LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA WORD SEARCH PUZZLE



PUTRI NURHALIZA 13520066

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER II 2021/202

Daftar Isi

Daftar Isi	i
A. Deskripsi Algoritma Brute Force	1
B. Source Code Program (Bahasa Pemograman C++)	
1. Parsing File ke Class	
2. Puzzle	4
3. Words	7
4. Search	8
5. Main	14
C. Input dan Output Hasil Eksekusi	16
D. Repository Program (Github)	31
E. Rangkuman Keberjalanan Program	31

A. Deskripsi Algoritma Brute Force

Penyelesaian Word Search Puzzle diimplementasi kan menggunakan algoritma brute force dengan langkah-langkah sebagai berikut.

- 1. Telusuri setiap elemen matriks secara traversal dengan iterasi baris dan kolom untuk mengetahui huruf yang ada pada posisi tersebut. Misal, matriks memiliki m baris dan n kolom. Kita dapat menempatkan posisi mulai dari matriks[0][0] dan berpindah ke posisi selanjutnya hingga matriks[m-1][n-1] jika kata belum ditemukan.
- 2. Cek apakah huruf pada posisi sekarang sama dengan huruf pertama pada kata yang akan dicari.
- 3. Jika sama, cek huruf kedua pada posisi setelahnya berdasarkan salah satu arah mata angin. Misal pada arah horizontal ke kanan, akan dicek apakah huruf kedua pada kata sama dengan huruf yang ada dikanan posisi sekarang.
 - Jika masih sama, maka teruskan penelusuran pada arah tersebut hingga huruf terakhir pada kata.
 - Jika ditemukan ketidaksesuaian, maka penelusuran pada arah tersebut dihentikan.

Lalu lakukan hal yang sama pada arah yang lain (horizontal ke kiri, vertikal ke atas, vertikal ke bawah, diagonal ke kanan atas, diagonal ke kanan bawah, diagonal ke kiri atas, dan diagonal ke kiri bawah) untuk mengecek kata dari awal. Jika kedelapan arah mata angin sudah ditelusuri namun kata belum ditemukan, maka ulangi dari poin 2 pada posisi selanjutnya di matriks.

4. Jika huruf pada posisi sekarang di matriks tidak sesuai dengan huruf pertama pada kata yang akan dicari, maka pindah ke posisi selanjutnya di matriks dan ulangi poin 2. Memungkinkan mencapai indeks terakhir pada matriks.

Singkatnya,

- Berhenti jika terdapat ketidaksesuaiian dalam penelusuran alur tersebut. Lanjutkan pengecekan dari awal kata pada alur penelusuran yang lain.
- Berhenti mencari begitu kata sudah berhasil ditemukan.

Kita dapat memperkirakan apakah penelusuran akan *out of bounds* dengan memperhitungkan panjang kata dan posisi huruf pertama pada matriks untuk melihat apakah akan cukup atau tidak pada arah tersebut. Jika tidak, kita tidak perlu melakukan penelusuran hingga *out of bounds*.

Berikut adalah ilustrasinya, misal kita ingin mencari kata "stima".

Cari huruf pertama yang sesuai

```
dasrcvr dasrcvr
ydmliaf ydmliaf
qwdiaxb qwdiaxb
fghdtxm fghdtxm
dhgtqsn dhgtqsn
```

"d" tidak sesuai, geser kotak biru ke "a" namun juga tidak sesuai, lalu geser ke "s".

```
dasrcvr dasrcvr
ydmliaf ydmliaf
qwdiaxb qwdiaxb
fghdtxm fghdtxm
dhgtqsn dhgtqsn
```

Ditemukan posisi huruf pertama yang sesuai. Lalu cek ke 8 arah yang mungkin untuk huruf kedua yaitu "t", namun tidak ada yang sesuai. Pindah kotak biru ke posisi berikutnya, hingga akhirnya mencapai "s" lagi.

```
dasrcvr
ydmliaf
qwdiaxb
fghdtxm
dhgtqSn
```

Ditemukan "s", lalu cek ke 8 arah yang mungkin dengan kotak hijau.

```
dasrcvrdasrcvr
ydmliafydmliaf
qwdiaxbqwdiaxb
fghdtxmfghdtxm
dhgtQSndhgtqSn
```

Pada arah horizontal ke kiri dan vertikal ke atas ditemukan huruf yang tidak sesuai dengan huruf kedua, pada 5 arah lain *out of bounds*. Sementara pada arah diagonal ke kiri atas,

```
dasrcvr
          dasrcvr
                    dasrcvr
                             dasrcvr
ydmliaf
          ydmliaf
                    ydmliaf ydmliaf
qwdiaxb
                    qwdiaxb qwdiaxb
        q w d <u>ii</u> a x b
fghdtxm
                    fghdtxm fghdtxm
          fghdtxm
dhgtqsn
          dhgtqSn
                    dhgtqSn
                            dhgtqSn
```

Ditemukan kesesuaian mulai dari huruf kedua hingga akhir. Sehingga kata ditemukan di indeks matriks[4][5] dengan arah diagonal ke kiri atas.

B. Source Code Program (Bahasa Pemograman C++)

Source code program dibuat secara modular, terdiri atas empat kelompok (header dan driver) serta satu file main.

1. Parsing File ke Class

Berfungsi untuk mem-*parsing* inputan dari file (.txt) dan menyimpannya ke struktur data *Puzzle* dan *Words*.

fileToData.h

```
#ifndef __FILETODATA_H
#define __FILETODATA_H

#include "puzzle.h"
#include "words.h"

void fileToData(string filename, Puzzle *p, Words *w);
#endif
```

fileToData.cpp

```
#include <iostream>
#include <stdlib.h>
#include <string>
#include <bits/stdc++.h>
#include "fileToData.h"
using namespace std;
void fileToData(string filename, Puzzle *p, Words *w)
    ifstream file(filename.c_str());
    string line;
    char letter;
    bool flag = false;
    int rows = 0, cols = 0, nWord=0;
    int i, j=0;
    // parsing puzzle baris pertama untuk menemukan banyak kolom
    getline(file, line);
    istringstream strline(line);
    while (strline>>letter)
        p->setElmt(rows, cols, letter);
        cols++;
    rows++;
```

```
// parsing puzzle baris kedua sampai akhir puzzle
while (!flag) {
   getline(file, line);
   istringstream strline(line);
   rows++;
   while (strline>>letter)
       if (j >= cols) j = 0;
       i = rows-1;
       p->setElmt(i, j, letter);
   if (file.peek() == '\n') flag = true;  // parse puzzle done
// untuk ngeskip blank line pemisah puzzle dan words
getline(file, line);
// parsing words yang mau dicari
while (getline(file, line)) {
   w->setWord(nWord, line);
   nWord++;
p->cols = cols;
p->rows = rows;
w->total = nWord;
```

