

**Laporan Tugas Kecil 1 IF2211 Strategi Algoritma  
Semester II Tahun Akademik 2021/2022**

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



**Disusun oleh:  
Ahmad Alfani Handoyo  
13520023  
K-02**

**Program Studi S1 Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung**

### A. Algoritma *Brute Force* untuk menyelesaikan *Word Search Puzzle*

Algoritma yang digunakan untuk menyelesaikan *word search puzzle* adalah metode *brute force* dengan teknik heuristik untuk hanya mengecek indeks yang memuat seluruh panjang kata yang diterima (pada masing-masing arah mata angin) sehingga tidak ada masalah *out-of-bounds checking*.

Langkah-langkah algoritma yang digunakan dalam menyelesaikan masalah tersebut adalah sebagai berikut:

1. Telusuri seluruh indeks berelemen karakter pada matriks (mulai dari pojok kiri atas hingga pojok kiri kanan)
2. Bandingkan karakter pada indeks matriks dengan huruf pertama kata yang dicari. Bila sama, maka cek huruf-huruf kata berikutnya pada semua 8 mata angin. Bila tidak, geser ke indeks berikutnya.
3. Pada pengecekan setiap indeks mata angin, bila huruf sama geser ke posisi berikutnya pada matriks sesuai dengan arah mata angin. Bila huruf tidak sama cek pada arah mata angin yang lain.
4. Bila ditemukan satu mata angin yang mana seluruh huruf pada kata yang dicari sesuai, maka solusi untuk kata tersebut ditemukan untuk indeks tersebut. Bila tidak, geser pencarian ke indeks berikutnya.
5. Bila hingga indeks terakhir tidak ditemukan indeks yang memuat solusi kata, maka solusi tidak ditemukan.

### B. *Source Code Program*

Program ditulis pada bahasa C dengan semua program diletakkan pada satu *file* bernama *main.c* pada folder */src*. Terlampir di bawah *source code* program seperti pada *file main.c*:

```
/* NAMA : Ahmad Alfani Handoyo
   NIM : 13520023
   Kelas : K02
   Tugas Kecil 1 IF2211 Strategi Algoritma Semester II tahun 2021/2022 */

#include <stdio.h>
#include <string.h>
#include <time.h>

typedef struct {
    char word[1000];
    int length;
```

```

} Word;

typedef struct {
    char matrix[100][100];
    int row;
    int col;
} Matrix;

// Print matrix
void PrintMatrix (Matrix matrix) {
    int i, j;
    for (i=0; i < matrix.row; i++) {
        for (j=0; j < matrix.col; j++) {
            printf("%c ", matrix.matrix[i][j]);
        }
        printf("\n");
    }
}

// Make a copy of a matrix with '-' in every index
void CopyEmptyMatrix (Matrix original, Matrix *copy) {
    Matrix temp = original;
    int i, j;
    for (i=0; i < temp.row; i++) {
        for (j=0; j < temp.col; j++) {
            temp.matrix[i][j] = '-';
        }
    }
    *copy = temp;
}

// Check first letter of word
void solveFirst (Matrix Puzzle, int pRow, int pCol, Word word, int *compare, int *firstFound) {
    int bool;
    if (Puzzle.matrix[pRow][pCol] != word.word[0]) {
        bool = 0;
    } else {
        bool = 1;
    }
    *compare = *compare + 1;
    *firstFound = bool;
}

// Check for direction East
void solveE (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

```

```

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pCol < (Puzzle.col - word.length + 1)) {
            int i = pRow, j = pCol+1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                j++; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

// Check for direction South-East
void solveSE (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol < (Puzzle.col - word.length + 1)) && (pRow < (Puzzle.row - word.length + 1))) {
            int i = pRow+1, j = pCol+1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
            }
        }
    }
}

```

```

        }
        i++; j++; k++; counter++;
    }

    if (allFound) {
        PrintMatrix(Solution);
        printf("\n");
    }

    *foundBool = allFound;
    *compare = counter;
}
}

// Check for direction South
void solveS (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pRow < (Puzzle.row - word.length + 1)) {
            int i = pRow+1, j = pCol, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
            }
            i++; k++; counter++;
        }

        if (allFound) {
            PrintMatrix(Solution);
            printf("\n");
        }

        *foundBool = allFound;
        *compare = counter;
    }
}
}

