

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

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

Nayotama Pradipta - 13520089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

I. Pendahuluan

A. Word Search Puzzle

Word Search Puzzle merupakan permainan yang melibatkan sebuah kumpulan huruf acak dan beberapa kata yang harus dicari. Kumpulan huruf acak ini disusun pada papan/kotak/matriks yang berbentuk segi-empat atau dapat disebut juga sebagai matriks huruf. Kata-kata yang harus dicari dapat ditemukan dalam delapan arah, yaitu vertikal ke atas, vertikal ke bawah, horizontal ke kanan, horizontal ke kiri, diagonal ke kanan atas, diagonal ke kanan bawah, diagonal ke kiri atas, dan diagonal ke kiri bawah. Tujuan akhir dari permainan ini adalah menemukan seluruh kata-kata yang harus dicari pada matriks huruf.

WORD SEARCH																
C	L	A	S	S	E	S	M	O	P							ruby
H	T	I	D	M	I	Y	U	I	E							blocks
H	E	B	Z	B	E	P	T	T	L							heredocs
E	S	R	L	M	M	P	A	E	B							classes
A	A	A	E	O	D	A	B	R	I							iterator
R	F	C	H	D	C	H	L	A	X							module
R	U	O	H	U	O	K	E	T	E							objects
A	R	B	H	L	A	C	S	O	L							flexible
Y	P	P	Y	E	I	N	S	R	F							each
E	S	S	T	C	E	J	B	O	S							happy
																mutable
																lambda
																hash
																array

Gambar 1.1 Ilustrasi Word Search Puzzle

B. Algoritma Brute Force

Algoritma Brute Force merupakan algoritma pada suatu program yang memiliki karakteristik *straightforward*, sederhana, dan *naïve*. Algoritma ini dapat diimplementasikan dengan sangat mudah karena tidak ada unsur kreativitas dan tidak memandang efisiensi. Algoritma ini secara garis besar akan selalu memakan waktu lebih banyak dan boros memori. Brute Force seringkali digunakan sebagai basis pembandingan dengan algoritma lain yang lebih efisien.

II. Langkah-langkah Algoritma Brute Force

Algoritma brute force pada *word search puzzle* adalah dengan cara mencocokkan huruf pada kata yang dicari dengan huruf yang ada di matriks. Agar perbandingan dapat dilakukan, data yang ada di text file harus diubah terlebih dahulu menjadi matriks/2D array dan 1D array. Perbandingan ini dilakukan secara delapan arah jika huruf pertama pada matriks sama dengan huruf pertama kata yang akan dicari. Program akan mengecek huruf kedua pada semua arah terlebih dahulu sebelum lanjut mencari mencari huruf pertama yang sama. Jika huruf kedua kata sama dengan huruf kedua kata yang dicari, maka selanjutnya akan dicek huruf ketiga, keempat, dst. pada arah yang sama. Pencocokan dilakukan hingga salah satu dari dua kondisi berikut terjadi:

1. Huruf tidak cocok dan panjang pencarian lebih kecil daripada panjang kata
2. Huruf selalu cocok dan jumlah huruf sama dengan panjang kata yang dicari

Jika kondisi pertama terjadi, maka program akan melanjutkan pencarian pada huruf selanjutnya pada matriks (mengular). Jika kondisi kedua terjadi, maka program sukses mencari kata, letak kata yang ditemukan akan ditampilkan dan program akan melanjutkan mencari kata selanjutnya. Proses ini akan diulang hingga seluruh kata ditemukan. Program juga memiliki fitur berupa menghitung waktu eksekusi beserta jumlah perbandingan yang dilakukan.