2. Puzzle

Berisi atribut serta *method* yang diperlukan pada *puzzle* (matriks)

puzzle.h

```
Puzzle();
Puzzle(int x, int y);

//setter isi Puzzle
void setElmt(int x, int y, char val);
void setDefault();
void setResult(int frow, int fcol, string found, string w);

//getter
char getElmt(int x, int y);

//display
void display();
};

#endif
```

puzzle.cpp

```
#include <iostream>
#include <stdlib.h>
#include <string>
#include <bits/stdc++.h>
#include "puzzle.h"
using namespace std;
Puzzle::Puzzle()
    rows = 0;
    cols = 0;
Puzzle::Puzzle(int x, int y)
    rows = x;
    cols = y;
    matrix = new char *[rows];
    for (int i=0; i<rows; i++)</pre>
        matrix[i] = new char[cols];
void Puzzle::setElmt(int x, int y, char val)
    matrix[x][y] = val;
void Puzzle::setDefault()
```

```
for (int i=0; i<rows; i++)</pre>
        for (int j=0; j<cols; j++)</pre>
        {
             setElmt(i,j,'-');
char Puzzle::getElmt(int x, int y)
    return matrix[x][y];
void Puzzle::display()
    for (int i=0; i<rows; i++)</pre>
        for (int j=0; j<cols; j++) cout << matrix[i][j] << " ";</pre>
        cout << endl;</pre>
// SET BERDASAR HASIL PENCARIAN
void Puzzle::setResult(int frow, int fcol, string found, string w)
    setDefault();
    if (found == "LR") for (int j=fcol; j<fcol+w.length(); j++)</pre>
setElmt(frow,j,w[j-fcol]);
    else if (found == "RL") for (int j=fcol-w.length()+1; j<fcol+1; j++)</pre>
setElmt(frow,j,w[fcol-j]);
    else if (found == "UD") for (int i=frow; i<frow+w.length(); i++)</pre>
setElmt(i,fcol,w[i-frow]);
    else if (found == "DU") for (int i=frow-w.length()+1; i<frow+1; i++)</pre>
setElmt(i,fcol,w[frow-i]);
    else if (found == "ULDR") for (int i=frow; i<frow+w.length(); i++)</pre>
setElmt(i,fcol-frow+i,w[i-frow]);
    else if (found == "DRUL") for (int j=fcol-w.length()+1; j<fcol+1; j++)</pre>
setElmt(frow-fcol+j,j,w[fcol-j]);
```

```
else if (found == "URDL") for (int i=frow; i<frow+w.length(); i++)
setElmt(i,fcol+frow-i, w[i-frow]);

else if (found == "DLUR") for (int j=fcol; j<fcol+w.length(); j++)
setElmt(frow+fcol-j,j,w[j-fcol]);

else cout << "\nKata tidak ditemukan pada puzzle";
}</pre>
```

3. Words

Berisi atribut dan *method* yang diperlukan pada *array* kata yang akan dicari.

words.h

```
#ifndef __WORD_H
#define __WORD_H
#include <string>
using namespace std;
class Words{
    public:
        int total;
        string * words;
        //constructor
        Words();
        Words(int new_total);
        //setter
        void setWord(int i, string newWord);
        //getter
        string getWord(int i);
        //display
        void display();
};
#endif
```

words.cpp

```
#include <iostream>
#include <stdlib.h>
#include <string>
#include "words.h"
using namespace std;
```

```
Words::Words()
{
    total = 0;
}

Words::Words(int new_total)
{
    total = new_total;
    words = new string[total];
}

void Words::setWord(int i, string newWord)
{
    words[i] = newWord;
}

string Words::getWord(int i)
{
    return words[i];
}

void Words::display()
{
    for (int i=0; i<total; i++) cout << words[i] << endl;
}</pre>
```

4. Search

Berisi atribut dan *method* yang diperlukan untuk algoritma *brute force* pencarian kata pada puzzle.

search.h

```
// jumlah total perbandingan huruf untuk pencarian 1
        int count;
kata pada puzzle
        //constructor
        Search();
        void find(Puzzle *p, string w);
        void horizontal_LR(Puzzle *p, string w, int cRow, int cCol);
        void horizontal_RL(Puzzle *p, string w, int cRow, int cCol);
        void vertical_UD(Puzzle *p, string w, int cRow, int cCol);
        void vertical_DU(Puzzle *p, string w, int cRow, int cCol);
        void diagonal_ULDR(Puzzle *p, string w, int cRow, int cCol);
        void diagonal_DRUL(Puzzle *p, string w, int cRow, int cCol);
        void diagonal_URDL(Puzzle *p, string w, int cRow, int cCol);
        void diagonal_DLUR(Puzzle *p, string w, int cRow, int cCol);
};
#endif
```

search.cpp

```
#include <iostream>
#include <stdlib.h>
#include <string>
#include "puzzle.h"
#include "words.h"
#include "search.h"
#include<bits/stdc++.h>
using namespace std;
//constructor
Search::Search()
    found = "none";
    count = 0;
// ALGORITMA BRUTEFORCE TRAVERSAL MATRIX
// Untuk pencarian huruf pertama yang sesuai
void Search::find(Puzzle *p, string w)
    int i=0;
    while (found == "none" && i<p->rows)
        while (found == "none" && j<p->cols)
```

```
count++;
            if (p->getElmt(i,j) == w[0])
                horizontal_LR(p, w, i, j);
                horizontal_RL(p, w, i, j);
                vertical_UD(p, w, i, j);
                vertical_DU(p, w, i, j);
                diagonal_ULDR(p, w, i, j);
                diagonal_DRUL(p, w, i, j);
                diagonal_URDL(p, w, i, j);
                diagonal_DLUR(p, w, i, j);
            j++;
        i++;
    frow = i-1;
    fcol = j-1;
// Jika ditemukan huruf pertama yang sesuai, bruteforce akan dilanjutkan ke arah
berikut
// Pencarian pada arah horizontal ke kanan
void Search::horizontal_LR(Puzzle *p, string w, int cRow, int cCol)
    // batasan tidak out of bounds
    if (cCol < p->cols - w.length() + 1)
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
            count++;
            if (p->getElmt(cRow, cCol+k) == w[k])
                if (k==w.length()-1) found = "LR";
            else foundtmp = false;
            k++;
// Pencarian pada arah horizontal ke kiri
void Search::horizontal_RL(Puzzle *p, string w, int cRow, int cCol)
    // batasan tidak out of bounds
```

```
if (cCol >= w.length() - 1)
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
            count++;
            if (p->getElmt(cRow, cCol-k) == w[k])
                if (k==w.length()-1) found = "RL";
            else foundtmp = false;
            k++;
// Pencarian pada arah vertikal ke bawah
void Search::vertical_UD(Puzzle *p, string w, int cRow, int cCol)
    if (cRow < p->rows - w.length() + 1)
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
            count++;
            if (p->getElmt(cRow+k, cCol) == w[k])
                if (k==w.length()-1) found = "UD";
            else foundtmp = false;
            k++;
// Pencarian pada arah vertikal ke atas
void Search::vertical_DU(Puzzle *p, string w, int cRow, int cCol)
    // batasan tidak out of bounds
    if (cRow >= w.length() - 1)
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
```

```
count++;
            if (p->getElmt(cRow-k, cCol) == w[k])
                if (k==w.length()-1) found = "DU";
            else foundtmp = false;
            k++;
// Pencarian pada arah diagonal ke kanan bawah
void Search::diagonal_ULDR(Puzzle *p, string w, int cRow, int cCol)
    // batasan tidak out of bounds
    if ((cCol < p->cols - w.length() + 1) && (cRow < p->rows - w.length() + 1))
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
            if (p->getElmt(cRow+k, cCol+k) == w[k])
                count++;
                if (k==w.length()-1) found = "ULDR";
            else foundtmp = false;
            k++;
// Pencarian pada arah diagonal ke kiri atas
void Search::diagonal_DRUL(Puzzle *p, string w, int cRow, int cCol)
    // batasan tidak out of bounds
    if ((cRow >= w.length() - 1) && (cCol >= w.length() - 1))
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
            count++;
            if (p->getElmt(cRow-k, cCol-k) == w[k])
                if (k==w.length()-1) found = "DRUL";
            else foundtmp = false;
```

```
k++;
// Pencarian pada arah diagonal ke kiri bawah
void Search::diagonal_URDL(Puzzle *p, string w, int cRow, int cCol)
    // batasan tidak out of bounds
    if ((cRow < p->rows - w.length() + 1) && (cCol >= w.length() - 1))
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
            count++;
            if (p->getElmt(cRow+k, cCol-k) == w[k])
                if (k==w.length()-1) found = "URDL";
            else foundtmp = false;
            k++;
// Pencarian pada arah diagonal ke kanan atas
void Search::diagonal_DLUR(Puzzle *p, string w, int cRow, int cCol)
    // batasan tidak out of bounds
    if ((cRow >= w.length() - 1) && (cCol < p->cols - w.length() + 1))
        int k=1;
        bool foundtmp = true;
        while (foundtmp && k<w.length())</pre>
            count++;
            if (p->getElmt(cRow-k, cCol+k) == w[k])
                if (k==w.length()-1) found = "DLUR";
            else foundtmp = false;
            k++;
    }
```

