Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Word Search Puzzle dengan Algoritma Brute Force

Disusun oleh:

13520016 - Gagas Praharsa Bahar

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

BAB 1
DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma Brute Force

Algoritma *brute force* merupakan sebuah pendekatan dari penyelesaian sebuah masalah algoritmik dengan menggunakan pendekatan yang lempeng (*straightforward)* untuk menyelesaikan sebuah persoalan yang telah didefinisikan. Algoritma *brute force* didasarkan pada pernyataan pada persoalan (*problem statement*) dan juga konsep yang dilibatkan oleh persoalan yang sedang dibahas.

Algoritma dengan pendekatan *brute force* biasanya memiliki ciri khas mempunyai konsep penyelesaian yang sederhana, langsung, jelas, dan intuitif. Pendekatan *brute force* seringkali melibatkan enumerasi semua kemungkinan solusi, sebelum akhirnya menghilangkan jawaban yang tidak memenuhi syarat dan mengambil solusi terbaik (apabila ada). Algoritma dengan pendekatan *brute force* dijamin akan menemukan sebuah solusi apabila solusi tersebut ada. Namun, algoritma dengan pendekatan *brute force* seringkali tidak mangkus atau tidak efektif, dengan O(n) yang lebih buruk dari polinomial.

1.2 Word Search Puzzle

Word search puzzle adalah permainan kata dimana pemain harus menemukan beberapa kata tersembunyi dalam kumpulan huruf acak. Kumpulan huruf tersebut biasa diletakkan pada “papan” (atau matriks) berbentuk segi empat atau dapat disebut juga matriks huruf. Kata-kata pada matriks huruf ini dapat ditemukan dalam delapan arah yang mungkin, yaitu vertikal ke atas, vertikal ke bawah, horizontal ke kanan, horizontal ke kiri, diagonal ke kanan atas, diagonal ke kanan bawah, diagonal ke kiri atas, dan diagonal ke kiri bawah. *Word search puzzle* pertama kali dibuat oleh Noeman E. Gibat yang mempublikasikan *puzzle* ini dalam the *Selenby Digest* pada 1 Maret 1968 di Norman, Oklahoma, Amerika Serikat. Setelah itu, *word search puzzle* perlahan menumbuhkan popularitasnya dan mulai terkenal di daerah asalnya. Tidak lama dari itu,

beberapa guru meminta salinan dari *word search puzzle* untuk kemudian digunakan sebagai alat untuk mengajar. Akhirnya, *puzzle* ini tersebar setelah ada guru yang meneruskan salinan *puzzle* ini kepada kerabatnya. Sekarang, *word search puzzle* bisa ditemukan dengan mudah di berbagai media, baik cetak maupun digital.

1.3 Algoritma Penyelesaian Word Search Puzzle dengan Pendekatan Brute Force

Dalam menyelesaikan permasalahan *word search puzzle* secara algoritmik, penulis menggunakan pendekatan *brute force*. Adapun langkah-langkah penyelesaian permasalahan dalam algoritma dapat dijelaskan secara deskriptif sebagai berikut:

1. Terima input dari file teks berupa puzzle yang sesuai dengan ketentuan yang ditentukan sebelumnya.
2. Tampung input pada struktur data agar dapat diakses oleh program.
3. Untuk setiap kata yang akan dicari, lakukan delapan pencarian ke segala arah.
    a. Pencocokan kata dimulai dengan menyiapkan kata yang akan dicari (pattern) dan puzzle itu sendiri (text)
    b. Setiap satuan string di enumerasi (baik vertikal, horizontal, maupun diagonal dihitung semua kemungkinannya) agar dapat dibaca oleh fungsi lineMatcher.
4. Dari kedelapan luaran fungsi, identifikasi fungsi manakah yang memiliki luaran (ada solusi)
5. Keluarkan solusi tersebut ke terminal.

BAB 2
IMPLEMENTASI ALGORITMA DALAM BAHASA JAVA

Dalam pembuatan program ini, penulis menggunakan bahasa pemrograman Java. Struktur dari program ini sendiri terbagi menjadi 4 file, yaitu Main.java, Puzzle.java, Result.java, dan Solver.java.

2.1 Main.java

File ini berisi *driver* utama dari program sehingga tidak memiliki fungsi di dalamnya.

2.2 Puzzle.java

File ini berisi *class* dari objek *Puzzle*.

- **Attributes**

Attribute	Description
`private char[][] puzzle`	Atribut ini berisi puzzle word search dalam bentuk array of array of char.
`private String[] words`	Atribut ini berisi daftar kata yang harus dicari.
`private int row`	Atribut ini berisi jumlah baris pada puzzle.
`private int col`	Atribut ini berisi jumlah kolom pada puzzle.
`private int wordsLength`	Atribut ini berisi jumlah kata yang harus dicari.
`private String dir`	Atribut ini menyimpan directory file

	testing.
`private int CAPACITY`	Atribut ini menyimpan kapasitas maksimal (karena menggunakan array statis)
`private int comparison`	Atribut ini menyimpan jumlah perbandingan.

- **Constructors**

Constructor (Parameter)	Description
`public Puzzle()`	Konstruktor ini akan menginisialisasi puzzle kosong.

- **Methods**

Methods(Parameter)	Description
`public char[][] getPuzzle()`	Metode ini berguna untuk mengambil puzzle.
`public String[] getWords()`	Metode ini berfungsi untuk mengambil daftar kata.
`public int getRow()`	Metode ini berfungsi untuk mengambil jumlah baris.
`public int getCol()`	Metode ini berfungsi untuk mengambil jumlah kolom.

`public int getWordsLength()`	Metode ini berfungsi untuk mengambil jumlah kata.
`public int getCapacity()`	Metode ini berfungsi untuk mengambil kapasitas maksimal.
`public int getComparison()`	Metode ini berfungsi untuk mengambil jumlah perbandingan.
`public void setComparison(int x)`	Metode ini berfungsi untuk mengganti nilai atribut comparison pada objek.
`public void fileInput(String filename)`	Metode ini berfungsi untuk mengisi puzzle dengan input dari file.
`public void printPuzzle()`	Metode ini berfungsi untuk menampilkan puzzle pada terminal.

2.3 Result.java

File ini berisi class dari objek Result.

- **Attributes**

Attribute	Description
`private final int firstCol`	Atribut ini berisi kolom pertama dari jawaban.
`private final int firstRow`	Atribut ini berisi baris pertama dari jawaban.
`private final int ansLength`	Atribut ini berisi panjang jawaban.

`private final String type`	Atribut ini berisi jenis jawaban (kiri, kanan, diagonal, dll)

- **Constructors**

Constructor (Parameter)	Description
`public Result(int firstRow, int firstCol, int ansLength, String type)`	Konstruktor ini akan membuat Result baru dengan atribut yang sesuai dengan parameternya masing-masing.
`public Result()`	Konstruktor ini akan membuat Result kosong (yang menandakan tidak ada solusi)

- **Methods**

Methods(Parameter)	Description
`public int getFirstCol()`	Metode ini berfungsi untuk mendapatkan atribut firstCol.
`public int getFirstRow()`	Metode ini berfungsi untuk mendapatkan atribut firstRow.
`public int getAnsLength()`	Metode ini berfungsi untuk mendapatkan atribut ansLength.
`public String getType()`	Metode ini berfungsi untuk mendapatkan atribut type.

`public void printResult(Puzzle p)`	Metode ini berfungsi untuk menampilkan jawaban pada layar satu per satu setiap kata.

2.4 Solver.java

File ini berisi *class* dari objek *Solver*.

- **Attributes**

  Class ini tidak memiliki atribut.

- **Constructors**

  Class ini tidak memiliki konstruktor.

- **Methods**

Methods(Parameter)	Description
`public static int lineMatcher(String text, String pattern, Puzzle p)`	Metode ini berfungsi sebagai metode *pattern matching* utama dengan menggunakan pendekatan *brute force*.
`private static String reverseCharArray(char[] s)`	Metode ini berfungsi untuk membalikkan array of char menjadi string.
`public static String reverseString(String s)`	Metode ini berfungsi untuk membalikkan sebuah string.
`public static Result wordCheckRight(Puzzle p, String word)`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah horizontal ke kanan
`public static Result`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah horizontal ke kiri

`wordCheckLeft(Puzzle p, String word)`	
`public static Result wordCheckDown(Puzzle p, String word)`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah vertikal ke bawah
`public static Result wordCheckUp(Puzzle p, String word)`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah vertikal ke atas
`public static Result wordCheckRightUp(Puzzle p, String word)`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah diagonal ke kanan atas
`public static Result wordCheckLeftDown(Puzzle p, String word)`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah diagonal ke kiri bawah
`public static Result wordCheckRightDown(Puzzle p, String word)`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah diagonal ke kanan bawah
`public static Result wordCheckLeftUp(Puzzle p, String word)`	Metode ini berfungsi untuk mencari kata pada puzzle dengan arah diagonal ke kiri atas
`public static Result allWordCheck(Puzzle p, String word)`	Metode ini berfungsi untuk menggabungkan kedelapan fungsi pengecekan kata.

`public static String[] diagonalize(Puzzle p)`	Metode ini berfungsi untuk mengembalikan string dari kemungkinan diagonal kanan atas.
`public static String[] diagonalizeRightDown(Puzzle p)`	Metode ini berfungsi untuk mengembalikan string dari kemungkinan diagonal kanan bawah.

2.5 PrintColor.java

Berisi atribut untuk print dengan warna.

`public static void printColor(char s, String color)`	Metode ini berfungsi untuk print char sesuai dengan warna yang diinginkan.

