Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2021/2022

Penyelesaian *Word Search Puzzle* dengan Algoritma *Brute Force*

Disusun oleh:

Rayhan Kinan Muhannad 13520065

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

1. Penjelasan Algoritma *Brute Force*

Strategi *brute force* merupakan salah satu strategi pemecahan masalah yang dapat dibilang cukup sederhana. Dengan menggunakan strategi *brute force* dalam penyelesaian suatu masalah, maka solusi dari permasalahan tersebut pasti ditemukan walaupun dengan waktu eksekusi yang lama. *Brute force* bukan merupakan suatu algoritma khusus, melainkan cara pendekatan dan strategi yang diambil oleh pemrogram untuk mencari solusi paling optimum dari suatu permasalahan.

Prinsip utama dari strategi *brute force* adalah program akan mencoba semua kemungkinan yang mungkin terjadi untuk mengidentifikasi dari seluruh kemungkinan tersebut, kemungkinan mana saja yang memenuhi syarat untuk menjadi solusi permasalahan. Hal tersebut memastikan bahwa algoritma *brute force* akan menghasilkan solusi, tetapi akan membuat waktu eksekusi menjadi lambat. Normalnya, algoritma *brute force* digunakan saat masukan pengguna tidak terlalu besar dan waktu eksekusi bukanlah hal yang diutamakan. Permasalahan yang dapat diselesaikan dengan algoritma *brute force* umumnya terdapat subpersoalan permutasi, kombinasi, subbagian, ataupun subhimpunan.

Untuk mempercepat waktu eksekusi, suatu algoritma *brute force* dapat dioptimisasi dengan melakukan *pruning*. *Pruning* merupakan optimisasi suatu algoritma dengan mengurangi kemungkinan pencarian yang sudah pasti salah. Dengan dilakukannya *pruning* kompleksitas waktu maupun memori dari algoritma *brute force* tidak akan berubah, namun secara *amortized analysis* algoritma tersebut akan memiliki performa lebih baik. Untuk membuat kompleksitas waktu dari suatu algoritma *brute force* menjadi lebih baik, maka diperlukan teknik *dynamic programming*. Teknik tersebut akan dibahas pada materi-materi selanjutnya.

Permasalahan yang diangkat pada tugas kecil ini adalah bagaimana suatu algoritma dengan pendekatan *brute force* dapat menjadi semua solusi kata di dalam *word search puzzle*. *Word search puzzle* adalah suatu permainan dimana pemain harus mencari seluruh kata yang diberikan di dalam matriks yang berisi karakter. Kata tersebut dapat terletak secara horizontal, vertikal, diagonal, maupun secara terbalik. Jika kata tersebut ditemukan, maka pemain harus menandai kata tersebut dengan garis ataupun lingkaran. Pada program yang telah dibuat penulis, penulis menggunakan *font color* yang berbeda dengan warna asalnya untuk menandai kata yang telah ditemukan.

Untuk menyimulasikan *word search puzzle* ke dalam bentuk informasi yang dapat diproses oleh komputer, maka data yang terdapat pada *puzzle* tersebut haruslah direpresentasikan ke dalam bentuk *abstract data type* (ADT). Penulis merancang dua jenis

ADT yang akan digunakan oleh program ini, yaitu ADT Matriks serta ADT Trie. ADT Matriks digunakan untuk menyimulasikan *grid* karakter tempat kata-kata tersembunyi berada. Penulis memilih ADT Matriks dikarenakan mudahnya melakukan *two dimension cell traversing* ketika melakukan *searching* karakter pada grid. Pada implementasi ADT Matriks, penulis menggunakan struktur data *dynamic list* yang sudah didefinisikan oleh Java. Hal tersebut dikarenakan masukan yang diketik oleh pengguna tidak didefinisikan terlebih dahulu dimensinya. Selain ADT Matriks, penulis juga menggunakan ADT Trie untuk menyimpan dan menguraikan kata yang diberikan. Penulis memilih ADT Trie dikarenakan mudahnya melakukan *retrieval* suatu *string* yang diberikan. Pada ADT Trie, kompleksitas waktu dari algoritma *string retrieval* tidak bergantung dari jumlah *string* yang disimpan pada Trie tersebut. Hal tersebut berbeda dengan penyimpanan *string* menggunakan ADT List, dimana untuk melakukan *retrieval string* diperlukan penyocokkan setiap elemen pada ADT tersebut. Pada ADT Trie, algoritma *string retrieval* bekerja layaknya algoritma *tree traversing*.