5. Main

File utama pada program yang berfungsi untuk:

- Menerima inputan nama file
- Menggunakan *method* fileToData
- Melakukan pencarian pada setiap *keywords* di *puzzle* menggunakan algoritma *brute force*
- Menghitung waktu eksekusi algoritma brute force
- Menghitung total perbandingan huruf
- Menampilkan output

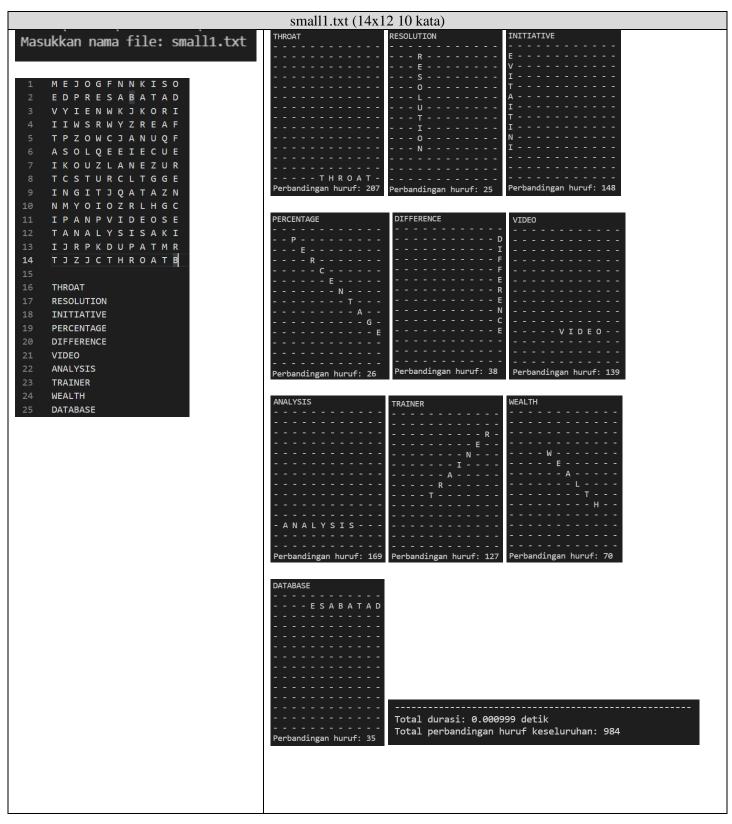
main.cpp

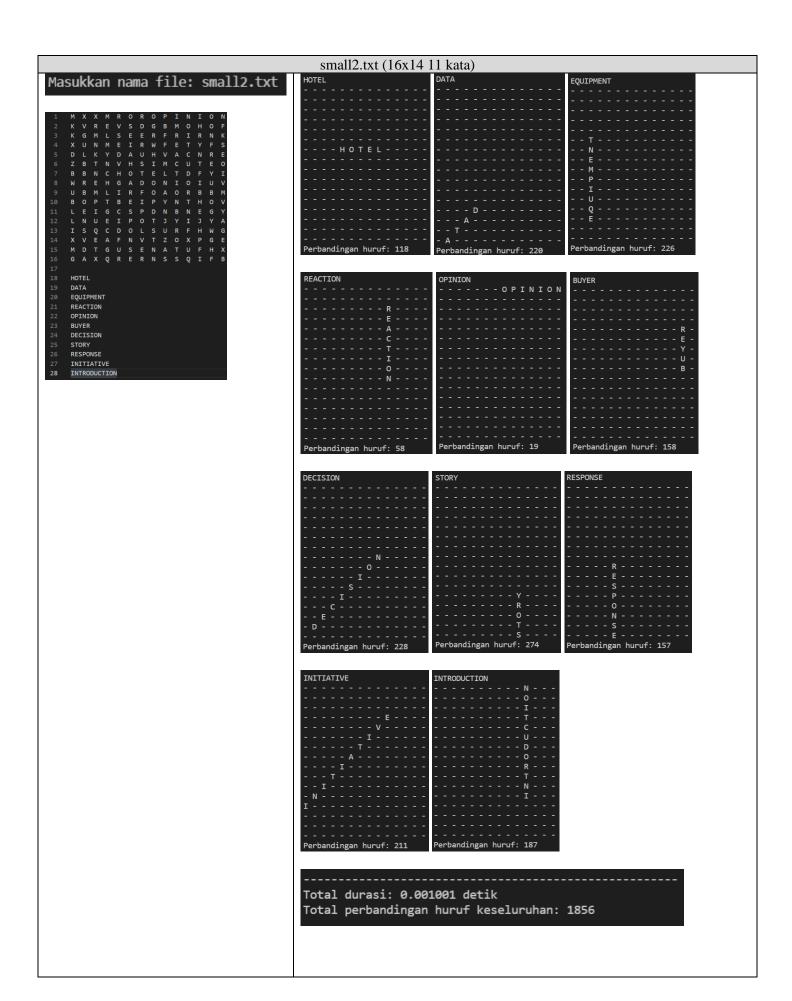
```
#include <iostream>
#include <stdlib.h>
#include <string>
#include <bits/stdc++.h>
#include "puzzle.h"
#include "words.h"
#include "search.h"
#include "fileToData.h"
using namespace std;
int main()
   Puzzle* puzzle1 = new Puzzle(50,50);
   Words* words1 = new Words(20);
    double durationTotal = 0;
    int countTotal = 0;
    string filename;
    cout << "Masukkan nama file: ";</pre>
    cin >> filename;
    filename = "../test/" + filename;
    fileToData(filename, puzzle1, words1);
    for (int i=0; i<words1->total; i++)
        Search* s = new Search();
        string str = words1->getWord(i);
        Puzzle* out = new Puzzle(puzzle1->rows,puzzle1->cols);
        chrono::steady_clock sc;
        auto start = sc.now();
        // ALGORITMA BRUTEFORCE PENCARIAN KATA
        // menghasilkan indeks ditemukannya huruf pertama kata dan arah
        // yang kemudian akan di set ke matrix hasil
        s->find(puzzle1, str);
```

