

LAPORAN TUGAS KECIL

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

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma
pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

Sarah Azka Arief (K2)

13520083



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG**

2022

BAB I

Deskripsi Algoritma Brute Force

Algoritma brute force merupakan teknik penyelesaian masalah yang mana eksistensi solusi dari suatu permasalahan akan diketahui dengan cara mencoba setiap jawaban yang mungkin. Oleh karena itu, metode ini hanya bergantung pada kemampuan komputasi dan mencoba semua kemungkinan, kebalikan dari efisiensi dan optimasi. Pada tugas kecil ini, program menggunakan bahasa Java untuk menerapkan algoritma brute force demi mencari penyelesaian dari *word search puzzle* yang diberikan dalam bentuk file txt yang sesuai dengan spesifikasi dari tugas kecil.

Program utama dimulai dengan pembacaan input *user* berupa nama file txt yang akan dicari penyelesaiannya. Kemudian file tersebut dibaca dan isi file yang berupa matriks berisi huruf disimpan dalam suatu 2D ArrayList berisi String yang kemudian diubah menjadi 2D *array of String* bernama 'matrix', sementara isi file yang berupa list kata yang harus dicari disimpan dalam ArrayList berisi *array of char* bernama 'words'. Setelah itu, 'matrix' dan 'words' diproses menggunakan 8 fungsi yang dibedakan berdasarkan arah pencariannya untuk mencari penyelesaian dari *word search puzzle* tersebut. Berikut 8 fungsi tersebut:

1. searchUp(matrix, word)
Mencari kata pada matriks dari bawah ke atas
2. searchDown(matrix, word)
Mencari kata pada matriks dari atas ke bawah
3. searchLeft(matrix, word)
Mencari kata pada matriks dari kanan ke kiri
4. searchRight(matrix, word)
Mencari kata pada matriks dari kiri ke kanan
5. searchDiagUpLeft(matrix, word)
Mencari kata pada matriks dari bawah ke atas dan kanan ke kiri
6. searchDiagUpRight(matrix, word)
Mencari kata pada matriks dari bawah ke atas dan kiri ke kanan
7. searchDiagDownLeft(matrix, word)
Mencari kata pada matriks dari atas ke bawah dan kanan ke kiri
8. searchDiagDownRight(matrix, word)
Mencari kata pada matriks dari atas ke bawah dan kiri ke kanan

Setiap fungsi di atas akan dimulai dengan looping terhadap huruf pertama dari setiap kata yang ada pada list 'word'. Kemudian, di dalam loop tersebut terjadi looping pada 'matrix' agar matriks teriterasi sesuai arah dari masing-masing fungsi. Setiap huruf pada matriks yang diiterasi akan dicek kesamaannya dengan huruf pertama pada kata yang sedang diiterasi. Apabila sama, koordinat dari huruf tersebut akan disimpan dalam suatu ArrayList dan akan dicek kesamaan dari huruf berikutnya dengan huruf berikutnya dari kata yang sedang diiterasi. Apabila setelahnya ditemukan perbedaan huruf antara yang sedang diiterasi pada matriks dan huruf yang sedang dicek pada kata, maka ArrayList berisi koordinat huruf akan dibersihkan menggunakan fungsi clear.

Proses perbandingan huruf akan berakhir hingga seluruh huruf pada matriks telah dicek atau telah ditemukan kata yang sedang dicari. Apabila kata yang sedang dicari telah ditemukan, kata tersebut akan dihapus dari 'words' dan hasil matriks yang menunjukkan lokasi dari kata tersebut akan diprint pada terminal beserta jumlah perbandingan yang dilakukan untuk mendapatkan kata tersebut. Penentuan kapan kata yang sedang dicari telah ditemukan atau belum adalah dengan menggunakan panjang kata yang sedang dicari sebagai parameter ditemukannya kata tersebut atau tidak.

Jumlah perbandingan pada program ini bertambah setiap kali terjadi proses perbandingan antara huruf pada matriks dan huruf dari kata yang sedang dicari dengan catatan bahwa apabila ditemukan dua huruf berturut-turut pada matriks yang sama dengan huruf pertama kata yang dicari maka akan dilakukan pengecekan dua kali terhadap huruf kedua di matriks tersebut.

Setelah ke-8 fungsi tersebut telah selesai dijalankan atau seluruh pada 'words' telah ditemukan, program akan menampilkan ukuran matriks, waktu yang dibutuhkan, serta kata yang tersisa alias belum ditemukan.

