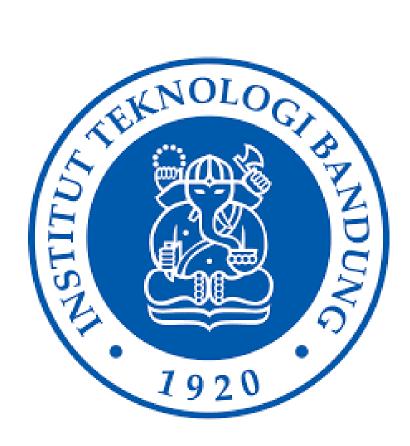
# Tugas Kecil IF2211 Strategi Algoritma

Penyelesaian Word Search Puzzle dengan Algoritma Brute Force



Ilham Pratama 13520041 Kelas K-02

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

## **DAFTAR ISI**

| Algoritma Brute Force         | 3  |
|-------------------------------|----|
| Source Code Program           | 3  |
| Screen Shoot Input dan Output | 8  |
| Input dan Output 1            | 8  |
| Input dan Output 2            | 8  |
| Input dan Output 3            | 9  |
| Input dan Output 4            | 9  |
| Input dan Output 5            | 10 |
| Input dan Output 6            | 11 |
| Input dan Output 7            | 11 |
| Input dan Output 8            | 12 |
| Input dan Output 9            | 14 |
| Alamat Source Code Program    | 16 |

#### **Algoritma Brute Force**

Penyelesaian masalah tersebut secara garis besar mengikuti langkah-langkah sebagai berikut

- 1. Program akan memuat file, bagian file akan dipisah menjadi 2 bagian, bagian yang pertama merupakan puzzle, dan bagian kedua adalah kata yang akan dicari. Bagian puzzle akan disimpan program dalam bentuk array of array(matriks) dengan nama matriks *puzzle* dan bagian yang akan dicari akan disimpan juga dalam array dengan nama *words*
- 2. Setelah disimpan ke dalam bentuk matriks, selanjutnya program akan melakukan pencocokan, program akan mulai *looping*, jika ditemukan ada elemen *puzzle* yang bernilai sama dengan elemen yang ada di *words*, maka elemen pada *puzzle* akan di cek pada sebelah kanan, kiri, atas, bawah, diagonal kanan atas, diagonal kanan bawah, diagonal kiri atas dan diagonal kanan bawahnya.
- 3. Setelah dilakukan pengecekan dan ternyata elemennya sama dengan elemen yang ada di *words*, maka lokasi dari elemennya yang sama akan disimpan dalam suatu matriks yang bernama *match* dengan nilai 1. Selanjutnya pengecekan akan terus dilanjutkan sampai elemen pada *words* berada pada kolom terakhir dari baris yang dicari.
- 4. Jika semua nilainya sama, maka program akan pengecek apakah ada kemungkinan lain
- 5. Jika tidak ada, maka program akan lanjut ke elemen baris selanjutnya yang ada di words
- 6. Hal tersebut akan terus berulang hingga semua baris pada words habis.
- 7. Jika baris pada *words* sudah habis maka program akan menampilkan elemen pada *puzzle* yang mempunyai match 1, dan jika elemen mempunyai match selain 1 maka elemen akan diubah menjadi " ".
- 8. Selanjutnya program akan menampilkan waktu dan jumlah perbandingan kata yang dilakukan

## **Source Code Program**

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<unistd.h>
#include<time.h>
#define ROWS 100
#define COLS 100
#define MAXWORD 100
void InitMatched(int match[ROWS][COLS]) {
    /*digunakan untuk inisialisasi matrik untuk matched, elemen diisi
dengan 0 */
    int i, j;
    for(i = 0 ; i < ROWS ; i++) {</pre>
        for(j = 0 ; j < COLS; j++) {</pre>
            match[i][j] = 0;
        }
    }
```

