

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

DAFTAR ISI	2
BAB 1	3
Algoritma Brute Force	3
Word Search Puzzle	3
Algoritma Penyelesaian Word Search Puzzle dengan Pendekatan Brute Force	4
BAB 2	5
Main.java	5
Puzzle.java	5
Result.java	7
Solver.java	9
PrintColor.java	11
Dependencies	11
BAB 3	12
Repository Program	12
Source Code Program	12
BAB 4	19
REFERENSI	30

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
<code>private char[][] puzzle</code>	Atribut ini berisi puzzle word search dalam bentuk array of array of char.
<code>private String[] words</code>	Atribut ini berisi daftar kata yang harus dicari.
<code>private int row</code>	Atribut ini berisi jumlah baris pada puzzle.
<code>private int col</code>	Atribut ini berisi jumlah kolom pada puzzle.
<code>private int wordsLength</code>	Atribut ini berisi jumlah kata yang harus dicari.
<code>private String dir</code>	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(Parameters)	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.

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

2.3 Result.java

File ini berisi class dari objek Result.

- **Attributes**

Attribute	Description
<code>private final int firstCol</code>	Atribut ini berisi kolom pertama dari jawaban.
<code>private final int firstRow</code>	Atribut ini berisi baris pertama dari jawaban.
<code>private final int ansLength</code>	Atribut ini berisi panjang jawaban.

<code>private final String type</code>	Atribut ini berisi jenis jawaban (kiri, kanan, diagonal, dll)
----------------------------------------	---------------------------------------------------------------

- Constructors

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

- Methods

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

<pre>public void printResult(Puzzle p)</pre>	<p>Metode ini berfungsi untuk menampilkan jawaban pada layar satu per satu setiap kata.</p>
----------------------------------------------	---------------------------------------------------------------------------------------------

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

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

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

2.5 PrintColor.java

Berisi atribut untuk print dengan warna.

<pre>public static void printColor(char s, String color)</pre>	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/gagaspahar/bruteforce-word-puzzle-solver>
- Google Drive:

3.2. Source Code Program

3.2.1. Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static Scanner sc;
5
6     public static void main(String[] args) {
7         sc = new Scanner(System.in);
8
9         System.out.println("Masukkan nama file berisi puzzle:");
10        String filename = sc.nextLine();
11        Puzzle puzzle = new Puzzle();
12        puzzle.fileInput(filename);
13
14        long start = System.nanoTime();
15        Result[] resultlist = new Result[puzzle.getWordsLength()];
16        for (int i = 0; i < puzzle.getWordsLength(); i++) {
17            resultlist[i] = Solver.allWordCheck(puzzle, puzzle.getWords()[i]);
18        }
19
20        long end = System.nanoTime();
21
22        System.out.println("Pilih output: ");
23        System.out.println("1. Single, berwarna");
24        System.out.println("2. Multiple matrix");
25        String prompt = sc.nextLine();
26
27        switch (prompt) {
28            case "1":
29                Result.printPuzzleWithColor(puzzle, resultlist);
30                break;
31            case "2":
32                for (int i = 0; i < puzzle.getWordsLength(); i++) {
33                    System.out.printf("%d. %s\n", i + 1, puzzle.getWords()[i]);
34                    resultlist[i].printResult(puzzle);
35                }
36            default:
37                Result.printPuzzleWithColor(puzzle, resultlist);
38                break;
39        }
40        long time = (end - start);
41        double timeElapsed = (double) time / 1_000_000_000;
42
43        System.out.println("Time taken: " + timeElapsed + " seconds");
44        System.out.printf("Comparison made: %d\n", puzzle.getComparison());
45        sc.close();
46    }
47 }
```

3.2.2. Puzzle.java

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.Scanner;
4
5 public class Puzzle {
6     private char[][] puzzle;
7     private String[] words;
8     private int row;
9     private int col;
10    private int wordsLength;
11    private String dir = "..\\\\test";
12    private int CAPACITY = 500;
13    private int comparison;
14
15    public Puzzle() {
16        this.puzzle = new char[CAPACITY][CAPACITY];
17        this.words = new String[CAPACITY];
18        this.row = 0;
19        this.col = 0;
20        this.wordsLength = 0;
21        this.comparison = 0;
22    }
23
24    public char[][] getPuzzle() {
25        return this.puzzle;
26    }
27
28    public String[] getWords() {
29        return this.words;
30    }
31
32    public int.getRow() {
33        return this.row;
34    }
35
36    public int.getCol() {
37        return this.col;
38    }
39
40    public int.getWordsLength() {
41        return this.wordsLength;
42    }
43
44    public int.getCapacity() {
45        return this.CAPACITY;
46    }
47
48    public int.getComparison() {
49        return this.comparison;
50    }
51
52    public void.setComparison(int x) {
53        this.comparison = x;
54    }
55}
```

```
1     public void fileInput(String filename) {
2         try {
3             String path = dir + "\\\" + filename;
4             File file = new File(path);
5             Scanner scfile = new Scanner(file);
6             int i = 0;
7             while (scfile.hasNextLine()) {
8                 String line = scfile.nextLine();
9                 if (line.isEmpty()) {
10                     break;
11                 }
12                 String[] lineArray = line.split(" ");
13                 this.col = lineArray.length;
14                 for (int j = 0; j < lineArray.length; j++) {
15                     this.puzzle[i][j] = lineArray[j].charAt(0);
16                 }
17                 i++;
18             }
19             this.row = i;
20             i = 0;
21             while (scfile.hasNextLine()) {
22                 String line = scfile.nextLine();
23                 String noSpaceStr = line.replaceAll("\\s", " ");
24                 this.words[i] = noSpaceStr.trim();
25                 i++;
26             }
27             this.wordsLength = i;
28             scfile.close();
29         } catch (FileNotFoundException e) {
30             System.out.println("File tidak ditemukan.");
31             System.exit(0);
32         }
33     }
34
35     public void.printPuzzle() {
36         for (int i = 0; i < this.row; i++) {
37             for (int j = 0; j < this.col; j++) {
38                 if (j == col - 1) {
39                     System.out.println(this.puzzle[i][j]);
40                 } else {
41                     System.out.print(this.puzzle[i][j] + " ");
42                 }
43             }
44         }
45     }
46 }
47 }
```