BAB II

Source Program

```
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3 import java.util.*;
4
5
6 public class WordSearch{
7     public static String getLetter(Arraylist<char[]> words, int row, int col){
8         return Character.toString(words.get(row)[col]);
9     }
10
11     public static void printWords(Arraylist<char[]> words){
12         for (int i = 0; i < words.size(); i++){
13             System.out.println(words.get(i));
14         }
15     }
16
17     public static void printMatrix(String[][] matrix, Arraylist<String> toPrint, int comparison){
18         int matrixRow = matrix.length, matrixCol = matrix[0].length;
19         for (int m = 0; m < matrixRow; m++){
20             for (int n = 0; n < matrixCol; n++){
21                 String temp = String.valueOf(m) + "," + String.valueOf(n);
22                 if (toPrint.contains(temp)){
23                     System.out.print(matrix[m][n]);
24                 }
25                 else{
26                     System.out.print("-");
27                 }
28                 System.out.print(" ");
29             }
30             System.out.println();
31             System.out.println("Total Comparisons: " + comparison);
32             System.out.println();
33         }
34     }
35
36     public static void searchMn(String[][] matrix, Arraylist<char[]> words){
37         Arraylist<Integer> toDelete = new Arraylist<>();
38         Arraylist<String> toPrint = new Arraylist<String>();
39         int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
40         for (int i = 0; i < words.size(); i++){
41             int wordCheck = 0;
42             boolean diff = true;
43             for (int j = matrixCol - 1; j >= 0; j--){
44                 for (int k = matrixRow - 1; k >= 0; k--){
45                     comparison++;
46                     if (matrix[k][j].equals(getLetter(words, i, wordCheck))){
47                         if (diff == true){
48                             diff = false;
49                         }
50                         String temp = String.valueOf(k) + "," + String.valueOf(j);
51                         toPrint.add(temp);
52                         wordCheck++;
53                     }
54                     else{
55                         if (diff == false){
56                             diff = true;
57                         }
58                         wordCheck = 0;
59                         toPrint.clear();
60                         comparison++;
61                         if (matrix[k][j].equals(getLetter(words, i, 0))){
62                             String temp = String.valueOf(j) + "," + String.valueOf(k);
63                             toPrint.add(temp);
64                             wordCheck++;
65                         }
66                     }
67                 }
68             }
69             if (wordCheck == words.get(i).length){
70                 printMatrix(matrix, toPrint, comparison);
71             }
72         }
73     }
74 }
```

```
1         j = -1;
2         k = -1;
3         toDelete.add(i);
4     }
5
6     wordCheck = 0;
7     diff = true;
8     toPrint.clear();
9
10 }
11
12 for (int i = toDelete.size() - 1; i >= 0; i--){
13     words.remove(toDelete.get(i)).intValue();
14 }
15
16 return;
17 }
18
19 public static void searchMn(String[][] matrix, Arraylist<char[]> words){
20     Arraylist<Integer> toDelete = new Arraylist<>();
21     Arraylist<String> toPrint = new Arraylist<String>();
22     int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
23     for (int i = 0; i < words.size(); i++){
24         int wordCheck = 0;
25         boolean diff = true;
26         for (int j = 0; j < matrixCol; j++){
27             for (int k = 0; k < matrixRow; k++){
28                 comparison++;
29                 if (matrix[k][j].equals(getLetter(words, i, wordCheck))){
30                     if (diff == true){
31                         diff = false;
32                     }
33                     String temp = String.valueOf(k) + "," + String.valueOf(j);
34                     toPrint.add(temp);
35                     wordCheck++;
36                 }
37                 else{
38                     if (diff == false){
39                         diff = true;
40                     }
41                     wordCheck = 0;
42                     toPrint.clear();
43                     comparison++;
44                     if (matrix[k][j].equals(getLetter(words, i, 0))){
45                         diff = false;
46                         String temp = String.valueOf(k) + "," + String.valueOf(j);
47                         toPrint.add(temp);
48                         wordCheck++;
49                     }
50                 }
51             }
52             if (wordCheck == words.get(i).length){
53                 printMatrix(matrix, toPrint, comparison);
54                 j = matrixCol;
55                 k = matrixRow;
56                 toDelete.add(i);
57             }
58         }
59     }
60     wordCheck = 0;
61     diff = true;
62     toPrint.clear();
63
64 }
65
66 for (int i = toDelete.size() - 1; i >= 0; i--){
67     words.remove(toDelete.get(i)).intValue();
68 }
69
70 return;
71 }
72
73 public static void searchLeft(String[][] matrix, Arraylist<char[]> words){
74     Arraylist<Integer> toDelete = new Arraylist<>();
75     Arraylist<String> toPrint = new Arraylist<String>();
76 }
```

```

1 int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
2 for (int i = 0; i < words.size(); i++){
3     int wordCheck = 0;
4     boolean diff = true;
5     for (int j = matrixRow - 1; j >= 0; j--){
6         for (int k = matrixCol - 1; k >= 0; k--){
7             comparison++;
8             if (matrix[j][k].equals(getLetter(words, i, wordCheck))){
9                 if (diff == true){
10                     diff = false;
11                 }
12                 String temp = String.valueOf(j) + "," + String.valueOf(k);
13                 toPrint.add(temp);
14                 wordCheck++;
15             }
16             else{
17                 if (diff == false){
18                     diff = true;
19                     wordCheck = 0;
20                     toPrint.clear();
21                     comparison++;
22                     if (matrix[j][k].equals(getLetter(words, i, 0))){
23                         diff = false;
24                         String temp = String.valueOf(j) + "," + String.valueOf(k);
25                         toPrint.add(temp);
26                         wordCheck++;
27                     }
28                 }
29             }
30             if (wordCheck == words.get(i).length){
31                 printMatrix(matrix, toPrint, comparison);
32                 j = -1;
33                 k = 0;
34                 toDelete.add(i);
35             }
36         }
37         wordCheck = 0;
38         diff = true;
39         toPrint.clear();
40     }
41 }
42 for (int i = toDelete.size() - 1; i >= 0; i--){
43     words.remove((toDelete.get(i)).intValue());
44 }
45 return;
46 }
47 public static void searchRight(String[][] matrix, ArrayList<char[]> words){
48     ArrayList<Integer> toDelete = new ArrayList<>();
49     ArrayList<String> toPrint = new ArrayList<>();
50     int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
51     for (int i = 0; i < words.size(); i++){
52         int wordCheck = 0;
53         boolean diff = true;
54         for (int j = 0; j < matrixRow; j++){
55             for (int k = 0; k < matrixCol; k++){
56                 comparison++;
57                 if (matrix[j][k].equals(getLetter(words, i, wordCheck))){
58                     if (diff == true){
59                         diff = false;
60                     }
61                     String temp = String.valueOf(j) + "," + String.valueOf(k);
62                     toPrint.add(temp);
63                     wordCheck++;
64                 }
65             }
66             else{
67                 if (diff == false){