```
void Max(int current, int *max) {
    // menentukan bilangan yang terbesar antara 2 bilangan
    if(current > *max) {
        *max = current;
}
int WordLen(char *word) {
    // mencari panjang dari Word
    int i;
    for (i = 0 ; word[i] != '\0'; i++){}
    i--;
    return i;
}
void Strip(char* word) {
    // mengupas string di word
    int i;
    for(i = 0 ; word[i] != '\0'; i++) {
        if(word[i] == '\n'){
            word[i] = '\0';
        }
    }
int main() {
    char puzzle[ROWS][COLS];
    int match[ROWS][COLS];
    char words[MAXWORD][COLS];
    char cc;
    int row = 0;
    int col = 0;
    int maxcol = 0;
    int numword = 0;
    int i, j, k, l;
    FILE *puzzlefile;
    FILE *keysearch;
    double time spend = 0.0;
    int jumlah banding kata = 0;
    char filename[50], keyname[50];
    /*inisialisasi matcher*/
    InitMatched(match);
    printf("masukkan nama file puzzle (e.x. puzzle1.txt) : \n");
    scanf("%s", &filename);
    printf("masukkan nama file kata yang dicari (e.x. key1.txt) : \n");
    scanf("%s", &keyname);
    puzzlefile = fopen(filename, "r");
    if (puzzlefile == NULL) {
        printf("gagal membuka file, periksa dulu apakah file yang anda
masukkan benar \n");
        return 0;
    }
```

```
while(!feof(puzzlefile)){
        cc = fgetc(puzzlefile);
        if(cc == '\n') {
            row++;
            col = 0;
        }else if(isalnum(cc)){
            puzzle[col][row] = toupper(cc);
            col++;
            Max(col, &maxcol);
        }
    }
    keysearch = fopen(keyname, "r");
    if(keysearch == NULL) {
        printf("tidak ada kata yang akan dicari\n");
        return 0;
    while(!feof(keysearch)){
        fgets(words[numword], 100, keysearch);
        Strip(words[numword]);
        numword++;
    numword--;
    //perhitungan waktu dimulai
    clock t begin = clock();
    /*algoritma matching */
    for(i = 0; i < numword; i++) {</pre>
        for (k = 0 ; k < row; k++) {
            for(j = 0; j < maxcol; j++) {</pre>
                 if(puzzle[j][k] == words[i][0]){
                     if (maxcol > (WordLen(words[i]) + j)) {
                         for(l = 0; words[i][l] == puzzle[j+l][k]; l++){
                             if (words[i][l+1] == '\0'){
                                  for(l = 0; words[i][l] == puzzle[j+l][k];
1++) {
                                      match[j+l][k] = 1;
                                      jumlah banding kata++;
                                 break;
                             }else{
                                  jumlah banding kata++;
                             }
                         if (row > (WordLen(words[i]) + k)) {
                             for(l=0; words[i][l] == puzzle[j+l][k+l];
1++) {
                                 if (words[i][l+1]=='\0') {
                                      for(l=0; words[i][l] ==
puzzle[j+l][k+l]; l++)
                                          match[j+l][k+l] = 1;
                                          jumlah banding kata++;
                                      break;
                                  }else{
                                      jumlah banding kata++;
                                  }
```

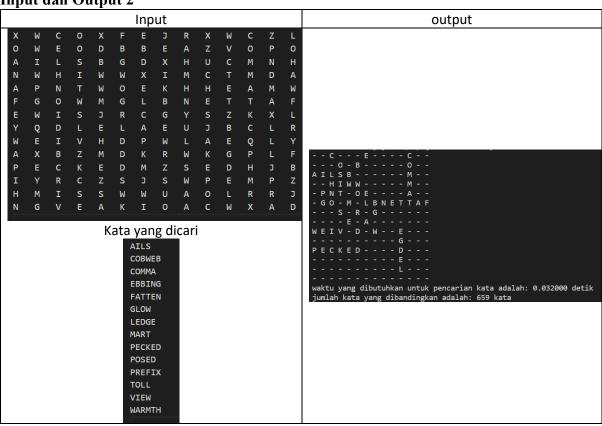
```
if (0 <= (k - WordLen(words[i]))) {</pre>
                             for(l=0; words[i][l] == puzzle[j+l][k-l];
1++) {
                                  if (words[i][l+1]=='\0') {
                                      for(l=0; words[i][l] ==
puzzle[j+l][k-l]; l++)
                                          match[j+1][k-1] = 1;
                                          jumlah banding kata++;
                                     break;
                                 }else{
                                      jumlah banding kata++;
                             }
                         }
                     }
                     if (0 <= (j - WordLen(words[i]))){</pre>
                         for(l=0; words[i][l] == puzzle[j-l][k]; l++) {
                             if (words[i][l+1]=='\0') {
                                 for(l=0; words[i][l] == puzzle[j-l][k];
1++)
                                      match[j-l][k] = 1;
                                      jumlah banding kata++;
                                 break;
                             }else{
                                  jumlah banding kata++;
                         if (row > (WordLen(words[i]) + k)) {
                             for(l=0; words[i][l] == puzzle[j-l][k+l];
1++) {
                                  if (words[i][l+1]=='\0') {
                                      for(l=0; words[i][l] == puzzle[j-
1][k+1]; 1++)
                                          match[j-l][k+l] = 1;
                                          jumlah_banding_kata++;
                                      break;
                                  }else{
                                      jumlah banding kata++;
                             }
                         if (0 <= (k - WordLen(words[i]))) {</pre>
                             for(l=0; words[i][l] == puzzle[j-l][k-l];
1++) {
                                 if (words[i][l+1]=='\0') {
                                      for(l=0; words[i][l] == puzzle[j-
l][k-l]; l++)
                                          match[j-1][k-1] = 1;
                                          jumlah banding kata++;
                                      break;
                                  }else{
                                      jumlah banding kata++;
                             }
                         }
```