2.6 Dependencies

- `java.util.Scanner`
- `java.io.File`
- `java.io.FileNotFoundException`
- `java.util.Arrays`

BAB 3

SOURCE CODE PROGRAM

3.1. Repository Program

Repository program dapat diakses melalui:

- Github: https://github.com/gagaspbahar/bruteforce-word-puzzle-solver
- Google Drive:

3.2. Source Code Program

3.2.1. Main.java

```java
import java.util.Scanner;

public class Main {
    public static Scanner sc;

    public static void main(String[] args) {
        sc = new Scanner(System.in);

        System.out.println("Masukkan nama file berisi puzzle:");
        String filename = sc.nextLine();
        Puzzle puzzle = new Puzzle();
        puzzle.fileInput(filename);

        long start = System.nanoTime();
        Result[] resultList = new Result[puzzle.getWordsLength()];
        for (int i = 0; i < puzzle.getWordsLength(); i++) {
            resultList[i] = Solver.allWordCheck(puzzle, puzzle.getWords()[i]);
        }

        long end = System.nanoTime();

        System.out.println("Pilih output: ");
        System.out.println("1. Single, berwarna");
        System.out.println("2. Multiple matrix");
        String prompt = sc.nextLine();

        switch (prompt) {
            case "1":
                Result.printPuzzleWithColor(puzzle, resultList);
                break;
            case "2":
                for (int i = 0; i < puzzle.getWordsLength(); i++) {
                    System.out.printf("%d. %s\n", i + 1, puzzle.getWords()[i]);
                    resultList[i].printResult(puzzle);
                }
            default:
                Result.printPuzzleWithColor(puzzle, resultList);
                break;
        }
        long time = (end - start);
        double timeElapsed = (double) time / 1_000_000_000;

        System.out.println("Time taken: " + timeElapsed + " seconds");
        System.out.printf("Comparison made: %d\n", puzzle.getComparison());
        sc.close();
    }
}
```

3.2.2. Puzzle.java

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

public class Puzzle {
    private char[][] puzzle;
    private String[] words;
    private int row;
    private int col;
    private int wordsLength;
    private String dir = "..\\test";
    private int CAPACITY = 500;
    private int comparison;

    public Puzzle() {
        this.puzzle = new char[CAPACITY][CAPACITY];
        this.words = new String[CAPACITY];
        this.row = 0;
        this.col = 0;
        this.wordsLength = 0;
        this.comparison = 0;
    }

    public char[][] getPuzzle() {
        return this.puzzle;
    }

    public String[] getWords() {
        return this.words;
    }

    public int getRow() {
        return this.row;
    }

    public int getCol() {
        return this.col;
    }

    public int getWordsLength() {
        return this.wordsLength;
    }

    public int getCapacity() {
        return this.CAPACITY;
    }

    public int getComparison() {
        return this.comparison;
    }

    public void setComparison(int x) {
        this.comparison = x;
    }

```

```java
    public void fileInput(String filename) {
        try {
            String path = dir + "\\" + filename;
            File file = new File(path);
            Scanner scfile = new Scanner(file);
            int i = 0;
            while (scfile.hasNextLine()) {
                String line = scfile.nextLine();
                if (line.isEmpty()) {
                    break;
                }
                String[] lineArray = line.split(" ");
                this.col = lineArray.length;
                for (int j = 0; j < lineArray.length; j++) {
                    this.puzzle[i][j] = lineArray[j].charAt(0);
                }
                i++;
            }
            this.row = i;
            i = 0;
            while (scfile.hasNextLine()) {
                String line = scfile.nextLine();
                String noSpaceStr = line.replaceAll("\\s", "");
                this.words[i] = noSpaceStr.trim();
                i++;
            }
            this.wordsLength = i;
            scfile.close();
        } catch (FileNotFoundException e) {
            System.out.println("File tidak ditemukan.");
            System.exit(0);
        }
    }

    public void printPuzzle() {
        for (int i = 0; i < this.row; i++) {
            for (int j = 0; j < this.col; j++) {
                if (j == col - 1) {
                    System.out.println(this.puzzle[i][j]);
                } else {
                    System.out.print(this.puzzle[i][j] + " ");
                }
            }
        }
    }

}
```

3.2.3. Solver.java

```java
import java.util.Arrays;

public class Solver {

    public static int lineMatcher(String text, String pattern, Puzzle p) {
        int ans = -1;
        text = text.trim();
        int tLen = text.trim().length();
        int pLen = pattern.length();
        int comp = 0;
        if (pLen <= tLen) {
            for (int i = 0; i <= tLen - pLen; i++) {
                int j = 0;
                while (j < pLen) {
                    comp++;
                    if (text.charAt(i + j) == pattern.charAt(j)) {
                        j++;
                    } else {
                        break;
                    }
                }
                if (j == pLen) {
                    ans = i;
                }
            }
        }
        p.setComparison(p.getComparison() + comp);
        return ans;
    }

    private static String reverseCharArray(char[] s) {
        String ans = "";
        for (int i = s.length - 1; i >= 0; i--) {
            ans += s[i];
        }
        return ans;
    }

    public static String reverseString(String s) {
        String ans = "";
        for (int i = s.length() - 1; i >= 0; i--) {
            ans += s.charAt(i);
        }
        return ans;
    }
```

```java
public static Result wordCheckRight(Puzzle p, String word) {
        Result ans = new Result();
        int first;
        int ansFirst = -1;
        int ansRow = -1;
        for (int j = 0; j < p.getRow(); j++) {
            first = lineMatcher(String.valueOf(p.getPuzzle()[j]), word, p);
            if (first != -1) {
                ansRow = j;
                ansFirst = first;
                break;
            }
        }
        if (ansFirst != -1) {
            ans = new Result(ansRow, ansFirst, word.length(), "right");
        }
        return ans;
    }

    public static Result wordCheckLeft(Puzzle p, String word) {
        Result ans = new Result();
        int first;
        int ansFirst = -1;
        int ansRow = -1;
        for (int j = 0; j < p.getRow(); j++) {
            first = lineMatcher(reverseCharArray(p.getPuzzle()[j]), word, p);
            if (first != -1) {
                ansRow = j;
                ansFirst = first;
                break;
            }
        }
        if (ansFirst != -1) {
            ans = new Result(ansRow, ansFirst, word.length(), "left");
        }
        return ans;
    }
```

```java
public static Result wordCheckDown(Puzzle p, String word) {
        Result ans = new Result();
        int first;
        int ansFirst = -1;
        int ansCol = -1;
        for (int j = 0; j < p.getCol(); j++) {
            String text = "";
            for (int i = 0; i < p.getRow(); i++) {
                text += p.getPuzzle()[i][j];
            }
            first = lineMatcher(text, word, p);
            if (first != -1) {
                ansCol = j;
                ansFirst = first;
                break;
            }
        }
        if (ansFirst != -1) {
            ans = new Result(ansFirst, ansCol, word.length(), "down");
        }
        return ans;
    }

    public static Result wordCheckUp(Puzzle p, String word) {
        Result ans = new Result();
        int first;
        int ansFirst = -1;
        int ansCol = -1;
        for (int j = 0; j < p.getCol(); j++) {
            String text = "";
            for (int i = 0; i < p.getRow(); i++) {
                text += p.getPuzzle()[i][j];
            }
            first = lineMatcher(reverseString(text), word, p);
            if (first != -1) {
                ansCol = j;
                ansFirst = first;
                break;
            }
        }
        if (ansFirst != -1) {
            ans = new Result(ansFirst, ansCol, word.length(), "up");
        }
        return ans;
    }
```

```java
public static Result wordCheckRightUp(Puzzle p, String word) {
        Result ans = new Result();
        String[] check = diagonalize(p);
        int first;
        int ansCol = -1;
        int ansRow = -1;
        int i = 0;
        int j = 0;
        for (int k = 0; k < check.length; k++) {
            first = lineMatcher(String.valueOf(check[k]), word, p);
            if (first != -1) {
                ansRow = i - first;
                ansCol = j + first;
                break;
            }
            if (i != p.getRow() - 1) {
                i++;
            } else {
                j++;
            }
        }
        if (ansCol != -1) {
            ans = new Result(ansRow, ansCol, word.length(), "rightUp");
        }
        return ans;
    }

    public static Result wordCheckLeftDown(Puzzle p, String word) {
        Result ans = new Result();
        String[] check = diagonalize(p);
        int first;
        int ansCol = -1;
        int ansRow = -1;
        int i = 0;
        int j = 0;
        for (int k = 0; k < check.length; k++) {
            first = lineMatcher(String.valueOf(reverseString(check[k])), word, p);
            if (first != -1) {
                ansRow = i - (check[k].trim().length() - 1) + first;
                ansCol = j + (check[k].trim().length() - 1) - first;
                break;
            }
            if (i != p.getRow() - 1) {
                i++;
            } else {
                j++;
            }
        }
        if (ansCol != -1) {
            ans = new Result(ansRow, ansCol, word.length(), "leftDown");
        }
        return ans;
    }
```

```java
public static Result wordCheckRightDown(Puzzle p, String word) {
        Result ans = new Result();
        String[] check = diagonalizeRightDown(p);
        int first;
        int ansCol = -1;
        int ansRow = -1;
        int i = p.getRow() - 1;
        int j = 0;
        for (int k = 0; k < check.length; k++) {
            first = lineMatcher(String.valueOf(check[k]), word, p);
            if (first ≠ -1) {
                ansRow = i + first;
                ansCol = j + first;
                break;
            }
            if (i ≠ 0) {
                i--;
            } else {
                j++;
            }
        }
        if (ansCol ≠ -1) {
            ans = new Result(ansRow, ansCol, word.length(), "rightDown");
        }
        return ans;
    }

    public static Result wordCheckLeftUp(Puzzle p, String word) {
        Result ans = new Result();
        String[] check = diagonalizeRightDown(p);
        int first;
        int ansCol = -1;
        int ansRow = -1;
        int i = p.getRow() - 1;
        int j = 0;
        for (int k = 0; k < check.length; k++) {
            first = lineMatcher(String.valueOf(reverseString(check[k])), word, p);
            if (first ≠ -1) {
                ansRow = i + (check[k].trim().length() - 1 - first);
                ansCol = j + (check[k].trim().length() - 1 - first);
                break;
            }
            if (i ≠ 0) {
                i--;
            } else {
                j++;
            }
        }
        if (ansCol ≠ -1) {
            ans = new Result(ansRow, ansCol, word.length(), "leftUp");
        }
        return ans;
    }
```

```java
public static Result allWordCheck(Puzzle p, String word) {
    Result res = new Result();
    while (res.getType() == "") {
        res = wordCheckRight(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        res = wordCheckLeft(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        res = wordCheckUp(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        res = wordCheckDown(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        res = wordCheckRightUp(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        res = wordCheckLeftDown(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        res = wordCheckRightDown(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        res = wordCheckLeftUp(p, word);
        if(res.getType() ≠ ""){
            break;
        }
        break;
    }
    return res;
}
```