3.2.3. Solver.java

```

1 import java.util.Arrays;
2
3 public class Solver {
4
5     public static int lineMatcher(String text, String pattern, Puzzle p) {
6         int ans = -1;
7         text = text.trim();
8         int tLen = text.trim().length();
9         int pLen = pattern.length();
10        int comp = 0;
11        if (pLen <= tLen) {
12            for (int i = 0; i < tLen - pLen; i++) {
13                int j = 0;
14                while (j < pLen) {
15                    comp++;
16                    if (text.charAt(i + j) == pattern.charAt(j)) {
17                        j++;
18                    } else {
19                        break;
20                    }
21                }
22                if (j == pLen) {
23                    ans = 1;
24                }
25            }
26        }
27        p.setComparison(p.getComparison() + comp);
28        return ans;
29    }
30
31    private static String reverseCharArray(char[] s) {
32        String ans = "";
33        for (int i = s.length - 1; i >= 0; i--) {
34            ans += s[i];
35        }
36        return ans;
37    }
38
39    public static String reverseString(String s) {
40        String ans = "";
41        for (int i = s.length() - 1; i >= 0; i--) {
42            ans += s.charAt(i);
43        }
44        return ans;
45    }

```

```

1 public static Result wordCheckRight(Puzzle p, String word) {
2     Result ans = new Result();
3     int first;
4     int ansFirst = -1;
5     int ansRow = -1;
6     for (int j = 0; j < p.getRow(); j++) {
7         first = lineMatcher(String.valueOf(p.getPuzzle()[j]), word, p);
8         if (first != -1) {
9             ansRow = j;
10            ansFirst = first;
11            break;
12        }
13    }
14    if (ansFirst != -1) {
15        ans = new Result(ansRow, ansFirst, word.length(), "right");
16    }
17    return ans;
18}
19
20 public static Result wordCheckLeft(Puzzle p, String word) {
21     Result ans = new Result();
22     int first;
23     int ansFirst = -1;
24     int ansRow = -1;
25     for (int j = 0; j < p.getRow(); j++) {
26         first = lineMatcher(reverseCharArray(p.getPuzzle()[j]), word, p);
27         if (first != -1) {
28             ansRow = j;
29             ansFirst = first;
30             break;
31         }
32     }
33     if (ansFirst != -1) {
34         ans = new Result(ansRow, ansFirst, word.length(), "left");
35     }
36     return ans;
37 }