```

```

1 wordCheck = 0;
2 toPrint.clear();
3 comparison++;
4 if (matrix[j][k].equals(getLetter(words, i, 0))){
5     diff = false;
6     String temp = String.valueOf(j) + "," + String.valueOf(k);
7     toPrint.add(temp);
8     wordCheck++;
9 }
10 }
11 }
12 if (wordCheck == words.get(i).length){
13     printMatrix(matrix, toPrint, comparison);
14     j = matrixRow;
15     k = matrixCol;
16     toDelete.add(i);
17 }
18 }
19 wordCheck = 0;
20 diff = true;
21 toPrint.clear();
22 }
23 }
24 for (int i = toDelete.size() - 1; i >= 0; i--){
25     words.remove((toDelete.get(i)).intValue());
26 }
27 return;
28 }
29 }
30 public static void searchDiagUpLeft(String[][] matrix, ArrayList<char[]> words){
31     ArrayList<Integer> toDelete = new ArrayList<>();
32     ArrayList<String> toPrint = new ArrayList<>();
33     int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
34     for (int i = 0; i < words.size(); i++){
35         int wordCheck = 0;
36         boolean diff = true;
37         for (int j = 0; j < matrixCol; j++){
38             int row = matrixRow - 1;
39             int col = j;
40             while (col >= 0 && row >= 0){
41                 comparison++;
42                 if (matrix[row][col].equals(getLetter(words, i, wordCheck))){
43                     if (diff == true){
44                         diff = false;
45                     }
46                     String temp = String.valueOf(row) + "," + String.valueOf(col);
47                     toPrint.add(temp);
48                     wordCheck++;
49                 }
50             }
51             else{
52                 if (diff == false){
53                     diff = true;
54                     wordCheck = 0;
55                     toPrint.clear();
56                     comparison++;
57                     if (matrix[row][col].equals(getLetter(words, i, 0))){
58                         diff = false;
59                         String temp = String.valueOf(row) + "," + String.valueOf(col);
60                         toPrint.add(temp);
61                         wordCheck++;
62                     }
63                 }
64             }
65             if (wordCheck == words.get(i).length){
66                 printMatrix(matrix, toPrint, comparison);
67                 toDelete.add(i);
68                 j = matrixCol;

```

```

1 break;
2 }
3 row--;
4 col--;
5 }
6 wordCheck = 0;
7 diff = true;
8 toPrint.clear();
9 }
10 for (int j = 0; j < matrixRow - 1; j++){
11     int col = matrixCol - 1;
12     int row = j;
13     while (col >= 0 && row >= 0){
14         comparison++;
15         if (matrix[row][col].equals(getLetter(words, i, wordCheck))){
16             if (diff == true){
17                 diff = false;
18             }
19             String temp = String.valueOf(row) + "," + String.valueOf(col);
20             toPrint.add(temp);
21             wordCheck++;
22         }
23         else{
24             if (diff == false){
25                 diff = true;
26                 wordCheck = 0;
27                 toPrint.clear();
28                 comparison++;
29                 if (matrix[row][col].equals(getLetter(words, i, 0))){
30                     String temp = String.valueOf(row) + "," + String.valueOf(col);
31                     toPrint.add(temp);
32                     wordCheck++;
33                 }
34             }
35         }
36     }
37     if (wordCheck == words.get(i).length){
38         printMatrix(matrix, toPrint, comparison);
39         toDelete.add(i);
40         j = matrixCol;
41         break;
42     }
43     row--;
44     col--;
45 }
46 wordCheck = 0;
47 diff = true;
48 toPrint.clear();
49 }
50 }
51 for (int i = toDelete.size() - 1; i >= 0; i--){
52     words.remove((toDelete.get(i)).intValue());
53 }
54 return;
55 }
56 public static void searchDiagUpRight(String[][] matrix, ArrayList<char[]> words){
57     ArrayList<Integer> toDelete = new ArrayList<>();
58     ArrayList<String> toPrint = new ArrayList<>();
59     int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
60     for (int i = 0; i < words.size(); i++){
61         int wordCheck = 0;
62         boolean diff = true;
63         for (int j = matrixRow - 1; j >= 0; j--){
64             int row = matrixRow - 1;
65             int col = j;
66             while (col < matrixCol && row >= 0){
67                 comparison++;

```

```

1 if (matrix[row][col].equals(getLetter(words, i, wordCheck))){
2     if (diff == true){
3         diff = false;
4     }
5     String temp = String.valueOf(row) + "," + String.valueOf(col);
6     toPrint.add(temp);
7     wordCheck++;
8 }
9 }
10 else{
11     if (diff == false){
12         diff = true;
13         wordCheck = 0;
14         toPrint.clear();
15         comparison++;
16         if (matrix[row][col].equals(getLetter(words, i, 0))){
17             diff = false;
18             String temp = String.valueOf(row) + "," + String.valueOf(col);
19             toPrint.add(temp);
20             wordCheck++;
21         }
22     }
23 }
24 if (wordCheck == words.get(i).length){
25     printMatrix(matrix, toPrint, comparison);
26     toDelete.add(i);
27     j = -1;
28     break;
29 }
30 row--;
31 col++;
32 }
33 wordCheck = 0;
34 diff = true;
35 toPrint.clear();
36 }
37 }
38 for (int j = matrixRow - 2; j >= 0; j--){
39     int col = 0;
40     int row = j;
41     while (col < matrixCol && row >= 0){
42         comparison++;
43         if (matrix[row][col].equals(getLetter(words, i, wordCheck))){
44             if (diff == true){
45                 diff = false;
46             }
47             String temp = String.valueOf(row) + "," + String.valueOf(col);
48             toPrint.add(temp);
49             wordCheck++;
50         }
51         else{
52             if (diff == false){
53                 diff = true;
54                 wordCheck = 0;
55                 toPrint.clear();
56                 comparison++;
57                 if (matrix[row][col].equals(getLetter(words, i, 0))){
58                     diff = false;
59                     String temp = String.valueOf(row) + "," + String.valueOf(col);
60                     toPrint.add(temp);
61                     wordCheck++;
62                 }
63             }
64         }
65     }
66     if (wordCheck == words.get(i).length){
67         printMatrix(matrix, toPrint, comparison);
68         toDelete.add(i);
69         j = -1;
70         break;
71 }