Setelah ADT didefinisikan, maka langkah selanjutnya adalah merancang algoritma *brute force* menggunakan ADT Matriks dan ADT Trie. Pertama-tama, sebelum ADT digunakan di dalam algoritma terlebih dahulu ADT tersebut diinstasiasi serta diisi dengan file masukan pengguna. ADT Matriks diisi dengan *grid* karakter serta ADT Trie diisi dengan kumpulan kata yang hendak dicari oleh algoritma. Jika masukan pengguna sudah sesuai dengan ketentuan ADT, maka program akan melanjutkan ke dalam tahap *searching* dan *matching*. Jika tidak, maka program akan berhenti pada tahap ini. Pada tahap *searching* dan *matching*, program akan melakukan enumerasi pada setiap *cell* pada ADT Matriks. Pada setiap enumerasi *cell* tersebut, akan dilakukan kembali enumerasi arah *cell traversing* dengan penunjuk baru sesuai dengan arah mata angin. Setiap kali *cell* berpindah ke *cell* baru pada *cell traversing*, maka program akan mencari *child node* pada ADT Trie yang bersesuaian dengan karakter pada *cell* tersebut. Jika ternyata *child node* tersebut merupakan penanda akhir kata yang hendak dicari, maka program melakukan *backtracking* untuk menandakan bahwa kumpulan *cell* tersebut membentuk kata. Untuk mengefisienkan memori, penulis menggunakan id integer yang berbeda-beda pada setiap kata masukan untuk menandai *cell* tersebut. Tahap *cell traversing* akan berhenti apabila penunjuk sudah diluar jangkauan ADT Matriks atau *child node* pada ADT Trie bernilai *null* (menandakan akhir dari algoritma *trie traversing*). Pada akhir program, seharusnya setiap kemungkinan kumpulan karakter yang dapat membentuk kata pada ADT Matriks sudah dilakukan dikunjungi oleh enumerasi dan dilakukan *string matching* oleh ADT Trie.

Algoritma yang telah dipaparkan dapat dikatakan sebagai realisasi teknik *brute force* dikarenakan adanya enumerasi setiap kemungkinan untuk mencari kemungkinan mana yang berpotensi menjadi solusi pada permasalahan. Meskipun teknik *brute force* bersifat *straightforward*, penulis mengoptimisasi beberapa bagian seperti penggunaan ADT Trie untuk *string matching* agar tidak perlu mencocokkan seluruh kata masukan pengguna serta menambahkan teknik heuristik pemberhentian *cell traversing* ketika tidak ada kemungkinan kata yang *matching* pada ADT Trie dan penunjuk *cell traversing* belum mencapai akhir jangkauan ADT Matriks.

2. *Source Code* dengan Bahasa Java

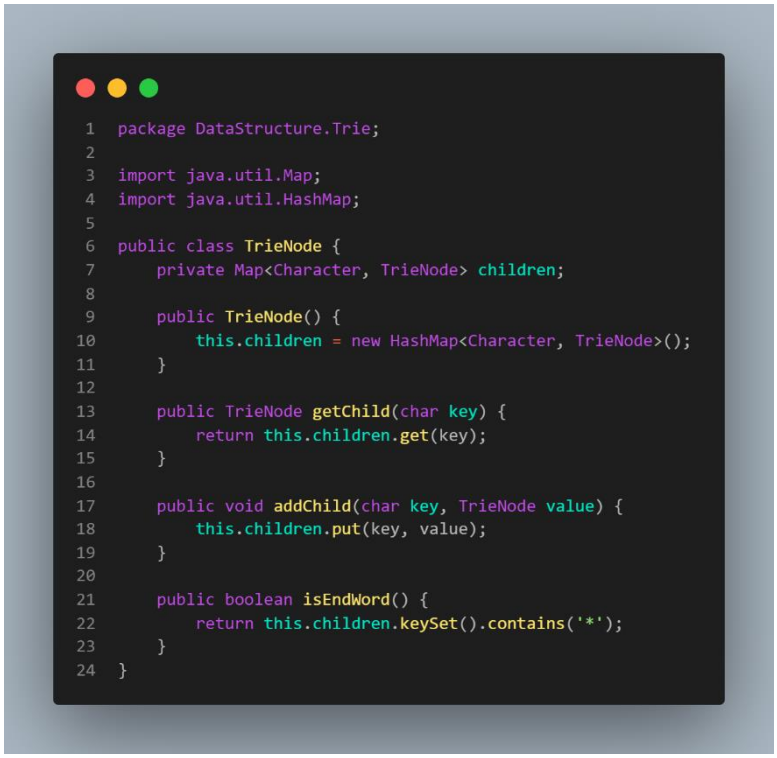
a. ConsoleWriter/ConsoleWriter.java

```java
package ConsoleWriter;

import DataStructure.WordGrid.WordGrid;

public class ConsoleWriter {
    private static final String[] ANSI_COLOR_CODE = {"\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m", "\u001B[35m", "\u001B[36m"};
    private static final String ANSI_RESET = "\u001B[0m";
    private WordGrid wordGrid;

    public ConsoleWriter(WordGrid wordGrid) {
        this.wordGrid = wordGrid;
    }

    public void write() {
        int i, j, length, width;

        length = this.wordGrid.getLength();
        width = this.wordGrid.getWidth();

        for (i = 0; i < length; i++) {
            for (j = 0; j < width; j++) {
                if (this.wordGrid.getGrid(i, j).getWordIndex() != -1) {
                    System.out.print(ANSI_COLOR_CODE[this.wordGrid.getGrid(i, j).getWordIndex() % ANSI_COLOR_CODE.length] + this.wordGrid.getGrid(i, j).getContent() + ANSI_RESET + " ");
                } else {
                    System.out.print(this.wordGrid.getGrid(i, j).getContent() + " ");
                }
            }
            System.out.println();
        }
    }
}
```

