Laporan Tugas Kecil Strategi Algoritma 2021/2022 Word Search Puzzle



Disusun Oleh:

Andreas Indra Kurniawan 13520091/K-1

# **Langkah-Langkah Algoritma Brute Force**

1. Program menerima masukan *file text* yang berisi matrix segiempat berisi huruf dan juga kata yang akan dicari pada matrix tersebut(matrix dan kata dipisahkan satu enter kosong).
2. Program akan memproses *file* dan membuat *object* untuk menampung informasi di atas lalu akan dijalankan timer untuk mengetahui waktu awal ketika program dijalankan.
3. Program akan melangkah dari matrix kiri atas mengecek tiap baris terlebih dahulu.
4. Untuk setiap huruf yang sedang dicek, akan dilakukan cek untuk setiap kata yang ada. Jika huruf pertama matrix cocok dengan huruf pertama kata maka akan dilakukan pengecekan lebih lanjut yaitu ke arah kanan, kiri, atas, bawah, dan diagonal(kanan bawah, kiri bawah, kiri atas, dan kanan atas). Jika dari huruf pertama sudah tidak cocok, maka akan langsung dilanjutkan ke pengecekan kata selanjutnya.
5. Ketika kata sudah cocok 100%, program akan menandai posisi kata menggunakan matrix angka yang besarnya sudah disesuaikan dengan besar matrix puzzle. Angka 8 di matrix angka ini berarti warna netral, sedangkan 0-7 merupakan warna dengan kode ansi yang sudah dipilih.
6. Setelah menandai posisi kata, matrix akan menampilkan jumlah perbandingan yang digunakan untuk mencari satu kata dan waktu yang digunakan sehingga kata didapatkan.
7. Ketika seluruh kata berhasil ditemukan, maka program akan menambahkan kode ansi ke matrix puzzle sesuai dengan matrix angka yang ditandai.
8. Program akan menampilkan matrix puzzle yang telah diwarnai beserta total perbandingan, total waktu, dan total kata yang ditemukan.

# **Source Program Dalam Bahasa Java**

**File mainProg.Java**

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

class mainProg{

    public static void printPuzzle(String[][] puzzle,int width,int length){

        int i,j;

        for(i=0;i<length;i++){

            for(j=0;j<width;j++){

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

            }

            System.out.println();

        }

        System.out.print("\u001B[0m");

    }

    public static void printMatColor(puzzle objP){

        int i,j;

        for(i=0;i<objP.length;i++){

            for(j=0;j<objP.width;j++){

                System.out.print(objP.getIsiMatPC(i, j)+" ");

            }

            System.out.println();

        }

        System.out.println();

    }

    public static void printWords(String[][] words, int length){

        int i,j;

        for(i=0;i<length;i++){

            int wordLen = words[i].length;

            for(j=0;j<wordLen;j++){

                System.out.print(words[i][j]);

            }

            System.out.println();

        }

    }

    public static void printWord(String[] word){

        int i;

        for(i=0;i<word.length;i++){

            System.out.print(word[i]);

        }

        System.out.println();

    }

    public static String arrToString(String[] word){

        int i;

        String x = "";

        for(i=0;i<word.length;i++){

            x += word[i];

        }

        return x;

    }

    public static void initDash(String[][] dash,int length, int width){

        int i,j;

        for(i=0;i<length;i++){

            for(j=0;j<width;j++){

                dash[i][j] = "-";

            }

        }

    }

    public static puzzle checkR(puzzle objP, String[] word ,int xAwal, int yAwal, int width,int k){        //Mengecek kata di bagian horizontal

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal;

        int yUp = yAwal+1;

        objP.setStatus(false);

        while(depanMatch && i<word.length && yUp<width){

            objP.incrArrCount(k);

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                depanMatch = false;

                return objP;

            }

            i++;

            yUp++;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle checkL(puzzle objP, String[] word ,int xAwal, int yAwal, int width, int k){        //Mengecek kata di bagian horizontal ke kiri

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal;

        int yUp = yAwal-1;

        objP.setStatus(false);

        while(depanMatch && i<word.length && yUp>=0){

            objP.incrArrCount(k);