```

```

1      }
2      row--;
3      col++;
4      }
5      wordCheck = 0;
6      diff = true;
7      toPrint.clear();
8      }
9      }
10     for (int i = toDelete.size() - 1; i >= 0; i--) {
11         words.remove((toDelete.get(i)).intValue());
12     }
13     return;
14 }
15 public static void searchDiagDownLeft(String[][] matrix, ArrayList<char>[] words) {
16     ArrayList<Integer> toDelete = new ArrayList<>();
17     ArrayList<String> toPrint = new ArrayList<>();
18     int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
19     for (int i = 0; i < words.size(); i++) {
20         int wordCheck = 0;
21         boolean diff = true;
22         for (int j = matrixCol - 1; j >= 0; j--) {
23             int col = matrixCol - i;
24             int row = j;
25             while (col >= 0 && row < matrixRow) {
26                 comparison++;
27                 if (matrix[row][col].equals(getLetter(words, i, wordCheck))) {
28                     if (diff == true) {
29                         diff = false;
30                     }
31                     String temp = String.valueOf(row) + "," + String.valueOf(col);
32                     toPrint.add(temp);
33                     wordCheck++;
34                 }
35             }
36             if (diff == false) {
37                 diff = true;
38                 wordCheck = 0;
39                 toPrint.clear();
40                 comparison++;
41                 if (matrix[row][col].equals(getLetter(words, i, 0))) {
42                     diff = false;
43                     String temp = String.valueOf(row) + "," + String.valueOf(col);
44                     toPrint.add(temp);
45                     wordCheck++;
46                 }
47             }
48             if (wordCheck == words.get(i).length() {
49                 printMatrix(matrix, toPrint, comparison);
50                 toDelete.add(i);
51                 j = -1;
52                 break;
53             }
54             row++;
55             col--;
56         }
57         wordCheck = 0;
58         diff = true;
59         toPrint.clear();
60     }
61     for (int j = matrixCol - 2; j >= 0; j--) {
62         int row = 0;
63         int col = j;
64         while (col >= 0 && row < matrixRow) {
65             comparison++;
66             if (matrix[row][col].equals(getLetter(words, i, wordCheck))) {
67

```

```

1         if (diff == true) {
2             diff = false;
3         }
4         String temp = String.valueOf(row) + "," + String.valueOf(col);
5         toPrint.add(temp);
6         wordCheck++;
7     }
8 }
9 else {
10     if (diff == false) {
11         diff = true;
12         wordCheck = 0;
13         toPrint.clear();
14         comparison++;
15         if (matrix[row][col].equals(getLetter(words, i, 0))) {
16             diff = false;
17             String temp = String.valueOf(row) + "," + String.valueOf(col);
18             toPrint.add(temp);
19             wordCheck++;
20         }
21     }
22     if (wordCheck == words.get(i).length() {
23         printMatrix(matrix, toPrint, comparison);
24         toDelete.add(i);
25         j = -1;
26         break;
27     }
28     row++;
29     col--;
30 }
31 wordCheck = 0;
32 diff = true;
33 toPrint.clear();
34 }
35 }
36 for (int i = toDelete.size() - 1; i >= 0; i--) {
37     words.remove((toDelete.get(i)).intValue());
38 }
39 return;
40 }
41 public static void searchDiagDownRight(String[][] matrix, ArrayList<char>[] words) {
42     ArrayList<Integer> toDelete = new ArrayList<>();
43     ArrayList<String> toPrint = new ArrayList<>();
44     int matrixRow = matrix.length, matrixCol = matrix[0].length, comparison = 0;
45     for (int i = 0; i < words.size(); i++) {
46         int wordCheck = 0;
47         boolean diff = true;
48         for (int j = matrixCol - 1; j >= 0; j--) {
49             int row = 0;
50             int col = j;
51             while (col < matrixCol && row < matrixRow) {
52                 comparison++;
53                 if (matrix[row][col].equals(getLetter(words, i, wordCheck))) {
54                     if (diff == true) {
55                         diff = false;
56                     }
57                     String temp = String.valueOf(row) + "," + String.valueOf(col);
58                     toPrint.add(temp);
59                     wordCheck++;
60                 }
61             }
62             if (diff == false) {
63                 diff = true;
64                 wordCheck = 0;
65                 toPrint.clear();
66                 comparison++;
67                 if (matrix[row][col].equals(getLetter(words, i, 0))) {
68

```

```

1         diff = false;
2         String temp = String.valueOf(row) + "," + String.valueOf(col);
3         toPrint.add(temp);
4         wordCheck++;
5     }
6 }
7 }
8 if (wordCheck == words.get(i).length() {
9     printMatrix(matrix, toPrint, comparison);
10    toDelete.add(i);
11    j = -1;
12    break;
13 }
14 row++;
15 col--;
16 }
17 wordCheck = 0;
18 diff = true;
19 toPrint.clear();
20 }
21 for (int j = matrixCol - 1; j >= 0; j--) {
22     int col = 0;
23     int row = j;
24     while (col < matrixCol && row < matrixRow) {
25         comparison++;
26         if (matrix[row][col].equals(getLetter(words, i, wordCheck))) {
27             if (diff == true) {
28                 diff = false;
29             }
30             String temp = String.valueOf(row) + "," + String.valueOf(col);
31             toPrint.add(temp);
32             wordCheck++;
33         }
34     }
35     if (diff == false) {
36         diff = true;
37         wordCheck = 0;
38         toPrint.clear();
39         comparison++;
40         if (matrix[row][col].equals(getLetter(words, i, 0))) {
41             diff = false;
42             String temp = String.valueOf(row) + "," + String.valueOf(col);
43             toPrint.add(temp);
44             wordCheck++;
45         }
46     }
47     if (wordCheck == words.get(i).length() {
48         printMatrix(matrix, toPrint, comparison);
49         toDelete.add(i);
50         j = 0;
51         break;
52     }
53     row++;
54     col++;
55 }
56 wordCheck = 0;
57 diff = true;
58 toPrint.clear();
59 }
60 }
61 for (int i = toDelete.size() - 1; i >= 0; i--) {
62     words.remove((toDelete.get(i)).intValue());
63 }
64 return;
65 }
66 public static void main(String[] args) {