b. DataStructure/Trie/Trie.java

```java
package DataStructure.Trie;

public class Trie {
    private TrieNode root;

    public Trie() {
        this.root = new TrieNode();
    }

    public TrieNode getRoot() {
        return this.root;
    }

    public void insertWord(String word) {
        TrieNode curTrieNode = this.root;

        for (char c : word.toCharArray()) {
            if (curTrieNode.getChild(c) != null) {
                curTrieNode = curTrieNode.getChild(c);
            } else {
                TrieNode newTrieNode = new TrieNode();
                curTrieNode.addChild(c, newTrieNode);
                curTrieNode = newTrieNode;
            }
        }

        curTrieNode.addChild('*', null);
    }
}
```

c. DataStructure/Trie/TrieNode.java

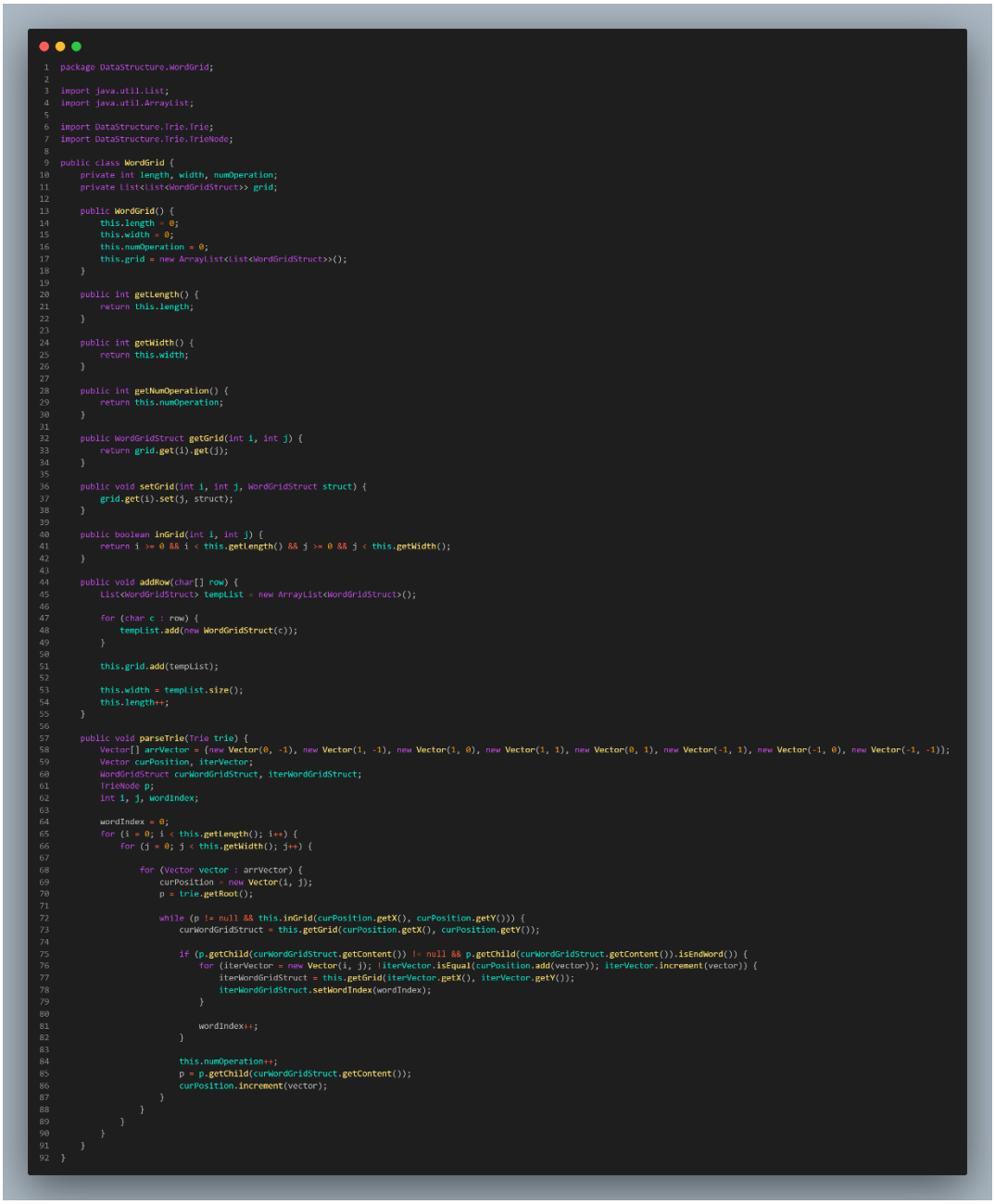


```java
package DataStructure.Trie;

import java.util.Map;
import java.util.HashMap;

public class TrieNode {
    private Map<Character, TrieNode> children;

    public TrieNode() {
        this.children = new HashMap<Character, TrieNode>();
    }

    public TrieNode getChild(char key) {
        return this.children.get(key);
    }

    public void addChild(char key, TrieNode value) {
        this.children.put(key, value);
    }

    public boolean isEndWord() {
        return this.children.keySet().contains('*');
    }
}
```