III. Source Program

Program yang saya buat menggunakan bahasa C++ dan dibagi menjadi tiga file sesuai dengan konsep modularitas. Ketiga file ini bernama `function.hpp`, `function.cpp`, dan `main.cpp`. Berikut adalah source program:

File `function.hpp`

```
extern int xAxis[];
extern int yAxis[];

bool isValid(std::string userInput);
void getSpec(std::string userInput, int *matrixCol, int *matrixRow, int *words);
bool searchMatrix(char *matrix, std::string word, int row, int col, int matrixRow, int matrixCol, int *numOfCheck, int *x, int *y);
void findPattern(char *matrix, std::string word, int matrixRow, int matrixCol, int *numOfCheck, int *x, int *y);
void printSolution(int matrixRow, int matrixCol, int row, int col, int x, int y);
```

File function.cpp

```
#include <iostream>
#include "function.hpp"

int xAxis[] = { -1, -1, -1, 0, 0, 1, 1, 1 };
int yAxis[] = { -1, 0, 1, -1, 1, -1, 0, 1 };

bool isValid(std::string userInput){
    return (userInput == "small.txt" || userInput == "medium1.txt" ||
userInput == "medium2.txt" || userInput == "medium3.txt" || userInput ==
"medium4.txt" || userInput == "medium5.txt" || userInput == "hard1.txt" ||
userInput == "hard2.txt");
}

void getSpec(std::string userInput, int *matrixCol, int *matrixRow, int
*words){
    // Get array and matrix size depending on the user input
    if (userInput == "small.txt"){ // P.S. Small Crossword Puzzle saya ambil
contoh dari spesifikasi tucil karena tidak ada di web pada pdf file
        *matrixCol = 8;
        *matrixRow = 7;
        *words = 8;
    }
    else if (userInput == "medium1.txt" || userInput == "medium2.txt" ||
userInput == "medium3.txt" || userInput == "medium4.txt" || userInput ==
"medium5.txt"){ // Medium berukuran 22x20
        *matrixCol = 15;
        *matrixRow = 15;
        *words = 15;
    }
    else { // Large berukuran 34x32
        *matrixCol = 22;
        *matrixRow = 20;
        *words = 20;
    }
}

bool searchMatrix(char *matrix, std::string word, int row, int col, int
matrixRow, int matrixCol, int *numOfCheck, int *x, int *y){
```

```

// Core/Main function to search and find the words inside the matrix
// If first letter is not equal -> break
if (*(matrix+row*matrixCol+col) != word[0]){
    return false;
}
int path;
for (path = 0; path < 8; path++){
    int a;
    int ra = row + xAxis[path];
    int ca = col + yAxis[path];
    for (a = 1; a < word.length(); a++){
        if (ra >= matrixRow || ca >= matrixCol || ra < 0 || ca < 0){
            *numOfCheck += 1;
            break;
        }
        if (*(matrix+ra*matrixCol+ca) != word[a]){
            *numOfCheck += 1;
            break;
        }
        ra += xAxis[path];
        ca += yAxis[path];
    }
    if (a == word.length()){
        *x = xAxis[path];
        *y = yAxis[path];
        return true;
    }
}
return false;
}

void printSolution(std::string word, int matrixRow, int matrixCol, int row,
int col, int x, int y){
    // This function is used to display the solution in a matrix form (Assume
that the word has already been found)
    char matrixSolution[matrixRow][matrixCol];
    for (int i = 0; i < matrixRow; i++){
        for (int j = 0; j < matrixCol; j++){
            matrixSolution[i][j] = '-';

```

```

    }
}
matrixSolution[row][col] = word[0];
for (int i = 1; i < word.length(); i++){
    matrixSolution[row+x*i][col+y*i] = word[i];
}
for (int i = 0; i < matrixRow; i++){
    for (int j = 0; j < matrixCol; j++){
        std::cout << matrixSolution[i][j] << " ";
    }
    std::cout << "\n";
}
std::cout << "\n";
}

void findPattern(char *matrix, std::string word, int matrixRow, int matrixCol,
int *numOfCheck, int *x, int *y){
    for (int row = 0; row < matrixRow; row++){
        for (int col = 0; col < matrixCol; col++){
            if (searchMatrix(matrix, word, row, col, matrixRow, matrixCol,
numOfCheck, x, y)){
                printSolution(word, matrixRow, matrixCol, row, col, *x, *y);
            }
            *numOfCheck += 1;
        }
        *numOfCheck += 1;
    }
}
}