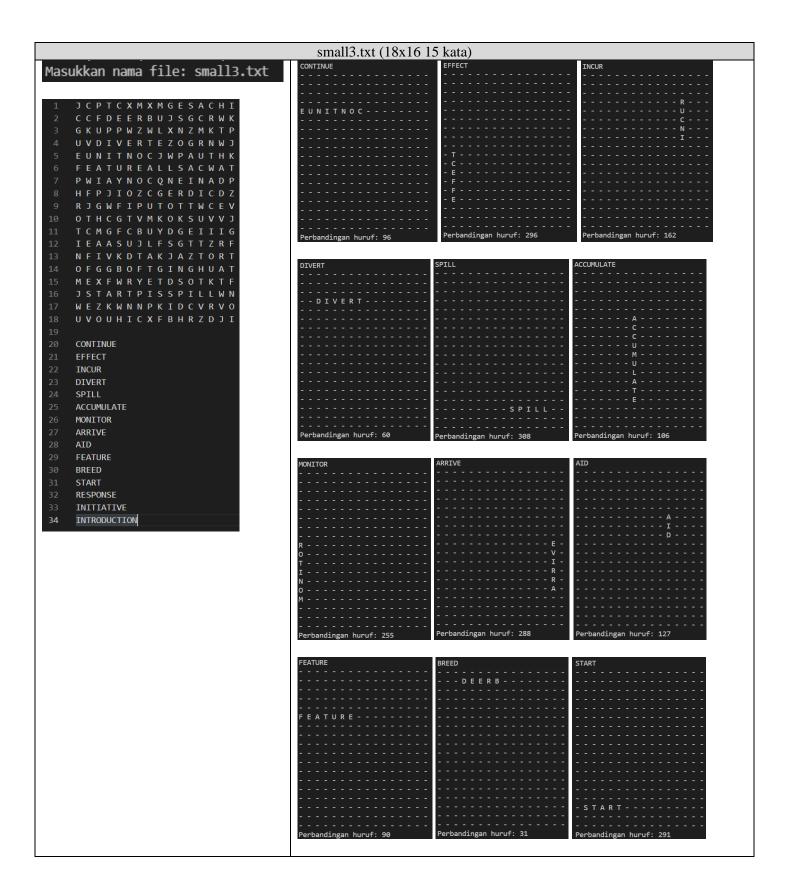
```
out->setResult(s->frow,s->fcol, s->found, str);
   // cout << endl << s->frow << " " << s->fcol << " " << s->found;
   auto end = sc.now();
   cout << endl << str << endl;</pre>
   out->display();
   cout << "Perbandingan huruf: " << s->count << endl;</pre>
   auto duration = static_cast<std::chrono::duration<double>>(end-start);
   durationTotal += duration.count();
   countTotal += s->count;
cout << "\n----\n";</pre>
cout << "Total durasi: " << durationTotal << " detik\n";</pre>
cout << "Total perbandingan huruf keseluruhan: " << countTotal << "\n";</pre>
string dummy;
cout << "\nPress anything to exit (then enter): ";</pre>
cin >> dummy;
return 0;
```