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                depanMatch = false;

                return objP;

            }

            i++;

            yUp--;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle checkT(puzzle objP, String[] word ,int xAwal, int yAwal, int width, int k){        //Mengecek kata di bagian vertikal ke atas

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal-1;

        int yUp = yAwal;

        objP.setStatus(false);

        while(depanMatch && i<word.length && xUp>=0){

            objP.incrArrCount(k);

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                depanMatch = false;

                return objP;

            }

            i++;

            xUp--;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle checkB(puzzle objP, String[] word ,int xAwal, int yAwal, int width,int k){        //Mengecek kata di bagian vertikal ke bawah

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal+1;

        int yUp = yAwal;

        objP.setStatus(false);

        while(depanMatch && i<word.length && xUp<objP.length){

            objP.incrArrCount(k);

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                depanMatch = false;

                return objP;

            }

            i++;

            xUp++;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle checkBR(puzzle objP, String[] word ,int xAwal, int yAwal, int width, int k){        //Mengecek kata di bagian diagonal ke tenggara

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal+1;

        int yUp = yAwal+1;

        objP.setStatus(false);

        while(depanMatch && i<word.length && xUp<objP.length && yUp<objP.width){

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                objP.incrArrCount(k);

                depanMatch = false;

                return objP;

            }

            i++;

            xUp++;

            yUp++;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle checkBL(puzzle objP, String[] word ,int xAwal, int yAwal, int width,int k){        //Mengecek kata di bagian diagonal ke barat daya

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal+1;

        int yUp = yAwal-1;

        objP.setStatus(false);

        while(depanMatch && i<word.length && xUp<objP.length && yUp>-1){

            objP.incrArrCount(k);

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                depanMatch = false;

                return objP;

            }

            i++;

            xUp++;

            yUp--;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle checkTL(puzzle objP, String[] word ,int xAwal, int yAwal, int width, int k){        //Mengecek kata di bagian diagonal ke barat laut

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal-1;

        int yUp = yAwal-1;

        objP.setStatus(false);

        while(depanMatch && i<word.length && xUp>-1 && yUp>-1){

            objP.incrArrCount(k);

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                depanMatch = false;

                return objP;

            }

            i++;

            xUp--;

            yUp--;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle checkTR(puzzle objP, String[] word ,int xAwal, int yAwal, int width, int k){        //Mengecek kata di bagian diagonal ke timur laut

        boolean depanMatch = true;

        int i=1;

        int xUp = xAwal-1;

        int yUp = yAwal+1;

        objP.setStatus(false);

        while(depanMatch && i<word.length && xUp>-1 && yUp<objP.width){

            objP.incrArrCount(k);

            if(!objP.getIsiMatP(xUp, yUp).equals(word[i])){

                depanMatch = false;

                return objP;

            }

            i++;

            xUp--;

            yUp++;

        }

        if(i==word.length){

            objP.setStatus(true);

        }

        return objP;

    }

    public static puzzle markR(puzzle objP, int xAwal, int yAwal, int wordLen){         //Menandai matrix warna ke arah kanan

        int j;

        for(j=yAwal;j<wordLen+yAwal;j++){

            objP.setMatPC(xAwal, j, objP.getColorCode());

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle markL(puzzle objP, int xAwal, int yAwal, int wordLen){

        int j;

        for(j=yAwal;j>yAwal-wordLen;j--){

            objP.setMatPC(xAwal, j, objP.getColorCode());

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle markT(puzzle objP, int xAwal, int yAwal, int wordLen){

        int i;

        for(i=xAwal;i>xAwal-wordLen;i--){

            objP.setMatPC(i, yAwal, objP.getColorCode());

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle markB(puzzle objP, int xAwal, int yAwal, int wordLen){

        int i;

        for(i=xAwal;i<xAwal+wordLen;i++){

            objP.setMatPC(i, yAwal, objP.getColorCode());

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle markBR(puzzle objP, int xAwal, int yAwal, int wordLen){

        int i;

        int j = yAwal;

        for(i=xAwal;i<xAwal+wordLen;i++){

            objP.setMatPC(i, j, objP.getColorCode());

            j++;

