LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2021/2022

**Penyelesaian Word Search Puzzle dengan**

**Algoritma Brute Force**

Raka Wirabuana Ninagan || 13520134 || K-02

Tugas ini merupakan tugas untuk mengimplementasikan algoritma *brute force*, berupa pembuatan *Word Search Puzzle*.

1. **Algoritma *Brute Force***

Misalkan program memiliki matriks yang berisi huruf-huruf. Beberapa urutan huruf dengan berbagai variasi arahnya merupakan kata yang diinginkan, sehingga kata-kata tersebut disimpan di *array of strings.* Pendekatan yang dilakukan oleh saya adalah memilih arah pengecekan sebagai dasar pergerakan pencocokan strings, yang saya implementasikan sebagai 8 arah (kanan, kiri, atas, bawah, kanan bawah, kanan atas, kiri bawah, kiri atas). Setiap kata yang dicari akan dicek sesuai giliran arah, contohnya, ada 5 kata yang dicari, satu arah akan cek kelima kata tersebut, lalu ganti arah, cek kelima kata tersebut, ulangi sampai arah yang paling terakhir.

Pada pergerakan arah horizontal, setiap baris akan dicek apakah ada dari kata-kata yang dicari terbentuk di baris tersebut. Ketika tidak ada, maka akan lanjut ke baris selanjutnya sampai paling bawah.

Pada pergerakan arah vertikal, secara algoritma mirip dengan yang horizontal, tetapi acuan pengecekannya berdasarkan kolom. Tiap kolom dicek keberadaan kata-kata yang dicari sampai kolom terakhir.

Pada pergerakan arah diagonal, pendekatan yang dilakukan adalah membagi pergerakan pengecekan menjadi 2, yaitu dari indeks paling awal/akhir ke kanan/kiri dilanjutkan dengan pergerakan ke atas/bawah (untuk bentuk persegi panjang mendatar), dan dari indeks paling awal/akhir ke atas/bawah dilanjutkan dengan pergerakan ke kanan/kiri (untuk bentuk persegi panjang tegak). Contoh, misalkan ada matriks dengan ukuran persegi panjang 5 x 4 (indeks baris dari 0-4 dan kolom dari 0-3). Maka pergerakan pengecekan per diagonalnya adalah (0,0) ke (0,1) ke (0,2) ke (0,3) ke (0,4), lalu geser baris ke indeks 1 sehingga lanjutnyannya adalah (1,0) ke (2,0) ke (3,0). Setiap indeks diatas adalah acuan untuk melakukan pengecekan diagonal, sehingga tiap indeks baris atau kolom dijumlahkan 1.

Tiap keberhasilan pencocokan akan ditulis ke layar beserta posisi melalui visualisas matriks. Ketika kata yang dicari berhasil ditemukan, maka kata tersebut akan dihilangkan dari list agar tidak dapat dicek kembali (mengingat bahwa 1 kata cukup hanya mencari 1 kali ditemukan saja).

1. *Source Program*

import java.util.Scanner;

import java.io.FileNotFoundException;

public class Main {

    public static void main(String[] args) throws FileNotFoundException{

        System.out.println("| WORD SEARCH PUZZLE |\n");

        System.out.println("Masukkan nama file .txt yang diinginkan. (contoh ketikan: large1.txt)");

        System.out.println("Mohon pastikan tidak ada spasi di tempat yang tidak sesuai aturan masukan.\n");

*// Membaca nama file (beserta extensionnya)*

*// Contoh : large1.txt*

        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan nama file : ");

        String file = input.nextLine();

        input.close();

        System.out.println();

*// Mendeklarasikan matriks dan array sekaligus memasukkan isinya*

*// Detail file processing ada di file FileProcess.java*

        char[][] puzzlewordmatrix = FileProcess.matrixProcessing(file);

        String[] targetwordlist = FileProcess.arrayProcessing(file);

        System.out.println("Masukan Puzzle :");

        displayMat(puzzlewordmatrix, "-");

        System.out.println("Daftar kata yang dicari :");

        for (int s = 0; s < targetwordlist.length; s++){

            System.out.println("- " + targetwordlist[s]);

        }

        for (int s = 0; s < puzzlewordmatrix[0].length; s++){

            System.out.print("---");