```

```

1     System.out.println("\n-----\nWORD SEARCH PUZZLE SOLVER\n-----");
2     System.out.print("Enter filename:");
3     Scanner sc = new Scanner(System.in);
4     String filename = "../test/" + sc.next();
5     sc.close();
6     try {
7         ArrayList<ArrayList<String>> tempMatrix = new ArrayList<>();
8         ArrayList<char>[] words = new ArrayList<>();
9         boolean isWord = false;
10
11         BufferedReader br = new BufferedReader(new FileReader(filename));
12         while (rl != null) {
13             String rl = br.readLine();
14             if (rl.length() == 0) {
15                 isWord = true;
16             }
17             else {
18                 String[] rls = rl.split(" ");
19                 if (isWord == false) {
20                     ArrayList<String> temp = new ArrayList<>();
21                     for (String x : rls) {
22                         temp.add(x);
23                     }
24                     tempMatrix.add(temp);
25                 }
26                 else {
27                     String all = "";
28                     for (String x : rls) {
29                         all += x;
30                     }
31                     char[] ch = all.toCharArray();
32                     words.add(ch);
33                 }
34             }
35             rl = br.readLine();
36         }
37         br.close();
38         String[][] matrix = tempMatrix.stream().map(s -> s.toArray(new String[0])).toArray(String[][]::new);
39         int searchWords = words.size();
40         long startTime = System.nanoTime();
41         searchLeft(matrix, words);
42         searchRight(matrix, words);
43         searchDown(matrix, words);
44         searchUp(matrix, words);
45         searchDiagUpLeft(matrix, words);
46         searchDiagUpRight(matrix, words);
47         searchDiagDownLeft(matrix, words);
48         searchDiagDownRight(matrix, words);
49         long endTime = System.nanoTime();
50         long totalTime = endTime - startTime;
51         double elapsedTimeSecond = (double) totalTime / 1,000,000,000;
52         System.out.println("Matrix size: " + matrix.length + " x " + matrix[0].length);
53         System.out.println("Elapsed Time: " + elapsedTimeSecond + " second(s)");
54         if (words.size() == 0) {
55             System.out.println("Words left: 0" + searchWords);
56             System.out.println("All words were found during the process");
57         }
58         else {
59             System.out.println("Words left: " + words.size() + "/" + searchWords);
60             printWords(words);
61         }
62     }
63     catch (Exception ie) {
64         System.out.println("gasabi");
65     }
66 }
67 }

```

BAB III

Screenshot Input/Output

No.	Screenshot Terminal	Keterangan
1.	<pre> PS C:\Users\Sarah Azka A\Desktop\if\sem-4\tubes\bin> java WordSearch ----- WORD SEARCH PUZZLE SOLVER ----- Enter Filename <filename>.txt: small1.txt ----- - G - - N - - I - - V - - I - - G - - S - - P - - A - - L - - S - Total Comparison: 1207 - - - - - M - - - A - - - R - - - S - - - H - - - A - - - L - - - L - ----- Total Comparison: 52 ----- - - - - - L - - - A - - - W - - - Y - - - E - - - R - ----- Total Comparison: 181 ----- - T - - E - - A - - C - - H - - E - - R - ----- Total Comparison: 734 ----- - - - - - B - - - A - - - R - - - N - - - E - - - Y - ----- Total Comparison: 800 ----- - - - - - L - - - A - - - W - - - Y - - - E - - - R - ----- Total Comparison: 810 ----- - - - - - L - - - I - - - L - - - Y - ----- Total Comparison: 968 ----- - - - - - E - - - R - - - I - - - K - - - S - - - E - - - N - ----- Total Comparison: 1145 </pre>	<p>Size: Small</p> <p>Matrix Size: 14 x 14</p> <p>Words to Search: 14</p> <p>Elapsed Time: 0.2642827 second(s)</p>

2.

PS C:\Users\Sarah Azka A\Desktop\if\sem-4\tubes\bin> java WordSearch

WORD SEARCH PUZZLE SOLVER

Enter Filename

<filename>.txt: small2.txt

```

-----
- S -
- Y -
- R -
- E -
- S -
- I -
- V -
-----

```

Total Comparison: 2175

```

-----
- S -
- O -
- S -
- S -
- E -
-----

```

Total Comparison: 2213

```

-----
- D -
- O -
- T -
- H -
- R -
- A -
- K -
- I -
-----
- Y -
- S -
- A -
- T -
- N -
- A -
- F -
-----
Total Comparison: 2579

```

Total Comparison: 763

```

-----
- D -
- A -
- E -
- N -
- E -
- R -
- Y -
- S -
-----
Total Comparison: 392

```

Total Comparison: 1171

```

-----
- S -
- T -
- A -
- R -
- K -
-----

```

Total Comparison: 694

Total Comparison: 1364

Size: Small

Matrix Size: 14 x 14

Words to Search: 16

Elapsed Time:

0.3751578 second(s)

