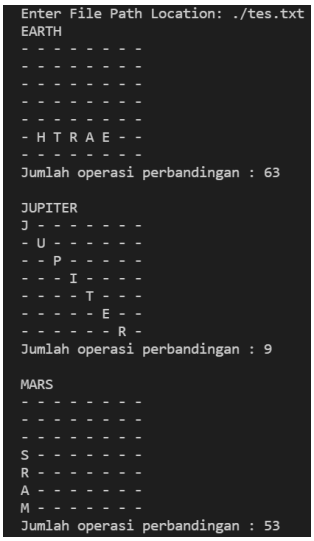
```java
public static void printMatrix(String[][] wordsMatrix) {
    for (int i = 0; i < wordsMatrix.length; i++) {
        for (int j = 0; j < wordsMatrix[i].length; j++) {
            System.out.print(wordsMatrix[i][j] + " ");
        }
        System.out.println("");
    }
}

public static void printFoundMatrix(String[][] wordsMatrix, Integer i1, Integer i2, Integer j1, Integer j2,
        Integer type) {
    for (int i = 0; i < wordsMatrix.length; i++) {
        for (int j = 0; j < wordsMatrix[i].length; j++) {
            if (type == 0) {
                if ((i >= i1) && (i <= i2) && (j >= j1) && (j <= j2)) {
                    System.out.print(wordsMatrix[i][j] + " ");
                } else {
                    System.out.print("- ");
                }
            } else if (type == 1) {
                if ((i >= i1) && (i <= i2) && (j >= j1) && (j <= j2)) {
                    if ((i1 - i == j1 - j) && (i2 - i == j2 - j)) {
                        System.out.print(wordsMatrix[i][j] + " ");
                    } else {
                        System.out.print("- ");
                    }
                } else {
                    System.out.print("- ");
                }
            } else {
                if ((i <= i1) && (i >= i2) && (j >= j1) && (j <= j2)) {
                    if ((i - i1 == j1 - j) && (i - i2 == j2 - j)) {
                        System.out.print(wordsMatrix[i][j] + " ");
                    } else {
                        System.out.print("- ");
                    }
                } else {
                    System.out.print("- ");
                }
            }
        }
        System.out.println("");
    }
}
```

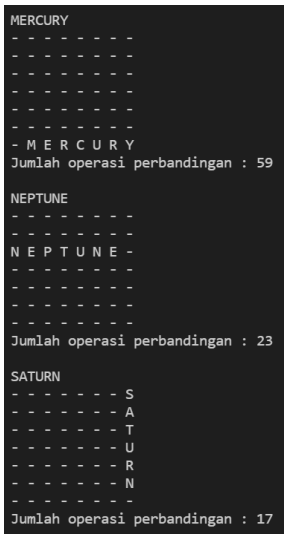
III. INPUT DAN OUTPUT PROGRAM

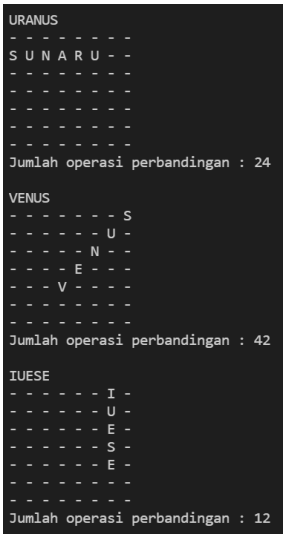
Input Program

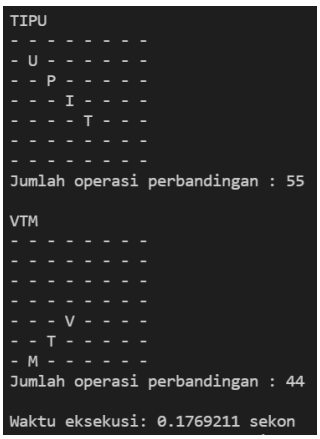


Output Program









IV. LINK KODE PROGRAM

https://github.com/adellinekania/Tucil-Word-Search-Puzzle

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