```

File main.cpp:

```
#include <iostream>
#include "function.hpp"
#include <bits/stdc++.h>

using namespace std;
int main(){
    clock_t start, end;
    string userInput;
    string output;
    char alfabet = 0;
    bool allfound = false;
    int matrixCol = 0;
    int matrixRow = 0;
    int line = 0;
    int i = 0;
    int j = 0;
    int k = -1;
    int words;
    int numOfCheck = 0;
    int checktest = 0;
    int x = 0;
    int y = 0;
    cout << "Welcome to Word Search!\n";
    cout << "Please enter an input file: ";
    cin >> userInput;

    // Invalid input
    while (!isValid(userInput)){
        cout << "Invalid file name! Please input the correct file: ";
        cin >> userInput;
    }

    getSpec(userInput, &matrixCol, &matrixRow, &words);
    // Create 2D and 1D Array to store text file into array
    char crossWordMatrix[matrixRow][matrixCol];
    string wordsToSearch[words];
    // Read File
```

```

FILE* inputFile = fopen(userinput.c_str(), "r");
// Read crossword and store inside matrix/2D array
while (line != matrixRow){
    alfabet = getc(inputFile);
    if (alfabet != '\n'){
        crossWordMatrix[i][j] = alfabet;
    }
    if (alfabet == '\n'){
        line++;
        i++;
        j=0;
    } else {
        if (alfabet != ' '){
            j++;
        }
    }
}
// Read words and store inside 1D array
while (!feof(inputFile)){
    alfabet = getc(inputFile);
    if (alfabet != '\n'){
        wordsToSearch[k] += alfabet;
        if (alfabet == ' '){
            wordsToSearch[k].pop_back();
        }
    } else {
        k++;
    }
}
// Erase last character of the last word
wordsToSearch[k].pop_back();
// Close file
fclose(inputFile);
// Algoritma Brute Force
cout << "CrossWord Puzzle solution for " << userinput << ": \n\n";
start = clock();
for (int m = 0; m < words; m++){

```

```
        findPattern((char *)crossWordMatrix, wordsToSearch[m], matrixRow,
matrixCol, &numOfCheck, &x, &y);
    }
    // Testing Display Array of String
    end = clock();
    double time_taken = double(end - start)/ double(CLOCKS_PER_SEC);
    cout << "Number of Checks: " << numOfCheck << "\n";
    cout << "Execution time: " << fixed << time_taken << setprecision(5)
<< " second(s)";
    cout << "\n";
}
```


IV. Screenshot Input & Output

1. Input & Output Small/Easy (Diambil dari contoh spesifikasi Tugas Kecil):

```
Tucil 1 > ≡ small.txt
 1 J S O L U T I S
 2 S U N A R U U A
 3 N E P T U N E T
 4 S O N I E I S U
 5 R C E V T R E R
 6 A H T R A E S N
 7 M M E R C U R Y
 8
 9 EARTH
10 JUPITER
11 MARS
12 MERCURY
13 NEPTUNE
14 SATURN
15 URANUS
16 VENUS

Welcome to Word Search!
Please enter an input file: small.txt
CrossWord Puzzle solution for small.txt:

- - - - -
- - - - -
- - - - -
- - - - -
- H T R A E -
- - - - -

J - - - - -
- U - - - - -
- - P - - - -
- - - I - - -
- - - - T - -
- - - - - E -
- - - - - R -

- - - - -
- - - - -
- - - - -
S - - - - -
R - - - - -
A - - - - -
M - - - - -

- - - - -
- - - - -
- - - - -
- - - - -
- M E R C U R Y

- - - - -
- - - - -
- - - - -
N E P T U N E -
- - - - -
- - - - -
- - - - -

- - - - - S
- - - - - A
- - - - - T
- - - - - U
- - - - - R
- - - - - N
- - - - -

- - - - -
S U N A R U -
- - - - -
- - - - -
- - - - -
- - - - -

- - - - - S
- - - - - U
- - - - - N
- - - - - E
- - - V - - -
- - - - -
- - - - -

Number of Checks: 718
Execution time: 0.056000 second(s)
```

