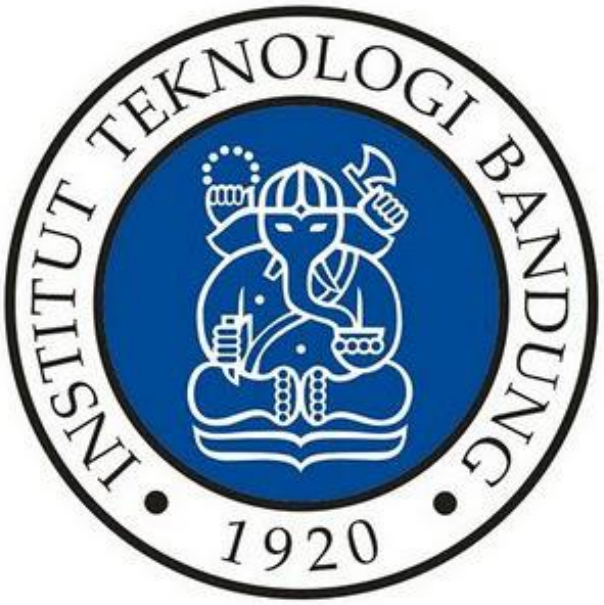
Tugas Kecil 1 - Word Search Puzzle

IF2211 Strategi Algoritma



Oleh

Bayu Samudra
13520128

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

A. Pendahuluan

Permainan *Word Search Puzzle* adalah permainan yang mengharuskan pemainnya mencari kata yang tersembunyi pada sekumpulan kata acak pada tabel. Untuk menemukan kata ini, dapat dilakukan dalam 8 arah mengikuti arah mata angin. Permainan ini pertama kali dipublikasikan oleh Noeman E. Gibat dalam the *Seleby Digest* pada 1 Maret 1968. Permainan ini sejak pertama kali diluncurkan mulai terkenal dan banyak digunakan oleh guru-guru untuk mengajarkan kepada murid-muridnya.

B. Deskripsi Algoritma yang Digunakan

Solusi yang dilakukan untuk menyelesaikan adalah menggunakan pendekatan algoritma brute force. Langkah yang dilakukan yaitu melakukan iterasi dimulai dari awal matriks yang berisi karakter puzzle hingga akhir. Untuk tiap iterasi, dilakukannya iterasi dengan 8 arah mata angin dengan utara menunjuk ke arah atas layar dan barat ke arah kiri layar. Pada tiap iterasi arah mata angin, dilakukan pemeriksaan string dengan cara memeriksa mencari manakah karakter awal yang cocok dengan pattern yang diinginkan. Jika ditemukan karakter awal, maka diperiksa apakah sesuai dengan pattern. Apabila tidak sesuai, dicari awal maka cari karakter lain yang diduga sebagai karakter pertama. Jika ditemukan kata yang cocok, maka tandai pada tabel warna untuk digunakan saat pencetakan.

Proses iterasi pada tiap arah mata angin dapat dilakukan dengan menyimpan lokasi saat ini menggunakan tuple. Perpindahan posisi hanya perlu mengubah lokasi ini sesuai dengan arah yang diinginkan. Bila posisi sudah tidak ada pada tabel, ubahlah posisi menjadi (-1,-1) dan keluar dari proses iterasi.

Dengan algoritma diatas, dapat diprediksi bahwa kompleksitas yang didapatkan adalah $O(n^3)$ dikarenakan iterasi tiap arah mata angin dilakukan hingga posisi tidak tersedia pada matriks.

Algoritma ini pada dasarnya masih dapat dioptimasi dengan cara menganggap titik iterasi merupakan awal karakter dari kata di pattern. Pemeriksaan hanya dilakukan sepanjang karakter yang pada string. Oleh karena itu, kompleksitas dapat dikurangi hingga $O(m.n^2)$ dengan $m < n$.

C. Kode program yang digunakan

Dalam implementasi algoritma ini, saya menggunakan bahasa C++ dengan pustaka penyimpanan dinamis seperti `vector` dan `pair`. Saya menggunakan paradigma berorientasi objek-prosedural untuk *matcher* dari tiap katakter.