```java
public static String[] diagonalize(Puzzle p) {
        char[][] puzzle = p.getPuzzle();
        int min = Math.min(p.getCol(), p.getRow()) * 2;
        String[] ans = new String[min+5];
        Arrays.fill(ans, "");
        String temp = "";
        int i = 0;
        int j = 0;
        int k = 0;
        int l = 1;
        while (k ≠ p.getRow()) {
            temp = "";
            do {
                temp += puzzle[i][j];
                i--;
                j++;
            } while (i ⩾ 0);
            ans[k] = temp;
            k++;
            i = k;
            j = 0;
        }
        i = p.getRow() - 1;
        j = l;
        while (l ≠ p.getCol()) {
            temp = "";
            do {
                temp += puzzle[i][j];
                i--;
                j++;
            } while (i ⩾ 0);
            ans[k] = temp;
            k++;
            l++;
            i = p.getRow() - 1;
            j = l;
        }
        return ans;
    }

    public static String[] diagonalizeRightDown(Puzzle p) {
        char[][] puzzle = p.getPuzzle();
        int min = Math.min(p.getCol(), p.getRow()) * 2;
        if (p.getCol() == p.getRow()) {
            min--;
        }
        String[] ans = new String[min+5];
        Arrays.fill(ans, "");
        String temp = "";
        int i = p.getRow() - 1;
        int j = 0;
        int count = 0;
        int k = i;
        int l = 1;
        while (k ≠ -1) {
            temp = "";
            do {
                temp += puzzle[i][j];
                i++;
                j++;
            } while (i < p.getRow());
            ans[count] = temp;
            count++;
            k--;
            i = k;
            j = 0;
        }
        i = 0;
        j = l;
        while (l ≠ p.getCol()) {
            temp = "";
            do {
                temp += puzzle[i][j];
                i++;
                j++;
            } while (i < p.getRow());
            ans[count] = temp;
            count++;
            l++;
            i = 0;
            j = l;
        }
        return ans;
    }
}
```

3.2.4. Result.java

```java
public class Result {
    private final int firstCol;
    private final int firstRow;
    private final int ansLength;
    private final String type;

    public Result(int firstRow, int firstCol, int ansLength, String type) {
        this.firstCol = firstCol;
        this.firstRow = firstRow;
        this.ansLength = ansLength;
        this.type = type;
    }

    public Result() {
        this.firstCol = -1;
        this.firstRow = -1;
        this.ansLength = -1;
        this.type = "";
    }

    public int getFirstCol() {
        return this.firstCol;
    }

    public int getFirstRow() {
        return this.firstRow;
    }

    public int getAnsLength() {
        return this.ansLength;
    }

    public String getType() {
        return this.type;
    }

```

```java
public void printResult(Puzzle p) {
        int row = p.getRow();
        int col = p.getCol();
        int start;
        int r;
        int c;
        char[][] puzzle = p.getPuzzle();
        boolean puzzleTable[][] = new boolean[row][col];
        switch (this.type) {
            case "right":
                for (int i = this.firstCol; i < this.firstCol + this.ansLength; i++) {
                    puzzleTable[this.firstRow][i] = true;
                }
                break;
            case "left":
                start = (p.getCol() - 1) - this.firstCol;
                for (int i = start; i >= start - (this.ansLength - 1); i--) {
                    puzzleTable[this.firstRow][i] = true;
                }
                break;
            case "down":
                for (int i = this.firstRow; i < this.firstRow + this.ansLength; i++) {
                    puzzleTable[i][this.firstCol] = true;
                }
                break;
            case "up":
                start = (p.getRow() - 1) - this.firstRow;
                for (int i = start; i >= start - (this.ansLength - 1); i--) {
                    puzzleTable[i][this.firstCol] = true;
                }
                break;
            case "rightUp":
                r = this.firstRow;
                c = this.firstCol;
                for (int i = 0; i < this.ansLength; i++) {
                    puzzleTable[r][c] = true;
                    r--;
                    c++;
                }
                break;
            case "leftDown":
                r = this.firstRow;
                c = this.firstCol;
                for (int i = 0; i < this.ansLength; i++) {
                    puzzleTable[r][c] = true;
                    r++;
                    c--;
                }
                break;
            case "rightDown":
                r = this.firstRow;
                c = this.firstCol;
                for (int i = 0; i < this.ansLength; i++) {
                    puzzleTable[r][c] = true;
                    r++;
                    c++;
                }
                break;
            case "leftUp":
                r = this.firstRow;
                c = this.firstCol;
                for (int i = 0; i < this.ansLength; i++) {
                    puzzleTable[r][c] = true;
                    r--;
                    c--;
                }
                break;

            default:
                System.out.println("not implemented.");
                break;
        }

        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (puzzleTable[i][j]) {
                    if (j == col - 1) {
                        System.out.println(puzzle[i][j]);
                    } else {
                        System.out.print(puzzle[i][j] + " ");
                    }
                } else {
                    if (j == col - 1) {
                        System.out.println("~");
                    } else {
                        System.out.print("~" + " ");
                    }
                }
            }
        }
    }
```

3.2.5. PrintColor.java

```java
public class PrintColor {
    // Reset
    public static final String RESET = "\033[0m"; // Text Reset

    // Regular Colors
    public static final String BLACK = "\033[0;30m"; // BLACK
    public static final String RED = "\033[0;31m"; // RED
    public static final String GREEN = "\033[0;32m"; // GREEN
    public static final String YELLOW = "\033[0;33m"; // YELLOW
    public static final String BLUE = "\033[0;34m"; // BLUE
    public static final String PURPLE = "\033[0;35m"; // PURPLE
    public static final String CYAN = "\033[0;36m"; // CYAN
    public static final String WHITE = "\033[0;37m"; // WHITE

    // High Intensity
    public static final String BLACK_BRIGHT = "\033[0;90m"; // BLACK
    public static final String RED_BRIGHT = "\033[0;91m"; // RED
    public static final String GREEN_BRIGHT = "\033[0;92m"; // GREEN
    public static final String YELLOW_BRIGHT = "\033[0;93m"; // YELLOW
    public static final String BLUE_BRIGHT = "\033[0;94m"; // BLUE
    public static final String PURPLE_BRIGHT = "\033[0;95m"; // PURPLE
    public static final String CYAN_BRIGHT = "\033[0;96m"; // CYAN
    public static final String WHITE_BRIGHT = "\033[0;97m"; // WHITE


    public static void printColor(char s, String color) {
        String colorcode = "";
        switch (color) {
            case "RED":
                colorcode = RED;
                break;
            case "GREEN":
                colorcode = GREEN;
                break;
            case "YELLOW":
                colorcode = YELLOW;
                break;
            case "BLUE":
                colorcode = BLUE;
                break;
            case "PURPLE":
                colorcode = PURPLE;
                break;
            case "CYAN":
                colorcode = CYAN;
                break;
            case "WHITE":
                colorcode = WHITE;
                break;
            case "RED_BRIGHT":
                colorcode = RED_BRIGHT;
                break;
            case "GREEN_BRIGHT":
                colorcode = GREEN_BRIGHT;
                break;
            case "YELLOW_BRIGHT":
                colorcode = YELLOW_BRIGHT;
                break;
            case "BLUE_BRIGHT":
                colorcode = BLUE_BRIGHT;
                break;
            case "PURPLE_BRIGHT":
                colorcode = PURPLE_BRIGHT;
                break;
            case "CYAN_BRIGHT":
                colorcode = CYAN_BRIGHT;
                break;
            case "WHITE_BRIGHT":
                colorcode = WHITE_BRIGHT;
                break;

            default:
                break;
        }
        System.out.print(colorcode + s + RESET);
    }
}
```

BAB 4

MASUKAN DAN LUARAN PROGRAM

1. small1.txt

```
G G B Q T P H S I S A S Q D S J
N W G E T P A S L E E P M A R O
J O T N L N A A O T H U E E V S
H H O K I P U C K C R E C C R R
T E X T D X O R G C G T N R Z M
F D U P E J A M B Z J Y A U F Q
O G D Y F L Q W B R T C R O Q F
G E T Z A T O O Z Y R X P S O Y
X W Q S C T D P T D K O P S V E
X B K P E M O L A T P P W I P V
J T C C Z B E K C D A K L E H N
L F G Q X W D L K R D G Z T R O
I S P Q X R I G T C N Y S O A C
N L Z X S G V C G M P B K D Z T

ASLEEP
PART
BRUNT
PLEB
CONVEY
PRANCE
DEFACE
ROWER
HEDGE
SOURCE
OOZY
VIDEO
PADDY
WAXING
```

```
small1.txt
Pilih output:
1. Single, berwarna
2. Multiple matrix
1
G G B Q T P H S I S A S Q D S J
N W G E T P A S L E E P M A R O
J O T N L N A A O T H U E E V S
H H O K I P U C K C R E C C R R
T E X T D X O R G C G T N R Z M
F D U P E J A M B Z J Y A U F Q
O G D Y F L Q W B R T C R O Q F
G E T Z A T O O Z Y R X P S O Y
X W Q S C T D P T D K O P S V E
X B K P E M O L A T P P W I P V
J T C C Z B E K C D A K L E H N
L F G Q X W D L K R D G Z T R O
I S P Q X R I G T C N Y S O A C
N L Z X S G V C G M P B K D Z T
Time taken: 0.0328424 seconds
Comparison made: 9566
```

2. small2.txt

```
S P T F S E L E C S O S I M
Y C E H U Y S H K C H C T O
X J A R G G J E L K E M E R
I J L L I I E C U J U T T O
E R A N E M R S I F R Y R C
T A R V H N E G U I R T A P
U E E U A E E T A T H M H R
C R T D G R C N E S B I E O
A A A P T I G M E R M O D J
I C L G K L O L L E U V R G
Z E I C E E G L V X W Y O J
L I U O G N C Q W N U P N D
Q N Q D A E E D P Y N Z B J
N A E F H E T U E V Q D U F

ACUTE
GEOMETRY
RIGHT
ANGLES
ISOSCELES
SCALENE
AREA
OBTUSE
TETRAHEDRON
EQUILATERAL
PERIMETER
TRIANGLE
```

```
S P T F S E L E C S O S I M
Y C E H U Y S H K C H C T O
X J A R G G J E L K E M E R
I J L L I I E C U J U T T O
E R A N E M R S I F R Y R C
T A R V H N E G U I R T A P
U E E U A E E T A T H M H R
C R T D G R C N E S B I E O
A A A P T I G M E R M O D J
I C L G K L O L L E U V R G
Z E I C E E G L V X W Y O J
L I U O G N C Q W N U P N D
Q N Q D A E E D P Y N Z B J
N A E F H E T U E V Q D U F
Time taken: 0.0331555 seconds
Comparison made: 5690
```

3. small3.txt