```

```

1 public static Result wordCheckDown(Puzzle p, String word) {
2     Result ans = new Result();
3     int first;
4     int ansFirst = -1;
5     int ansCol = -1;
6     for (int j = 0; j < p.getCol(); j++) {
7         String text = "";
8         for (int i = 0; i < p.getRow(); i++) {
9             text += p.getPuzzle()[i][j];
10        }
11        first = lineMatcher(text, word, p);
12        if (first != -1) {
13            ansCol = j;
14            ansFirst = first;
15            break;
16        }
17    }
18    if (ansFirst != -1) {
19        ans = new Result(ansFirst, ansCol, word.length(), "down");
20    }
21    return ans;
22}
23
24 public static Result wordCheckUp(Puzzle p, String word) {
25     Result ans = new Result();
26     int first;
27     int ansFirst = -1;
28     int ansCol = -1;
29     for (int j = 0; j < p.getCol(); j++) {
30         String text = "";
31         for (int i = 0; i < p.getRow(); i++) {
32             text += p.getPuzzle()[i][j];
33         }
34         first = lineMatcher(reverseString(text), word, p);
35         if (first != -1) {
36             ansCol = j;
37             ansFirst = first;
38             break;
39         }
40     }
41     if (ansFirst != -1) {
42        ans = new Result(ansFirst, ansCol, word.length(), "up");
43    }
44    return ans;
45}

```

```

1 public static Result wordCheckRightUp(Puzzle p, String word) {
2     Result ans = new Result();
3     String[] check = diagonalize(p);
4     int first;
5     int ansCol = -1;
6     int ansRow = -1;
7     int i = 0;
8     int j = 0;
9     for (int k = 0; k < check.length; k++) {
10        first = lineMatcher(String.valueOf(check[k]), word, p);
11        if (first != -1) {
12            ansRow = i - first;
13            ansCol = j + first;
14            break;
15        }
16        if (i != p.getRow() - 1) {
17            i++;
18        } else {
19            j++;
20        }
21    }
22    if (ansCol != -1) {
23        ans = new Result(ansRow, ansCol, word.length(), "rightUp");
24    }
25    return ans;
26}
27
28 public static Result wordCheckLeftDown(Puzzle p, String word) {
29     Result ans = new Result();
30     String[] check = diagonalize(p);
31     int first;
32     int ansCol = -1;
33     int ansRow = -1;
34     int i = 0;
35     int j = 0;
36     for (int k = 0; k < check.length; k++) {
37         first = lineMatcher(String.valueOf(reverseString(check[k])), word, p);
38         if (first != -1) {
39             ansRow = i - (check[k].trim().length() - 1) + first;
40             ansCol = j + (check[k].trim().length() - 1) - first;
41             break;
42         }
43         if (i != p.getRow() - 1) {
44             i++;
45         } else {
46             j++;
47         }
48    }
49    if (ansCol != -1) {
50        ans = new Result(ansRow, ansCol, word.length(), "leftDown");
51    }
52    return ans;
53}