        }

        System.out.println();

        System.out.println("HASIL PENCARIAN :");

*// Mendeklarasikan matrix display dan inisialisasi dengan simbol -*

        char[][] foundedMat = new char[puzzlewordmatrix.length][puzzlewordmatrix[0].length];

        initializeMat(foundedMat);

*// Mengaktifkan Penghitung Waktu*

        long start = System.nanoTime();

        targetwordlist = horizontalRight(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Horizontal Kiri*

        targetwordlist = horizontalLeft(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Vertikal Bawah*

        targetwordlist = verticalDown(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Vertikal Atas*

        targetwordlist = verticalUp(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Kanan Bawah*

        targetwordlist = southEast(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Kanan Atas*

        targetwordlist = northEast(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Kiri Bawah*

        targetwordlist = southWest(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Kiri Atas*

        targetwordlist = northWest(puzzlewordmatrix, foundedMat, targetwordlist);

        System.out.println();

*// Mencatat durasi kerja pencocokan dan penampilan matriks ke layar*

        long elapsedTime = System.nanoTime() - start;

        double seconds = (double)elapsedTime / 1\_000\_000\_000.0;

        System.out.print("Matching time : ");

        System.out.print(seconds);

        System.out.print(" s.");

    }

*// Menampilkan matrix*

    public static void displayMat(char[][] matrix, String word){

        System.out.println(">> WORD : " + word);

        for (int i = 0; i < matrix.length; i++){

            System.out.print(" |");

            for (int j = 0; j < matrix[0].length; j++){

                System.out.print(" " + matrix[i][j] + " ");

            }

            System.out.print("|");

            System.out.println();

        }

        System.out.println();

    }

*// Menginisialisasi matrix untuk hasil pencocokan kata*

*// Mengembalikan matrix display yang terisi menjadi seperti semula*

    public static void initializeMat(char[][] matrix){

        for (int i = 0; i < matrix.length; i++){

            for (int j = 0; j < matrix[0].length; j++){

                if (matrix[i][j] != '-'){

                    matrix[i][j] = '-';

                }

            }

        }

    }

*// Delete elemen array setelah ditemukan*

    public static String[] deleteWord(String[] wordlist, String word){

        String[] newWordlist = new String[wordlist.length];

        int i = 0, j = 0;

        while (i < newWordlist.length){

            if (wordlist[j] != word){

                newWordlist[i] = wordlist[j];

                j++;

            }

            else{

                newWordlist[i] = null;

                j += 1;

            }

            i++;

        }

        return newWordlist;

    }

*// Horizontal Kanan*

    public static String[] horizontalRight(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                int h = 0;

                while ((h < puzzlemat.length)  && (!found)){

                    int i = 0;

                    while (i <= puzzlemat[0].length - twlength  && (!found)){

                        int j = 0;

                        while ((j < twlength) && (puzzlemat[h][i+j] == targetword.charAt(j)) && (!found)){

                            filled = true;

                            resultmat[h][i+j] = puzzlemat[h][i+j];

                            j += 1;

                        }

                        if (j == twlength){

                            displayMat(resultmat, targetword);

                            initializeMat(resultmat);

                            targetarr[g] = null;

                            found = true;

                        }

                        if (filled){

                            initializeMat(resultmat);

                            filled = false;

                        }

                        i++;

                    }

                    h++;

                }

            }

        }

        return targetarr;

    }

*// Horizontal Kiri*

    public static String[] horizontalLeft(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                int h = 0;

                while ((h < puzzlemat.length)  && (!found)){

                    int i = puzzlemat[0].length - 1;

                    while (i >= twlength - 1){

                        int j = 0;

                        while ((j < twlength) && (puzzlemat[h][i-j] == targetword.charAt(j)) && (!found)){

                            filled = true;

                            resultmat[h][i-j] = puzzlemat[h][i-j];

                            j += 1;

                        }

                        if (j == twlength){

                            displayMat(resultmat, targetword);

                            initializeMat(resultmat);

                            targetarr[g] = null;

                            found = true;

                        }

                        if (filled){

                            initializeMat(resultmat);

                            filled = false;

                        }

                        i -= 1;

                    }

                    h++;

                }

            }

        }

        return targetarr;