3.

```
PS C:\Users\Sarah Azka A\Desktop\if\sem-4\tubes\bin> java WordSearch
```

```
WORD SEARCH PUZZLE SOLVER
```

```
Enter Filename
```

```
<filename>.txt: small3.txt
```

```
-----
- T
- R
- I
- B
- B
- I
- A
- N
- I
-----
```

```
Total Comparison: 604
```

```
- C N A L B E L T T A M -
```

```
Total Comparison: 713
```

```
-----
- A C I N O M
-----
Total Comparison: 932
```

```
WORDUKASIL
```

```
Total Comparison: 1189
```

```
- K R O Y W E N
```

```
Total Comparison: 1825
```

```
-----
- E B E O H P -
-----
Total Comparison: 2374
```

```
G U N T H E R
```

```
Total Comparison: 550
```

```
B U F F A Y
```

```
Total Comparison: 730
```

Size: Small

Matrix Size: 14 x 14

Words to Search: 14

Elapsed Time:

0.3587642 second(s)

4

```
PS C:\Users\Sarah Azka A\Desktop\if\sem-4\tubes\bin> java WordSearch
```

WORD SEARCH PUZZLE SOLVER

Enter Filename

```
<filename>.txt: medium1.txt
```

TESE
OL
C



Total Comparison: 464

-	-	-	-	-	-	P	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	M	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	L	-	-	-	-	-	-	-	-	-

Total Comparison: 2119

A large, dark gray grid with the words 'SUNIT' and 'DUVET' written vertically in white capital letters on the left and right sides respectively.

```

-----
Total Comparison: 7609

```

Total Comparison: 565

REMO

	E
D	
O	
9	

[illegible]

Total Comparison: 467

Total Comparison: 3579

Size: Medium

Matrix Size: 20 x 22

Words to Search: 24

Elapsed Time:

0.9024866 second(s)

13

5.

PS C:\Users\Sarah Azka A\Desktop\if\sem-4\tubes\bin> java WordSearch

WORD SEARCH PUZZLE SOLVER

Enter Filename

<filename>.txt: medium2.txt

```

-----
- E -
- N -
- I -
- L -
- E -
- F -
-----

```

Total Comparison: 2260

```

----- R -----
----- E -----
----- T -----
----- N -----
----- U -----
----- H -----
-----

```

Total Comparison: 2508

```

----- C -----
----- A -----
----- R -----
----- N -----
----- I -----
----- V -----
----- A -----
----- O -----
----- F -----
----- R -----
----- E -----
----- I -----
----- C -----
----- A -----
----- N -----
-----

```

Total Comparison: 426

Total Comparison: 5737

```

----- C ----- S -----
----- L ----- U -----
----- A ----- M -----
----- W ----- A -----
----- S ----- T -----
----- R -----
----- A -----
----- N -----
-----

```

Total Comparison: 1609

Total Comparison: 9037

Size: Medium

Matrix Size: 20 x 22

Words to Search: 24

Elapsed Time:

0.9488121 second(s)

	<div> <div> <div>ELGNUJ</div> <div>NAISA</div> <div>Total Comparison: 317</div> <div>Total Comparison: 821</div> </div> <div> <div>LAGNEB</div> <div>Total Comparison: 450</div> </div> <div> <div>GNIRAOR</div> <div>Total Comparison: 1746</div> </div> </div> <div> <div> <div>LAMMAM</div> <div>LIAT</div> <div>Total Comparison: 2755</div> <div>Total Comparison: 5580</div> </div> <div> <div>DLIW</div> <div>RETAENAM</div> <div>Total Comparison: 3320</div> <div>Total Comparison: 5659</div> </div> </div>	
--	---	--

6.

```
PS C:\Users\Sarah Azka A\Desktop\if\sem-4\tubes\bin> java WordSearch
```

WORD SEARCH PUZZLE SOLVER

Enter Filename

```
<filename>.txt: medium3.txt
```

S
U
O
I
C
I
L
E
D
D
E
R

Total Comparison: 10949

C
O
R
T
L
A
N
D

Total Comparison: 860

```

I
D
A
R
E
D
R
O
M
E
Total Comparison: 3642
Total Comparison: 5788

```

Total Comparison: 3642

Total Comparison: 5788

```

- - - - - G
- - - - - O
- - - - - L
- - - - - D
-   O     E
-   P     N
-   A     D
-   L     E
- - - - - L
- - - - - I
- - - - - C
- - - - - I
- - - - - O
- - - - - U
- - - - - S
- - - - -
- - - - -
- - - - -
- - - - -
Total Comparison: 5267                Total Comparison: 6732

```

Total Comparison: 5267

Total Comparison: 6732

Size: Medium

Matrix Size: 20 x 22

Words to Search: 24

Elapsed Time:

0.9007778 second(s)

7.

PS C:\Users\Sarah Azka A\Desktop\if\sem-4\tubes\bin> java WordSearch

WORD SEARCH PUZZLE SOLVER

Enter Filename

<filename>.txt: large1.txt

A

B

O

V

E

Total Comparison: 5

```

I      H      T      A      T      E      S      N      I      H      I      L      I      S      T
M      I      T      A      T      E      S      N      I      H      I      L      I      S      T
I      T      A      T      E      S      N      I      H      I      H      I      L      I      S      T
A      T      E      S      N      I      H      I      H      I      H      I      L      I      S      T
T      E      S      N      I      H      I      H      I      H      I      H      I      L      I      S      T
E      S      N      I      H      I      H      I      H      I      H      I      H      I      L      I      S      T
S      N      I      H      I      H      I      H      I      H      I      H      I      H      I      L      I      S      T
Total Comparison: 14373
Total Comparison: 22333
M      O      R      I      A      T      O      R      S      O      R      N      I      B      U      S
E      D      I      A      T      O      R      S      O      R      N      I      B      U      S
I      A      T      O      R      S      O      R      N      I      B      U      S
Total Comparison: 21406
Total Comparison: 23273

```

```

LACITEHHTIRA      SPEELB
Total Comparison: 1015
Total Comparison: 2999

```

Size: Large

Matrix Size: 30 x 32

Words to Search:

251/26/2022

Elapsed Time:

1.8509556 second(s)