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle markBL(puzzle objP, int xAwal, int yAwal, int wordLen){

        int i;

        int j = yAwal;

        for(i=xAwal;i<xAwal+wordLen;i++){

            objP.setMatPC(i, j, objP.getColorCode());

            j--;

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle markTL(puzzle objP, int xAwal, int yAwal, int wordLen){

        int i;

        int j = yAwal;

        for(i=xAwal;i>xAwal-wordLen;i--){

            objP.setMatPC(i, j, objP.getColorCode());

            j--;

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle markTR(puzzle objP, int xAwal, int yAwal, int wordLen){

        int i;

        int j = yAwal;

        for(i=xAwal;i>xAwal-wordLen;i--){

            objP.setMatPC(i, j, objP.getColorCode());

            j++;

        }

        objP.updateColor();

        return objP;

    }

    public static puzzle color(puzzle objP){                                    //Mewarnain matrix puzzle sesuai warna dari matrix color

        int i,j;

        boolean color = false;

        for(i=0;i<objP.length;i++){

            for(j=0;j<objP.width;j++){

                if(objP.getIsiMatPC(i, j) != 8){

                    color = true;

                }else{

                    color = false;

                }

                if(color){

                    objP.setColorCode(objP.getIsiMatPC(i, j));

                    objP.setMatP(i, j, objP.getColor()+objP.getIsiMatP(i, j));

                }else{

                    objP.setMatP(i, j, objP.getResetC()+objP.getIsiMatP(i, j));

                }

            }

        }

        return objP;

    }

    public static puzzle checkPuzzle(puzzle pObj, int k, int i, int j){

        if(!pObj.getArrFound(k) && pObj.getIsiMatP(i, j).equals(pObj.getLetter(k, 0))){

            pObj.incrArrCount(k);

            pObj = checkR(pObj, pObj.getWord(k),i,j,pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") mendatar ke kanan dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markR(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

            pObj = checkL(pObj, pObj.getWord(k), i, j, pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") mendatar ke kiri dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markL(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

            pObj = checkT(pObj, pObj.getWord(k), i, j, pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") vertikal ke atas dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markT(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

            pObj = checkB(pObj, pObj.getWord(k), i, j, pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") vertikal ke bawah dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markB(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

            pObj = checkBR(pObj, pObj.getWord(k), i, j, pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") diagonal ke kanan bawah dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markBR(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

            pObj = checkBL(pObj, pObj.getWord(k), i, j, pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") diagonal ke kiri bawah dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markBL(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

            pObj = checkTL(pObj, pObj.getWord(k), i, j, pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") diagonal ke kiri atas dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markTL(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

            pObj = checkTR(pObj, pObj.getWord(k), i, j, pObj.width, k);

            if(pObj.getStatus()){

                // printR(pObj.getMatP(), matrixDash, i, j, pObj.getWord(k), width, length);

                System.out.println("Kata " + arrToString(pObj.getWord(k)) + " ditemukan pada koordinat (" + i + "," + j + ") diagonal ke kanan atas dalam :");

                System.out.println(pObj.getArrCount(k) + " perbandingan dan waktu " + pObj.getTime() + "ms\n");

                markTR(pObj, i, j, pObj.getWord(k).length);

                pObj.incrWordGot();

                pObj.setArrFound(k, true);

            }

        }

        return pObj;

    };

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        String filePath = sc.next();

        String path = System.getProperty("user.dir");

        path = path + "\\..\\test\\" + filePath;

        System.out.println(path);

        puzzle pObj = new puzzle();

        int i,j;

        i = 0; j = 0;

        int length = 0;

        int width = 0;

        int countWords = 0;

        try{

            File file = new File(path);

            Scanner readData = new Scanner(file);

            Boolean puzzEnd = false;

            while(readData.hasNextLine()){

                String lineData = readData.nextLine();

                if(lineData.isEmpty()){

                    puzzEnd = true;

                    lineData = readData.nextLine();

                }

                if(!puzzEnd){

                    String[] row = lineData.split(" ",0);

                    for(j = 0;j<row.length;j++){

                        pObj.setMatP(i,j,row[j]);