```
N O R T H P O L E O E L N D
E V R O Y V G Y H R L A I E
V Q N V Y Q D X E J L Y P K
U M U C D M O H K E A X I S
W Z X A R T P L D M F J S X
L Q P S T S P C E E L G N A
F A C I I O B W A G L O B E
D L N N D M R E V K I A X D
L Z A D F E L O P H T U O S
J L J F M L E V A R T P N G
P N X Y O A G Y R D O S M J
E A R T H P R C Y X P P J X
Z T O S U F R K N P H A R O
X M D L E V T T S X C M X J

ANGLE
GLOBE
PLANISPHERE
AXIS
LANDMARKS
SOUTHPOLE
EARTH
MAPS
TRAVEL
EQUATOR
NORTHPOLE
```

4. medium1.txt

```
O H E C W W T E I C B Q V O L K H M N D I T
A S S E R T I N G I V Y W S H Y D B U N P Y
G B W R I C N F H S X O R C M J M U N J R N
P L K A S C Q S S E L E C R U O S W M T M J
U O A J Y R Q E E J G N I S I P S E D D T Y
P A I N A B R Q D W W P S W A D S Q F B E Q
I I J C C P R G U U N P U X D Q K G D W B L
O H S O M E F U I C V D H N E W S F L A S H
U O X O U T P M J W F W F G O Z B C N L C G
X G C D D Y T S C B B I X K L Y V Y N V S K
G E A N Y K R G T P G R Y L X A Y P T U O D
D F J A A M O K H C S R W B V A R R U F N R
D R S U M I P A G W Q U S C O Y B B H D E Z
Q H X J H G S S U F E W R G N X S A C X L L
C U E Z T H S U A H W F A U L A L T S C K I
Z B G E S T A G R G D Q X R A Y I F U H Q R
X L X R A D P Q D U O K Y O P S M P N G K J
A U F B K P B F E A Q L P S L E E R K V P F
Y W K Z E X O L Q L S B H G K X D H S S X K
K X I K O S U I S L E C J T L E E S T T T T

ABASH
ASSERTING
ASTHMA
CELSIUS
DECOMPRESS
DESPISING
DRAUGHT
GLANCE
LAUGH
LEES
LEFTY
MIGHT
NEWSFLASH
OSCAR
PASSPORT
SLIME
SMUG
SOURCELESS
THESAURUS
WARPED
```

```
O H E C W W T E I C B Q V O L K H M N D I T
A S S E R T I N G I V Y W S H Y D B U N P Y
G B W R I C N F H S X O R C M J M U N J R N
P L K A S C Q S S E L E C R U O S W M T M J
U O A J Y R Q E E J G N I S I P S E D D T Y
P A I N A B R Q D W W P S W A D S Q F B E Q
I I J C C P R G U U N P U X D Q K G D W B L
O H S O M E F U I C V D H N E W S F L A S H
U O X O U T P M J W F W F G O Z B C N L C G
X G C D D Y T S C B B I X K L Y V Y N V S K
G E A N Y K R G T P G R Y L X A Y P T U O D
D F J A A M O K H C S R W B V A R R U F N R
D R S U M I P A G W Q U S C O Y B B H D E Z
Q H X J H G S S U F E W R G N X S A C X L L
C U E Z T H S U A H W F A U L A L T S C K I
Z B G E S T A G R G D Q X R A Y I F U H Q R
X L X R A D P Q D U O K Y O P S M P N G K J
A U F B K P B F E A Q L P S L E E R K V P F
Y W K Z E X O L Q L S B H G K X D H S S X K
K X I K O S U I S L E C J T L E E S T T T T
Time taken: 0.0440895 seconds
Comparison made: 22166
```

5. medium2.txt

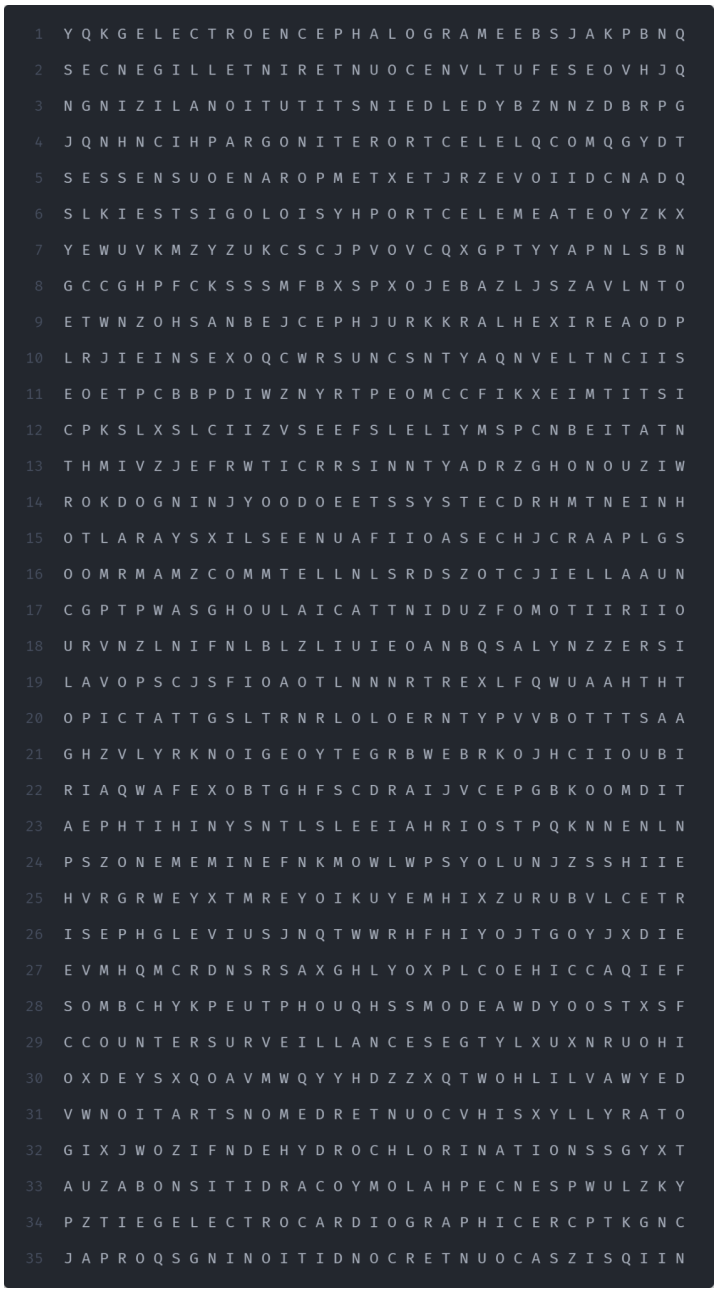
```
N S T R I N I D A D T Q D L B K M R K T B K R Z I
E A E X C O L Y Z G H A V A N A A P U Y S B L D O
W N Q O N J A F O R R M O W F G N R C P H U J B H
D T D A U A Z Z Q X S H R J Z P I G D N B E S E E
E O H T H L C W N O T T A W A E L A K B W N H I L
L D L J D A M A S C U S O A B N A T S O O O A J S
H O D G T Z J O L S V S W C I G V H A Z K S Q I I
I M H F P L J J M M A J C L K S Q E G J H A R N N
R I T B A G H D A D D F R L M H F N H B Q I E G K
S N N Q L Y Y B B X J E F Q P S O S F O T R B J I
A G Z G X M C T R E B V B T B M Z L R F G E D W T
N O E H E P S N O J W K R L E S Z Z M R L S K I O
T X L U X E M B O U R G U I O T U U G Y I W U O E
I L Y X P C D O L O I L S F V N C P F U B Y W M O
A W S A S C A L W P V R S R B I D S M C N T A A X
G A D L W A R S A W E E E A N G R O Y I Z E I D Z
O U P C U H U V B D R M L K N E V B N Z T H T R H
B W J Y C K L U A P A X S L I S N G H N P R P I M
N I V G O J H R R D H N Y G B O A Q I W P A Y D O
W V T F F N G L R F V Z L V C O C L C N E N S F S
X J U D T L G E B K C A Q A O F B J V I E N N A C
X Q F R E O T Y F W R T N K U U F G C A I R O I O
M T G B Y S K V A U N O O M D Q B Y T K D O E Z W
S B C Q M D F Y A N M W H L C F W E I S W O Y E Z
F B M A H A W R O I G C O P E N H A G E N T R E Z

ALGIERS
AMSTERDAM
ATHENS
BAGHDAD
BANGKOK
BEIJING
BEIRUT
BELGRADE
BERLIN
BRUSSELS
BUDAPEST
BUENOSAIRES
CAIRO
COPENHAGEN
DAMASCUS
DUBLIN
HANOI
HAVANA
HELSINKI
KUWAIT
LONDON
LUXEMBOURG
MADRID
MANILA
MONACO
MOSCOW
NEWDELHI
OTTAWA
PYONGYANG
RIYADH
SANSALVADOR
SANTIAGO
SANTODOMINGO
SEOUL
STOCKHOLM
TEHRAN
TOKYO
TRINIDAD
VIENNA
WARSAW
```