```

```

// Check for direction South-West
void solveSW (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol >= (word.length - 1)) && (pRow < (Puzzle.row - word.length + 1))) {
            int i = pRow+1, j = pCol-1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i++; j--; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

```

```

// Check for direction West
void solveW (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pCol >= (word.length - 1)) {
            int i = pRow, j = pCol-1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];

```

```

        if (Puzzle.matrix[i][j] != word.word[k]) {
            bool = 0;
        } else {
            if (k == word.length-1) {
                allFound = 1;
            }
        }
        j--; k++; counter++;
    }

    if (allFound) {
        PrintMatrix(Solution);
        printf("\n");
    }

    *foundBool = allFound;
    *compare = counter;
}

}

// Check for direction North-West
void solveNW (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol >= (word.length - 1)) && (pRow >= (word.length - 1))) {
            int i = pRow-1, j = pCol-1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i--; j--; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

```

```

        *foundBool = allFound;
        *compare = counter;
    }
}

// Check for direction North
void solveN (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;
        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if (pRow >= (word.length - 1)) {
            int i = pRow-1, j = pCol, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
            }
            i--; k++; counter++;
        }

        if (allFound) {
            PrintMatrix(Solution);
            printf("\n");
        }

        *foundBool = allFound;
        *compare = counter;
    }
}

// Check for direction North-East
void solveNE (Matrix Puzzle, int pRow, int pCol, Word word, int *foundBool, int *compare) {
    Matrix Solution;
    CopyEmptyMatrix(Puzzle, &Solution);

    if (!*foundBool) {
        int bool, allFound, counter;

```



```

        counter = *compare; bool = 1; allFound = 0;
        Solution.matrix[pRow][pCol] = Puzzle.matrix[pRow][pCol];
        if ((pCol < (Puzzle.col - word.length + 1)) && (pRow >= (word.length - 1))) {
            int i = pRow-1, j = pCol+1, k = 1;
            while ((k < word.length) && bool) {
                Solution.matrix[i][j] = Puzzle.matrix[i][j];
                if (Puzzle.matrix[i][j] != word.word[k]) {
                    bool = 0;
                } else {
                    if (k == word.length-1) {
                        allFound = 1;
                    }
                }
                i--; j++; k++; counter++;
            }

            if (allFound) {
                PrintMatrix(Solution);
                printf("\n");
            }

            *foundBool = allFound;
            *compare = counter;
        }
    }
}

int main() {
    char repeatprogram = 'Y';
    while (repeatprogram == 'Y') {
        clock_t time;
        int i = 0, j = 0, k = 0, l = 0;

        printf("=====\n");
        printf("WELCOME TO WORD SEARCH PUZZLE SOLVER\n");
        printf("by 13520023 - Ahmad Alfani Handoyo\n");
        printf("=====\n\n");

        // Read filename of puzzle
        char directory[207] = "../test/";
        char filename[200];
        printf("Insert filename of puzzle (including the file format): ");
        scanf("%s", filename);
        strcat(directory,filename);

        // Read file
        char c, cPrev;

```

```

int Row = 0, Col = 0, foundRow = 0;
FILE *fp = fopen(directory, "r");

if (fp == NULL) {
    printf("No such file with name %s is found!\n\n", filename);
} else {
    printf("-----\n");
    printf("      WORD SEARCH PUZZLE\n");
    printf("-----\n");

    // Read rows and columns of puzzle matrix
    c = cPrev = getc(fp);
    while ((c != '\n') || (cPrev != '\n')) {
        if (c == '\n') {
            foundRow = 1;
            Row++;
        }
        if (!foundRow && c != ' ') {
            Col++;
        }
        cPrev = c;
        c = getc(fp);
    }
    fclose(fp);
    printf("%d ROWS and %d COLUMNS\n\n", Row, Col);

    // Read puzzle matrix
    Matrix Puzzle;
    Puzzle.row = Row; Puzzle.col = Col;

    fp = fopen(directory, "r");
    c = cPrev = getc(fp);
    while ((c != '\n') || (cPrev != '\n')) {
        if (c != '\n' && c != ' ') {
            Puzzle.matrix[i][j] = c;
            j++;
            if (j == Col) {
                j = 0;
                i++;
            }
        }
        cPrev = c;
        c = getc(fp);
    }

    // Print puzzle
    PrintMatrix(Puzzle);
}