Berikut ini adalah file header tipe yang digunakan:

```
#pragma once

enum Direction {
 NORTH,
 NORTHEAST,
 EAST,
 SOUTHEAST,
 SOUTH,
 SOUTHWEST,
 WEST,
 NORTHWEST
};

enum Color {
 NORMAL,
 HIDDEN,
 RED,
 GREEN,
 YELLOW,
 BLUE,
 MAGENTA,
 CYAN,
 WHITE
};
```

Berikut ini adalah kode sumber untuk matcher yang digunakan:

Header

```
#pragma once

#include <string>
#include <vector>

#include "types.hpp"
using namespace std;
```

```
namespace Matcher {
class StringMatcher {
private:
 vector<vector<Color>> colorTable;
 vector<string> charTable;
 pair<int, int> size;
 int colorNumber = 0;

 bool isPositionExist(pair<int, int> pos);
 int directionLength(pair<int, int> pos, Direction dir);
 bool isAdjecentIdxAvailable(pair<int, int> idx, Direction dir);
 pair<int, int> getAdjecentIdx(pair<int, int> idx, Direction dir);
 void matchDirection(pair<int, int> pos, Direction dir, string
pattern,
                     bool heuristic);

public:
 StringMatcher(vector<string> table);
 StringMatcher(vector<string> table, Color defaultColor);

 void match(string s);
 void match(string s, bool heuristic);
 pair<int, int> getSize();
 vector<vector<Color>> getColorizedResult();
};
}  // namespace Matcher
```