    }

*// Vertikal Bawah*

    public static String[] verticalDown(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                int h = 0;

                while ((h < puzzlemat[0].length)  && (!found)){

                    int i = 0;

                    while (i <= puzzlemat.length - twlength){

                        int j = 0;

                        while ((j < twlength) && (puzzlemat[i+j][h] == targetword.charAt(j)) && (!found)){

                            filled = true;

                            resultmat[i+j][h] = puzzlemat[i+j][h];

                            j += 1;

                        }

                        if (j == twlength){

                            displayMat(resultmat, targetword);

                            initializeMat(resultmat);

                            targetarr[g] = null;

                            found = true;

                        }

                        if (filled){

                            initializeMat(resultmat);

                            filled = false;

                        }

                        i++;

                    }

                    h++;

                }

            }

        }

        return targetarr;

    }

*// Vertikal Atas*

    public static String[] verticalUp(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                int h = 0;

                while ((h < puzzlemat[0].length)  && (!found)){

                    int i = puzzlemat.length - 1;

                    while (i >= twlength - 1){

                        int j = 0;

                        while ((j < twlength) && (puzzlemat[i-j][h] == targetword.charAt(j)) && (!found)){

                            filled = true;

                            resultmat[i-j][h] = puzzlemat[i-j][h];

                            j += 1;

                        }

                        if (j == twlength){

                            displayMat(resultmat, targetword);

                            initializeMat(resultmat);

                            targetarr[g] = null;

                            found = true;

                        }

                        if (filled){

                            initializeMat(resultmat);

                            filled = false;

                        }

                        i--;

                    }

                    h++;

                }

            }

        }

        return targetarr;

    }

*// Arah tenggara*

    public static String[] southEast(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                if (puzzlemat.length > puzzlemat[0].length){

                    int f = 0;

                    while (f < puzzlemat[0].length && (!found)){

                        int h = 0, i = f;

                        while (h <= puzzlemat[0].length - twlength && i <= puzzlemat.length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i+j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h+j][i+j] = puzzlemat[h+j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i++;

                        }

                        f++;

                    }

                    f = 1;

                    while (f < puzzlemat.length && (!found)){

                        int h = f, i = 0;

                        while (h <= puzzlemat[0].length - twlength && i <= puzzlemat.length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i+j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h+j][i+j] = puzzlemat[h+j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i++;

                        }

                        f++;

                    }

                }

                else{

                    int f = 0;

                    while (f < puzzlemat[0].length && (!found)){

                        int h = 0, i = f;

                        while (h <= puzzlemat[0].length - twlength && i <= puzzlemat.length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i+j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h+j][i+j] = puzzlemat[h+j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i++;

                        }

                        f++;

                    }

                    f = 1;

                    while (f < puzzlemat.length && (!found)){

                        int h = f, i = 0;

                        while (h <= puzzlemat.length - twlength && i <= puzzlemat[0].length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i+j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h+j][i+j] = puzzlemat[h+j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i++;

                        }

                        f++;

                    }

                }

            }

        }

        return targetarr;

    }

*// Arah timur laut*

    public static String[] northEast(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                if (puzzlemat.length > puzzlemat[0].length){

                    int f = puzzlemat.length - 1;

                    while (f >= twlength - 1 && (!found)){

                        int h = f, i = 0;

                        while (h >= twlength - 1 && i <= puzzlemat[0].length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i+j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h-j][i+j] = puzzlemat[h-j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i++;

                        }

                        f--;

                    }

                    f = 1;

                    while (f < puzzlemat[0].length && (!found)){

                        int h = puzzlemat.length - 1, i = f;

                        while (h >= twlength - 1 && i <= puzzlemat[0].length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i+j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h-j][i+j] = puzzlemat[h-j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i++;

                        }

                        f++;

                    }

                }

                else{

                    int f = 0;

                    while (f < puzzlemat[0].length && (!found)){

                        int h = puzzlemat.length - 1, i = f;

                        while (h >= twlength - 1 && i <= puzzlemat[0].length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i+j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h-j][i+j] = puzzlemat[h-j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i++;

                        }

                        f++;

                    }

                    f = puzzlemat.length - 2;

                    while (f >= twlength - 1 && (!found)){

                        int h = f, i = 0;