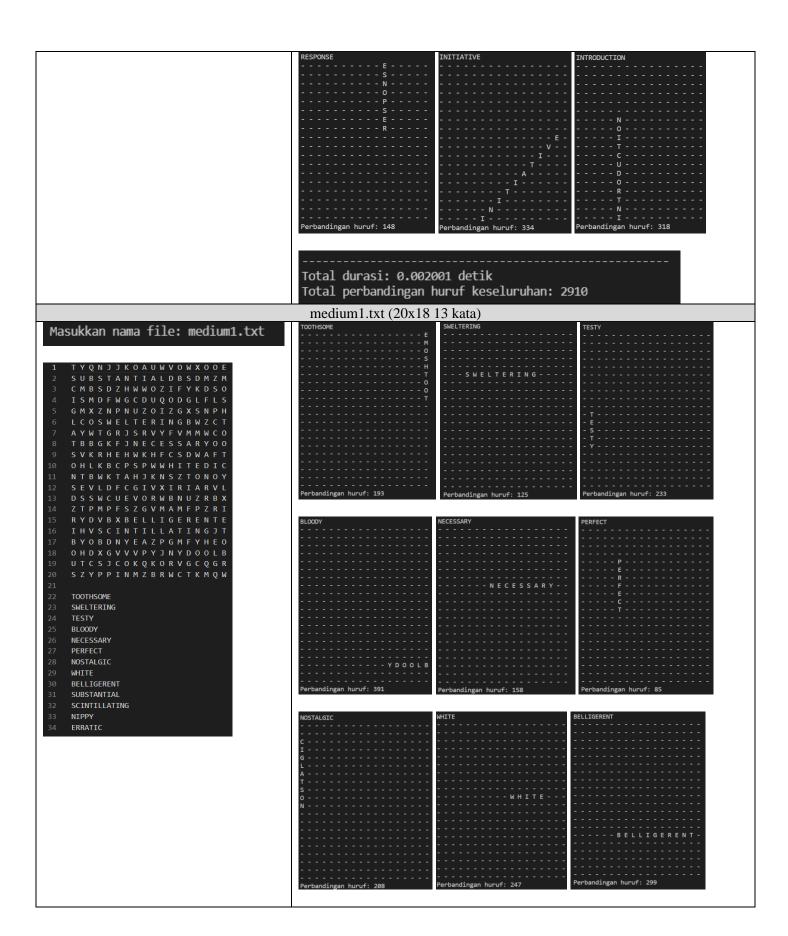
C. Input dan Output Hasil Eksekusi

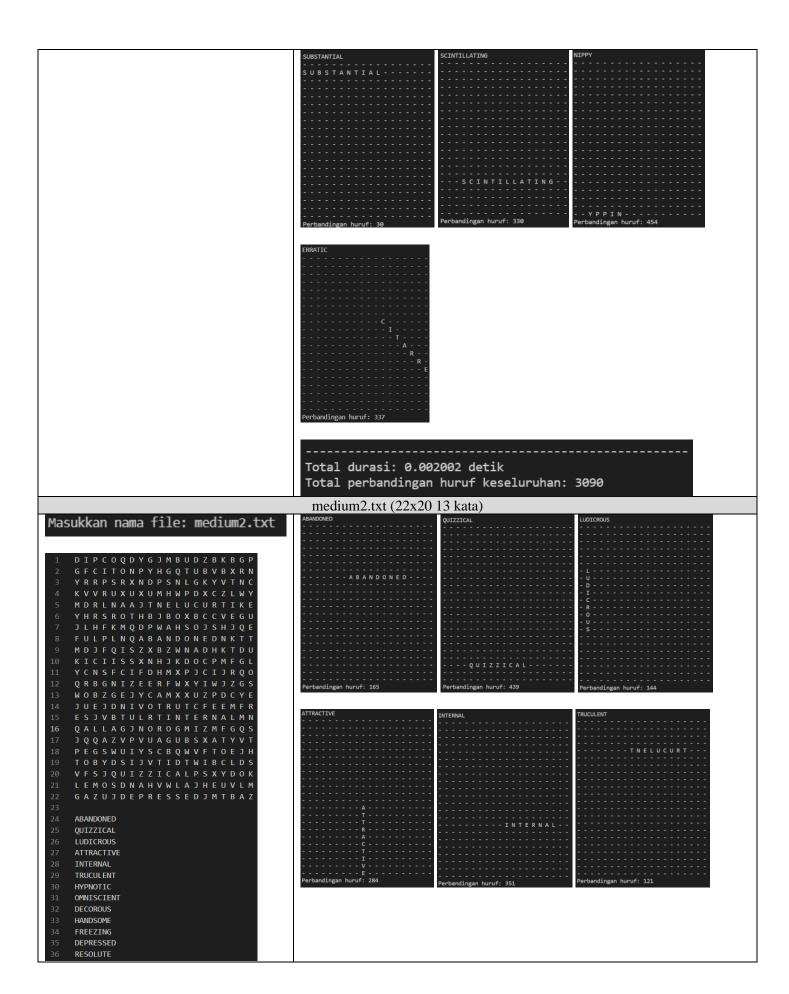
Berikut adalah hasil eksekusi 9 *test case* yang terdapat pada folder /test. Total durasi yang kurang stabil disebabkan kecepatan eksekusi yang juga dipengerahui beberapa faktor lain saat program sedang berjalan, sehingga bisa saja didapatkan total durasi yang berbeda untuk *test case* yang sama.)