```

```

1 public static Result wordCheckRightDown(Puzzle p, String word) {
2     Result ans = new Result();
3     String[] check = diagonalizeRightDown(p);
4     int first;
5     int ansCol = -1;
6     int ansRow = -1;
7     int i = p.getRow() - 1;
8     int j = 0;
9     for (int k = 0; k < check.length; k++) {
10         first = lineMatcher(String.valueOf(check[k]), word, p);
11         if (first != -1) {
12             ansRow = i + first;
13             ansCol = j + first;
14             break;
15         }
16         if (i != 0) {
17             i--;
18         } else {
19             j++;
20         }
21     }
22     if (ansCol != -1) {
23         ans = new Result(ansRow, ansCol, word.length(), "rightDown");
24     }
25     return ans;
26 }
27
28 public static Result wordCheckLeftUp(Puzzle p, String word) {
29     Result ans = new Result();
30     String[] check = diagonalizeRightDown(p);
31     int first;
32     int ansCol = -1;
33     int ansRow = -1;
34     int i = p.getRow() - 1;
35     int j = 0;
36     for (int k = 0; k < check.length; k++) {
37         first = lineMatcher(String.valueOf(reverseString(check[k])), word, p);
38         if (first != -1) {
39             ansRow = i + (check[k].trim().length() - 1 - first);
40             ansCol = j + (check[k].trim().length() - 1 - first);
41             break;
42         }
43         if (i != 0) {
44             i--;
45         } else {
46             j++;
47         }
48     }
49     if (ansCol != -1) {
50         ans = new Result(ansRow, ansCol, word.length(), "leftUp");
51     }
52     return ans;
53 }
```

```

1 public static Result allWordCheck(Puzzle p, String word) {
2     Result res = new Result();
3     while (res.getType() == "") {
4         res = wordCheckRight(p, word);
5         if(res.getType() != ""){
6             break;
7         }
8         res = wordCheckLeft(p, word);
9         if(res.getType() != ""){
10            break;
11        }
12        res = wordCheckUp(p, word);
13        if(res.getType() != ""){
14            break;
15        }
16        res = wordCheckDown(p, word);
17        if(res.getType() != ""){
18            break;
19        }
20        res = wordCheckRightUp(p, word);
21        if(res.getType() != ""){
22            break;
23        }
24        res = wordCheckLeftDown(p, word);
25        if(res.getType() != ""){
26            break;
27        }
28        res = wordCheckRightDown(p, word);
29        if(res.getType() != ""){
30            break;
31        }
32        res = wordCheckLeftUp(p, word);
33        if(res.getType() != ""){
34            break;
35        }
36        break;
37    }
38    return res;
39 }
```

```

1  public static String[] diagonalize(Puzzle p) {
2      char[][] puzzle = p.getPuzzle();
3      int min = Math.min(p.getCol(), p.getRow()) * 2;
4      String[] ans = new String[min+5];
5      Arrays.fill(ans, "");
6      String temp = "";
7      int i = 0;
8      int j = 0;
9      int k = 0;
10     int l = 1;
11     while (k != p.getRow()) {
12         temp = "";
13         do {
14             temp += puzzle[i][j];
15             i--;
16             j++;
17         } while (i >= 0);
18         ans[k] = temp;
19         k++;
20         i = k;
21         j = 0;
22     }
23     i = p.getRow() - 1;
24     j = l;
25     while (l != p.getCol()) {
26         temp = "";
27         do {
28             temp += puzzle[i][j];
29             i--;
30             j++;
31         } while (i >= 0);
32         ans[k] = temp;
33         k++;
34         l++;
35         i = p.getRow() - 1;
36         j = l;
37     }
38     return ans;
39 }
40
41 public static String[] diagonalizeRightDown(Puzzle p) {
42     char[][] puzzle = p.getPuzzle();
43     int min = Math.min(p.getCol(), p.getRow()) * 2;
44     if (p.getCol() == p.getRow()) {
45         min--;
46     }
47     String[] ans = new String[min+5];
48     Arrays.fill(ans, "");
49     String temp = "";
50     int i = p.getRow() - 1;
51     int j = 0;
52     int count = 0;
53     int k = i;
54     int l = 1;
55     while (k != -1) {
56         temp = "";
57         do {
58             temp += puzzle[i][j];
59             i++;
60             j++;
61         } while (i < p.getRow());
62         ans[count] = temp;
63         count++;
64         k--;
65         i = k;
66         j = 0;
67     }
68     i = 0;
69     j = l;
70     while (l != p.getCol()) {
71         temp = "";
72         do {
73             temp += puzzle[i][j];
74             i++;
75             j++;
76         } while (i < p.getRow());
77         ans[count] = temp;
78         count++;
79         l++;
80         i = 0;
81         j = l;
82     }
83     return ans;
84 }
85 }
```