d. DataStructure/WordGrid/Vector.java

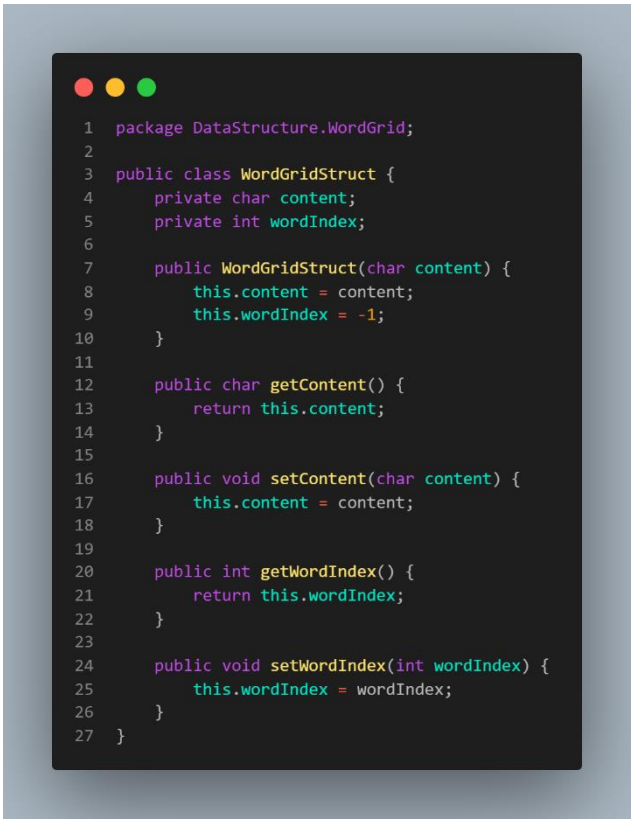
```java
package DataStructure.WordGrid;

public class Vector {
    private int x, y;

    public Vector(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return this.x;
    }

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

    public int getY() {
        return this.y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public boolean isEqual(Vector other) {
        return this.getX() == other.getX() && this.getY() == other.getY();
    }

    public Vector add(Vector other) {
        return new Vector(this.getX() + other.getX(), this.getY() + other.getY());
    }

    public void increment(Vector other) {
        this.setX(this.getX() + other.getX());
        this.setY(this.getY() + other.getY());
    }
}
```

e. DataStructure/WordGrid/WordGrid.java



```java
package DataStructure.WordGrid;

import java.util.List;
import java.util.ArrayList;

import DataStructure.Trie.Trie;
import DataStructure.Trie.TrieNode;

public class WordGrid {
    private int length, width, numOperation;
    private List<List<WordGridStruct>> grid;

    public WordGrid() {
        this.length = 0;
        this.width = 0;
        this.numOperation = 0;
        this.grid = new ArrayList<List<WordGridStruct>>();
    }

    public int getLength() {
        return this.length;
    }

    public int getWidth() {
        return this.width;
    }

    public int getNumOperation() {
        return this.numOperation;
    }

    public WordGridStruct getGrid(int i, int j) {
        return grid.get(i).get(j);
    }

    public void setGrid(int i, int j, WordGridStruct struct) {
        grid.get(i).set(j, struct);
    }

    public boolean inGrid(int i, int j) {
        return i >= 0 && i < this.getLength() && j >= 0 && j < this.getWidth();
    }

    public void addRow(char[] row) {
        List<WordGridStruct> tempList = new ArrayList<WordGridStruct>();

        for (char c : row) {
            tempList.add(new WordGridStruct(c));
        }

        this.grid.add(tempList);

        this.width = tempList.size();
        this.length++;
    }

    public void parseTrie(Trie trie) {
        Vector[] arrVector = {new Vector(0, -1), new Vector(1, -1), new Vector(1, 0), new Vector(1, 1), new Vector(0, 1), new Vector(-1, 1), new Vector(-1, 0), new Vector(-1, -1)};
        Vector curPosition, iterVector;
        WordGridStruct curWordGridStruct, iterWordGridStruct;
        TrieNode p;
        int i, j, wordIndex;

        wordIndex = 0;
        for (i = 0; i < this.getLength(); i++) {
            for (j = 0; j < this.getWidth(); j++) {

                for (Vector vector : arrVector) {
                    curPosition = new Vector(i, j);
                    p = trie.getRoot();

                    while (p != null && this.inGrid(curPosition.getX(), curPosition.getY())) {
                        curWordGridStruct = this.getGrid(curPosition.getX(), curPosition.getY());

                        if (p.getChild(curWordGridStruct.getContent()) != null && p.getChild(curWordGridStruct.getContent()).isEndWord()) {
                            for (iterVector = new Vector(i, j); !iterVector.isEqual(curPosition.add(vector)); iterVector.increment(vector)) {
                                iterWordGridStruct = this.getGrid(iterVector.getX(), iterVector.getY());
                                iterWordGridStruct.setWordIndex(wordIndex);
                            }

                            wordIndex++;
                        }

                        this.numOperation++;
                        p = p.getChild(curWordGridStruct.getContent());
                        curPosition.increment(vector);
                    }
                }
            }
        }
    }
}
```