```
if (row > (WordLen(words[i]) + k)) {
                         for(l=0; words[i][l] == puzzle[j][k+l]; l++) {
                             if (words[i][l+1]=='\0') {
                                 for(l=0; words[i][l] == puzzle[j][k+l];
1++)
                                     match[j][k+l] = 1;
                                     jumlah banding kata++;
                                 break;
                             }else{
                                 jumlah_banding_kata++;
                    if (0 <= (k - WordLen(words[i]))){</pre>
                     for(l=0; words[i][l] == puzzle[j][k-l]; l++) {
                             if (words[i][l+1]=='\0') {
                                 for(l=0; words[i][l] == puzzle[j][k-l];
1++)
                                     match[j][k-l] = 1;
                                     jumlah banding kata++;
                                 break;
                             }else{
                                 jumlah banding kata++;
                        }
                   }
               }
           }
       }
    }
    clock t end = clock();
    for (k = 0; k < row; k++) {
        for (j = 0; j < maxcol; j++) {</pre>
            if (match[j][k] == 1) {
                printf("%c ", puzzle[j][k]);
            }else {
                printf("- ");
        }
        printf("\n");
    }
    time spend = (double) (end-begin) / CLOCKS PER SEC;
    printf("waktu yang dibutuhkan untuk pencarian kata adalah:\n%f
detik\n", time spend );
    printf("jumlah kata yang dibandingkan adalah : %d",
jumlah banding kata);
    return 0;
```

## **Screen Shoot Input dan Output**

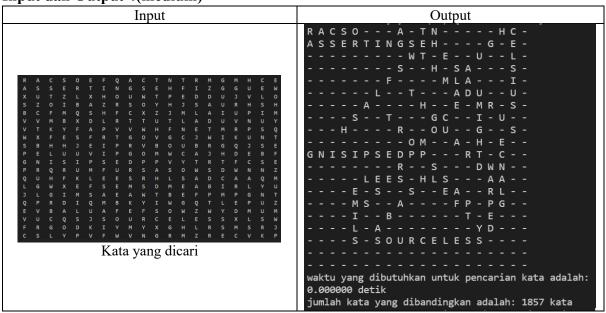
## Input dan Output 1

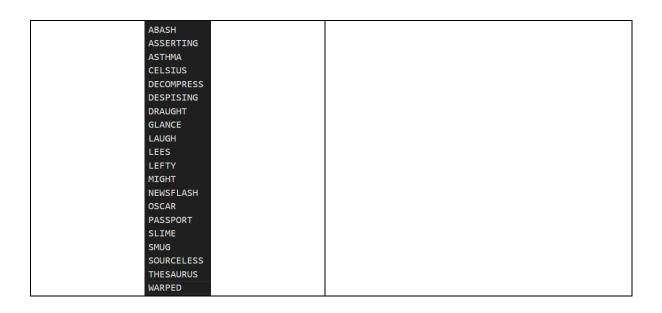
|   |   |  |   |                                      |  | Inp                                       | out  |  |  |  |  |  |                     | output  |
|---|---|--|---|--------------------------------------|--|---|--|--|--|--|--|--|---------------------|---|
| R<br>E<br>L<br>S                          | T<br>C<br>F<br>Y                          | 0<br>I<br>0<br>A                               | H<br>G<br>N<br>Q                          | I<br>C<br>G<br>T                     | E<br>K<br>U<br>O                               | J<br>D<br>D                               | B<br>S<br>E                                    | W<br>T<br>I<br>F                               | U<br>Q<br>Q<br>N                               | X<br>M<br>W                                    | О<br>Q<br>Н<br>W                               | C<br>J<br>T<br>C                               | P<br>E<br>E<br>L    | S aspect  |
