LAPORAN TUGAS KECIL IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2021/2022

PENYELESAIAN WORD SEARCH PUZZLE DENGAN ALGORITMA BRUTE FORCE

Disusun oleh:

13520156 Dimas Faidh Muzaki

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

Algoritma *brute force*

Algoritma *brute force* digunakan untuk menyelesaikan word search puzzle dijelaskan dalam urutan berikut:

1. Pertama, file konfigurasi yang berisikan puzzle dan kata-kata yang perlu dicari dibuka dengan library fstream pada program berbahasa C++.
2. Board/papan puzzle dimuat ke sebuah matriks yang merupakan vector of vectors of chars bernama board. Fstream membaca file konfigurasi perbaris, maka proses tersebut dilakukan secara traversal. Selain itu, kata-kata yang perlu dicari pada puzzle juga dimasukkan ke dalam sebuah vector of strings bernama words.
3. Setelah program selesai memuat file konfigurasi, program akan menampilkan konfigurasi ke terminal.
4. Proses *brute force* dimulai pada titik ini. Program akan mentraversal matriks board. Untuk setiap elemen char pada matriks, words juga akan ditraversal. Iterator board dan kata pada words yang medapatkan giliran akan dilemparkan ke sebuah prosedur.
5. Misal, iterator board yang dilempar ke dalam prosedur adalah (i,j) dan kata yang dilempar adalah “adaad”. Maka, prosedur dimulai dengan mengecek kesamaan character pada board[i][j] dengan huruf pertama pada “adaad”. Jika kondisinya sama, maka kata yang dicek selanjutnya adalah huruf selanjutnya pada kata “adaad” dengan character lain disekitar board[i][j]. Prosedur juga menerima parameter lain yang menunjukkan arah pencarian misalkan x:
    a. Jika x = 1, maka pengecekan akan mengarah ke Timur
    b. Jika x = 2, maka pengecekan akan mengarah ke Tenggara
    c. Jika x = 3, maka pengecekan akan mengarah ke Selatan
    d. Jika x = 4, maka pengecekan akan mengarah ke Barat Daya
    e. Jika x = 5, maka pengecekan akan mengarah ke Barat
    f. Jika x = 6, maka pengecekan akan mengarah ke Barat Laut
    g. Jika x = 7, maka pengecekan akan mengarah ke Utara
    h. Jika x = 8, maka pengecekan akan mengarah ke Timur Laut
6. Pengecekan akan dilakukan selama i dan j valid, yaitu tidak keluar batas dari luas papan/board/puzzle dan selama huruf dan character pada kata dan pada elemen matriks yang ditunjuk masih sama.
7. Apabila kecocokan keseluruhan kata ditemukan, program akan menampilkan sebuah board temporary yang dibuat selama proses *brute force* untuk memberitahu pengguna letak kata.
8. Langkah empat sampai tujuh akan diulang untuk semua elemen matriks dan seluruh kata yang dicari.
9. Langkah empat sampai delapan akan dicatat lama eksekusi dan banyak proses pengecekan karakter untuk ditampilkan kepada pengguna di akhir program.

Source program dalam Bahasa C++

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <chrono>
using namespace std;
```

```
using namespace std::chrono;

/**
 * Beberapa data esensial dibuat global
 * untuk memudahkan penggunaan
 */
int row, col = 0;
vector<vector<char>> board;
vector<string> words;
int counter = 0;

/* Deklarasi prosedur pembantu */
void readFile(string path);
void displayTest();
void checkWord(int i, int j, string word, int arah);
void generate(vector<vector<char>> &vec);

int main(int argc, const char **argv)
{
    string path;

    /* pembacaan file konfigurasi */
    readFile(argv[1]);
    /* menampilkan hasil yang dibaca */
    displayTest();

    /* mulai pencatatan waktu untuk brute force */
    auto start = high_resolution_clock::now();
    /* traversal matriks */
    for (int i = 0; i < board.size(); i++)
    {
        for (int j = 0; j < board[i].size(); j++)
        {
            for (string word : words)
            {
                /* pengecekan tiap elemen matriks */
                /*  untuk tiap kata pada words */
                checkWord(i, j, word, 1);
                checkWord(i, j, word, 2);
                checkWord(i, j, word, 3);
                checkWord(i, j, word, 4);
                checkWord(i, j, word, 5);
                checkWord(i, j, word, 6);
                checkWord(i, j, word, 7);
                checkWord(i, j, word, 8);
            }
        }
    }
```