                        width = j+1;

                    }

                    length++;

                    i++;

                }else{

                    pObj.setWord(countWords, lineData.split("",0)) ;

                    countWords++;

                }

            }

            pObj.setCountW(countWords);

            pObj.setLength(length);

            pObj.setWidth(width);

            pObj.initMatPC();

            pObj.initArrCount();

            pObj.initArrFound();

            readData.close();

        }catch (FileNotFoundException exception){

            System.out.println("File tidak ditemukan");

        }

        int k;

        pObj.startTimer();

        for(i=0;i<length;i++){

            for(j=0;j<width;j++){

                for(k=0;k<countWords;k++){

                    pObj = checkPuzzle(pObj, k, i, j);

                }

            }

        }

        long endTime = pObj.getTime();

        color(pObj);

        printPuzzle(pObj.getMatP(), pObj.getWidth(), pObj.getLength());

        System.out.println("\n" + "Total perbandingan yang dilakukan:");

        System.out.println(pObj.getCountSum());

        System.out.println("Total waktu yang digunakan:");

        System.out.println(endTime + "ms");

        System.out.println("Total kata yang ditemukan :");

        System.out.println(pObj.getWordGot());

        sc.close();

    }

}

**File puzzle.java**

public class puzzle {

    String[][] matrixPuzzle = new String[100][100];

    String[][] matrixWord = new String[100][100];

    int[][] matrixPuzzleC = new int[100][100];

    int[] arrayCount = new int[100];

    boolean[] arrayFound = new boolean[100];

    String[] arrayColor = new String[6];

    long startTime;

    int countWords = 0;

    int length;

    int width;

    int nthColor = 0;

    int getCountSum = 0;

    int wordGot = 0;

    boolean found = false;

    public static final String ANSI\_RED = "\u001B[31m";

    public static final String ANSI\_YELLOW = "\u001B[33m";

    public static final String ANSI\_GREEN = "\u001B[32m";

    public static final String ANSI\_BLUE = "\u001B[34m";

    public static final String ANSI\_CYAN = "\u001b[36m";

    public static final String ANSI\_MAGENTA = "\u001b[35m";

    public static final String ANSI\_BRED = "\033[38;5;213m";

    public static final String ANSI\_BBLUE = "\033[38;5;122m";

    public static final String ANSI\_RESET = "\u001B[0m";

    // public void setArrC(){

    //     this.arrayColor[0] = "\u001B[31m";  //Red

    //     this.arrayColor[1] = "\u001B[33m";  //Yellow

    //     this.arrayColor[2] = "\u001B[32m";  //Green

    //     this.arrayColor[3] = "\u001B[34m";  //Blue

    //     this.arrayColor[4] = "\u001B[37m";  //White

    //     this.arrayColor[5] = "\u001B[0m";  //reset

    // }

    public void initArrFound(){

        int i;

        for(i=0;i<countWords;i++){

            this.arrayFound[i] = false;

        }

    }

    public void setArrFound(int i, boolean x){

        this.arrayFound[i] = x;

    }

    public boolean getArrFound(int i){

        return this.arrayFound[i];

    }

    public void incrWordGot(){

        this.wordGot++;

    }

    public int getWordGot(){

        return this.wordGot;

    }

    public void startTimer(){

        this.startTime = System.nanoTime();

    }

    public long getTime(){

        return (System.nanoTime()-this.startTime)/1000000;

    }

    public void setArrCount(int i, int x){

        this.arrayCount[i] = x;

    }

    public void incrGetCount(int x){

        this.getCountSum += x;

    }

    public int getArrCount(int i){

        return this.arrayCount[i];

    }

    public void incrArrCount(int i){

        this.arrayCount[i]++;

    }

    public int getCountSum(){

        int i;

        int sum = 0;

        for(i=0;i<this.arrayCount.length;i++){

            sum += this.arrayCount[i];

        }

        return sum;

    }

    public void initArrCount(){

        int i;

        for(i=0;i<this.countWords;i++){

            setArrCount(i, 0);