| V<br>P<br>A<br>S<br>F<br>I<br>M<br>X<br>F | R<br>E<br>R<br>I<br>O<br>C<br>P<br>K<br>C | K<br>D<br>Y<br>E<br>L<br>V<br>U<br>M<br>F<br>M | K<br>Z<br>D<br>B<br>E<br>L<br>P<br>R<br>N | O<br>V<br>D<br>I<br>E<br>O<br>C<br>Z | T<br>F<br>A<br>Y<br>D<br>B<br>C<br>W<br>T<br>E | A<br>R<br>P<br>G<br>C<br>O<br>E<br>D<br>B | P<br>X<br>A<br>N<br>S<br>O<br>N<br>R<br>H<br>V | R<br>G<br>I<br>P<br>F<br>A<br>V<br>K<br>W<br>D | A<br>X<br>C<br>K<br>N<br>H<br>E<br>D<br>G<br>E | A<br>C<br>W<br>G<br>C<br>S<br>Y<br>U<br>Q<br>T | W<br>N<br>E<br>V<br>I<br>D<br>E<br>O<br>K<br>B | C<br>Y<br>N<br>Q<br>T<br>Y<br>J<br>K<br>R<br>K | S A F M F J U U C L | EP<br>C-DE<br>NEE<br>AOFW-L<br>-ROTAAS<br>PZ-RXCA   |
|   |   |  |   |                                      |  |   | ag d EEP NT VEY ACE GGE Y DY T B NCE ER RCE EO | icar   |  |  |  |  |                     | Y D D A P A I - E B N P V S - E - G I O - L H - D U P - C O N V E Y E U P - C O N V E Y E R O W E R - D - O C G E E waktu yang dibutuhkan untuk pencarian kata adalah: 0.000000 detik jumlah kata yang dibandingkan adalah: 711 kata PS E:\semester 4\strategi algoritma\tucil\Tucil 1> |

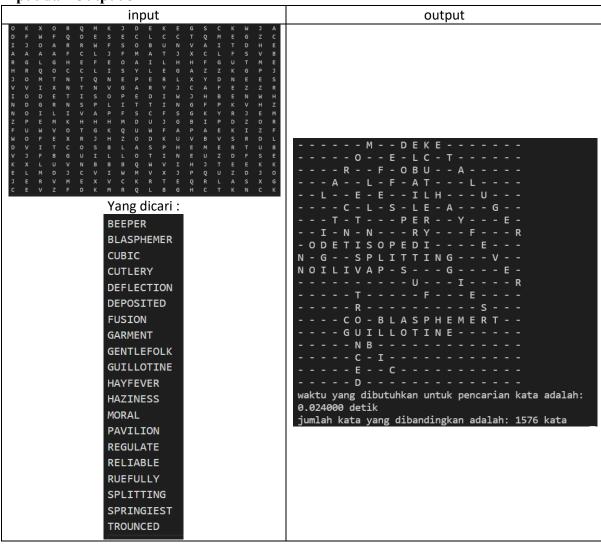


|          | uput dan Output 3   |  |  |  |  |  |  |  |  |
|----------|---|--|--|--|--|--|--|--|--|
| input    | output  |  |  |  |  |  |  |  |  |
| Input  S | Output  R E W - R E N A - E S L J - R - P S I N N E R E O N - P R M B Y - O I - E C N - A R S - I E L O P E C A L S P A N D I K - T |  |  |  |  |  |  |  |  |

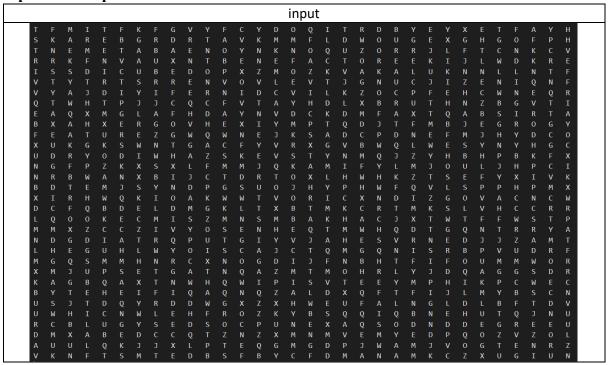
### Input dan Output 4(medium)





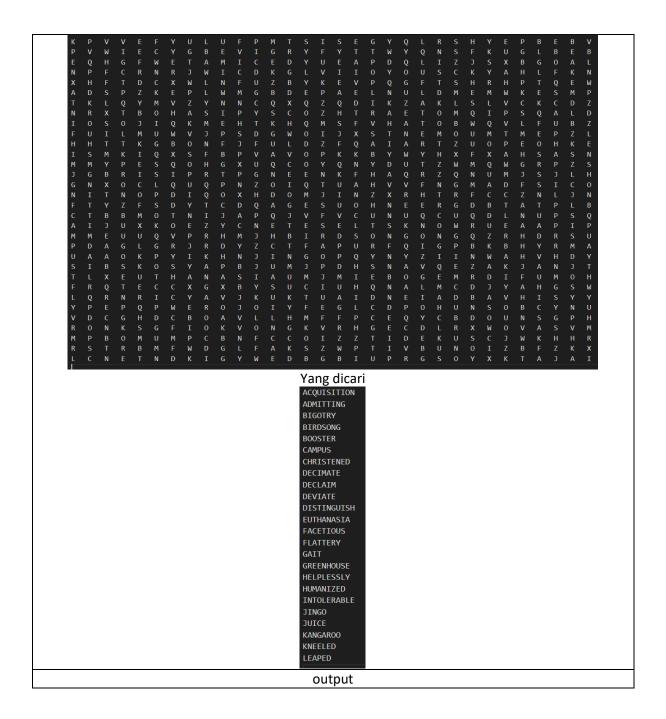


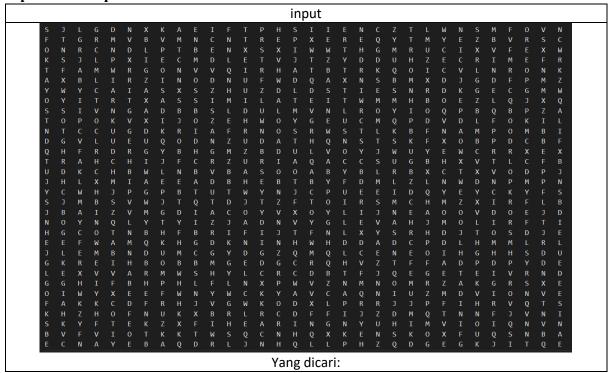
| input dan Output o  |   |  |  |  |  |  |  |  |
|---|---|--|--|--|--|--|--|--|
| input   | output  |  |  |  |  |  |  |  |
| W W U K E N H H O B K I F B B R R L S O V R B G O E L P G I Q C R R A E I S Z A P U U R V U A W S B F T A R W H E W L X A D H E R E N C E L P I S O E N R X V S Z P H A D Y I S L I D N H E T G P P U A L E E M R O U L K O T X L D N H G H C L T N O J J G H K A M Y N I R B M C U J R I U U B U A N S Z R G A T O Z Y G Q L B Y R L A N K C W I Q M D B E A P N O X F Y F Y A E J C P X D C E E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J M B Z D A T J F G M V L I O R G S F H T A C S E R P V J D A D A T J F G M V L I O R G S F H T A C S E R P V J D A D A T J F G M V L I O R G S F H T A C S E R P V J D A D A T J F G M V L I O R G S F H T A C S E R P V J D A D A T J F G M V L I O R G S F H T A C S E R P V J D A D A T J F G M V L I O R G S F H T A C S E R P V J D A D A T J F G M V L I O R G S F H T A C S E R P V J D A D A T J F G M V L I O R G S F H T A R D A T J F G M V L I O R G S F H T A R D A T J F G M V L I O R G S F H T A R D A T J F G M V L I O R G S F H T A R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M V L I O R D A T J F G M T J | B - I B - R - S L - I R R - E - S A B - T A R W - E A D H E R E N C E L - I S O E N I S - I D - H E T G P - U F - P C - S - R N E U R A - E - R O U L L T N - J J G H - A - Y N I R B I U U - U S - R R L A N I D E B I R C S E R P |  |  |  |  |  |  |  |