```

```

// Read words
int wordcount = 0, wordlength = 0;
Word words[30];
c = getc(fp);
while (c != EOF) {
    if (c == '\n') {
        words[wordcount].length = wordlength;
        wordcount++;
        wordlength = 0;
    } else {
        words[wordcount].word[wordlength] = c;
        wordlength++;
    }
    c = getc(fp);
}
words[wordcount].length = wordlength;
wordcount++;
fclose(fp);

// Print words
printf("\nKEYWORDS:\n");
for (i=0; i < wordcount; i++) {
    printf("%d. ", i+1);
    for (j = 0; j < words[i].length; j++) {
        printf("%c", words[i].word[j]);
    }
    printf("\n");
}
printf("\n");

// Brute-Force Algorithm
printf("-----\n");
printf("          SOLUTIONS\n");
printf("-----\n");
time = clock();
int found, firstFound, compare = 0;
for (i=0; i < wordcount; i++) {
    printf("%d. ", i+1);
    for (l = 0; l < words[i].length; l++) {
        printf("%c", words[i].word[l]);
    }
    printf("\n");

    found = 0;
    j = 0;
    while (j < Row && !found) {
        k = 0;
        while (k < Col && !found) {

```

```

        solveFirst(Puzzle, j, k, words[i], &compare, &firstFound);
        if (firstFound) {
            solveE(Puzzle, j, k, words[i], &found, &compare);
            solveSE(Puzzle, j, k, words[i], &found, &compare);
            solveS(Puzzle, j, k, words[i], &found, &compare);
            solveSW(Puzzle, j, k, words[i], &found, &compare);
            solveW(Puzzle, j, k, words[i], &found, &compare);
            solveNW(Puzzle, j, k, words[i], &found, &compare);
            solveN(Puzzle, j, k, words[i], &found, &compare);
            solveNE(Puzzle, j, k, words[i], &found, &compare);
        }
        k++;
    }
    j++;
}
if (!found) {
    printf("Word is not found in the puzzle\n\n");
}
}
time = clock() - time;
int time_consumed = (((double) time)/CLOCKS_PER_SEC)*1000;
printf("-----\n");
printf("          STATISTICS\n");
printf("-----\n");
printf("TOTAL LETTER COMPARISON: %d LETTERS\n", compare);
printf("TOTAL TIME TAKEN: %d milliseconds\n\n", time_consumed);
}

printf("Do you want to solve another puzzle? (Y/N): ");
scanf(" %c", &repeatprogram);
}
return 0;
}

```

### C. Screenshot Hasil Input dan Output Program

Mesin yang digunakan untuk menguji coba program berjalan pada Windows 11 build 22000.434. Mesin juga mempunyai spesifikasi prosesor AMD Ryzen 7 4800H dengan cache L1 512KB, L2 4MB, dan L3 8MB dan RAM teralokasi 23,4GB.

#### 1. Ukuran *small*

small1.txt (16x15, 11 kata)

```
Insert filename of puzzle (including the file format): small1.txt
```

WORD SEARCH PUZZLE

16 ROWS and 15 COLUMNS

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

KEYWORDS:

1. SIMPLICITY
2. LANGUAGE
3. CHASE
4. WARRANT
5. EVOLUTION
6. LEASH
7. BULLETIN
8. TOLERANT
9. WHEEL
10. MECHANISM
11. INTELLIGENCE

## SOLUTIONS

## 1. SIMPLICITY

## SIMPLICITY

## 2. LANGUAGE

- LANGUAGE

### 3. CHASE

#### 4. WARRANT

## 5. EVOLUTION

NO ITULOVE

## 6. LEASH

H  
S  
A  
E  
L

## 7. BULLETIN

7. BULLETIN

NITELLUB

## 8. TOLERANT

— T O L E R A N T —

9. WHEEL

LEEHW

10. MECHANISM

MSINAHCEM

```

11. INTELLIGENCE
-----
I
N
T
E
L
L
I
G
E
N
C
E
-----
STATISTICS
-----
TOTAL LETTER COMPARISON: 1502 LETTERS
TOTAL TIME TAKEN: 273 miliseconds
Do you want to solve another puzzle? (Y/N):