```
N S T R I N I D A D T Q D L B K M R K T B K R Z I
E A E X C O L Y Z G H A V A N A A P U Y S B L D O
W N Q O N J A F O R R M O W F G N R C P H U J B H
D T D A U A Z Z Q X S H R J Z P I G D N B E S E E
E O H T H L C W N O T T A W A E L A K B W N H I L
L D L J D A M A S C U S O A B N A T S O O O A J S
H O D G T Z J O L S V S W C I G V H A Z K S Q I I
I M H F P L J J M M A J C L K S Q E G J H A R N N
R I T B A G H D A D D F R L M H F N H B Q I E G K
S N N Q L Y Y B B X J E F Q P S O S F O T R B J I
A G Z G X M C T R E B V B T B M Z L R F G E D W T
N O E H E P S N O J W K R L E S Z Z M R L S K I O
T X L U X E M B O U R G U I O T U U G Y I W U O E
I L Y X P C D O L O I L S F V N C P F U B Y W M O
A W S A S C A L W P V R S R B I D S M C N T A A X
G A D L W A R S A W E E E A N G R O Y I Z E I D Z
O U P C U H U V B D R M L K N E V B N Z T H T R H
B W J Y C K L U A P A X S L I S N G H N P R P I M
N I V G O J H R R D H N Y G B O A Q I W P A Y D O
W V T F F N G L R F V Z L V C O C L C N E N S F S
X J U D T L G E B K C A Q A O F B J V I E N N A C
X Q F R E O T Y F W R T N K U U F G C A I R O I O
M T G B Y S K V A U N O O M D Q B Y T K D O E Z W
S B C Q M D F Y A N M W H L C F W E I S W O Y E Z
F B M A H A W R O I G C O P E N H A G E N T R E Z
Time taken: 0.0658819 seconds
Comparison made: 68992
```

6. medium3.txt

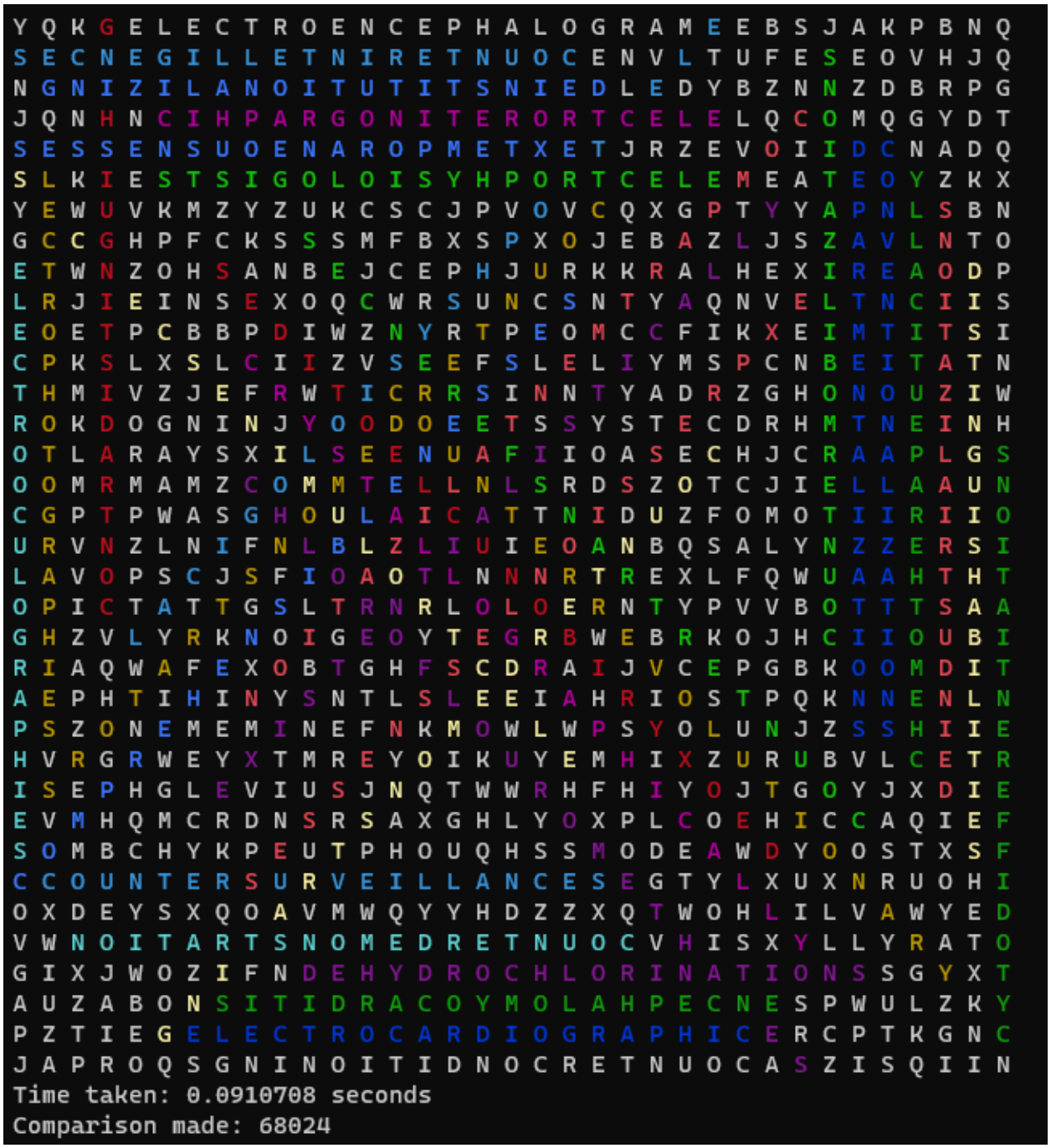
```
L E Z Y D N V Y C A N Q S T A R H K E
I N M E D I O L O B A R A P P A E E R
P I T Q G A O G E T I K G S S A P B A
X L T U O D Q Z Y D E C A G O N T U U
C Y L I N D E R E L S P H E R E A C Q
T W Z L E S M W K P O L C M O L G E S
X O N A L U K S E E A P A A O L O G R
R G O T C B W A I L L R U W A E N N N
W E M E R M S U F R G P T R F N L E O
B L O R I O S E E O P N E P R A F Z G
I G N A C H C L L X C T A N O L N O A
P N G L A R L E G E A O O T T P Q L N
Y A D P A I L T G L C G N X C A S C O
R I E V P L R C I D A S I E E E G D N
A R N S A A Y R U X N L O T S T R O R
M T E R E Y D A E R E U P S N D U T N
I R A H C A C H R H V I O P I I I C T
D P T S U L U N N A L E W R A S O S A
G P N Q A E T M O B I U S S T R I P C

ACUTE
HEART
POLYGON
ANNULUS
HELIX
PRISM
ARC
HEPTAGON
QUADRILATERAL
BIPYRAMID
HEXAGON
RAY
CIRCLE
ISOSCELES
RECTANGLE
CONE
KITE
RHOMBUS
CUBE
LINE
ROUND
CURVE
LOZENGE
SECTOR
CYLINDER
MOBIUSSTRIP
SHAPE
DECAGON
NONAGON
SPHERE
DISC
PARABOLOID
SQUARE
DOT
PARALLELOGRAM
STAR
ELLIPSE
PENTAGON
TRAPEZOID
EQUILATERAL
PLANE
TRIANGLE
GNOMON
POINT
WEDGE
```

7. large1.txt

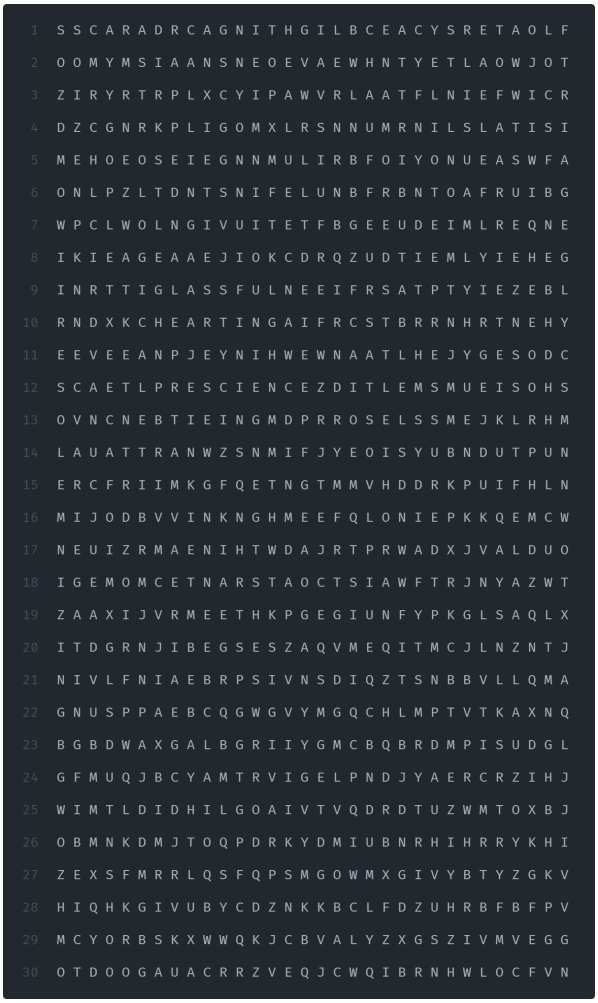


```
1  Y Q K G E L E C T R O E N C E P H A L O G R A M E E B S J A K P B N Q
2  S E C N E G I L L E T N I R E T N U O C E N V L T U F E S E O V H J Q
3  N G N I Z I L A N O I T U T I T S N I E D L E D Y B Z N N Z D B R P G
4  J Q N H N C I H P A R G O N I T E R O R T C E L E L Q C O M Q G Y D T
5  S E S S E N S U O E N A R O P M E T X E T J R Z E V O I I D C N A D Q
6  S L K I E S T S I G O L O I S Y H P O R T C E L E M E A T E O Y Z K X
7  Y E W U V K M Z Y Z U K C S C J P V O V C Q X G P T Y Y A P N L S B N
8  G C C G H P F C K S S S M F B X S P X O J E B A Z L J S Z A V L N T O
9  E T W N Z O H S A N B E J C E P H J U R K K R A L H E X I R E A O D P
10 L R J I E I N S E X O Q C W R S U N C S N T Y A Q N V E L T N C I I S
11 E O E T P C B B P D I W Z N Y R T P E O M C C F I K X E I M T I T S I
12 C P K S L X S L C I I Z V S E E F S L E L I Y M S P C N B E I T A T N
13 T H M I V Z J E F R W T I C R R S I N N T Y A D R Z G H O N O U Z I W
14 R O K D O G N I N J Y O O D O E E T S S Y S T E C D R H M T N E I N H
15 O T L A R A Y S X I L S E E N U A F I I O A S E C H J C R A A P L G S
16 O O M R M A M Z C O M M T E L L N L S R D S Z O T C J I E L L A A U N
17 C G P T P W A S G H O U L A I C A T T N I D U Z F O M O T I I R I I O
18 U R V N Z L N I F N L B L Z L I U I E O A N B Q S A L Y N Z Z E R S I
19 L A V O P S C J S F I O A O T L N N N R T R E X L F Q W U A A H T H T
20 O P I C T A T T G S L T R N R L O L O E R N T Y P V V B O T T T S A A
21 G H Z V L Y R K N O I G E O Y T E G R B W E B R K O J H C I I O U B I
22 R I A Q W A F E X O B T G H F S C D R A I J V C E P G B K O O M D I T
23 A E P H T I H I N Y S N T L S L E E I A H R I O S T P Q K N N E N L N
24 P S Z O N E M E M I N E F N K M O W L W P S Y O L U N J Z S S H I I E
25 H V R G R W E Y X T M R E Y O I K U Y E M H I X Z U R U B V L C E T R
26 I S E P H G L E V I U S J N Q T W W R H F H I Y O J T G O Y J X D I E
27 E V M H Q M C R D N S R S A X G H L Y O X P L C O E H I C C A Q I E F
28 S O M B C H Y K P E U T P H O U Q H S S M O D E A W D Y O O S T X S F
29 C C O U N T E R S U R V E I L L A N C E S E G T Y L X U X N R U O H I
30 O X D E Y S X Q O A V M W Q Y Y H D Z Z X Q T W O H L I L V A W Y E D
31 V W N O I T A R T S N O M E D R E T N U O C V H I S X Y L L Y R A T O
32 G I X J W O Z I F N D E H Y D R O C H L O R I N A T I O N S S G Y X T
33 A U Z A B O N S I T I D R A C O Y M O L A H P E C N E S P W U L Z K Y
34 P Z T I E G E L E C T R O C A R D I O G R A P H I C E R C P T K G N C
35 J A P R O Q S G N I N O I T I D N O C R E T N U O C A S Z I S Q I I N
```