Source Code

```
#include "Matcher.hpp"

#include <string>
using namespace std;
using namespace Matcher;

/* Private Method */
int StringMatcher::directionLength(pair<int, int> pos, Direction
dir) {
 if (!this->isPositionExist(pos)) {
```

```
    return -1;
  }

  switch (dir) {
    case NORTH:
      return pos.first + 1;
    case NORTHWEST:
      return min(this->directionLength(pos, NORTH),
                 this->directionLength(pos, WEST));
    case WEST:
      return pos.second + 1;
    case SOUTHWEST:
      return min(this->directionLength(pos, SOUTH),
                 this->directionLength(pos, WEST));
    case SOUTH:
      return this->size.first - pos.first;
    case SOUTHEAST:
      return min(this->directionLength(pos, SOUTH),
                 this->directionLength(pos, EAST));
    case EAST:
      return this->size.second - pos.second;
    case NORTHEAST:
      return min(this->directionLength(pos, NORTH),
                 this->directionLength(pos, EAST));
  }

  return -1;
}

bool StringMatcher::isPositionExist(pair<int, int> pos) {
  return pos.first >= 0 && pos.second >= 0 && pos.first < size.first
&&
         pos.second < size.second;
}

bool StringMatcher::isAdjecentIdxAvailable(pair<int, int> idx,
Direction dir) {
  switch (dir) {
    case NORTH:
      return this->isPositionExist(pair<int, int>(idx.first - 1,
idx.second));
```

```
    case EAST:
      return this->isPositionExist(pair<int, int>(idx.first,
idx.second + 1));
    case SOUTH:
      return this->isPositionExist(pair<int, int>(idx.first + 1,
idx.second));
    case WEST:
      return this->isPositionExist(pair<int, int>(idx.first,
idx.second - 1));
    case NORTHEAST:
      return this->isAdjecentIdxAvailable(idx, NORTH) &&
             this->isAdjecentIdxAvailable(idx, EAST);
    case NORTHWEST:
      return this->isAdjecentIdxAvailable(idx, NORTH) &&
             this->isAdjecentIdxAvailable(idx, WEST);
    case SOUTHEAST:
      return this->isAdjecentIdxAvailable(idx, SOUTH) &&
             this->isAdjecentIdxAvailable(idx, EAST);
    case SOUTHWEST:
      return this->isAdjecentIdxAvailable(idx, SOUTH) &&
             this->isAdjecentIdxAvailable(idx, WEST);
  }

  return false;
}

pair<int, int> StringMatcher::getAdjecentIdx(pair<int, int> idx,
                                             Direction dir) {
  if (!this->isAdjecentIdxAvailable(idx, dir)) {
    return pair<int, int>(-1, -1);
  }

  switch (dir) {
    case NORTH:
      return pair<int, int>(idx.first - 1, idx.second);
    case NORTHEAST:
      return pair<int, int>(idx.first - 1, idx.second + 1);
    case EAST:
      return pair<int, int>(idx.first, idx.second + 1);
    case SOUTHEAST:
      return pair<int, int>(idx.first + 1, idx.second + 1);
```

```
    case SOUTH:
      return pair<int, int>(idx.first + 1, idx.second);
    case SOUTHWEST:
      return pair<int, int>(idx.first + 1, idx.second - 1);
    case WEST:
      return pair<int, int>(idx.first, idx.second - 1);
    case NORTHWEST:
      return pair<int, int>(idx.first - 1, idx.second - 1);
  }

  return pair<int, int>(-1, -1);
}

/* Match Direction */
void StringMatcher::matchDirection(pair<int, int> pos, Direction
dir,
                                   string pattern, bool heuristic) {
  pair<int, int> currentPos = pos;
  while (this->directionLength(currentPos, dir) >=
(int)pattern.length() ||
         (!heuristic && this->directionLength(currentPos, dir) >= 0))
{
    int match = 0;
    pair<int, int> posIndex = currentPos;

    for (int i = 0; i < (int)pattern.length(); i++) {
      if (!this->isPositionExist(posIndex) ||
          this->charTable[posIndex.first][posIndex.second] !=
pattern[match]) {
        break;
      }

      match++;
      posIndex = this->getAdjecentIdx(posIndex, dir);
    }

    if (match == (int)pattern.length()) {
      // Colorize
      pair<int, int> posIndex = currentPos;
      for (int i = 0; i < (int)pattern.length(); i++) {
        this->colorTable[posIndex.first][posIndex.second] =
```

```
            (Color)(this->colorNumber % 7 + 2);
        posIndex = this->getAdjecentIdx(posIndex, dir);
      }

      this->colorNumber++;
    }

    currentPos = this->getAdjecentIdx(currentPos, dir);
  }
}

/* Constructor */
StringMatcher::StringMatcher(vector<string> table)
    : StringMatcher(table, HIDDEN) {}

StringMatcher::StringMatcher(vector<string> table, Color
defaultColor) {
  this->charTable = table;
  this->size.first = table.size();

  if (this->size.first > 0) {
    this->size.second = table[0].length();
  } else {
    this->size.second = 0;
  }

  for (int i = 0; i < this->size.first; i++) {
    vector<Color> row;
    for (int j = 0; j < this->size.second; j++) {
      row.push_back(defaultColor);
    }
    this->colorTable.push_back(row);
  }
}

/* Public Methods */
vector<vector<Color>> StringMatcher::getColorizedResult() {
  return this->colorTable;
}

void StringMatcher::match(string pattern, bool heuristic) {
```

```
 for (int i = 0; i < this->size.first; i++) {
   for (int j = 0; j < this->size.second; j++) {
     this->matchDirection(pair<int, int>(i, j), NORTH, pattern,
heuristic);
     this->matchDirection(pair<int, int>(i, j), NORTHWEST, pattern,
heuristic);
     this->matchDirection(pair<int, int>(i, j), WEST, pattern,
heuristic);
     this->matchDirection(pair<int, int>(i, j), SOUTHWEST, pattern,
heuristic);
     this->matchDirection(pair<int, int>(i, j), SOUTH, pattern,
heuristic);
     this->matchDirection(pair<int, int>(i, j), SOUTHEAST, pattern,
heuristic);
     this->matchDirection(pair<int, int>(i, j), EAST, pattern,
heuristic);
     this->matchDirection(pair<int, int>(i, j), NORTHEAST, pattern,
heuristic);
   }
 }
}

void StringMatcher::match(string pattern) { this->match(pattern,
false); }

pair<int, int> StringMatcher::getSize() { return this->size; }
```