3.2.4. Result.java

```

1  public class Result {
2      private final int firstCol;
3      private final int firstRow;
4      private final int ansLength;
5      private final String type;
6
7      public Result(int firstRow, int firstCol, int ansLength, String type) {
8          this.firstCol = firstCol;
9          this.firstRow = firstRow;
10         this.ansLength = ansLength;
11         this.type = type;
12     }
13
14     public Result() {
15         this.firstCol = -1;
16         this.firstRow = -1;
17         this.ansLength = -1;
18         this.type = "";
19     }
20
21     public int getFirstCol() {
22         return this.firstCol;
23     }
24
25     public int getFirstRow() {
26         return this.firstRow;
27     }
28
29     public int getAnsLength() {
30         return this.ansLength;
31     }
32
33     public String getType() {
34         return this.type;
35     }
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
}
}

1  public void printResult(Puzzle p) {
2      int row = p.getRow();
3      int col = p.getCol();
4      int start;
5      int r;
6      int c;
7      char[][] puzzle = p.getPuzzle();
8      boolean puzzleTable[][] = new boolean[row][col];
9      switch (this.type) {
10         case "right":
11             for (int i = this.firstCol; i < this.firstCol + this.ansLength; i++) {
12                 puzzleTable[this.firstRow][i] = true;
13             }
14             break;
15         case "left":
16             start = (p.getCol() - 1) - this.firstCol;
17             for (int i = start; i > start - (this.ansLength - 1); i--) {
18                 puzzleTable[this.firstRow][i] = true;
19             }
20             break;
21         case "down":
22             for (int i = this.firstRow; i < this.firstRow + this.ansLength; i++) {
23                 puzzleTable[i][this.firstCol] = true;
24             }
25             break;
26         case "up":
27             start = (p.getRow() - 1) - this.firstRow;
28             for (int i = start; i > start - (this.ansLength - 1); i--) {
29                 puzzleTable[i][this.firstCol] = true;
30             }
31             break;
32         case "rightUp":
33             r = this.firstRow;
34             c = this.firstCol;
35             for (int i = 0; i < this.ansLength; i++) {
36                 puzzleTable[r][c] = true;
37                 r--;
38                 c++;
39             }
40             break;
41         case "leftDown":
42             r = this.firstRow;
43             c = this.firstCol;
44             for (int i = 0; i < this.ansLength; i++) {
45                 puzzleTable[r][c] = true;
46                 r++;
47                 c--;
48             }
49             break;
50         case "rightDown":
51             r = this.firstRow;
52             c = this.firstCol;
53             for (int i = 0; i < this.ansLength; i++) {
54                 puzzleTable[r][c] = true;
55                 r++;
56                 c++;
57             }
58             break;
59         case "leftUp":
60             r = this.firstRow;
61             c = this.firstCol;
62             for (int i = 0; i < this.ansLength; i++) {
63                 puzzleTable[r][c] = true;
64                 r--;
65                 c--;
66             }
67             break;
68         default:
69             System.out.println("not implemented.");
70             break;
71     }
72
73     for (int i = 0; i < row; i++) {
74         for (int j = 0; j < col; j++) {
75             if (puzzleTable[i][j]) {
76                 if (j == col - 1) {
77                     System.out.println(puzzle[i][j]);
78                 } else {
79                     System.out.print(puzzle[i][j] + " ");
80                 }
81             } else {
82                 if (j == col - 1) {
83                     System.out.println("-");
84                 } else {
85                     System.out.print("-" + " ");
86                 }
87             }
88         }
89     }
90 }
91 }
```