	<div> <div> VLLACITENEG </div> <div> CARILLONNED </div> <div> Total Comparison: 11968 </div> <div> Total Comparison: 1049 </div> <div> YSORPEL </div> <div> EXORCISMS </div> <div> Total Comparison: 16893 </div> <div> Total Comparison: 5184 </div> </div>	
	<div> <div> GIGGOLD </div> <div> V A R T H S A </div> <div> Total Comparison: 7179 </div> <div> Total Comparison: 751 </div> <div> INFUSING </div> <div> D E B U C </div> <div> Total Comparison: 9299 </div> <div> Total Comparison: 1229 </div> </div>	
	<div> <div> Y L G N I G A R U O C S I D </div> <div> S N O I S I L E </div> <div> Total Comparison: 2184 </div> <div> Total Comparison: 3355 </div> </div>	

	<div> <div> R E T S E </div> <div> G N I L E V E L </div> <div>Total Comparison: 4146</div> <div>Total Comparison: 8411</div> </div> <div> <div> G N I V A E H </div> <div> G W I R C A S A H </div> <div>Total Comparison: 6948</div> <div>Total Comparison: 9094</div> </div> <div> <div> S D E O T S A N </div> <div> F I N G E R P R E N T I N G </div> <div>Total Comparison: 10500</div> <div>Total Comparison: 882</div> </div> <div> <div> E X P E C T A N T L Y </div> <div> I N D U S T R I A L I S T S </div> <div>Total Comparison: 138</div> <div>Total Comparison: 2262</div> </div> <div> <p>Matrix size: 30 x 32</p> <p>Elapsed Time: 1.8589556 second(s)</p> <p>Words left: 0/25</p> <p>All words were found during the process</p> </div>	
--	---	--

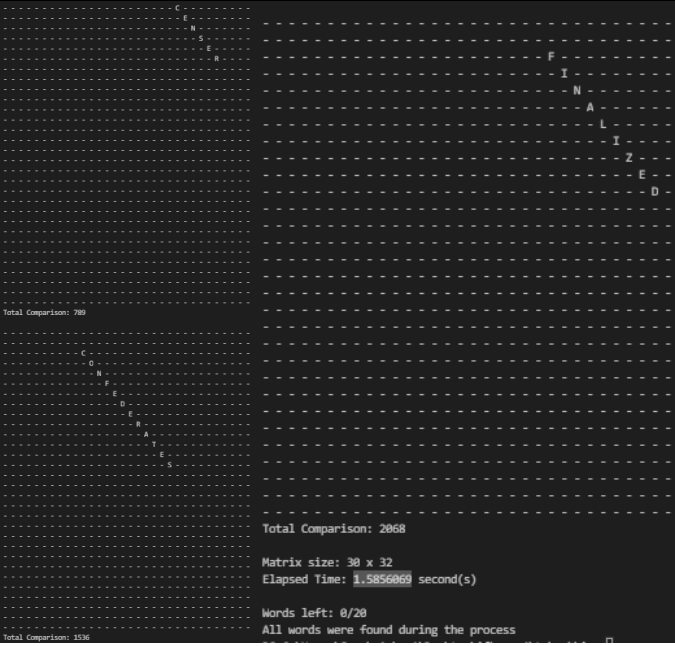

8.