2. Input & Output Medium 1

```
Tucil 1 > ≡ medium1.txt
1 R O D G Z D M M Z S M X T Y M
2 K L S T B H I V A K Z L M S H
3 L C I C R C Y I E G Z A P C U
4 G V O Z H E R R R Z N L I A R
5 A X O I Y O I E L I H D O S O
6 Q L G B T E A N T N A S I P N
7 T A B C K T L O D G U U R I A
8 N V I E S A B Y R E Q P A A C
9 C V F A R A F E V R E E T N A
10 Z B L X C T A M J Z H R N S C
11 X T P G F T D L D C A I O E I
12 A C S A B A H T A K M O C A T
13 M G A E S G I G V Z A R Q R I
14 V K A U Y L F C M Y E R Z F T
15 W R J W I N N I P E G B H J F
16
17 ALBERT
18 ATHABASCA
19 CASPIAN SEA
20 ERIE
21 GREAT BEAR
22 GREAT SALT
23 HURON
24 MANITOBA
25 MICHIGAN
26 ONTARIO
27 REINDEER
28 SUPERIOR
29 TITICACA
30 VICTORIA
31 WINNIPEG
```

```
Welcome to Word Search!  
Please enter an input file: medium1.txt  
CrossWord Puzzle solution for medium1.txt:
```



```
Number of Checks: 4940
Execution time: 0.388000 second(s)
```

3. Input & Output Medium 2

[illegible]

Y
L
S
A
E
M

ERUTAINIM

MINSULE

MINUTE

NANOSCOPIC

ETITEP

DEZISTNIP

YN

EET

YNIT

Number of Checks: 5072
Execution time: 0.421000 second(s)

4. Input & Output Medium 3

The screenshot displays a Word Search application interface. On the left, a list of 31 words is shown, with the first 15 words highlighted in blue. The words are: 1. IHSUMAMBJWWBTXR, 2. BTVMVMXKGAOKJU, 3. ARBOCEPACNDESCS, 4. SHHQITGADALBPSS, 5. LLTTMOREEMPNLHE, 6. LRSUEADHEPQWADL, 7. CGLTRKROVCPDTHL, 8. FGNAREPIVORUEVS, 9. AZJAPDEATHADDER, 10. WRIPOJ KINGCOBRA, 11. GTOQRATTTLESNAKE, 12. BCDGHTUOMNOTTOC, 13. REPIVDENROHDQZL, 14. GUDUQABMAMKCALB, 15. SDWRFEKANSGREGIT, 16. BANDED KRAIT, 17. BLACK MAMBA, 18. CAPE COBRA, 19. COPPERHEAD, 20. COTTONMOUTH, 21. DEATH ADDER, 22. EURO VIPER, 23. HORNED VIPER, 24. JARARACA, 25. KING COBRA, 26. MAMUSHI, 27. MULGA, 28. RATTLESNAKE, 29. RUSSELLS, 30. TIGER SNAKE.

In the center, a crossword puzzle grid is displayed, with the words from the list placed within the grid. The words are: 1. IHSUMAMBJWWBTXR, 2. BTVMVMXKGAOKJU, 3. ARBOCEPACNDESCS, 4. SHHQITGADALBPSS, 5. LLTTMOREEMPNLHE, 6. LRSUEADHEPQWADL, 7. CGLTRKROVCPDTHL, 8. FGNAREPIVORUEVS, 9. AZJAPDEATHADDER, 10. WRIPOJ KINGCOBRA, 11. GTOQRATTTLESNAKE, 12. BCDGHTUOMNOTTOC, 13. REPIVDENROHDQZL, 14. GUDUQABMAMKCALB, 15. SDWRFEKANSGREGIT, 16. BANDED KRAIT, 17. BLACK MAMBA, 18. CAPE COBRA, 19. COPPERHEAD, 20. COTTONMOUTH, 21. DEATH ADDER, 22. EURO VIPER, 23. HORNED VIPER, 24. JARARACA, 25. KING COBRA, 26. MAMUSHI, 27. MULGA, 28. RATTLESNAKE, 29. RUSSELLS, 30. TIGER SNAKE.