```

small2.txt (18x16, 13 kata)

```

=====
WELCOME TO WORD SEARCH PUZZLE SOLVER
by 13520023 - Ahmad Alfani Handoyo
=====

Insert filename of puzzle (including the file format): small2.txt

=====
WORD SEARCH PUZZLE
=====

18 ROWS and 16 COLUMNS

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

KEYWORDS:
1. BROADCAST
2. GOLD
3. IMPLICIT
4. FAITHFUL
5. INFLATION
6. ENTERTAIN
7. ROLE
8. SEPARATION
9. FUR
10. LAST
11. TRACK
12. SOLVE
13. PROPOSAL

```

[illegible]

3. IMPLICIT

4. FAITHFUL

5. INFLATION

6. ENTERTAIN

7. ROLE

8. SEPARATION

9. FUR

10. LAST

11. TRACK

12. SOLVE

13. PROPOSAL

STATISTICS

TOTAL LETTER COMPARISON: 2516 LETTERS  
TOTAL TIME TAKEN: 385 milliseconds

Do you want to solve another puzzle? (Y/N):

small3.txt (15x15, 10 kata)

```
=====
WELCOME TO WORD SEARCH PUZZLE SOLVER
by 13520023 - Ahmad Alfani Handoyo
=====

Insert filename of puzzle (including the file format): small3.txt
=====
WORD SEARCH PUZZLE
=====
15 ROWS and 15 COLUMNS

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

KEYWORDS:
1. ALLOWANCE
2. DETECTIVE
3. OVERCHARGE
4. ADVERTISE
5. APPOINTMENT
6. EXPERIMENT
7. CHOCOLATE
8. EXPECT
9. INTERFERENCE
10. ACCUMULATION

=====
SOLUTIONS
=====
1. ALLOWANCE
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

2. DETECTIVE
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

3. OVERCHARGE
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

4. ADVERTISE
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

5. APPOINTMENT
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

6. EXPERIMENT
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

7. CHOCOLATE
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

8. EXPECT
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
```





### 3. RESTLESS

- - - - - R - - - - -  
 - - - - - E - - - - -  
 - - - - - S - - - - -  
 - - - - - T - - - - -  
 - - - - - L - - - - -  
 - - - - - E - - - - -  
 - - - - - S - - - - -  
 - - - - - S - - - - -

#### 4. EXPLICIT

E  
 X  
 P  
 L  
 I  
 C  
 I  
 T

## 5. APPLIED

- - - - - A -  
 - - - - - P -  
 - - - - - P -  
 - - - - - L -  
 - - - - - I -  
 - - - - - E -  
 - - - - - D -

## 7. VIGOROUS

- - V I G O R O U S

## 6. ARCHITECTURE

- - - - E R U T C E T I H C R A - -

## 8. CONTRIBUTION

----- NOITUBIRTC

## 9. ANNOUNCEMENT

A  
N  
N  
O  
U  
N  
C  
E  
M  
E  
N  
T

## 10. THOUGHT

## 11. PARALYZED

----- PARALYZED -----

## 12. UNDERSTAND

D  
N  
A  
T  
S  
R  
E  
D  
N  
U

## STATISTICS

```
TOTAL LETTER COMPARISON: 2678 LETTERS
TOTAL TIME TAKEN: 446 miliseconds
```

Do you want to solve another puzzle? (Y/N):

medium2.txt (22x20, 14 kata)

WELCOME TO WORD SEARCH PUZZLE SOLVER  
by 13520023 - Ahmad Alfani Handoyo

```
Insert filename of puzzle (including the file format): medium2.txt
```