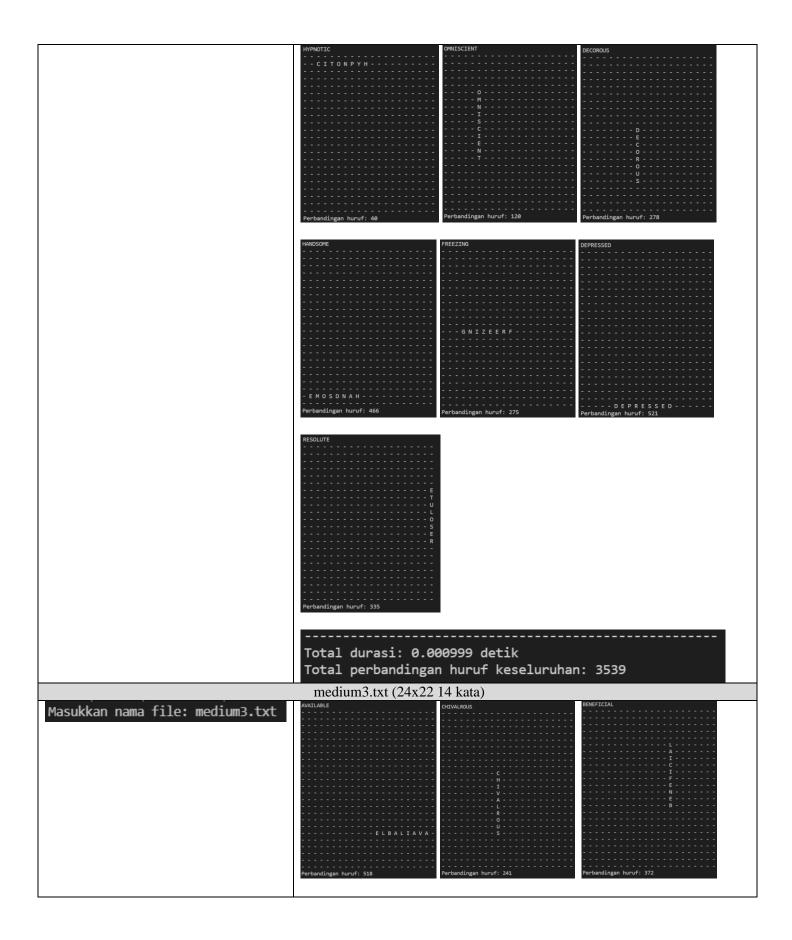


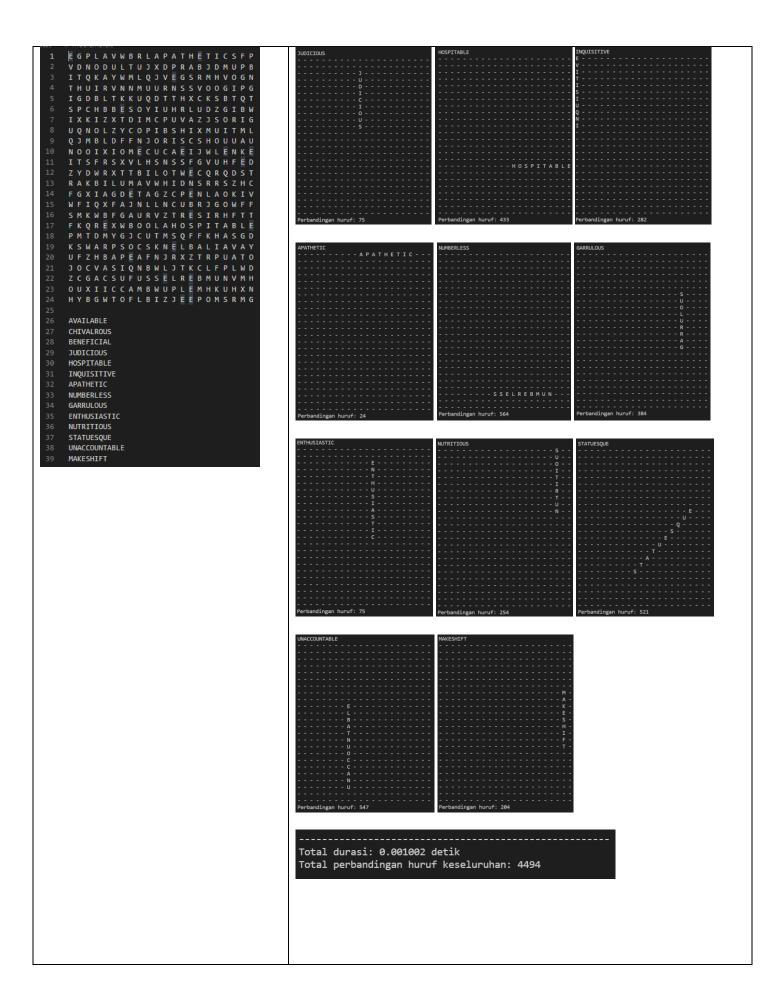


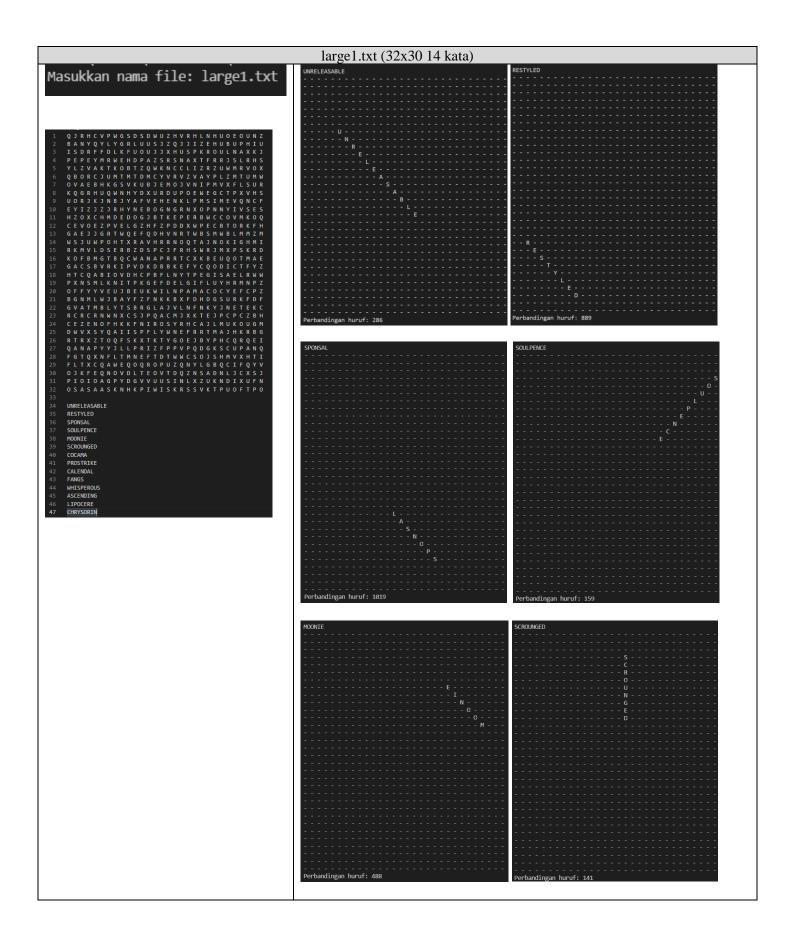


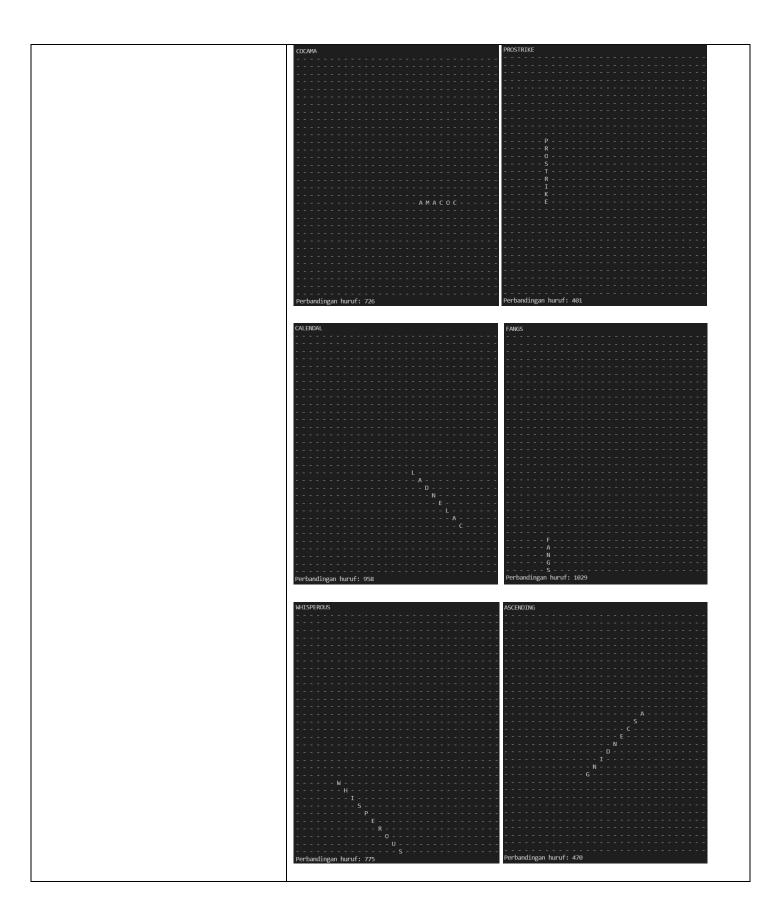


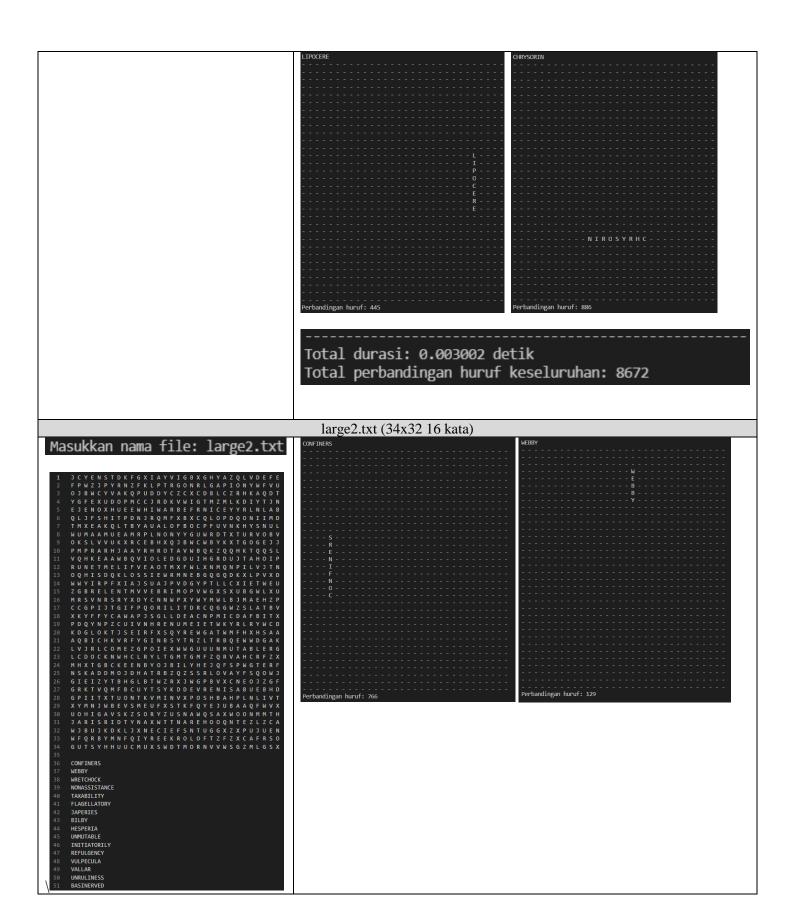