```
    auto stop = high_resolution_clock::now();
    cout << endl << "Pengecekan huruf dilakukan sebanyak: " << counter << "
kali" << endl;
    auto duration = duration_cast<microseconds>(stop - start);
    cout << "Puzzle diselesaikan dalam waktu: " << duration.count() << "
microseconds"<< endl;
    return 0;
}

void generate(vector<vector<char>> &vec)
{

    for (int i = 0; i < row; i++)
    {
        vector<char> v_row;
        for (int j = 0; j < col; j++)
        {
            v_row.push_back('_');
        }
        vec.push_back(v_row);
    }
}

void checkWord(int i, int j, string word, int arah)
{
    vector<vector<char>> vec;
    bool isRight = false;
    int k = 0;

    generate(vec);

    // cout << word << endl ;
    while (k < word.size() && i >= 0 && j >= 0 && i < row && j < col &&
counter++ && word[k] == board[i][j])
    {
        vec[i][j] = word[k];
        k += 1;
        switch (arah)
        {
        case 1:
            j = j + 1;
            i = i;
            break;
        case 2:
            j = j + 1;
            i = i + 1;
            break;
        case 3:
```

```
            j = j;
            i = i + 1;
            break;
        case 4:
            j = j - 1;
            i = i + 1;
            break;
        case 5:
            j = j - 1;
            i = i;
            break;
        case 6:
            j = j - 1;
            i = i - 1;
            break;
        case 7:
            j = j;
            i = i - 1;
            break;
        case 8:
            j = j + 1;
            i = i - 1;
            break;
        }
    }

    if (k == word.size())
    {
        cout << endl
             << word << endl;
        for (int l = 0; l < row; l++)
        {
            for (int m = 0; m < col; m++)
            {
                cout << vec[l][m] << " ";
            }
            cout << endl;
        }
    }
}

void readFile(string path)
{
    /* Kamus */
    fstream file;
    /* Algoritma */

    /* Membuka file konfigurasi */
```

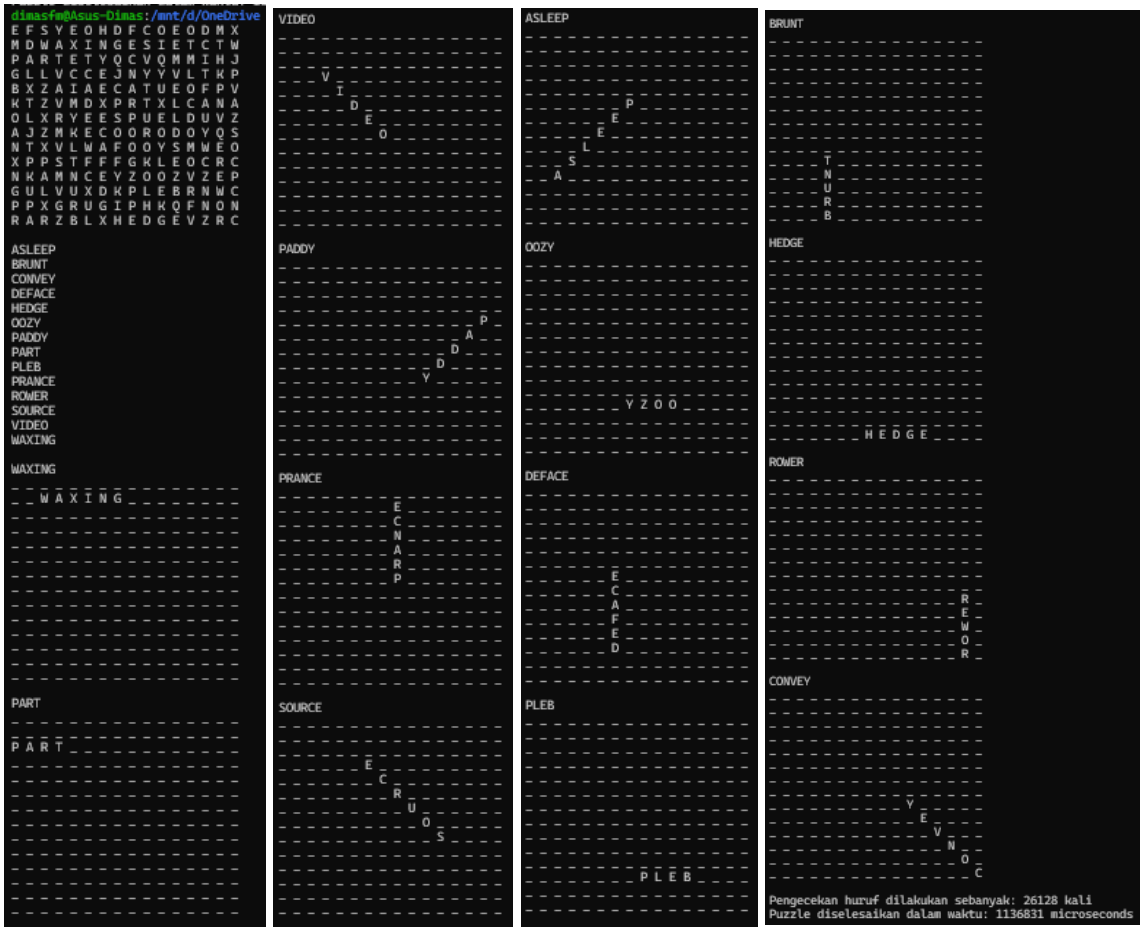
```
    file.open(path, ios::in);
    if (file.is_open())
    {
        string line;
        /* Proses pembacaan board permainan */
        while (getline(file, line))
        {
            vector<char> b_row;
            /* Panjang line 1 memiliki arti baris kosong */
            if (line.length() == 1)
                break;
            else
            {
                col = line.length() / 2;
                for (int i = 0; i < line.length(); i += 2)
                {
                    b_row.push_back(line[i]);
                }
                board.push_back(b_row);
                row += 1;
            }
        }