WORD SEARCH PUZZLE

22 ROWS and 20 COLUMNS

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

**KEYWORDS :**

1. RETIREE
2. INTEREST
3. REPLACEMENT
4. ENTERTAIN
5. WARRANT
6. WEAKNESS
7. PROVISION
8. LIBERTY
9. ACTIVATE
10. ANNOUNCEMENT
11. INHABITANT
12. DEPRIVATION
13. MIGRATION
14. COMPETITION

## SOLUTIONS

## 1. RETIREE

E  
E  
R  
I  
T  
E  
R

## 2. INTEREST

T  
S  
E  
R  
E  
T  
N  
I

### 3. REPLACEMENT

```

graph TD
    T --> N
    T --> E1[E]
    N --> M
    N --> E2[E]
    M --> E3[E]
    M --> C
    E3 --> A
    E3 --> L
    A --> P
    A --> E4[E]
    L --> E5[E]
    L --> R
  
```

#### 4. ENTERTAIN

N  
I  
A  
T  
R  
E  
T  
N  
E

## 5. WARRANT

WARRANT

## 6. WEAKNESS

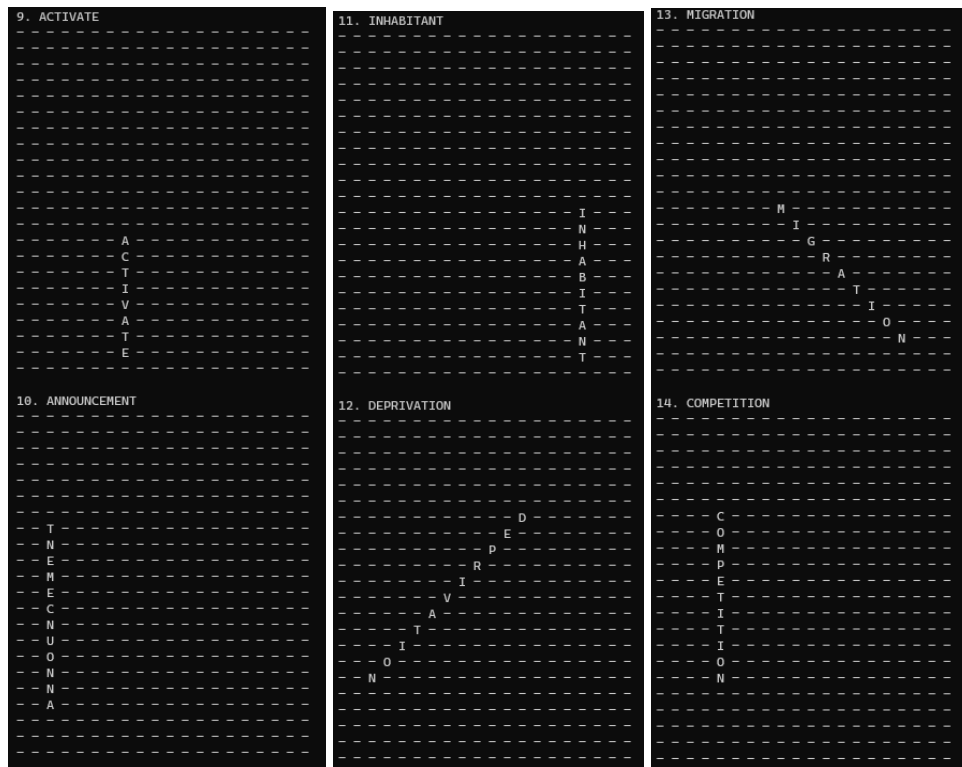
S  
S  
E  
N  
K  
A  
E  
W

## 7. PROVISION

NOI  
S  
I  
V  
O  
R  
P

## 8. LIBERTY

Y T R E B I L



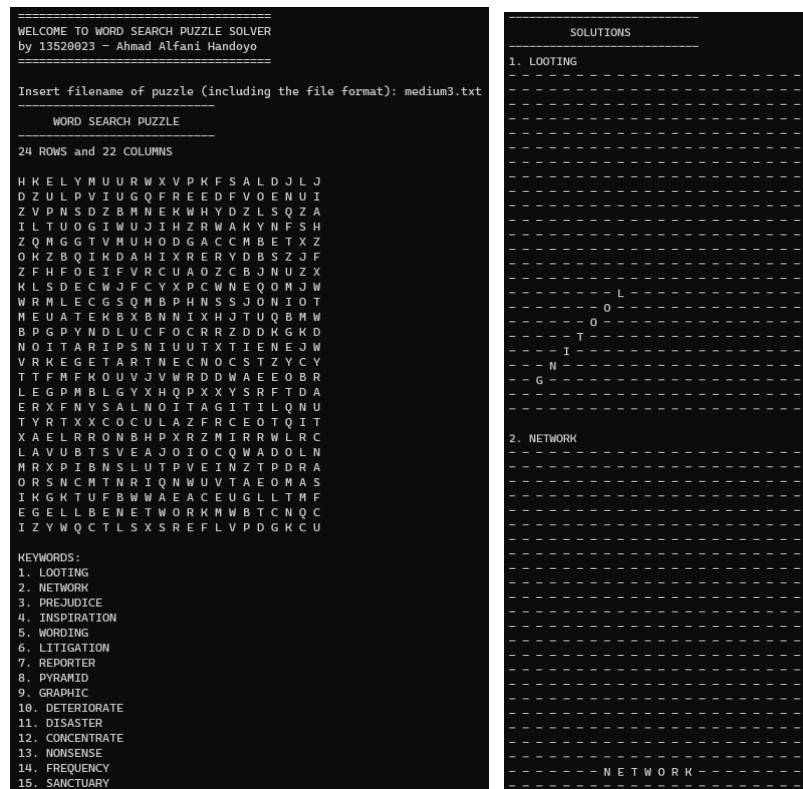
STATISTICS

TOTAL LETTER COMPARISON: 4162 LETTERS

TOTAL TIME TAKEN: 619 milliseconds

Do you want to solve another puzzle? (Y/N):

medium3.txt (24x22, 15 kata)