f. DataStructure/WordGrid/WordGridStruct.java



```java
package DataStructure.WordGrid;

public class WordGridStruct {
    private char content;
    private int wordIndex;

    public WordGridStruct(char content) {
        this.content = content;
        this.wordIndex = -1;
    }

    public char getContent() {
        return this.content;
    }

    public void setContent(char content) {
        this.content = content;
    }

    public int getWordIndex() {
        return this.wordIndex;
    }

    public void setWordIndex(int wordIndex) {
        this.wordIndex = wordIndex;
    }
}
```

g. FileReader/FileReader.java

```java
package FileReader;

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

import DataStructure.Trie.Trie;
import DataStructure.WordGrid.WordGrid;

public class FileReader {
    private File file;
    private Scanner scanner;
    private Trie trie;
    private WordGrid wordGrid;

    public FileReader(String filename) {
        String line;

        try {
            file = new File("test/" + filename);
            scanner = new Scanner(file);

            this.wordGrid = new WordGrid();

            line = scanner.nextLine();
            while (!line.equals("")) {
                wordGrid.addRow(line.replaceAll(" ", "").toCharArray());
                line = scanner.nextLine();
            }

            this.trie = new Trie();

            do {
                line = scanner.nextLine();
                this.trie.insertWord(line.replaceAll(" ", ""));
            } while (scanner.hasNextLine());

        } catch (Exception e) {
            this.wordGrid = null;
            this.trie = null;
        }
    }

    public Trie getTrie() {
        return this.trie;
    }

    public WordGrid getWordGrid() {
        return this.wordGrid;
    }

    public void close() {
        scanner.close();
    }
}
```

h. App.java

```java
import java.util.Scanner;
import java.math.BigDecimal;
import java.math.RoundingMode;

import DataStructure.Trie.Trie;
import DataStructure.WordGrid.WordGrid;
import FileReader.FileReader;
import ConsoleWriter.ConsoleWriter;

public class App {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan nama file: ");
        String filename = scanner.nextLine();

        FileReader fileReader = new FileReader(filename);
        if (fileReader.getWordGrid() != null && fileReader.getTrie() != null) {
            WordGrid wordGrid = fileReader.getWordGrid();
            Trie trie = fileReader.getTrie();
            fileReader.close();

            long startTime = System.nanoTime();
            wordGrid.parseTrie(trie);
            long endTime = System.nanoTime();

            double executionTime = ((double) (endTime - startTime)) * 1e-6;

            System.out.println();
            ConsoleWriter consoleWriter = new ConsoleWriter(wordGrid);
            consoleWriter.write();

            BigDecimal bigDecimal = new BigDecimal(executionTime).setScale(3, RoundingMode.HALF_EVEN);

            System.out.println();
            System.out.println("Waktu eksekusi: " + bigDecimal.doubleValue() + " milidetik");

            System.out.println();
            System.out.println("Jumlah operasi perbandingan: " + wordGrid.getNumOperation());

            scanner.close();

        } else {
            System.out.println("File tidak ditemukan pada folder test!");
        }
    }
}
```

3. *Screenshot* dari *Input* dan *Output* dari Program

a. *Small* (14 × 14)

- ***Input* 1**

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

```
MARSHALL
MANHATTAN
MOSBY
TED
TEACHER
SLAPSGIVING
BARNEY
LAWYER
LILY
ERIKSEN
STINSON
PRESENTER
ALDRIN
ROBIN
```

- ***Output* 1**

```
Masukkan nama file: small_1.txt

L N O S N I T S E T L A L A
M T S M N A E S T E B E R R
S L P A R T K N L A M E L I
L G R R B A I I I C N R R A
A N E S A S A R S H T I E E
W I S H R S E O M E H K Y L
Y V E A N B T B A R L S A B
E I N L E T D I N B I E E M
R G T L Y R E N H A L N O A
W S E Y E A O D A N Y S G L
A P R Y E R P R T A B O W D
P A N R B W R I T Y M T L R
B L N A A P M P A A B R I I
G S E L D A N E N E Y R E N

Waktu eksekusi: 3.854 milidetik

Jumlah operasi perbandingan: 2699
```

- ***Input* 2**

```
R N N O D S E N O R H T N S
E G A G O S E I E S R E C W
Y D K W T Y O T D R R E E E
R I S D H R E O R E L H R I
K R N O R E T S I N N A L R
H O C E A S S R A A E I T W
A N W A K I T O E T A R D O
L T Y E I V A T G E O G A O
D H S D S Y R S E F E A E D
R R A S S T K E D N S S N G
O O T D S R E A D E A O E A
G N N O T D E R F S R S R M
O E A N E R Y E O O R S Y E
S R F N D S A N R S T E S R
```

```
WESTEROS
DAENERYS
LANNISTER
STARK
DOTHRAKI
DREADFORT
GAME
WEIRWOOD
KHAL DROGO
THRONES
VISERYS
ESSOS
IRON THRONE
FANTASY
CERSEI
GENDRY
```