        /* Pembacaan kata-kata yang ingin dicari */
        while (file >> line)
        {
            words.push_back(line);
        }

        file.close();
    }
}

void displayTest()
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            cout << board[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
    for (auto i = words.begin(); i != words.end(); ++i)
    {
        cout << *i << endl;
    }
```

```
}
```

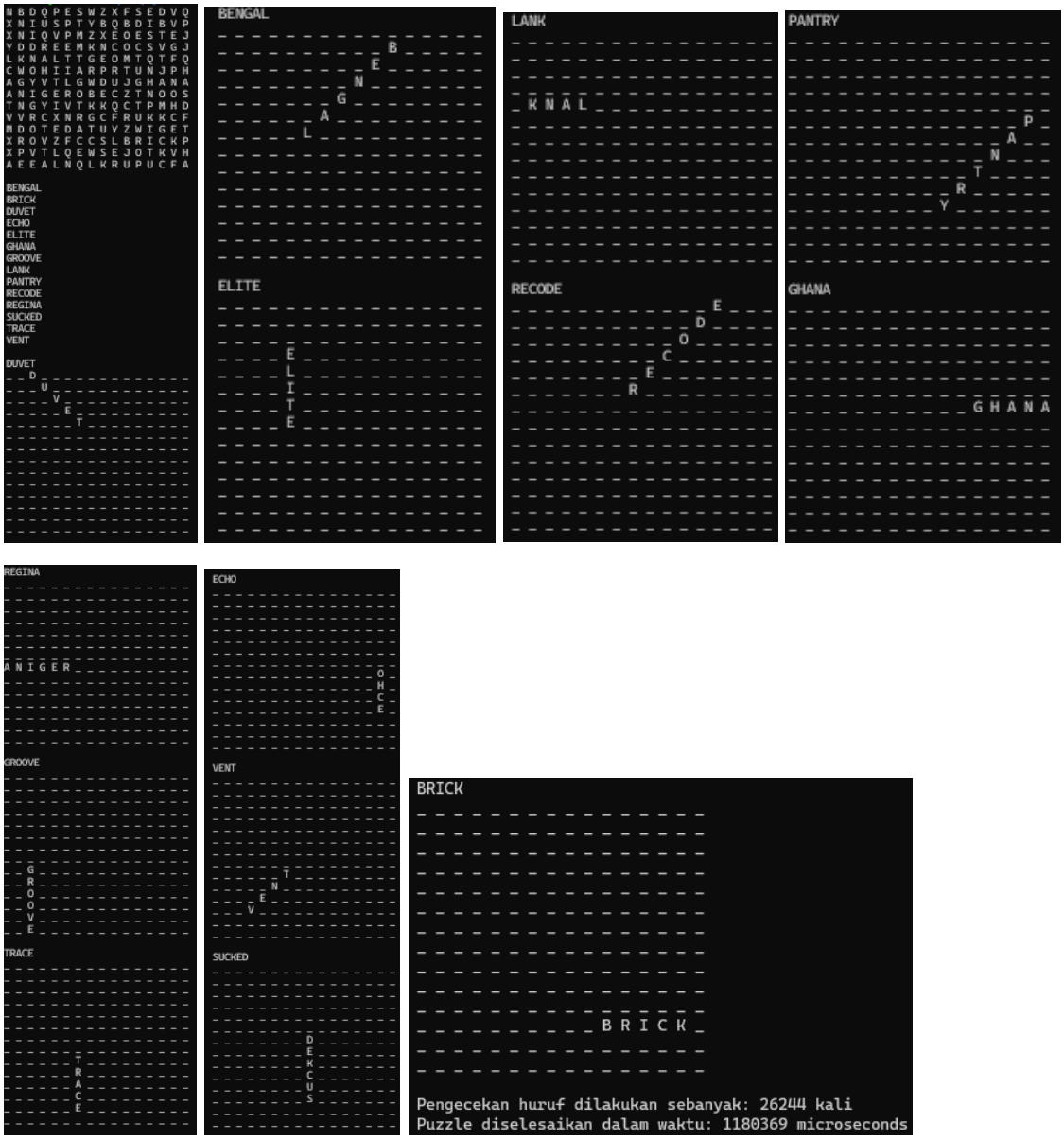
Screenshot eksekusi program

Terdapat 9 contoh test case yang terdiri dari 3 puzzle ukuran small(14x16), 3 medium(20x22), dan 3 large(32x34) yang konfigurasinya terletak pada folder test. Screenshot dibawah tidak mencakup semua hasil kata yang ditemukan karena sangat besar dan banyak.