```
1  CHEMOTHERAPEUTICALLY
2  COUNTERREVOLUTIONARY
3  ELECTROCARDIOGRAPHIC
4  CHLOROFLOUROMETHANES
5  COUNTERSURVEILLANCES
6  ELECTROENCEPHALOGRAM
7  COMPARTMENTALIZATION
8  COUNTERTRANSFERENCES
9  ELECTROLUMINESCENCES
10 COMPREHENSIBLENESSES
11 CRYSTALLOGRAPHICALLY
12 ELECTROOCULOGRAPHIES
13 CONTRADISTINGUISHING
14 CYTODIFFERENTIATIONS
15 ELECTROPHOTOGRAPHIES
16 CONVENTIONALIZATIONS
17 DEHYDROCHLORINATIONS
18 ELECTROPHSYSIOLOGICAL
19 COUNTERCONDITIONINGS
20 DEINDUSTRIALIZATIONS
21 ELECTROPHYSIOLOGISTS
22 COUNTERDEMONSTRATING
23 DEINSTITUTIONALIZING
24 ELECTRORETINOGRAPHIC
25 COUNTERDEMONSTRATION
26 DEOXYRIBONUCLEOTIDES
27 ENCEPHALOMYOCARDITIS
28 COUNTERDEMONSTRATORS
29 DEPARTMENTALIZATIONS
30 EXISTENTIALISTICALLY
31 COUNTERINTELLIGENCES
32 DIMETHYLNITROSAMINES
33 EXPRESSIONLESSNESSES
34 COUNTERMOBILIZATIONS
35 DISTINGUISHABILITIES
36 EXTEMPORANEOUSNESSES
```

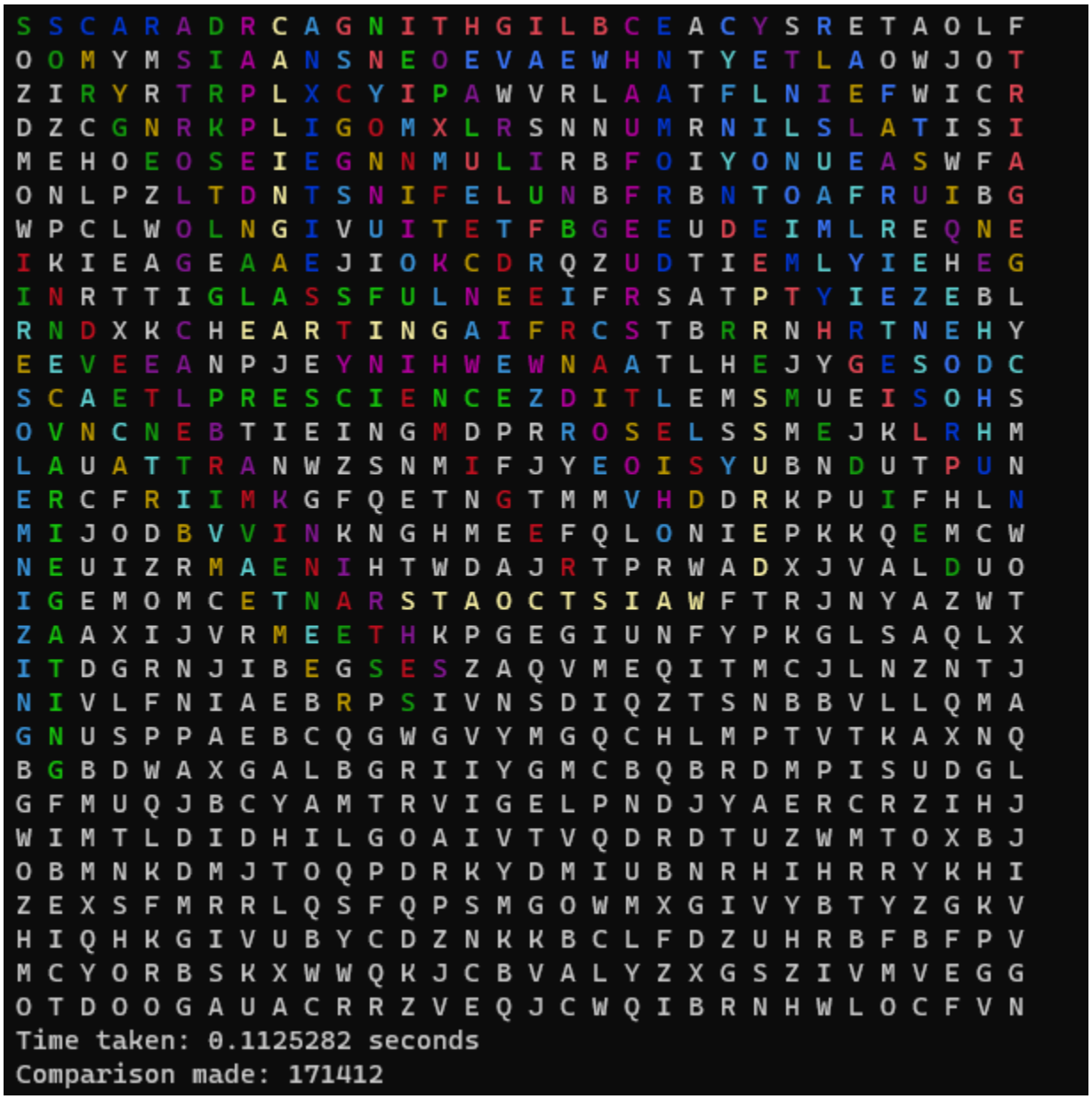

Y Q K G E L E C T R O E N C E P H A L O G R A M E E B S J A K P B N Q
S E C N E G I L L E T N I R E T N U O C E N V L T U F E S E O V H J Q
N G N I Z I L A N O I T U T I T S N I E D L E D Y B Z N N Z D B R P G
J Q N H N C I H P A R G O N I T E R O R T C E L E L Q C O M Q G Y D T
S E S S E N S U O E N A R O P M E T X E T J R Z E V O I I D C N A D Q
S L K I E S T S I G O L O I S Y H P O R T C E L E M E A T E O Y Z K X
Y E W U V K M Z Y Z U K C S C J P V O V C Q X G P T Y Y A P N L S B N
G C C G H P F C K S S S M F B X S P X O J E B A Z L J S Z A V L N T O
E T W N Z O H S A N B E J C E P H J U R K K R A L H E X I R E A O D P
L R J I E I N S E X O Q C W R S U N C S N T Y A Q N V E L T N C I I S
E O E T P C B B P D I W Z N Y R T P E O M C C F I K X E I M T I T S I
C P K S L X S L C I I Z V S E E F S L E L I Y M S P C N B E I T A T N
T H M I V Z J E F R W T I C R R S I N N T Y A D R Z G H O N O U Z I W
R O K D O G N I N J Y O O D O E E T S S Y S T E C D R H M T N E I N H
O T L A R A Y S X I L S E E N U A F I I O A S E C H J C R A A P L G S
O O M R M A M Z C O M M T E L L N L S R D S Z O T C J I E L L A A U N
C G P T P W A S G H O U L A I C A T T N I D U Z F O M O T I I R I I O
U R V N Z L N I F N L B L Z L I U I E O A N B Q S A L Y N Z Z E R S I
L A V O P S C J S F I O A O T L N N N R T R E X L F Q W U A A H T H T
O P I C T A T T G S L T R N R L O L O E R N T Y P V V B O T T T S A A
G H Z V L Y R K N O I G E O Y T E G R B W E B R K O J H C I I O U B I
R I A Q W A F E X O B T G H F S C D R A I J V C E P G B K O O M D I T
A E P H T I H I N Y S N T L S L E E I A H R I O S T P Q K N N E N L N
P S Z O N E M E M I N E F N K M O W L W P S Y O L U N J Z S S H I I E
H V R G R W E Y X T M R E Y O I K U Y E M H I X Z U R U B V L C E T R
I S E P H G L E V I U S J N Q T W W R H F H I Y O J T G O Y J X D I E
E V M H Q M C R D N S R S A X G H L Y O X P L C O E H I C C A Q I E F
S O M B C H Y K P E U T P H O U Q H S S M O D E A W D Y O O S T X S F
C C O U N T E R S U R V E I L L A N C E S E G T Y L X U X N R U O H I
O X D E Y S X Q O A V M W Q Y Y H D Z Z X Q T W O H L I L V A W Y E D
V W N O I T A R T S N O M E D R E T N U O C V H I S X Y L L Y R A T O
G I X J W O Z I F N D E H Y D R O C H L O R I N A T I O N S S G Y X T
A U Z A B O N S I T I D R A C O Y M O L A H P E C N E S P W U L Z K Y
P Z T I E G E L E C T R O C A R D I O G R A P H I C E R C P T K G N C
J A P R O Q S G N I N O I T I D N O C R E T N U O C A S Z I S Q I I N
Time taken: 0.0910708 seconds
Comparison made: 68024

8. large2.txt

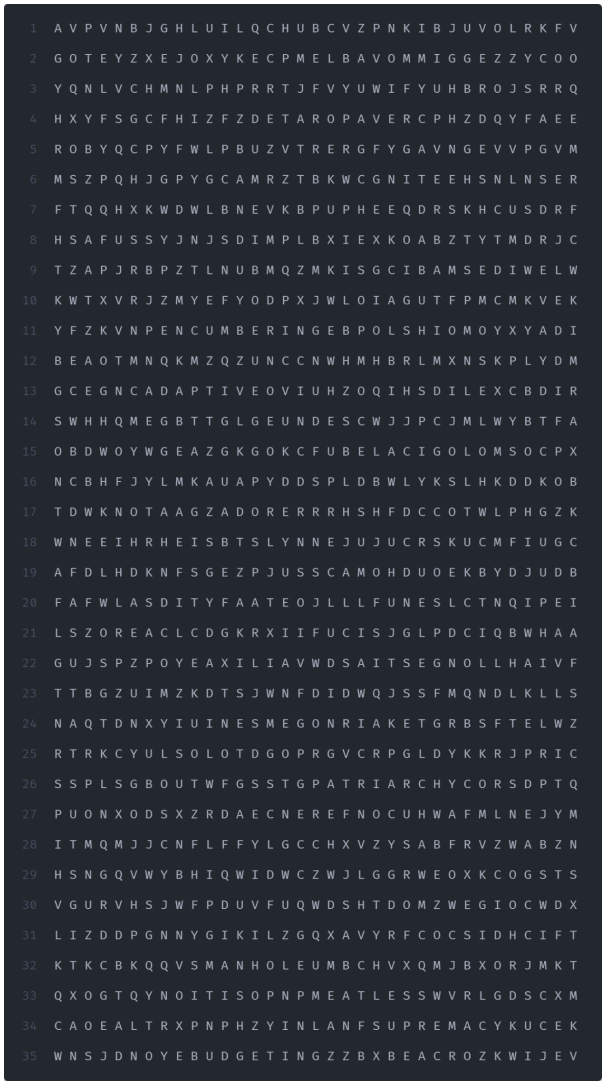