Berikut ini adalah kode untuk utilitas yang digunakan

Header File

```
#pragma once

#include <string>
#include <vector>
using namespace std;

namespace Util {
string trimWhitespace(string input);
void trimVectorString(vector<string> &input);
}  // namespace Util
```

Source Code

```
#include "Util.hpp"
#include <cctype>
using namespace std;

string Util::trimWhitespace(string input){
 string result = "";
  for(char i : input) {
   if(isalnum(i)){
     result += i;
   }
 }

 return result;
}

void Util::trimVectorString(vector<string> &input){
 for(int i = 0; i < (int) input.size(); i++) {
   input[i] = trimWhitespace(input[i]);
 }
}
```

Berikut ini adalah kode program untuk melakukan proses output yang berkaitan dengan terminal.

Header File

```
#pragma once

#include <string>
#include <vector>

#include "../types.hpp"

using namespace std;

namespace Console {
void printTable(vector<string> charTable, vector<vector<Color>>
colorTable);
```

```
string getColoredChar(Color color, char ch);
void printHeader();
}  // namespace Console
```

Source Code

```
#include "Console.hpp"

#include <iostream>
using namespace std;

string getColorANSI(Color color) {
 switch (color) {
   case RED:
     return "\x1B[31m";
   case GREEN:
     return "\x1B[32m";
   case YELLOW:
     return "\x1B[33m";
   case BLUE:
     return "\x1B[34m";
   case MAGENTA:
     return "\x1B[35m";
   case CYAN:
     return "\x1B[36m";
   case WHITE:
     return "\x1B[37m";
   case HIDDEN:
     return "\x1B[90m";
   case NORMAL:
   default:
     return "\x1B[0m";
 }
}

string Console::getColoredChar(Color color, char ch) {
 return getColorANSI(color) + ch + getColorANSI(NORMAL);
}

void Console::printTable(vector<string> charTable,
```

```
                        vector<vector<Color>> colorTable) {
 if (charTable.size() == 0) {
   cout << endl;
   return;
 }

 int rows = charTable.size();
 int cols = charTable[0].length();

 for (int i = 0; i < rows; i++) {
   for (int j = 0; j < cols; j++) {
     if (j > 0) {
       cout << " ";
     }

     cout << getColoredChar(colorTable[i][j], charTable[i][j]);
   }

   cout << endl;
 }

 cout << endl;
}

void Console::printHeader() {
 string headerTemplate =
     " __          __           _   _____                _      \n \\
\\       "
     " / /          | | |  __ \\             | |    \n  \\ \\  /\\
/ /__  _ "
     "__ __| | | |__) |   _ _______| | ___ \n   \\ \\/  \\/ / _ \\|
'__/ _` | "
     "|  ___/ | | |_  /_  / |/ _ \\\\\n    \\  /\\  / (_) | | | (_| |
| |   | "
     "|_| |/ / / /| |  __/\n     \\/  \\/ \\___/|_|  \\__,_| |_|
"
     "\\__,_/___/___|_|\\___|\n";

 cout << getColorANSI(GREEN) << headerTemplate << getColorANSI(CYAN)
<< endl
      << "Versi 1.0.0" << getColorANSI(NORMAL) << endl;
```

```
}
```

Berikut ini adalah kode sumber yang digunakan untuk melakukan manipulasi file

Header File

```
#pragma once
#include <string>
#include <vector>

#include "../types.hpp"
using namespace std;

namespace FileManagement {
class FileReader {
private:
 string filename;
 vector<string> charTable;
 vector<string> patterns;

public:
 void parse();
 FileReader(string filename);
 FileReader(ifstream* filestream);
 vector<string> getTable();
 vector<string> getPatterns();
};

void writeTableToFile(ofstream& filestream, vector<string>
charTable,
                      vector<vector<Color>> colorTable);
}  // namespace FileManagement
```