```
Yang dicari:
                              ABATEMENT
ADOPTED
                               AGAINST
                               AMBIDEXTROUS
                               BENEFACTOR
                              BIFOCAL
                               BUREAUCRAT
                              CANNERY
                               CITYSCAPE
                              CLERGY
                              CREAKING
                               DEGREE
                               DESPAIRED
                               DOORKNOB
                              DRAMATIZING
                              DRAT
                               EXACERBATE
                               EXCEEDING
                              EXERTING
                               FEATURE
                               FEEBLY
                               FONDLING
                               GRAVITATE
                                output
      - - - - - G - - - C - - - - - - D B Y - - - - F - - - - E - - R D - - A - - - - - - - O U G - - - - - O - - -
T N E M E T A B A E N - - - - - - - O R R - - - - - N - - - - - - V A - - N T - - - - - - - R E E - - - - D - - - - - - I - - B E - - P - - - - - - K A L - - - - L - - -
       T - - - R R - - - O - - - - - - N U C - - - - I
      - P - - T G - - - A Z - - - -
    waktu yang dibutuhkan untuk pencarian kata adalah: 0.048000 detik
jumlah kata yang dibandingkan adalah: 6350 kata
```

|  | • •   |  |
|--|-------|--|
|  | innit |  |
|  | HIDUL |  |
|  |       |  |
|  |       |  |