```
S S C A R A D R C A G N I T H G I L B C E A C Y S R E T A O L F
O O M Y M S I A A N S N E O E V A E W H N T Y E T L A O W J O T
Z I R Y R T R P L X C Y I P A W V R L A A T F L N I E F W I C R
D Z C G N R K P L I G O M X L R S N N U M R N I L S L A T I S I
M E H O E O S E I E G N N M U L I R B F O I Y O N U E A S W F A
O N L P Z L T D N T S N I F E L U N B F R B N T O A F R U I B G
W P C L W O L N G I V U I T E T F B G E E U D E I M L R E Q N E
I K I E A G E A A E J I O K C D R Q Z U D T I E M L Y I E H E G
I N R T T I G L A S S F U L N E E I F R S A T P T Y I E Z E B L
R N D X K C H E A R T I N G A I F R C S T B R R N H R T N E H Y
E E V E E A N P J E Y N I H W E W N A A T L H E J Y G E S O D C
S C A E T L P R E S C I E N C E Z D I T L E M S M U E I S O H S
O V N C N E B T I E I N G M D P R R O S E L S S M E J K L R H M
L A U A T T R A N W Z S N M I F J Y E O I S Y U B N D U T P U N
E R C F R I I M K G F Q E T N G T M M V H D D R K P U I F H L N
M I J O D B V V I N K N G H M E E F Q L O N I E P K K Q E M C W
N E U I Z R M A E N I H T W D A J R T P R W A D X J V A L D U O
I G E M O M C E T N A R S T A O C T S I A W F T R J N Y A Z W T
Z A A X I J V R M E E T H K P G E G I U N F Y P K G L S A Q L X
I T D G R N J I B E G S E S Z A Q V M E Q I T M C J L N Z N T J
N I V L F N I A E B R P S I V N S D I Q Z T S N B B V L L Q M A
G N U S P P A E B C Q G W G V Y M G Q C H L M P T V T K A X N Q
B G B D W A X G A L B G R I I Y G M C B Q B R D M P I S U D G L
G F M U Q J B C Y A M T R V I G E L P N D J Y A E R C R Z I H J
W I M T L D I D H I L G O A I V T V Q D R D T U Z W M T O X B J
O B M N K D M J T O Q P D R K Y D M I U B N R H I H R R Y K H I
Z E X S F M R R L Q S F Q P S M G O W M X G I V Y B T Y Z G K V
H I Q H K G I V U B Y C D Z N K K B C L F D Z U H R B F B F P V
M C Y O R B S K X W W Q K J C B V A L Y Z X G S Z I V M V E G G
O T D O O G A U A C R R Z V E Q J C W Q I B R N H W L O C F V N
```

```
ALLEGROS
ANTONYMS
ANXIETIES
ASTROLOGICAL
ASYMMETRICALLY
ATTRIBUTABLE
BLIGHTING
BULLPEN
CALLING
CENSER
CHAUFFEURS
CHEERFULLY
CONFEDERATES
DIRKS
DISINFECTING
ENAMORED
EQUALITY
FINALIZED
FLOATERS
FLUXING
GLASSFUL
HEARTING
HONEYMOON
HOODWINKING
HOSTILITY
INDETERMINATE
INVENTIVENESS
LEASING
NURSERYMEN
OARING
OVERZEALOUS
PHIALLED
PLIGHTED
PRESCIENCE
PRESSURED
RAFT
RAPPED
REACTIVATE
REGIMENTS
REMEDIED
REMEMBRANCE
SCAR
SHRINKABLE
SOLEMNIZING
TIEING
TRIAGE
VARIEGATING
WAISTCOATS
WEAVE
WHINY
```

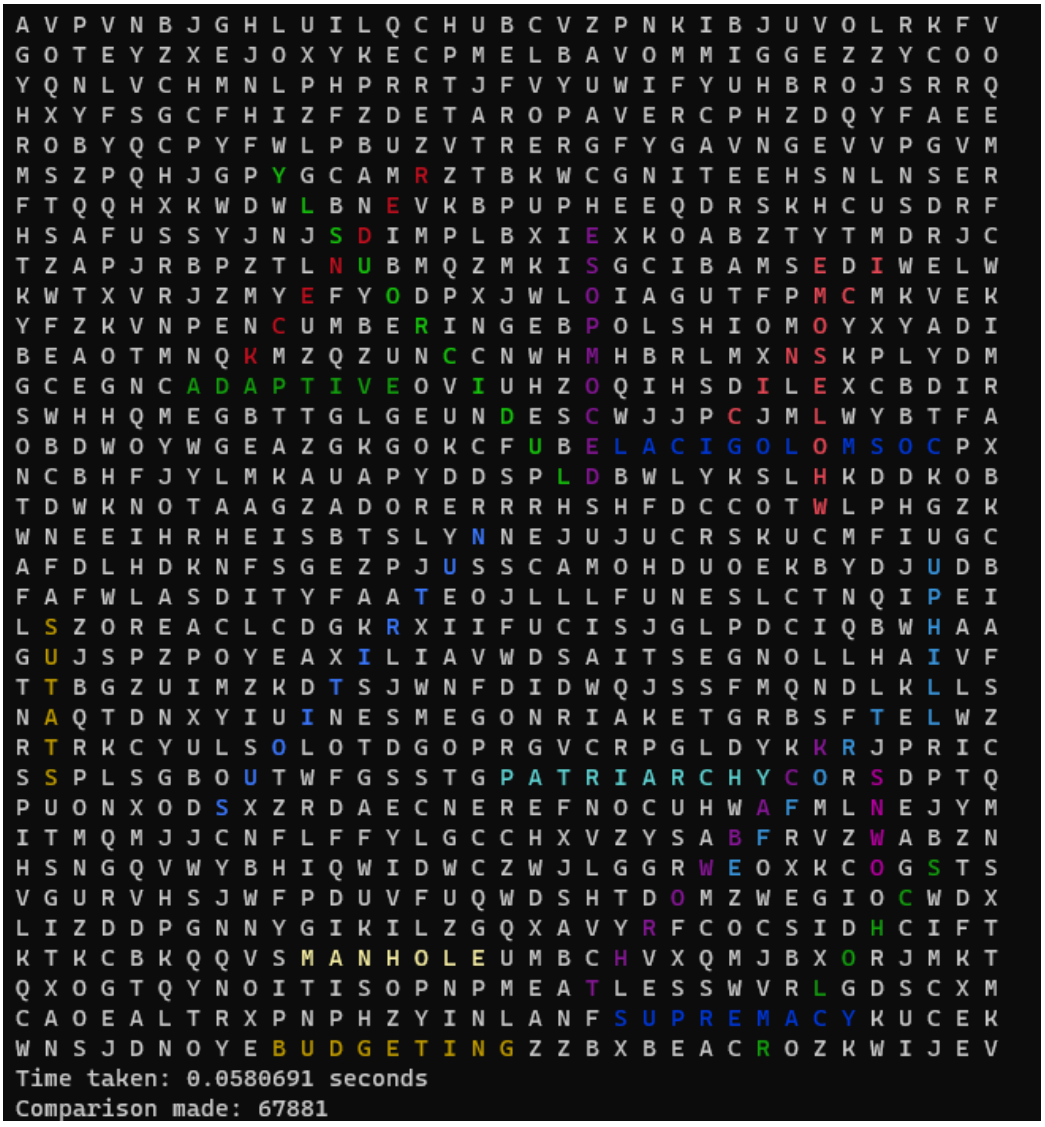

S S C A R A D R C A G N I T H G I L B C E A C Y S R E T A O L F
O O M Y M S I A A N S N E O E V A E W H N T Y E T L A O W J O T
Z I R Y R T R P L X C Y I P A W V R L A A T F L N I E F W I C R
D Z C G N R K P L I G O M X L R S N N U M R N I L S L A T I S I
M E H O E O S E I E G N N M U L I R B F O I Y O N U E A S W F A
O N L P Z L T D N T S N I F E L U N B F R B N T O A F R U I B G
W P C L W O L N G I V U I T E T F B G E E U D E I M L R E Q N E
I K I E A G E A A E J I O K C D R Q Z U D T I E M L Y I E H E G
I N R T T I G L A S S F U L N E E I F R S A T P T Y I E Z E B L
R N D X K C H E A R T I N G A I F R C S T B R R N H R T N E H Y
E E V E E A N P J E Y N I H W E W N A A T L H E J Y G E S O D C
S C A E T L P R E S C I E N C E Z D I T L E M S M U E I S O H S
O V N C N E B T I E I N G M D P R R O S E L S S M E J K L R H M
L A U A T T R A N W Z S N M I F J Y E O I S Y U B N D U T P U N
E R C F R I I M K G F Q E T N G T M M V H D D R K P U I F H L N
M I J O D B V V I N K N G H M E E F Q L O N I E P K K Q E M C W
N E U I Z R M A E N I H T W D A J R T P R W A D X J V A L D U O
I G E M O M C E T N A R S T A O C T S I A W F T R J N Y A Z W T
Z A A X I J V R M E E T H K P G E G I U N F Y P K G L S A Q L X
I T D G R N J I B E G S E S Z A Q V M E Q I T M C J L N Z N T J
N I V L F N I A E B R P S I V N S D I Q Z T S N B B V L L Q M A
G N U S P P A E B C Q G W G V Y M G Q C H L M P T V T K A X N Q
B G B D W A X G A L B G R I I Y G M C B Q B R D M P I S U D G L
G F M U Q J B C Y A M T R V I G E L P N D J Y A E R C R Z I H J
W I M T L D I D H I L G O A I V T V Q D R D T U Z W M T O X B J
O B M N K D M J T O Q P D R K Y D M I U B N R H I H R R Y K H I
Z E X S F M R R L Q S F Q P S M G O W M X G I V Y B T Y Z G K V
H I Q H K G I V U B Y C D Z N K K B C L F D Z U H R B F B F P V
M C Y O R B S K X W W Q K J C B V A L Y Z X G S Z I V M V E G G
O T D O O G A U A C R R Z V E Q J C W Q I B R N H W L O C F V N
Time taken: 0.1125282 seconds
Comparison made: 171412

9. large3.txt