3. PREJUDICE

P

R

E

J

U

D

I

C

E

4. INSPIRATION

NOITARIPSI

5. WORDING

G

N

I

D

R

O

W

6. LITIGATION

NOITAGITIL

7. REPORTER

R

E

P

O

R

T

E

R

8. PYRAMID

D

I

M

A

R

Y

P

9. GRAPHIC

G

R

A

P

H

I

C

10. DETERIORATE

D

E

T

E

R

I

O

R

A

T

E

11. DISASTER

D

I

S

A

S

T

E

R

12. CONCENTRATE

ETARTNECNOC

13. NONSENSE

E

S

N

E

S

N

O

N

14. FREQUENCY

Y

C

N

E

U

Q

E

R

F



1. NEGLECT

NEGLECT

2. CRYSTAL

CRYSTAL

3. SURVIVOR

SURVIVOR

4. CABINET

CABINET

5. MORNING

MORNING

6. SKELETON

SKELETON

7. SQUEEZE

S  
Q  
U  
E  
Z  
E

8. DEPRIVATION

D  
E  
P  
R  
I  
V  
A  
T  
I  
O  
N

9. SUPPRESS

S  
U  
P  
P  
R  
E  
S  
S

10. PERFUME

P  
E  
R  
F  
U  
M  
E

11. IMPULSE

I  
M  
P  
U  
L  
S  
E

12. MECHANICAL

M  
E  
C  
H  
A  
N  
I  
C  
A  
L





1. ENTERTAINMENT

E  
N  
T  
E  
R  
T  
A  
I  
N  
M  
E  
N  
T

2. CONTRIBUTION

NOITUBIRTNOC

3. INTRODUCTION

N  
O  
I  
T  
T  
C  
U  
D  
D  
O  
R  
T  
N  
I

4. ACCESSIBLE

A  
C  
C  
E  
S  
S  
I  
B  
L  
E

5. CONTINENTAL

C  
O  
N  
T  
I  
N  
E  
N  
T  
A  
L

6. DISTRIBUTOR

D  
I  
S  
T  
R  
I  
B  
U  
T  
O  
R

7. DEFICIENCY

D  
E  
F  
I  
C  
I  
E  
N  
C  
Y

8. DEMONSTRATION

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

9. ESTABLISHED

E  
S  
T  
A  
B  
L  
I  
S  
H  
E  
D

10. INSTITUTION

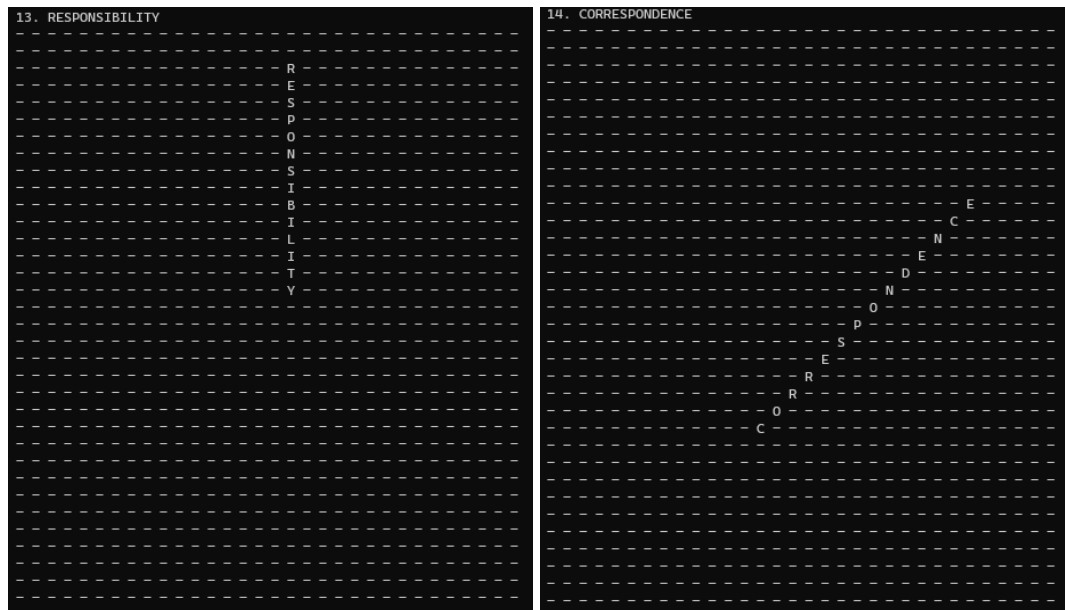
I  
N  
S  
T  
I  
T  
U  
T  
I  
O  
N

11. INSTITUTION

I  
N  
S  
T  
I  
T  
U  
T  
I  
O  
N

12. CONSERVATIVE

C  
O  
N  
S  
E  
R  
V  
A  
T  
I  
V  
E



```

-----
STATISTICS
-----
TOTAL LETTER COMPARISON: 8421 LETTERS
TOTAL TIME TAKEN: 1497 milliseconds

Do you want to solve another puzzle? (Y/N): |