On the right, a word search grid is displayed, with the words from the list placed within the grid. The words are: 1. IHSUMAMBJWWBTXR, 2. BTVMVMXKGAOKJU, 3. ARBOCEPACNDESCS, 4. SHHQITGADALBPSS, 5. LLTTMOREEMPNLHE, 6. LRSUEADHEPQWADL, 7. CGLTRKROVCPDTHL, 8. FGNAREPIVORUEVS, 9. AZJAPDEATHADDER, 10. WRIPOJ KINGCOBRA, 11. GTOQRATTTLESNAKE, 12. BCDGHTUOMNOTTOC, 13. REPIVDENROHDQZL, 14. GUDUQABMAMKCALB, 15. SDWRFEKANSGREGIT, 16. BANDED KRAIT, 17. BLACK MAMBA, 18. CAPE COBRA, 19. COPPERHEAD, 20. COTTONMOUTH, 21. DEATH ADDER, 22. EURO VIPER, 23. HORNED VIPER, 24. JARARACA, 25. KING COBRA, 26. MAMUSHI, 27. MULGA, 28. RATTLESNAKE, 29. RUSSELLS, 30. TIGER SNAKE.

5. Input & Output Medium 4

The screenshot displays a Word Search application interface. On the left, a list of 31 Greek mythological figures is provided: 1. ICJTTOQB DIONYSUS, 2. SLYWULBORPUTVYC, 3. YGOISSETIDORHPA, 4. WZYWIQUPOSEIDON, 5. IZHFRSSEPAPOLLO, 6. UNDEIEUXHXVOUSO, 7. FHISRUEQEDTTDP EE, 8. LVCARAPEZQEJ KCV, 9. NATSUEHPROMMHTR, 10. UTVPECRZHWCAOER, 11. BLWSTFIWEEOEWRK, 12. JAWOCGUCWSRNTHP, 13. LSURIKETRIUMTWA, 14. TIGEOUSQKKYE EIK, 15. ISXDVHADESTJ KSH, 16. (empty), 17. APHRODITE, 18. APOLLO, 19. ARES, 20. ATLAS, 21. CHAOS, 22. DIONYSUS, 23. EROS, 24. HADES, 25. HERA, 26. HERMES, 27. IRIS, 28. MORPHEUS, 29. POSEIDON, 30. PROMETHEUS, 31. ZEUS. The list is presented in a monospaced font with some letters in red. To the right of the list is a large, empty crossword puzzle grid. At the top right, a text box contains the message: 'Welcome to Word Search! Please enter an input file: medium4.txt Crossword Puzzle solution for medium4.txt:'. Below this text box is another large, empty crossword puzzle grid. The interface is dark-themed with a black background and white text.

S
O
R
E

HADES

H
E
R
A

S
O
R
E

HADES

H
E
R
A

S
O
R
E

HADES

H
E
R
A

HERMES
SIRI

SUEHPROM

HERMES
SIRI

SUEHPROM

HERMES
SIRI
SUEHPROM

POSEIDON

SEUHEZ

POSEIDON

SEUHEZ

POSEIDON

SEUHEZ

POSEIDON

SEUHEZ

6. Input & Output Medium 5

[illegible]

N O M E L

K
L
I
M

P
I
G
E
Y
E

SILKY

S
L
I
T
E
Y
E

T
I
G
E
R

```
- - - - -  
- - - - -  
- - - - -  
- - - - - W  
- - - - - E  
- - - - - A  
- - - - - S  
- - - - - E  
- - - - - L  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
  
- - - - -  
- - - - -  
- E - - - - -  
- T - - - - -  
- I - - - - -  
- H - - - - -  
- W - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
A - - - - -  
- R - - - - -  
- B - - - - -  
- E - - - - -  
- Z - - - - -  
- - - - -  
- - - - -
```