```
A V P V N B J G H L U I L Q C H U B C V Z P N K I B J U V O L R K F V
G O T E Y Z X E J O X Y K E C P M E L B A V O M M I G G E Z Z Y C O O
Y Q N L V C H M N L P H P R R T J F V Y U W I F Y U H B R O J S R R Q
H X Y F S G C F H I Z F Z D E T A R O P A V E R C P H Z D Q Y F A E E
R O B Y Q C P Y F W L P B U Z V T R E R G F Y G A V N G E V V P G V M
M S Z P Q H J G P Y G C A M R Z T B K W C G N I T E E H S N L N S E R
F T Q Q H X K W D W L B N E V K B P U P H E E Q D R S K H C U S D R F
H S A F U S S Y J N J S D I M P L B X I E X K O A B Z T Y T M D R J C
T Z A P J R B P Z T L N U B M Q Z M K I S G C I B A M S E D I W E L W
K W T X V R J Z M Y E F Y O D P X J W L O I A G U T F P M C M K V E K
Y F Z K V N P E N C U M B E R I N G E B P O L S H I O M O Y X Y A D I
B E A O T M N Q K M Z Q Z U N C C N W H M H B R L M X N S K P L Y D M
G C E G N C A D A P T I V E O V I U H Z O Q I H S D I L E X C B D I R
S W H H Q M E G B T T G L G E U N D E S C W J J P C J M L W Y B T F A
O B D W O Y W G E A Z G K G O K C F U B E L A C I G O L O M S O C P X
N C B H F J Y L M K A U A P Y D D S P L D B W L Y K S L H K D D K O B
T D W K N O T A A G Z A D O R E R R R H S H F D C C O T W L P H G Z K
W N E E I H R H E I S B T S L Y N N E J U J U C R S K U C M F I U G C
A F D L H D K N F S G E Z P J U S S C A M O H D U O E K B Y D J U D B
F A F W L A S D I T Y F A A T E O J L L L F U N E S L C T N Q I P E I
L S Z O R E A C L C D G K R X I I F U C I S J G L P D C I Q B W H A A
G U J S P Z P O Y E A X I L I A V W D S A I T S E G N O L L H A I V F
T T B G Z U I M Z K D T S J W N F D I D W Q J S S F M Q N D L K L L S
N A Q T D N X Y I U I N E S M E G O N R I A K E T G R B S F T E L W Z
R T R K C Y U L S O L O T D G O P R G V C R P G L D Y K K R J P R I C
S S P L S G B O U T W F G S S T G P A T R I A R C H Y C O R S D P T Q
P U O N X O D S X Z R D A E C N E R E F N O C U H W A F M L N E J Y M
I T M Q M J J C N F L F F Y L G C C H X V Z Y S A B F R V Z W A B Z N
H S N G Q V W Y B H I Q W I D W C Z W J L G G R W E O X K C O G S T S
V G U R V H S J W F P D U V F U Q W D S H T D O M Z W E G I O C W D X
L I Z D D P G N N Y G I K I L Z G Q X A V Y R F C O C S I D H C I F T
K T K C B K Q Q V S M A N H O L E U M B C H V X Q M J B X O R J M K T
Q X O G T Q Y N O I T I S O P N P M E A T L E S S W V R L G D S C X M
C A O E A L T R X P N P H Z Y I N L A N F S U P R E M A C Y K U C E K
W N S J D N O Y E B U D G E T I N G Z Z B X B E A C R O Z K W I J E V
```

```
ADAPTIVE
BUDGETING
COSMOLOGICAL
DECOMPOSE
EFFORT
FORNICATED
ICONIC
LUDICROUSLY
MANHOLE
NUTRITIOUS
OWNS
PATRIARCHY
REDNECK
SCHOLAR
STATUS
SUPREMACY
THROWBACK
UPHILL
VERBATIM
WHOLESOME
```


A V P V N B J G H L U I L Q C H U B C V Z P N K I B J U V O L R K F V
G O T E Y Z X E J O X Y K E C P M E L B A V O M M I G G E Z Z Y C O O
Y Q N L V C H M N L P H P R R T J F V Y U W I F Y U H B R O J S R R Q
H X Y F S G C F H I Z F Z D E T A R O P A V E R C P H Z D Q Y F A E E
R O B Y Q C P Y F W L P B U Z V T R E R G F Y G A V N G E V V P G V M
M S Z P Q H J G P Y G C A M R Z T B K W C G N I T E E H S N L N S E R
F T Q Q H X K W D W L B N E V K B P U P H E E Q D R S K H C U S D R F
H S A F U S S Y J N J S D I M P L B X I E X K O A B Z T Y T M D R J C
T Z A P J R B P Z T L N U B M Q Z M K I S G C I B A M S E D I W E L W
K W T X V R J Z M Y E F Y O D P X J W L O I A G U T F P M C M K V E K
Y F Z K V N P E N C U M B E R I N G E B P O L S H I O M O Y X Y A D I
B E A O T M N Q K M Z Q Z U N C C N W H M H B R L M X N S K P L Y D M
G C E G N C A D A P T I V E O V I U H Z O Q I H S D I L E X C B D I R
S W H H Q M E G B T T G L G E U N D E S C W J J P C J M L W Y B T F A
O B D W O Y W G E A Z G K G O K C F U B E L A C I G O L O M S O C P X
N C B H F J Y L M K A U A P Y D D S P L D B W L Y K S L H K D D K O B
T D W K N O T A A G Z A D O R E R R R H S H F D C C O T W L P H G Z K
W N E E I H R H E I S B T S L Y N N E J U J U C R S K U C M F I U G C
A F D L H D K N F S G E Z P J U S S C A M O H D U O E K B Y D J U D B
F A F W L A S D I T Y F A A T E O J L L L F U N E S L C T N Q I P E I
L S Z O R E A C L C D G K R X I I F U C I S J G L P D C I Q B W H A A
G U J S P Z P O Y E A X I L I A V W D S A I T S E G N O L L H A I V F
T T B G Z U I M Z K D T S J W N F D I D W Q J S S F M Q N D L K L L S
N A Q T D N X Y I U I N E S M E G O N R I A K E T G R B S F T E L W Z
R T R K C Y U L S O L O T D G O P R G V C R P G L D Y K K R J P R I C
S S P L S G B O U T W F G S S T G P A T R I A R C H Y C O R S D P T Q
P U O N X O D S X Z R D A E C N E R E F N O C U H W A F M L N E J Y M
I T M Q M J J C N F L F F Y L G C C H X V Z Y S A B F R V Z W A B Z N
H S N G Q V W Y B H I Q W I D W C Z W J L G G R W E O X K C O G S T S
V G U R V H S J W F P D U V F U Q W D S H T D O M Z W E G I O C W D X
L I Z D D P G N N Y G I K I L Z G Q X A V Y R F C O C S I D H C I F T
K T K C B K Q Q V S M A N H O L E U M B C H V X Q M J B X O R J M K T
Q X O G T Q Y N O I T I S O P N P M E A T L E S S W V R L G D S C X M
C A O E A L T R X P N P H Z Y I N L A N F S U P R E M A C Y K U C E K
W N S J D N O Y E B U D G E T I N G Z Z B X B E A C R O Z K W I J E V
Time taken: 0.0580691 seconds
Comparison made: 67881

10. Menu Input

```
Masukkan nama file berisi puzzle:
1.txt
Pilih output:
1. Single, berwarna
2. Multiple matrix
1
J S O L U T I S
S U N A R U U A
N E P T U N E T
S O N I E I S U
R C E V T R E R
A H T R A E S N
M M E R C U R Y
Time taken: 0.0197627 seconds
Comparison made: 472
```

```
Masukkan nama file berisi puzzle:
1.txt
Pilih output:
1. Single, berwarna
2. Multiple matrix
2
1. EARTH
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- H T R A E - -
- - - - - - - -
2. JUPITER
J - - - - - - -
- U - - - - - -
- - P - - - - -
- - - I - - - -
- - - - T - - -
- - - - - E - -
- - - - - - R -
3. MARS
- - - - - - - -
- - - - - - - -
- - - - - - - -
S - - - - - - -
R - - - - - - -
A - - - - - - -
M - - - - - - -
4. MERCURY
- - - - - - - -
- - - - - - - -
```

```
4. MERCURY
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- M E R C U R Y
5. NEPTUNE
- - - - - - - -
- - - - - - - -
N E P T U N E -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
6. SATURN
- - - - - - - S
- - - - - - - A
- - - - - - - T
- - - - - - - U
- - - - - - - R
- - - - - - - N
- - - - - - - -
7. URANUS
- - - - - - - -
S U N A R U - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
8. VENUS
- - - - - - - S
- - - - - - U -
- - - - - N - -
- - - - E - - -
- - - V - - - -
- - - - - - - -
- - - - - - - -
```

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf

http://www.whenwewordsearch.com/

LAMPIRAN

Checklist Fitur

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	