3.2.5. PrintColor.java

```
1  public class PrintColor {
2      // Reset
3      public static final String RESET = "\u001B[0m"; // Text Reset
4
5      // Regular Colors
6      public static final String BLACK = "\u001B[0;30m"; // BLACK
7      public static final String RED = "\u001B[0;31m"; // RED
8      public static final String GREEN = "\u001B[0;32m"; // GREEN
9      public static final String YELLOW = "\u001B[0;33m"; // YELLOW
10     public static final String BLUE = "\u001B[0;34m"; // BLUE
11     public static final String PURPLE = "\u001B[0;35m"; // PURPLE
12     public static final String CYAN = "\u001B[0;36m"; // CYAN
13     public static final String WHITE = "\u001B[0;37m"; // WHITE
14
15     // High Intensity
16     public static final String BLACK_BRIGHT = "\u001B[0;90m"; // BLACK
17     public static final String RED_BRIGHT = "\u001B[0;91m"; // RED
18     public static final String GREEN_BRIGHT = "\u001B[0;92m"; // GREEN
19     public static final String YELLOW_BRIGHT = "\u001B[0;93m"; // YELLOW
20     public static final String BLUE_BRIGHT = "\u001B[0;94m"; // BLUE
21     public static final String PURPLE_BRIGHT = "\u001B[0;95m"; // PURPLE
22     public static final String CYAN_BRIGHT = "\u001B[0;96m"; // CYAN
23     public static final String WHITE_BRIGHT = "\u001B[0;97m"; // WHITE
24
25
26     public static void printColor(char s, String color) {
27         String colorcode = "";
28         switch (color) {
29             case "RED":
30                 colorcode = RED;
31                 break;
32             case "GREEN":
33                 colorcode = GREEN;
34                 break;
35             case "YELLOW":
36                 colorcode = YELLOW;
37                 break;
38             case "BLUE":
39                 colorcode = BLUE;
40                 break;
41             case "PURPLE":
42                 colorcode = PURPLE;
43                 break;
44             case "CYAN":
45                 colorcode = CYAN;
46                 break;
47             case "WHITE":
48                 colorcode = WHITE;
49                 break;
50             case "RED_BRIGHT":
51                 colorcode = RED_BRIGHT;
52                 break;
53             case "GREEN_BRIGHT":
54                 colorcode = GREEN_BRIGHT;
55                 break;
56             case "YELLOW_BRIGHT":
57                 colorcode = YELLOW_BRIGHT;
58                 break;
59             case "BLUE_BRIGHT":
60                 colorcode = BLUE_BRIGHT;
61                 break;
62             case "PURPLE_BRIGHT":
63                 colorcode = PURPLE_BRIGHT;
64                 break;
65             case "CYAN_BRIGHT":
66                 colorcode = CYAN_BRIGHT;
67                 break;
68             case "WHITE_BRIGHT":
69                 colorcode = WHITE_BRIGHT;
70                 break;
71             default:
72                 break;
73         }
74         System.out.print(colorcode + s + RESET);
75     }
76 }
77 }
```

BAB 4

MASUKAN DAN LUARAN PROGRAM

1. small1.txt

```
1 G G B Q T P H S I S A S Q D S J
2 N W G E T P A S L E E P M A R O
3 J O T N L N A A O T H U E E V S
4 H H O K I P U C K C R E C C R R
5 T E X T D X O R G C G T N R Z M
6 F D U P E J A M B Z J Y A U F Q
7 O G D Y F L Q W B R T C R O Q F
8 G E T Z A T O O Z Y R X P S O Y
9 X W Q S C T D P T D K O P S V E
10 X B K P E M O L A T P P W I P V
11 J T C C Z B E K C D A K L E H N
12 L F G Q X W D L K R D G Z T R O
13 I S P Q X R I G T C N Y S O A C
14 N L Z X S G V C G M P B K D Z T
15
16 ASLEEP
17 PART
18 BRUNT
19 PLEB
20 CONVEY
21 PRANCE
22 DEFACE
23 ROWER
24 HEDGE
25 SOURCE
26 OOZY
27 VIDEO
28 PADDY
29 WAXING
```

```
masukan nama file diatas please
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

```
1 S P T F S E L E C S O S I M
2 Y C E H U Y S H K C H C T O
3 X J A R G G J E L K E M E R
4 I J L L I I E C U J U T T O
5 E R A N E M R S I F R Y R C
6 T A R V H N E G U I R T A P
7 U E E U A E E T A T H M H R
8 C R T D G R C N E S B I E O
9 A A A P T I G M E R M O D J
10 I C L G K L O L L E U V R G
11 Z E I C E E G L V X W Y O J
12 L I U O G N C Q W N U P N D
13 Q N Q D A E E D P Y N Z B J
14 N A E F H E T U E V Q D U F
15
16 ACUTE
17 GEOMETRY
18 RIGHT
19 ANGLES
20 ISOSCELES
21 SCALENE
22 AREA
23 OBTUSE
24 TETRAHEDRON
25 EQUILATERAL
26 PERIMETER
27 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

```
1 N O R T H P O L E O E L N D
2 E V R O Y V G Y H R L A I E
3 V Q N V Y Q D X E J L Y P K
4 U M U C D M O H K E A X I S
5 W Z X A R T P L D M F J S X
6 L Q P S T S P C E E L G N A
7 F A C I I O B W A G L O B E
8 D L N N D M R E V K I A X D
9 L Z A D F E L O P H T U O S
10 J L J F M L E V A R T P N G
11 P N X Y O A G Y R D O S M J
12 E A R T H P R C Y X P P J X
13 Z T O S U F R K N P H A R O
14 X M D L E V T T S X C M X J
15
16 ANGLE
17 GLOBE
18 PLANISPHERE
19 AXIS
20 LANDMARKS
21 SOUTHPOLE
22 EARTH
23 MAPS
24 TRAVEL
25 EQUATOR
26 NORTHPOLE
```

```
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
Time taken: 0.0244275 seconds
Comparison made: 2683
```