        }

    }

    public String getColor(){

        if(this.nthColor == 0){

            return ANSI\_RED;

        }else{

            if(this.nthColor == 1){

                return ANSI\_YELLOW;

            }else{

                if(this.nthColor == 2){

                    return ANSI\_GREEN;

                }else{

                    if(this.nthColor == 3){

                        return ANSI\_BLUE;

                    }else{

                        if(this.nthColor == 4){

                            return ANSI\_CYAN;

                        }else{

                            if(this.nthColor == 5){

                                return ANSI\_MAGENTA;

                            }else{

                                if(this.nthColor == 6){

                                    return ANSI\_BRED;

                                }else{

                                    if(this.nthColor==7){

                                        return ANSI\_BBLUE;

                                    }else{

                                        return ANSI\_RESET;

                                    }

                                }

                            }

                        }

                    }

                }

            }

        }

    }

    public void updateColor(){

        this.nthColor++;

        this.nthColor %= 8;

    }

    public String getResetC(){

        return ANSI\_RESET;

    }

    public void setMatP(int i, int j, String x){

        this.matrixPuzzle[i][j] = x;

    }

    public String[][] getMatP(){

        return this.matrixPuzzle;

    }

    public String getIsiMatP(int i, int j){

        return this.matrixPuzzle[i][j];

    }

    public void setMatW(int i, int j, String x){

        this.matrixWord[i][j] = x;

    }

    public String[][] getMatW(){

        return this.matrixWord;

    }

    public String[] getWord(int pos){

        return this.matrixWord[pos];

    }

    public void setWord(int pos, String[] S){

        this.matrixWord[pos] = S;

    }

    public String getLetter(int x, int y){

        return this.matrixWord[x][y];

    }

    public void setStatus(boolean status){

        this.found = status;

    }

    public boolean getStatus(){

        return this.found;

    }

    public void setCountW(int num){

        this.countWords = num;

    }

    public int getCountW(){

        return this.countWords;

    }

    public void setLength(int num){

        this.length = num;

    }

    public int getLength(){

        return this.length;

    }

    public void setWidth(int num){

        this.width = num;

    }

    public int getWidth(){

        return this.width;

    }

    public void setMatPC(int i, int j, int x){

        this.matrixPuzzleC[i][j] = x;

    }

    public void initMatPC(){

        int i,j;

        for(i=0;i<this.length;i++){

            for(j=0;j<this.width;j++){

                setMatPC(i,j,8);

            }

        }

    }

    public int getIsiMatPC(int i, int j){

        return this.matrixPuzzleC[i][j];

    }

    public int[][] getMatPC(){

        return this.matrixPuzzleC;

    }

    public int getColorCode(){

        return this.nthColor;

    }

    public void setColorCode(int x){

        this.nthColor = x;

    }

}

# **Hasil Skrinshut**

**Input(small1.txt):**

**A picture containing calendar

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Input(small2.txt):**

**A picture containing text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Arrow

Description automatically generated with medium confidence**

**Input(small3.txt):**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**A picture containing arrow

Description automatically generated**

**Input(medium1.txt):**

**A picture containing text, electronics, keyboard

Description automatically generated**

**Text

Description automatically generated with low confidence**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Arrow

Description automatically generated with medium confidence**

**Input(medium2.txt):**

**A picture containing text, electronics, keyboard

Description automatically generated**

**Graphical user interface

Description automatically generated with low confidence**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**A picture containing text

Description automatically generated**

**Input(medium3.txt):**

**A picture containing text, keyboard, file

Description automatically generated**

**Text

Description automatically generated with medium confidence**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

****

**A picture containing text

Description automatically generated**

**Input(large1.txt):**

**Background pattern

Description automatically generated**

**Text

Description automatically generated with low confidenceTable

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Background pattern

Description automatically generated**

**Text

Description automatically generated**

**Input(large2.txt):**

**Background pattern

Description automatically generated**

**A picture containing text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Background pattern

Description automatically generated**

**Text

Description automatically generated**

**Input(large3.txt):**

**Background pattern

Description automatically generated**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Background pattern

Description automatically generated**

**Text

Description automatically generated**

# **Link**