- ***Output* 2**

```
Masukkan nama file: small_2.txt

R N N O D S E N O R H T N S
E G A G O S E I E S R E C W
Y D K W T Y O T D R R E E E
R I S D H R E O R E L H R I
K R N O R E T S I N N A L R
H O C E A S S R A A E I T W
A N W A K I T O E T A R D O
L T Y E I V A T G E O G A O
D H S D S Y R S E F E A E D
R R A S S T K E D N S S N G
O O T D S R E A D E A O E A
G N N O T D E R F S R S R M
O E A N E R Y E O O R S Y E
S R F N D S A N R S T E S R

Waktu eksekusi: 2.268 milidetik

Jumlah operasi perbandingan: 2540
```

- ***Input* 3**

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

```
JOEY
GELLER
TRIBBIANI
GUNTHER
MATT LEBLANC
BUFFAY
MONICA
ROSS
LISA KUDROW
GREEN
APARTMENT
WAITRESS
NEW YORK
CENTRAL PERK
FRIENDS
PHOEBE
BING
```

- ***Output* 3**

```
Masukkan nama file: small_3.txt

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

Waktu eksekusi: 2.794 milidetik

Jumlah operasi perbandingan: 2591
```

b. ***Medium*** **(20 × 22)**

- ***Input*** **1**

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

```
BED
CLOSET
DOOR
DUVET
FAN
LAMP
LIGHT
MIRROR
PHOTO
PILLOWS
RADIO
REMOTE
ROBE
RUG
SHELF
ALARM CLOCK
COMFORTER
CURTAINS
DRESSER
DRESSING DOWN
SLIPPERS
TEDDY BEAR
TELEVISION
WINDOWS
```

- ***Output* 1**

- ***Input* 2**

```
AFRICAN
ASIAN
BENGAL
CLAWS
FELINE
HUNTER
JUNGLE
ORANGE
PREDATOR
ROARING
SIBERIAN
STRIPES
TAIL
TEETH
WILD
CARNIVORE
CONSERVATION
ENDANGERED
INDOCHINESE
MALAYAN
MAMMAL
MAN EATER
MEAT EATER
SUMATRAN
```

- ***Output* 2**

```
Masukkan nama file: medium_2.txt

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

Waktu eksekusi: 4.098 milidetik

Jumlah operasi perbandingan: 5950
```

- ***Input* 3**

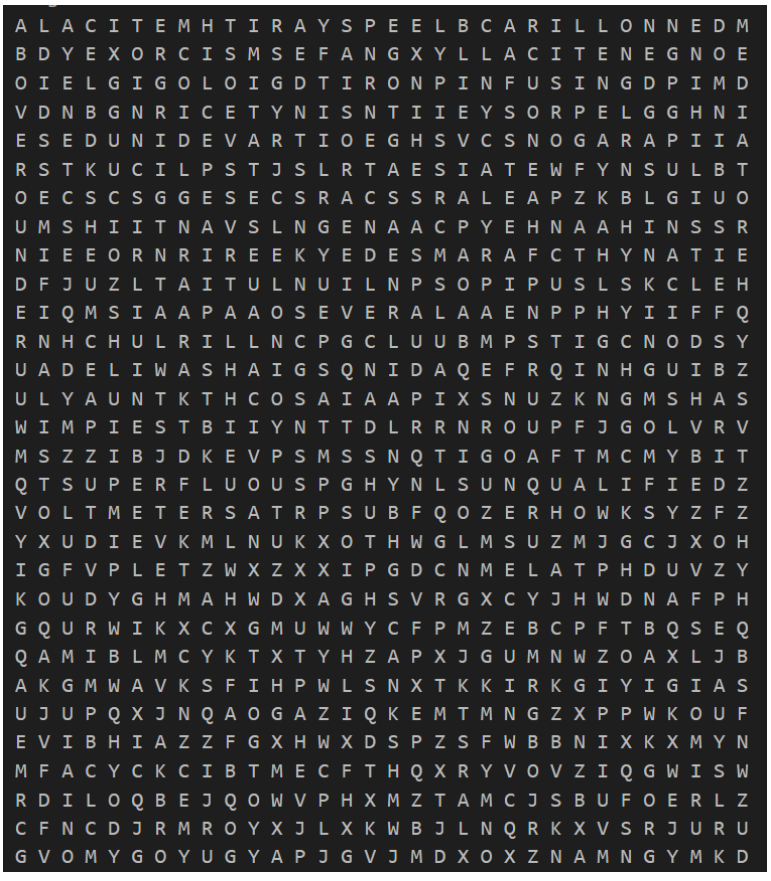
```
A S R R F S B G C C O Y A A A E W H Q A H D
Q K D O A P K I B M K M Q L K X R G B V E G
G M H M E R I P M E B G A V V G Q I U S R M
O M C E S D B J S R Z G C I B C A N Y A X O
L N I P S I R C O Z G N X M N C P L N V E W
D I N O E A N S L N O M O D B Z A N H M N S
E D F T R N I K M C C J P S F B Y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