                        while (h >= twlength - 1 && i <= puzzlemat[0].length - twlength && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i+j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h-j][i+j] = puzzlemat[h-j][i+j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i++;

                        }

                        f--;

                    }

                }

            }

        }

        return targetarr;

    }

*// Arah barat daya*

    public static String[] southWest(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                if (puzzlemat.length > puzzlemat[0].length){

                    int f = 0;

                    while (f < puzzlemat.length && (!found)){

                        int h = f, i = puzzlemat[0].length - 1;

                        while (h <= puzzlemat.length - twlength && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i-j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h+j][i-j] = puzzlemat[h+j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i--;

                        }

                        f++;

                    }

                    f = puzzlemat[0].length - 2;

                    while (f >= twlength - 1 && (!found)){

                        int h = 0, i = f;

                        while (h <= puzzlemat.length - twlength && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i-j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h+j][i-j] = puzzlemat[h+j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i--;

                        }

                        f--;

                    }

                }

                else{

                    int f = puzzlemat[0].length - 1;

                    while (f >= twlength - 1 && (!found)){

                        int h = 0, i = f;

                        while (h <= puzzlemat.length - twlength && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i-j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h+j][i-j] = puzzlemat[h+j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i--;

                        }

                        f--;

                    }

                    f = 1;

                    while (f < puzzlemat.length && (!found)){

                        int h = f, i = puzzlemat[0].length - 1;

                        while (h <= puzzlemat.length - twlength && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h+j][i-j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h+j][i-j] = puzzlemat[h+j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h++;

                            i--;

                        }

                        f++;

                    }

                }

            }

        }

        return targetarr;

    }

*// Arah Barat laut*

    public static String[] northWest(char[][] puzzlemat, char[][] resultmat, String[] targetarr){

        boolean filled = false;

        for (int g = 0; g  < targetarr.length; g++){

            if (targetarr[g] != null){

                String targetword = targetarr[g];

                int twlength = targetword.length();

                boolean found = false;

                if (puzzlemat.length > puzzlemat[0].length){

                    int f = puzzlemat.length - 1;

                    while (f >= twlength - 1 && (!found)){

                        int h = f, i = puzzlemat[0].length - 1;

                        while (h >= twlength - 1 && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i-j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h-j][i-j] = puzzlemat[h-j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i--;

                        }

                        f--;

                    }

                    f = puzzlemat[0].length - 2;

                    while (f >= twlength - 1 && (!found)){

                        int h = puzzlemat.length - 1, i = f;

                        while (h >= twlength - 1 && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i-j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h-j][i-j] = puzzlemat[h-j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i--;

                        }

                        f--;

                    }

                }

                else{

                    int f = puzzlemat[0].length - 1;

                    while (f >= twlength - 1 && (!found)){

                        int h = puzzlemat.length - 1, i = f;

                        while (h >= twlength - 1 && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i-j] == targetword.charAt(j))){

                                filled = true;

                                resultmat[h-j][i-j] = puzzlemat[h-j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i--;

                        }

                        f--;

                    }

                    f = puzzlemat.length - 2;

                    while (f >= twlength - 1 && (!found)){

                        int h = f, i = puzzlemat[0].length - 1;

                        while (h >= twlength - 1 && i >= twlength - 1 && (!found)){

                            int j = 0;

                            while ((j < twlength) && (puzzlemat[h-j][i-j] == targetword.charAt(j)) && (!found)){

                                filled = true;

                                resultmat[h-j][i-j] = puzzlemat[h-j][i-j];

                                j += 1;

                            }

                            if (j == twlength){

                                displayMat(resultmat, targetword);

                                initializeMat(resultmat);

                                targetarr[g] = null;

                                found = true;

                            }

                            if (filled){

                                initializeMat(resultmat);

                                filled = false;

                            }

                            h--;

                            i--;

                        }

                        f--;

                    }

                }

            }

        }

        return targetarr;

    }

}

1. *Screenshot* input dan output
2. Alamat *drive* berisi kode program.

Paket program dapat diakses melalui repository github saya:

<https://github.com/rkvilena/PuzzleWordSearch---13520134-IF2211-Strategi-Algoritma.git>