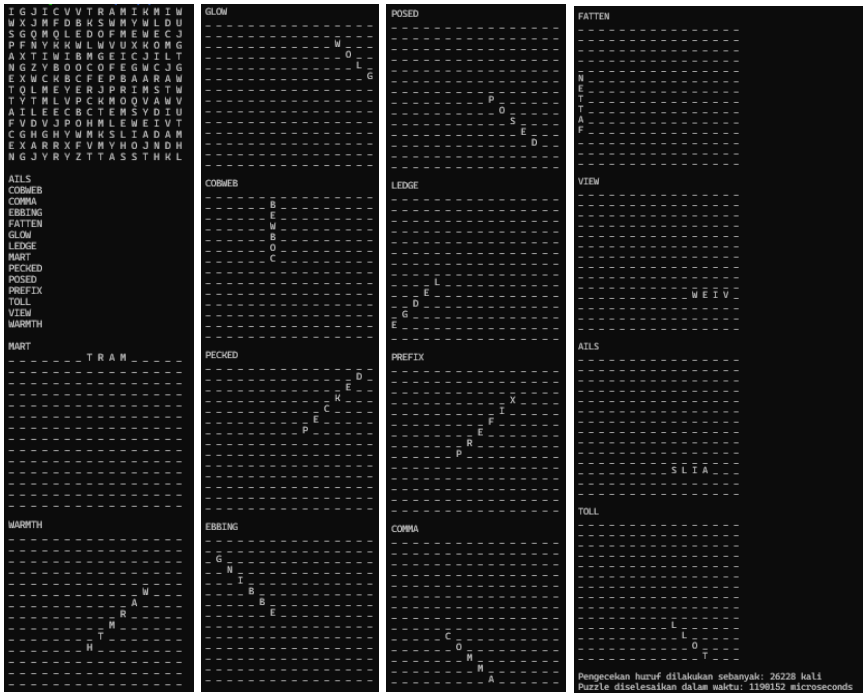
Small 1



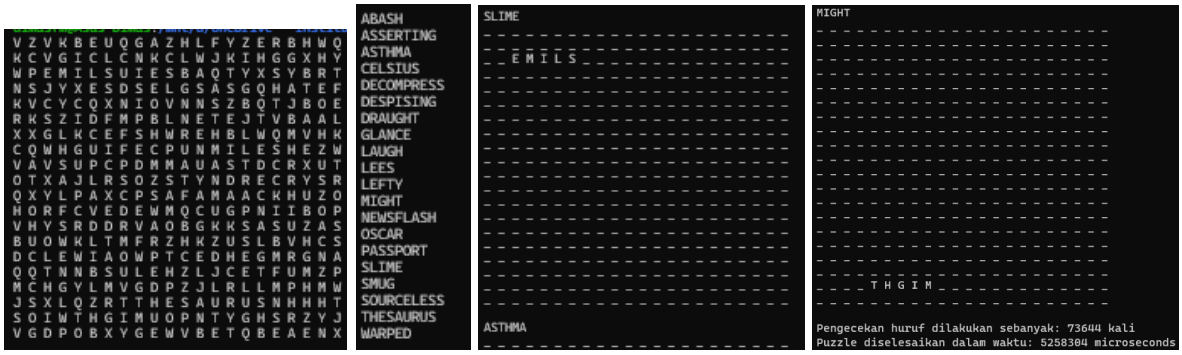
Small 2


BENGAL
BRICK
DUVET
ECHO
ELITE
GHANA
GROOVE
LANK
PANTRY
RECODE
REGINA
SUCKED
TRACE
VENT
LANK
PANTRY
ELITE
RECODE
GHANA
REGINA
ECHO
GROOVE
VENT
TRACE
SUCKED
BRICK
Pengecekan huruf dilakukan sebanyak: 26244 kali