```
CAMEO
CORTLAND
CRISPIN
EMPIRE
ENVY
FORTUNE
FUJI
GALA
IDARED
JONAMAC
KIKU
MCINTOSH
OPAL
PINK LADY
ROME
AMBROSIA
BRAEBURN
GOLDEN DELICIOUS
GRANNY SMITH
HONEYCRISP
JONAGOLD
MACOUN
PAULA RED
RED DELICIOUS
```

- ***Output* 3**

c. *Large* (30 × 32)

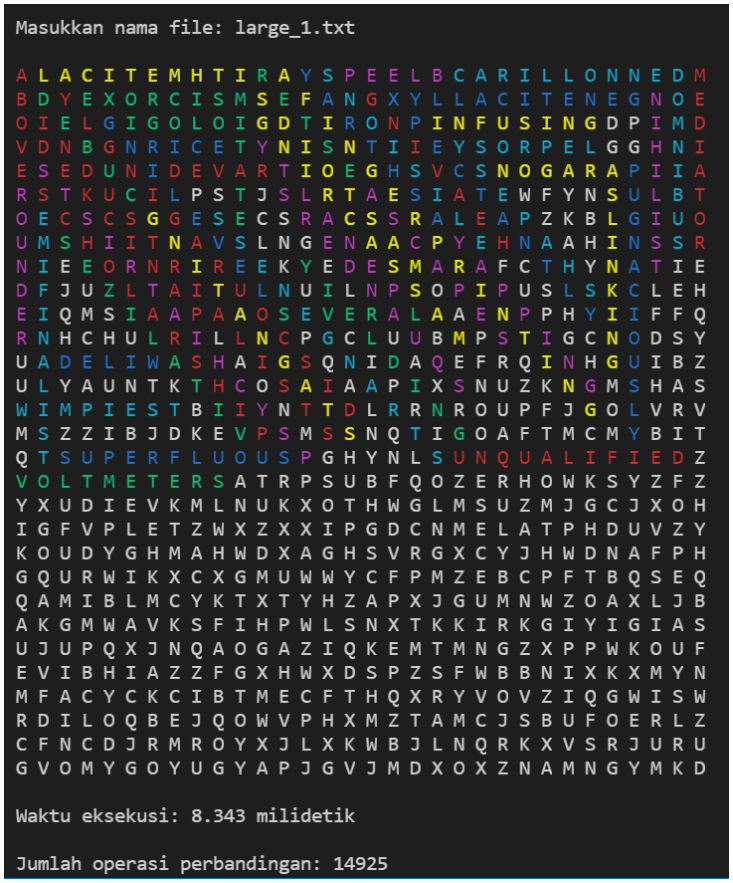
- ***Input* 1**



```
ABOVE
ARITHMETICAL
ASHTRAY
BLEEPS
CARILLONNED
CUBED
DISCOURAGINGLY
ELISIONS
ESTER
EXORCISMS
EXPECTANTLY
FINGERPRINTING
GENETICALLY
GIGOLO
HEAVING
IMITATES
INDUSTRIALISTS
INFUSING
LEPROSY
LEVELING
MASSACRING
MASTOIDS
MEDIATOR
NIHILIST
OMNIBUS
```