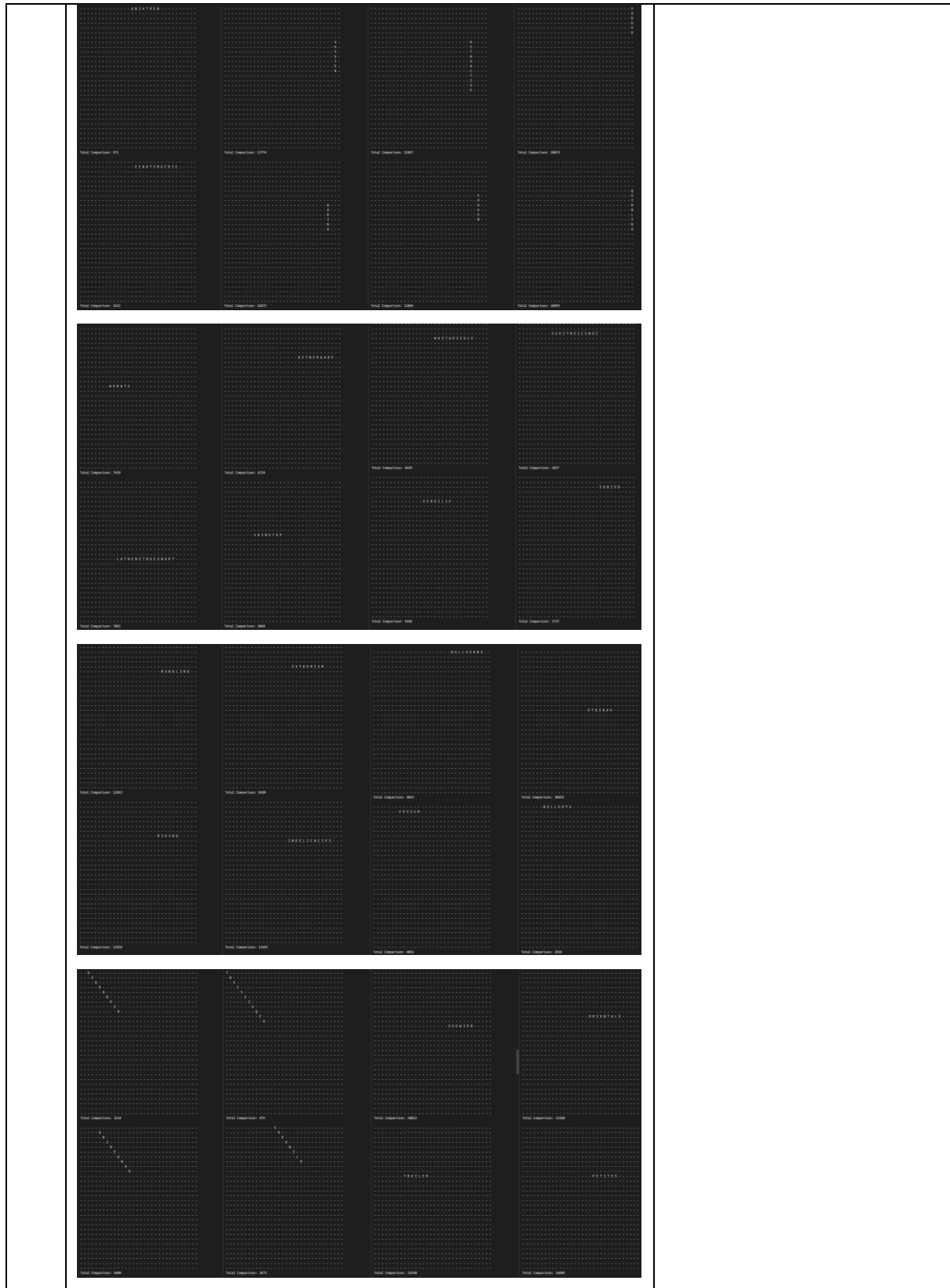
```

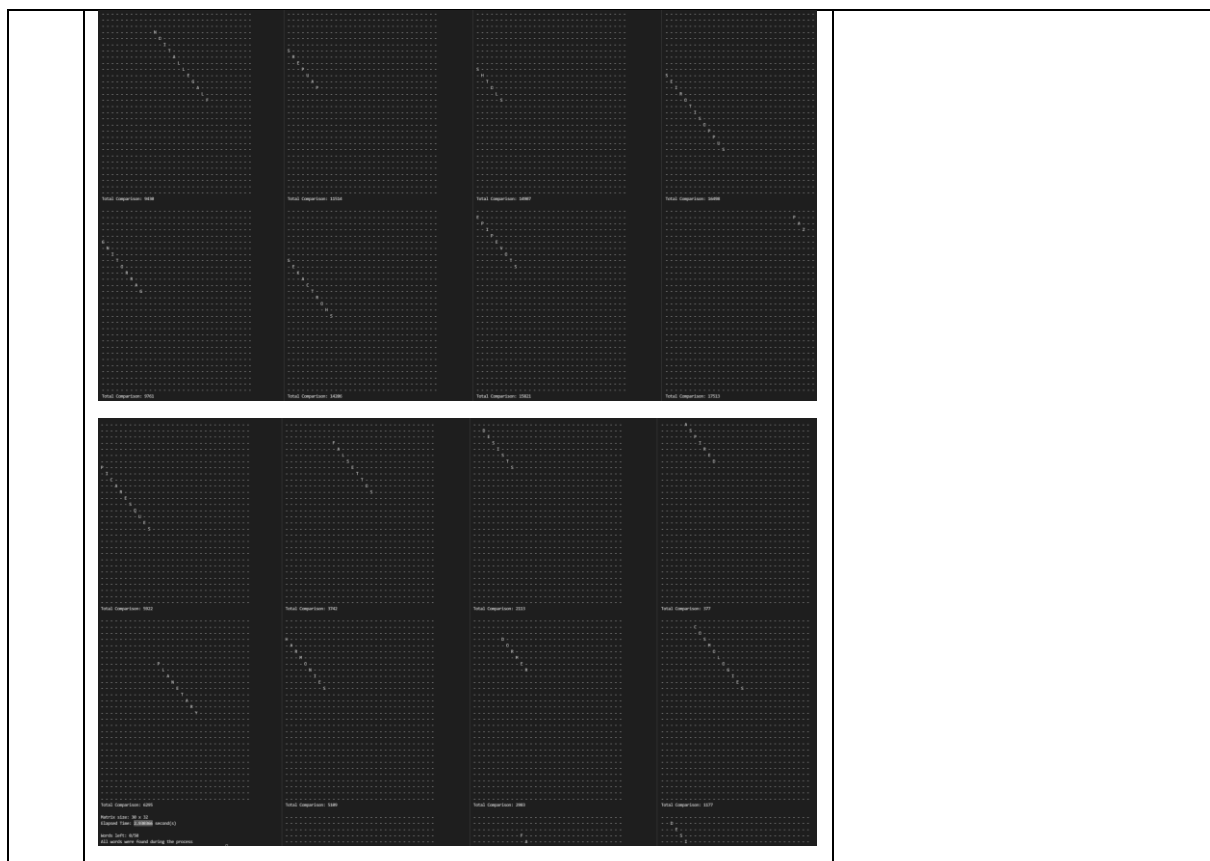
D
I
R
K
S
E
N
A
M
O
R
E
D

```

1.5856069 second(s)

	 <p>The screenshot shows a word search puzzle solver interface. It displays a 30x32 matrix of letters. The search results show the following words found: C, N, S, R, F, I, N, A, L, I, Z, E, D. The total comparison count is 789. The matrix size is 30 x 32. The elapsed time is 1.5850869 second(s). The words left are 0/20. All words were found during the process.</p>	
9.	 <p>The screenshot shows a word search puzzle solver interface. It displays a 30x32 matrix of letters. The search results show the following words found: A, G, R, I, E, V, I, N, G. The total comparison count is 165. The matrix size is 30 x 32. The elapsed time is 2.930366 second(s). The words left are 0/20. All words were found during the process.</p>	<p>Size: Large</p> <p>Matrix Size: 30 x 32</p> <p>Words to Search: 50</p> <p>Elapsed Time: 2.930366 second(s)</p>





LAMPIRAN

Github Repository: [azkazkazka/word-search-puzzle-solver \(github.com\)](https://github.com/azkazkazka/word-search-puzzle-solver)

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