Puzzle diselesaikan dalam waktu: 1180369 microseconds

Small 3



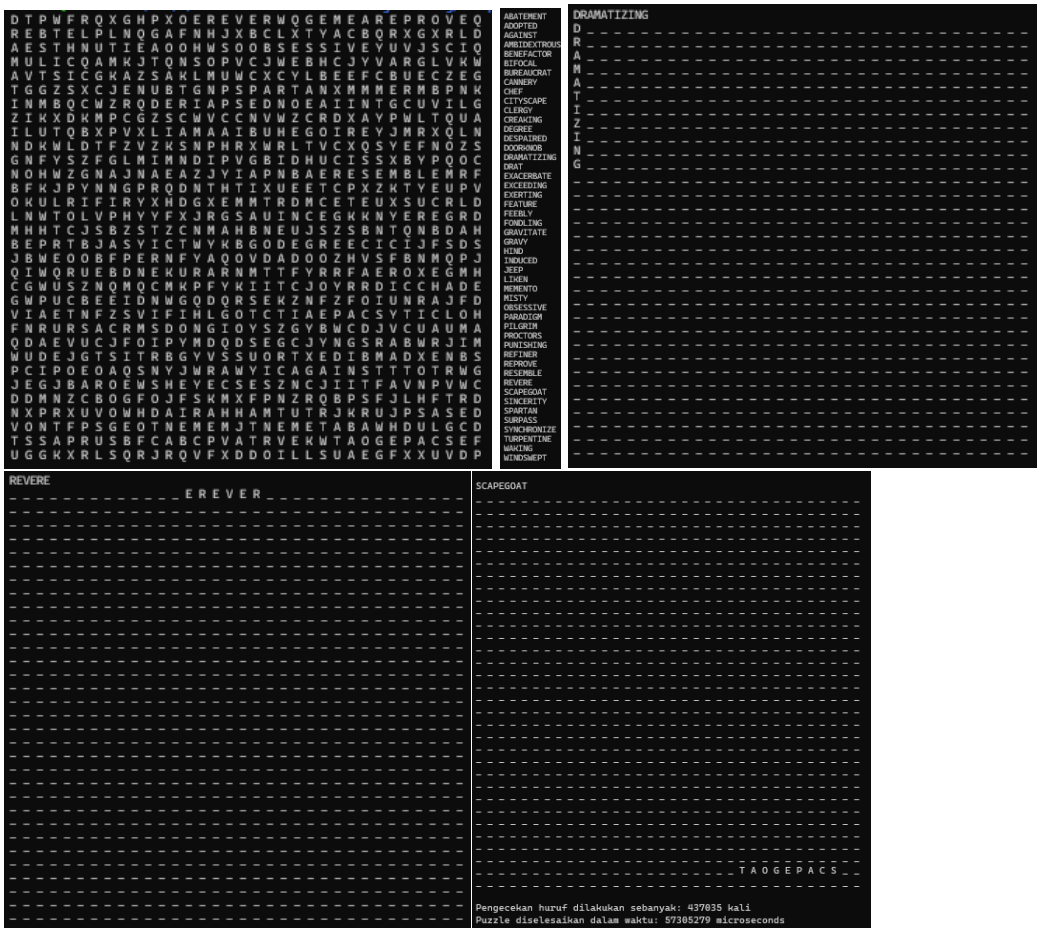
Medium 1



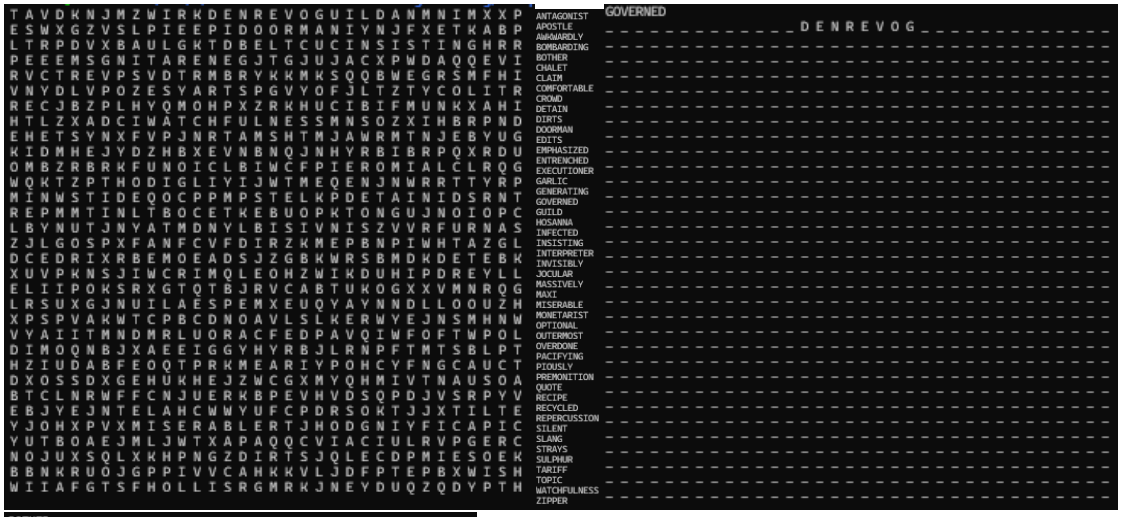
Medium 2

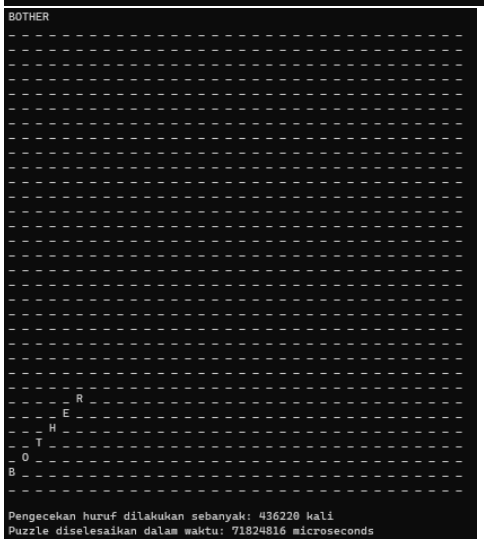
Medium 3

Large 1

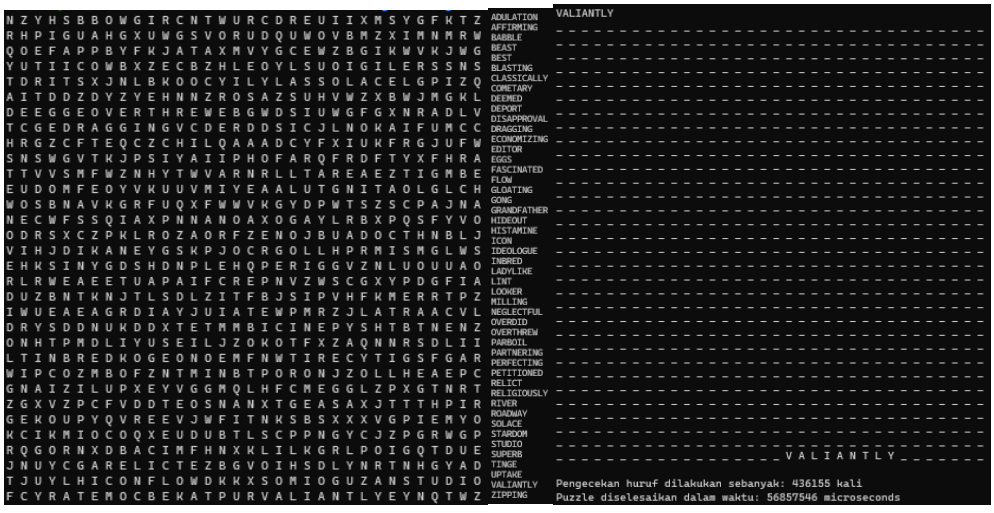


Large 2





Large 3



Link Drive

https://drive.google.com/drive/folders/1T5VBOCC7kxu-A268c0MUU2YegDCcc6Gp?usp=sharing