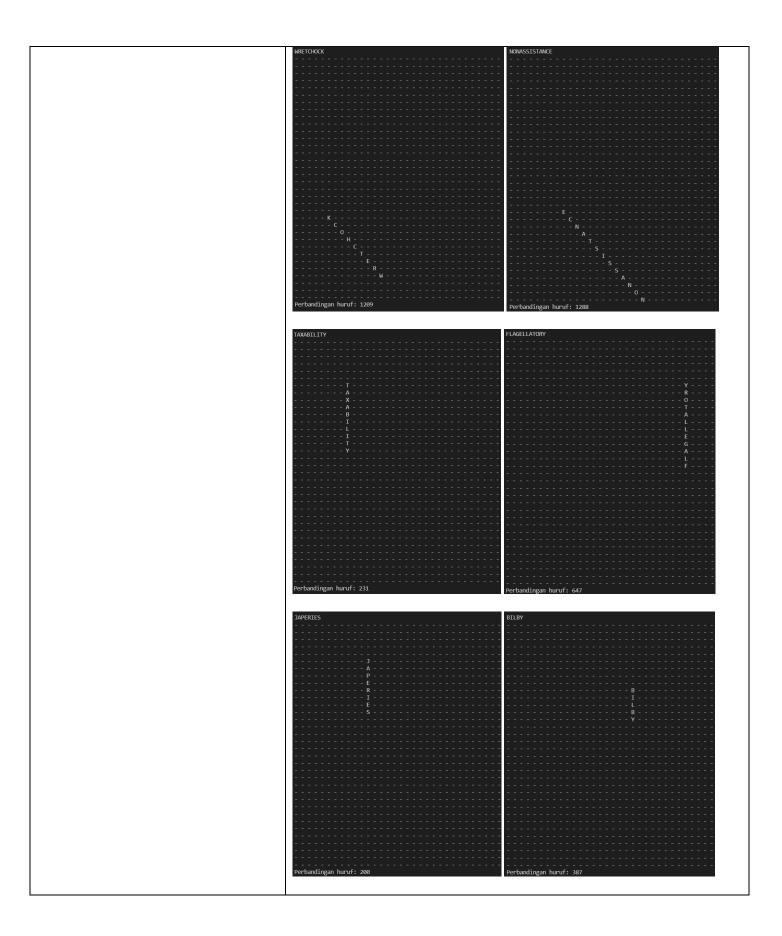


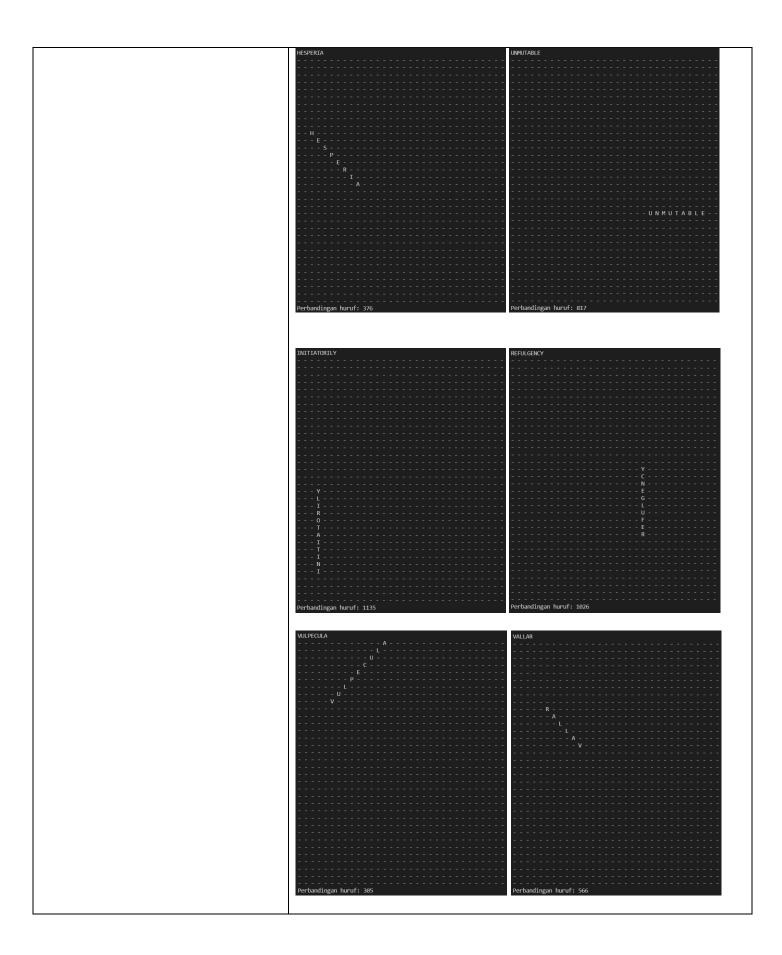


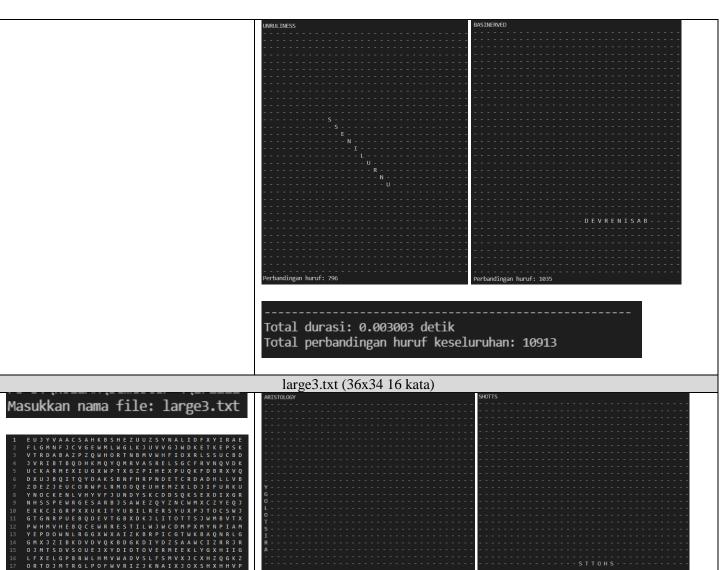


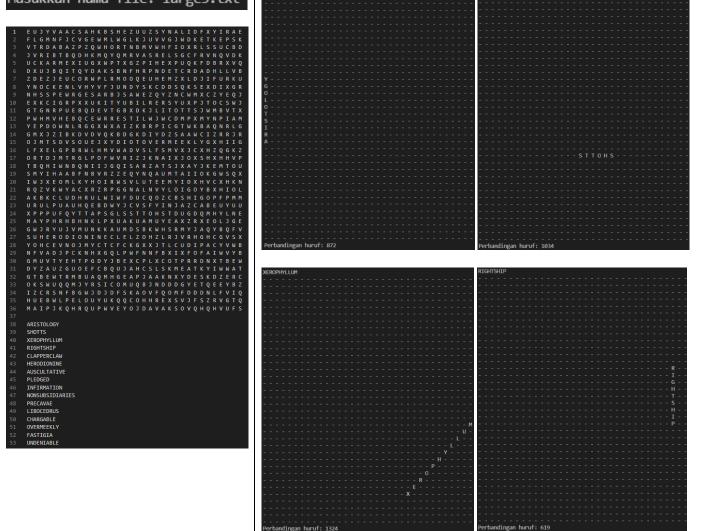


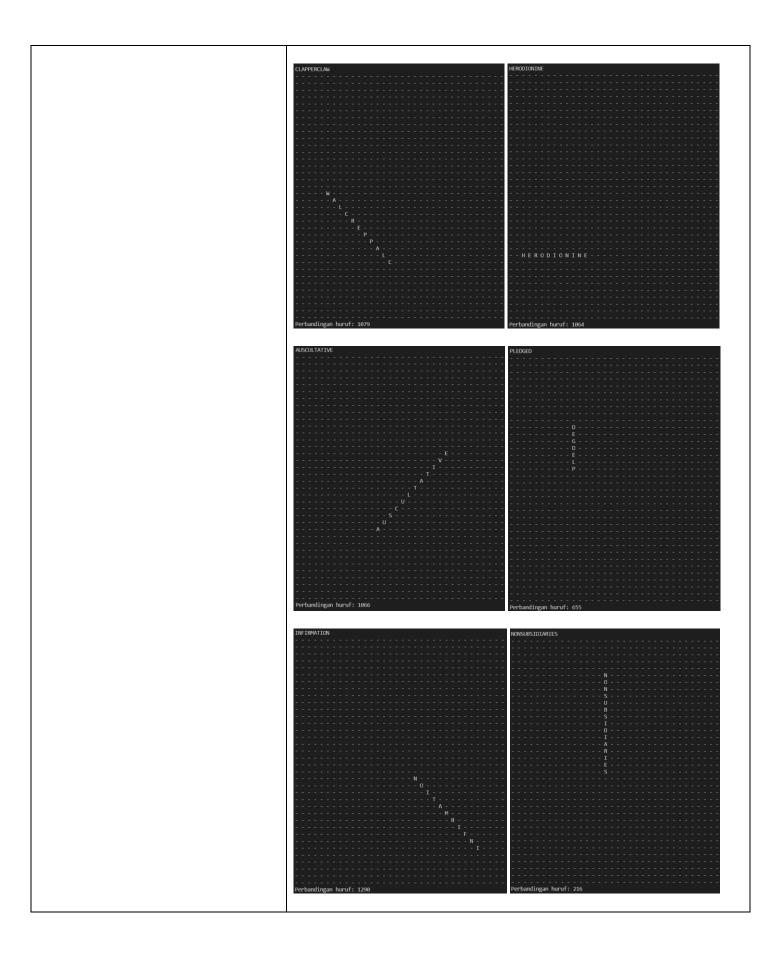