```
ABEYANCE
                          ADDRESS
                          ANISEED
                          ASHAMED
                          ASSIMILATE
                          BAPTIZING
                          BASIN
                          BIDED
                          CHART
                          CONDIMENT
                          CONSULT
                          CONTAINMENT
                          CRIME
                          DETERMINIST
                          DOWNSTAGE
                          EMPIRICIST
                          EXPERT
                          FRIGID
                          GERMANY
                          HEARING
                          HOOT
                          IDIOM
                          INTEGRATION
                            output
             --EI-T----
             - MN - NTREPXE -
                     ----BT
                              - A
                                    Ν -
                            - - S -
         T-ASSIMILATEIT-
 SI-N---B-S----NL-
                    - R -
                                - S
           D
                                 - N
                     - - D
                                 - 0
                                          - A O - - D -
                                    - - S - -
                                  - D A
                                                - N F - - - I
               ---HEARING--
E C N A Y E B A - - - - - - - - - - - - - - - - -
waktu yang dibutuhkan untuk pencarian kata adalah:
0.040000 detik
jumlah kata yang dibandingkan adalah: 6743 kata
```

# **Alamat Source Code Program**

Program dapat ditemukan pada link berikut. <a href="https://github.com/ilhampratama2109/word-seach-puzzle-solver">https://github.com/ilhampratama2109/word-seach-puzzle-solver</a>

| Point  | Ya                   | Tidak |
|--|----------------------|-------|
| Program berhasil dikompilasi tanpa kesalahan (no syntax error) | V                    |       |
| Program berhasil running                                       | V                    |       |
| Program dapat membaca file masukan dan                         | V, agak berbeda dari |       |
| menuliskan luaran.   | spesifikasi, di      |       |
|  | program saya file    |       |
|  | input ada 2, yaitu   |       |
|  | file puzzle dan file |       |
|  | kata yang akan       |       |
|  | dicari, saya belum   |       |
|  | bisa memadukannya    |       |
|  | kedalam satu file    |       |
|  | karena keterbasan    |       |
|  | saya                 |       |
| Program berhasil menemukan semua kata di dalam puzzle.         | V                    |       |