```

large3.txt (36x34, 13 kata)

```

=====
WELCOME TO WORD SEARCH PUZZLE SOLVER
by 13520023 - Ahmad Alfani Handoyo
=====

Insert filename of puzzle (including the file format): large3.txt

WORD SEARCH PUZZLE

36 ROWS and 34 COLUMNS

RGXQULUAIVCCTTRCTTPNVFBQQTNTRIDPIE
IQRJYAMINFRASCTUREJBHVZXLKSSJWW
RSYHNXPYSFUPELNBGHPFUNAEMCSLGYYHTL
LADGIGENSSSFZURUONRSDCPHVUESZHKKCJ
PKOCIKTMNUJQCMZKRIEKVQNNHIZVRTWOOH
ELWYCIYBFLDWGAVHSEONPDZSSKTNGQVWZL
ZLEXUWZBZFYZPNOITACILPMOCRHCSSLWUGP
ZRYYZNUSDTXRRRMALBCIPRITFFCOAMBKUQ
PZQULTGDRQACAJZJNCUXKGNFHHGIPOREFNU
VZPBJETYBWTJVVZNNKNDPNSPCRCOBPMREVFZ
CHKIGXNZFANSEHFOCZANFVPOVVEPCJHTPR
JGPXDRXSEIEYBQYIWOTBXXKETURNJXSSVNC
CRNTYKRRHKKMWSKTDANIBLSYUANFQVRDUII
ZCEIHHTZTVHEOBKPNHTOZBPATXETCOUZZAA
YGDWIDDQAHNLIVPHKSNWFMIINFORMATION
OCRBRMVPKRPWTSSTENHQSOZPJZMUZYMKNB
KPHFEYIIHYPADZAFFJWLNFEKXCFSSUSLNGZG
KUYFHAEEKVOUEXRHJPJVACCWVTEBMGRIUBF
OOHXFLUXJRSTLEKAPULIPKNRHTBNYNADBBH
YLIQPDROWIAPOGZOZXEWVAKTAAHOUPJEU
EOKVEFAOBXFLAIHJIFCZSNBNJRJCYIFTNNT
XVGRZKTTTPCKQSNQZNJGSKWFUTTIAIZENP
GVHHYOLAKSRAWTZFNURMUNGFFSRTKQODJW
CELECTRONICSERPFXXZIBVKTNNQNGCHPRKON
VDWQUOMEOEZIIABYTSJBMVQZTOUFEBESVC
SGQTNZKLKGBZETLTSOYUQNLKWMVJTXBYUF
SZQVOZRTGJNYIYIMYGLIAJTTJEMHRKZCNB
KHMJHGRVAXXTROOLKVVWGRAETPDJYZJCAKI
LDXZGLHULORFQNWIGOKFFKLMNRDWTFLUXQ
HQEXIQECQMVOZPABDSKLVZNFAGAMSVUHPD
IDNXBBMBVSULCFDIRVZYGYDFYDLJQFARDA
JHBKZKHUSOCODYTADTQZQWUZATSKMCUIUOR
CNHGHWSMCWHHNJPEGPWULRAEGSSRWZVTD
ACKKIJBYOGGKYVJRPSZQXLDLVJJVMVGRFR
UZYRSPXJMMKGTJKCIISYNIOPOKKLSJHQR
FTWIZCBQBFZHXBULFMOIIZKDDKGHGBMIET

```

```

KEYWORDS:
1. INTERACTIVE
2. ELECTRONICS
3. REGISTRATION
4. TRANSMISSION
5. COMPLICATION
6. COUNTRYSIDE
7. PREOCCUPATION
8. OPERATIONAL
9. INFORMATION
10. CREDIBILITY
11. DEMONSTRATE
12. INFRASTRUCTURE
13. SUPPLEMENTARY

SOLUTIONS

```

1. INTERACTIVE

E  
V  
I  
T  
C  
A  
R  
E  
T  
N  
I

2. ELECTRONICS

E L E C T R O N I C S

3. REGISTRATION

R  
E  
G  
I  
S  
T  
R  
A  
T  
I  
O  
N

4. TRANSMISSION

T  
R  
A  
N  
S  
M  
I  
S  
S  
I  
O  
N

5. COMPLICATION

N O I T A C I L P H O C

6. COUNTRYSIDE

C  
O  
U  
N  
T  
R  
Y  
S  
I  
D  
E

7. PREOCCUPATION

P  
R  
E  
O  
C  
C  
U  
P  
A  
T  
I  
O  
N

8. OPERATIONAL

O  
P  
E  
R  
A  
T  
I  
O  
N  
A  
L

9. INFORMATION

I  
N  
F  
O  
R  
M  
A  
T  
I  
O  
N

10. CREDIBILITY

C  
R  
E  
D  
I  
B  
I  
L  
I  
T  
Y

11. DEMONSTRATE

D  
E  
M  
O  
N  
S  
T  
R  
A  
T  
E

12. INFRASTRUCTURE

I  
N  
F  
R  
A  
S  
T  
R  
U  
C  
T  
U  
R  
E

```

13. SUPPLEMENTARY
-----
Y
R
A
T
N
E
H
E
L
P
P
U
S
-----

STATISTICS
-----
TOTAL LETTER COMPARISON: 8166 LETTERS
TOTAL TIME TAKEN: 1578 milliseconds
Do you want to solve another puzzle? (Y/N): |

```

#### D. Alamat Unduh Program

Program dapat diunduh pada *repository* berikut:

<https://github.com/blueguy42/Word-Search-Solver>

#### E. Tabel Keberjalanan Program

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.	✓	