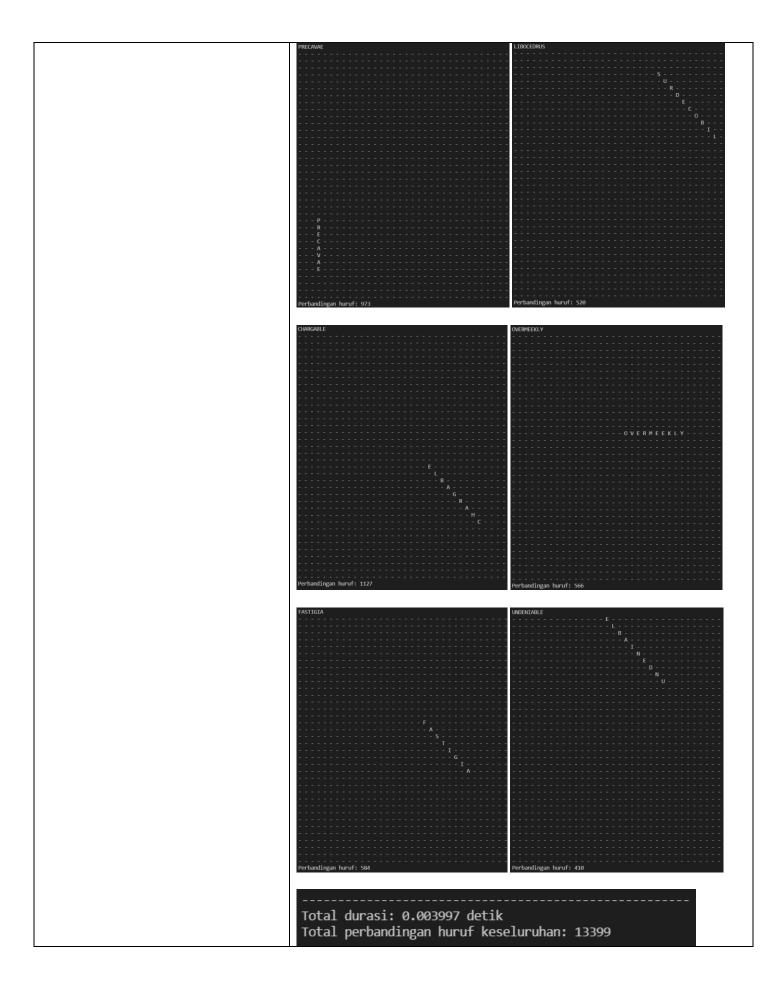












D. Repository Program (Github)

 $\underline{https://github.com/Putriliza/Tucil1Stima-WordSearchPuzzle}$

E. Rangkuman Keberjalanan Program

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