4. medium1.txt

```
1 O H E C W W T E I C B Q V O L K H M N D I T
2 A S S E R T I N G I V Y W S H Y D B U N P Y
3 G B W R I C N F H S X O R C M J M U N J R N
4 P L K A S C Q S S E L E C R U O S W M T M J
5 U O A J Y R Q E E J G N I S I P S E D D T Y
6 P A I N A B R Q D W W P S W A D S Q F B E Q
7 I I J C C P R G U U N P U X D Q K G D W B L
8 O H S O M E F U I C V D H N E W S F L A S H
9 U O X O U T P M J W F W F G O Z B C N L C G
10 X G C D D Y T S C B B I X K L Y V Y N V S K
11 G E A N Y K R G T P G R Y L X A Y P T U O D
12 D F J A A M O K H C S R W B V A R R U F N R
13 D R S U M I P A G W Q U S C O Y B B H D E Z
14 Q H X J H G S S U F E W R G N X S A C X L L
15 C U E Z T H S U A H W F A U L A L T S C K I
16 Z B G E S T A G R G D Q X R A Y I F U H Q R
17 X L X R A D P Q D U O K Y O P S M P N G K J
18 A U F B K P B F E A Q L P S L E E R K V P F
19 Y W K Z E X O L Q L S B H G K X D H S S X K
20 K X I K O S U I S L E C J T L E E S T T T T
21
22 ABASH
23 ASSERTING
24 ASTHMA
25 CELSIUS
26 DECOMPRESS
27 DESPIsing
28 DRAUGHT
29 GLANCE
30 LAUGH
31 LEES
32 LEFTY
33 MIGHT
34 NEWSFLASH
35 OSCAR
36 PASSPORT
37 SLIME
38 SMUG
39 SOURCELESS
40 THESAURUS
41 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

```

1 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
2 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
3 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
4 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
5 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
6 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
7 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
8 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
9 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
10 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
11 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
12 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
13 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
14 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
15 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
16 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
17 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
18 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
19 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
20 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
21 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
22 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
23 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
24 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
25 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
26
27 ALGIERS
28 AMSTERDAM
29 ATHENS
30 BAGHDAD
31 BANGKOK
32 BEIJING
33 BEIRUT
34 BELGRADE
35 BERLIN
36 BRUSSELS
37 BUDAPEST
38 BUENOSAIRES
39 CAIRO
40 COPENHAGEN
41 DAMASCUS
42 DUBLIN
43 HANOI
44 HAVANA
45 HELSINKI
46 KUWAIT
47 LONDON
48 LUXEMBOURG
49 MADRID
50 MANILA
51 MONACO
52 MOSCOW
53 NEWDELHI
54 OTTAWA
55 PYONGYANG
56 RIYADH
57 SANSALVADOR
58 SANTIAGO
59 SANTODOMINGO
60 SEOUL
61 STOCKHOLM
62 TEHRAN
63 TOKYO
64 TRINIDAD
65 VIENNA
66 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

```

1 LEZYDNVYCANQSTARHKE
2 INMEDIOLOBARAPPAEER
3 PITQGAOGETIKGSSAPBA
4 XLTUODQZYDECAGONTUU
5 CYLINDERELSPHEREACQ
6 TWZLESMWKPOLCMOLGES
7 XONALUKSEEAPAALOGR
8 RGOTCBWAILLRUWAENN
9 WEMERMMSUFRGPTRFNLEO
10 BLORIOSEEOPNEPRAFZG
11 IGNACHCLLXCTANOLNOA
12 PNLGLARLEGEOOTTTPQLN
13 YADPAILTGLCGNXCASCO
14 RIEVPLRCIDASIEEEGDN
15 ARNSAAYRUXNLOTSTROR
16 MTEREYDAEREUPSNNDUTN
17 IRAHCACHRHVIOPIIICT
18 DPTSULUNNALEWRASOSA
19 GPNQAETMOBIUSSTRIPC
20
21 ACUTE
22 HEART
23 POLYGON
24 ANNULUS
25 HELIX
26 PRISM
27 ARC
28 HEPTAGON
29 QUADRILATERAL
30 BIPYRAMID
31 HEXAGON
32 RAY
33 CIRCLE
34 ISOSCELES
35 RECTANGLE
36 CONE
37 KITE
38 RHOMBUS
39 CUBE
40 LINE
41 ROUND
42 CURVE
43 LOZENGE
44 SECTOR
45 CYLINDER
46 MOBIUSSTRIP
47 SHAPE
48 DECAGON
49 NONAGON
50 SPHERE
51 DISC
52 PARABOLOID
53 SQUARE
54 DOT
55 PARALLELOGRAM
56 STAR
57 ELLIPSE
58 PENTAGON
59 TRAPEZOID
60 EQUILATERAL
61 PLANE
62 TRIANGLE
63 GNOMON
64 POINT
65 WEDGE