Number of Checks: 4660
Execution time: 0.401000 second(s)

7. Input & Output Hard 1

```
Tucil 1 >  hard1.txt
1  O H E C W W T E I C B Q V O L K H M N D I T
2  A S S E R T I N G I V Y W S H Y D B U N P Y
3  G B W R I C N F H S X O R C M J M U N J R N
4  P L K A S C Q S S E L E C R U O S W M T M J
5  U O A J Y R Q E E J G N I S I P S E D D T Y
6  P A I N A B R Q D W W P S W A D S Q F B E Q
7  I I J C C P R G U U N P U X D Q K G D W B L
8  O H S O M E F U I C V D H N E W S F L A S H
9  U O X O U T P M J W F W F G O Z B C N L C G
10 X G C D D Y T S C B B I X K L Y V Y N V S K
11 G E A N Y K R G T P G R Y L X A Y P T U O D
12 D F J A A M O K H C S R W B V A R R U F N R
13 D R S U M I P A G W Q U S C O Y B B H D E Z
14 Q H X J H G S S U F E W R G N X S A C X L L
15 C U E Z T H S U A H W F A U L A L T S C K I
16 Z B G E S T A G R G D Q X R A Y I F U H Q R
17 X L X R A D P Q D U O K Y O P S M P N G K J
18 A U F B K P B F E A Q L P S L E E R K V P F
19 Y W K Z E X O L Q L S B H G K X D H S S X K
20 K X I K O S U I S L E C J T L E E S T T T T
21
22 ABASH
23 ASSERTING
24 ASTHMA
25 CELSIUS
26 DECOMPRESS
27 DESPISING
28 DRAUGHT
29 GLANCE
30 LAUGH
31 LEES
32 LEFTY
33 MIGHT
34 NEWSFLASH
35 OSCAR
36 PASSPORT
37 SLIME
38 SMUG
39 SOURCELESS
40 THESAURUS
41 WARPED
```