```
PANICKED
PARAGONS
PENALTY
PSYCHIATRISTS
PUGNACIOUSLY
RAVED
RICE
ROUNDER
SCHOLARSHIP
SEASCAPES
SEMIFINALIST
SEVERAL
SHAPES
SLINKING
SQUANDERS
STAGNATING
STRANGENESS
SUPERFLUOUS
UNQUALIFIED
VITALIZES
VOLTMETERS
WILED
WIMPIEST
YAPPING
YIELDING
```

- ***Output* 1**


Masukkan nama file: large_1.txt
A L A C I T E M H T I R A Y S P E E L B C A R I L L O N N E D M
B D Y E X O R C I S M S E F A N G X Y L L A C I T E N E G N O E
O I E L G I G O L O I G D T I R O N P I N F U S I N G D P I M D
V D N B G N R I C E T Y N I S N T I I E Y S O R P E L G G H N I
E S E D U N I D E V A R T I O E G H S V C S N O G A R A P I I A
R S T K U C I L P S T J S L R T A E S I A T E W F Y N S U L B T
O E C S C S G G E S E C S R A C S S R A L E A P Z K B L G I U O
U M S H I I T N A V S L N G E N A A C P Y E H N A A H I N S S R
N I E E O R N R I R E E E K Y E D E S M A R A F C T H Y N A T I E
D F J U Z L T A I T U L N U I L N P S O P I P U S L S K C L E H
E I Q M S I A A P A A O S E V E R A L A A E N P P H Y I I F F Q
R N H C H U L R I L L N C P G C L U U B M P S T I G C N O D S Y
U A D E L I W A S H A I G S Q N I D A Q E F R Q I N H G U I B Z
U L Y A U N T K T H C O S A I A A P I X S N U Z K N G M S H A S
W I M P I E S T B I I Y N T T D L R R N R O U P F J G O L V R V
M S Z Z I B J D K E V P S M S S N Q T I G O A F T M C M Y B I T
Q T S U P E R F L U O U S P G H Y N L S U N Q U A L I F I E D Z
V O L T M E T E R S A T R P S U B F Q O Z E R H O W K S Y Z F Z
Y X U D I E V K M L N U K X O T H W G L M S U Z M J G C J X O H
I G F V P L E T Z W X Z X X I P G D C N M E L A T P H D U V Z Y
K O U D Y G H M A H W D X A G H S V R G X C Y J H W D N A F P H
G Q U R W I K X C X G M U W W Y C F P M Z E B C P F T B Q S E Q
Q A M I B L M C Y K T X T Y H Z A P X J G U M N W Z O A X L J B
A K G M W A V K S F I H P W L S N X T K K I R K G I Y I G I A S
U J U P Q X J N Q A O G A Z I Q K E M T M N G Z X P P W K O U F
E V I B H I A Z Z F G X H W X D S P Z S F W B B N I X K X M Y N
M F A C Y C K C I B T M E C F T H Q X R Y V O V Z I Q G W I S W
R D I L O Q B E J Q O W V P H X M Z T A M C J S B U F O E R L Z
C F N C D J R M R O Y X J L X K W B J L N Q R K X V S R J U R U
G V O M Y G O Y U G Y A P J G V J M D X O X Z N A M N G Y M K D
Waktu eksekusi: 8.343 milidetik
Jumlah operasi perbandingan: 14925

- ***Input* 2**

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

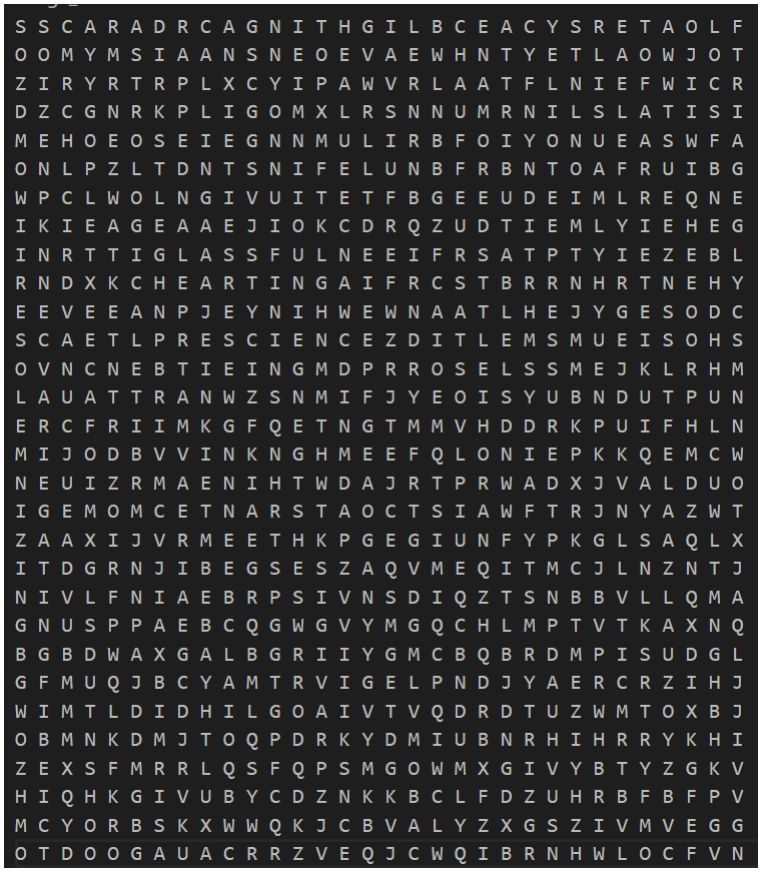
```
ACQUIESCENT
AGGRIEVING
ASPIRED
BELLHOPS
BERTHING
BLINKERS
BOMBARDED
BULLHORNS
CESIUM
CIRCUMSTANCE
CONSCIENTIOUS
COSMOLOGIES
DANGERING
DECOYED
DESISTS
DOINGS
DORMER
ELUCIDATION
EXTREMISM
FALSETTOS
FILIGREE
FLAGELLATION
FRAGMENTED
GARROTING
HARMONIES
```

```
INDELICACIES
MANGLING
MIKING
ORIENTALS
PAUPERS
PETITES
PETUNIAS
PICARESQUES
PLANETARY
PODDED
QUIBBLING
RETROACTIVE
SHAVEN
SHORTCAKES
SHOWIER
SHYSTER
SLOTHS
STAMEN
STOVEPIPE
SUPPOSITORIES
TRAILER
TRANSCONTINENTAL
VARIATE
WANING
ZAP
```

- ***Output* 2**

```
Masukkan nama file: large_2.txt

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

Waktu eksekusi: 9.956 milidetik

Jumlah operasi perbandingan: 14385
```

- ***Input* 3**



```
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
```

- ***Output* 3**

```
Masukkan nama file: large_3.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

Waktu eksekusi: 6.832 milidetik

Jumlah operasi perbandingan: 14469
```

4. ***Link Google Drive***

   a. **Google Drive**

      Tugas Kecil 1 Penyelesaian Word Search Puzzle dengan Algoritma Brute Force

   b. **Repository GitHub**

      Word Search Puzzle Solver

5. ***Checklist***

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