```

The grid contains the following text:

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 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

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 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 T 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 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 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 CHLOROFLUROMETHANES
- 5 COUNTERSURVEILLANCES
- 6 ELECTROENCEPHALogram
- 7 COMPARTMENTALIZATION
- 8 COUNTERTRANSFERENCES
- 9 ELECTROLUMINESCENCES
- 10 COMPREHENSIBILITIES
- 11 CRYSTALLOGRAPHICALLY
- 12 ELECTROOCULOGRAPHIES
- 13 CONTRADISTINGUISHING
- 14 CYTODIFFERENTIATIONS
- 15 ELECTROPHOTOGRAPHIES
- 16 CONVENTIONALIZATIONS
- 17 DEHYDROCHLORINATIONS
- 18 ELECTROPHYSIOLOGICAL
- 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 I 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

1	ALLEGROS
2	ANTONYMS
3	ANXIETIES
4	ASTROLOGICAL
5	ASYMMETRICALLY
6	ATTRIBUTABLE
7	BLIGHTING
8	BULLPEN
9	CALLING
10	CENSER
11	CHAUFFEURS
12	CHEERFULLY
13	CONFEDERATES
14	DIRKS
15	DISINFECTING
16	ENAMORED
17	EQUALITY
18	FINALIZED
19	FLOATERS
20	FLUXING
21	GLASSFUL
22	HEARTING
23	HONEYMOON
24	HOODWINKING
25	HOSTILITY
26	INDETERMINATE
27	INVENTIVENESS
28	LEASING
29	NURSERYMEN
30	OARING
31	OVERZEALOUS
32	PHIALLED
33	PLIGHTED
34	PRESCIENCE
35	PRESSED
36	RAFT
37	RAPPED
38	REACTIVATE
39	REGIMENTS
40	REMEDIED
41	REMEMBRANCE
42	SCAR
43	SHRINKABLE
44	SOLEMNIZING
45	TIEING
46	TRIAGE
47	VARIEGATING
48	WAISTCOATS
49	WEAVE
50	WHINNY

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

```
1 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
2 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
3 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
4 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
5 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
6 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
7 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
8 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
9 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
10 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
11 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
12 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
13 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
14 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
15 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
16 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
17 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
18 W N E E T 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
19 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
20 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
21 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
22 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
23 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
24 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
25 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
26 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
27 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
28 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
29 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
30 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
31 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
32 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
33 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
34 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
35 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
```

- 1 ADAPTIVE
- 2 BUDGETING
- 3 COSMOLOGICAL
- 4 DECOMPOSE
- 5 EFFORT
- 6 FORNICATED
- 7 ICONIC
- 8 LUDICROUSLY
- 9 MANHOLE
- 10 NUTRITIOUS
- 11 OWNS
- 12 PATRIARCHY
- 13 REDNECK
- 14 SCHOLAR
- 15 STATUS
- 16 SUPREMACY
- 17 THROWBACK
- 18 UPHILL
- 19 VERBATIM
- 20 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)-Bag1.pdf)

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-
Bag2.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 <i>running</i>	v	
3. Program dapat membaca file masukan dan menuliskan luaran.	v	
4. Program berhasil menemukan semua kata di dalam puzzle.	v	