AMH-TS-A

AMH-TSA

SEMPER PARATI
D

SEMPER PARATI
D

T
H
G
U
A
R
D

H
G
U
A
L

G

L

A

N

C

E

LEES

Y
T
F
E
L

NEWSFLASH

R
A
C
S
O

M
I
G
H
T

T
R
O
P
S
S
A
P

G
U
M
S

SSELECRUOS

S
L
I
M
E

```
Number of Checks: 12415
Execution time: 0.937000 second(s)
```

8. Input & Output Hard 2

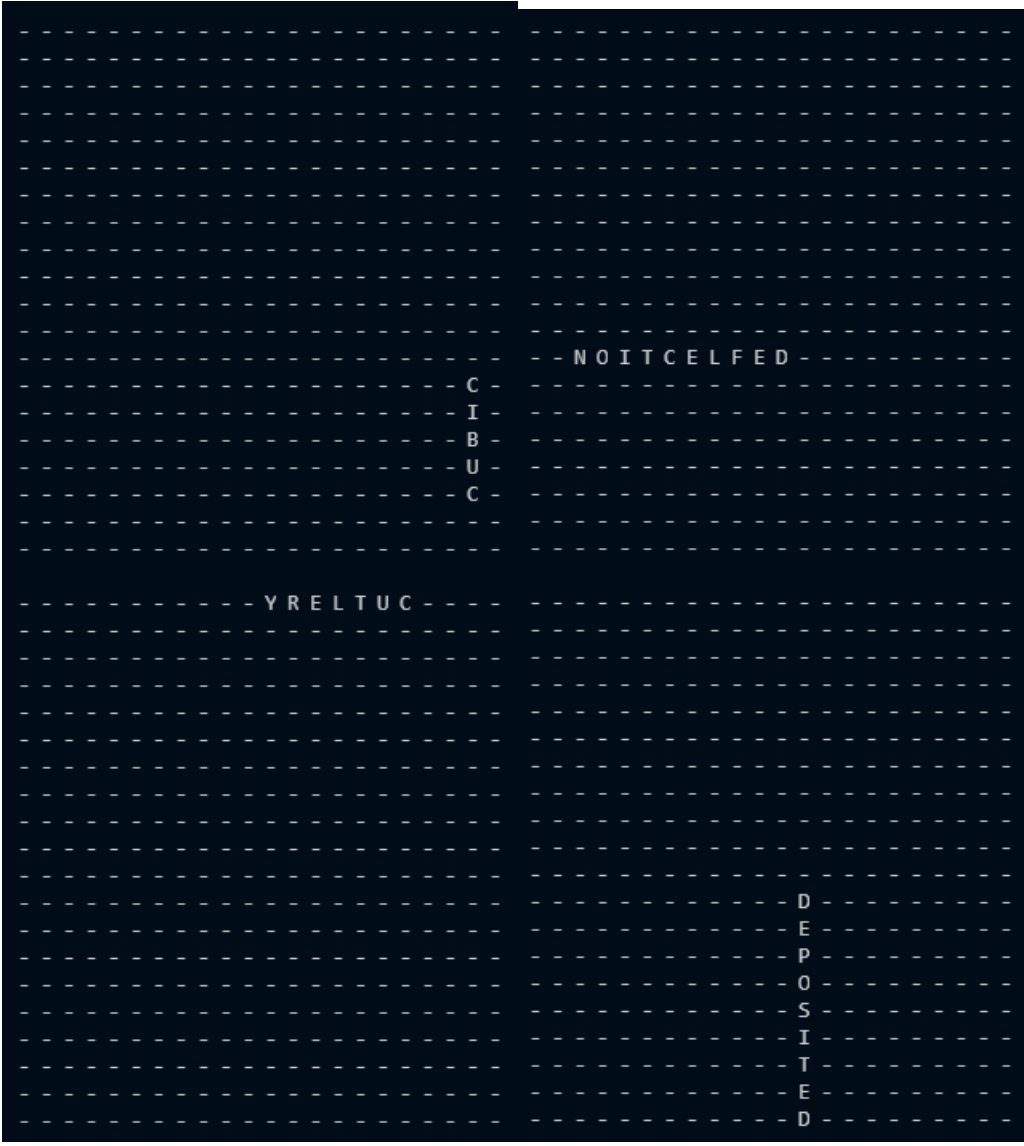
Tucil 1 > ≡ hard2.txt

21	
22	BEEPER
23	BLASPHEMER
24	CUBIC
25	CUTLERY
26	DEFLECTION
27	DEPOSITED
28	FUSION
29	GARMENT
30	GENTLEFOLK
31	GUILLOTINE
32	HAYFEVER
33	HAZINESS
34	MORAL
35	PAVILION
36	REGULATE
37	RELIABLE
38	RUEFULLY
39	SPLITTING
40	SPRINGIEST
41	TROUNCED

Welcome to Word Search!

```
Please enter an input file: hard2.txt
```

CrossWord Puzzle solution for hard2.txt:





- H A Y F E V E R -

L

A

R

O

M

- N O I L I V A P -

- H A Z I N E S S -



- - - S P R I N G I E S T - - - - -

- - - - D E C N U O R T - - - - -

```
Number of Checks: 11744
Execution time: 0.937000 second(s)
```


V. Checklist Fitur 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	√	

VI. Link Drive Kode Program

<https://drive.google.com/drive/folders/1miRwRQNcH1bkfWazhA18YJcyrsQjmdU0?usp=sharing>

*** File yang terdapat di drive diatas sudah saya bagi menjadi empat folder demi kerapihan, akan tetapi untuk mencoba/testing dengan .txt file, file yang akan di test harus dipindahkan terlebih dahulu ke directory yang sama dengan directory executable. ***