Source Code

```
#include "File.hpp"
```

```cpp
#include <cctype>
#include <fstream>
#include <iostream>
#include <string>

#include "../Util.hpp"
using namespace std;

FileManagement::FileReader::FileReader(string filename) {
 this->filename = filename;
 this->parse();
}

bool isStringSpaceOnly(string str) {
 for (char c : str) {
   if (!isspace(c)) {
     return false;
   }
 }

 return true;
}

void FileManagement::FileReader::parse() {
 ifstream fileStream(this->filename);
 vector<string> tables;
 string line;

 while (getline(fileStream, line) && !isStringSpaceOnly(line)) {
   tables.push_back(line);
 }

 Util::trimVectorString(tables);
 this->charTable = tables;

 while (getline(fileStream, line)) {
   if (!isStringSpaceOnly(line)) this->patterns.push_back(line);
 }

 Util::trimVectorString(this->patterns);
```

```
 fileStream.close();
}

vector<string> FileManagement::FileReader::getTable() {
 return this->charTable;
}

vector<string> FileManagement::FileReader::getPatterns() {
 return this->patterns;
}

void FileManagement::writeTableToFile(ofstream &filestream,
                                      vector<string> charTable,
                                      vector<vector<Color>>
colorTable) {
 if (charTable.size() == 0) {
   filestream << "--- Tabel Kosong ---" << endl;
   return;
 } else {
   filestream << "" << endl;
 }

 for (int i = 0; i < (int)charTable.size(); i++) {
   for (int j = 0; j < (int)charTable[0].length(); j++) {
     if (j > 0) {
       filestream << " ";
     }

     if (colorTable[i][j] == HIDDEN) {
       filestream << "-";
     } else {
       filestream << charTable[i][j];
     }
   }

   filestream << endl;
 }
}
```

Berikut ini adalah kode program utama

```cpp
#include <ctime>
#include <fstream>
#include <iostream>

#include "lib/Matcher.hpp"
#include "lib/io/Console.hpp"
#include "lib/io/File.hpp"

using namespace std;
using namespace FileManagement;

int main(int argc, char* argv[]) {
 Console::printHeader();
 cout << endl;

 string filename;
 if (argc > 1) {
   filename = argv[1];
   cout << "Membaca file " << filename << endl << endl;
 } else {
   cout << "Insert file path : ";
   getline(cin, filename);
 }

 FileReader reader(filename);
 reader.parse();

 clock_t timeStart = clock();

 vector<string> table = reader.getTable();
 vector<string> patterns = reader.getPatterns();

 Matcher::StringMatcher matcher(table);

 for (string i : patterns) {
   matcher.match(i);
 }

 clock_t timeStop = clock();
 vector<vector<Color>> colorTable = matcher.getColorizedResult();
```

```
 Console::printTable(table, colorTable);

 cout << "Waktu Eksekusi : "
      << (1000.0 * (timeStop - timeStart)) / CLOCKS_PER_SEC << " ms"
<< endl
      << endl;

 cout << "Apakah anda ingin menyimpan file hasil pemrosesan [Y/n] ?
";

 string ans;
 getline(cin, ans);

 if (tolower(ans[0]) == 'y') {
   cout << "Masukan path file : ";

   string path;
   getline(cin, path);
   ofstream filestream(path);

   filestream << "-- Hasil Pemrosesan --" << endl;

   writeTableToFile(filestream, table, colorTable);

   filestream << endl;
   filestream << "Waktu Eksekusi : "
              << (1000.0 * (timeStop - timeStart)) / CLOCKS_PER_SEC
<< " ms"
              << endl;

   filestream.close();

   cout << endl << "Berhasil menyimpan data" << endl;
   cout << endl;
 }

 return 0;
}
```

D. Data Status Implementasi

Berikut ini adalah data status dari hasil implementasi program ini

Poin	**Status**
1. Program berhasil dikompilasi tanpa error	Ya
2. Program berhasil run	Ya
3. Program dapat membaca file dan menuliskan luaran	Ya
4. Program berhasil menemukan semua kata di puzzle	Ya, dengan catatan kata harus ada di puzzle

E. Tangkapan Layar Output

Berikut ini adalah tangkapan layar output dan masukan yang digunakan pada program

- Large 1

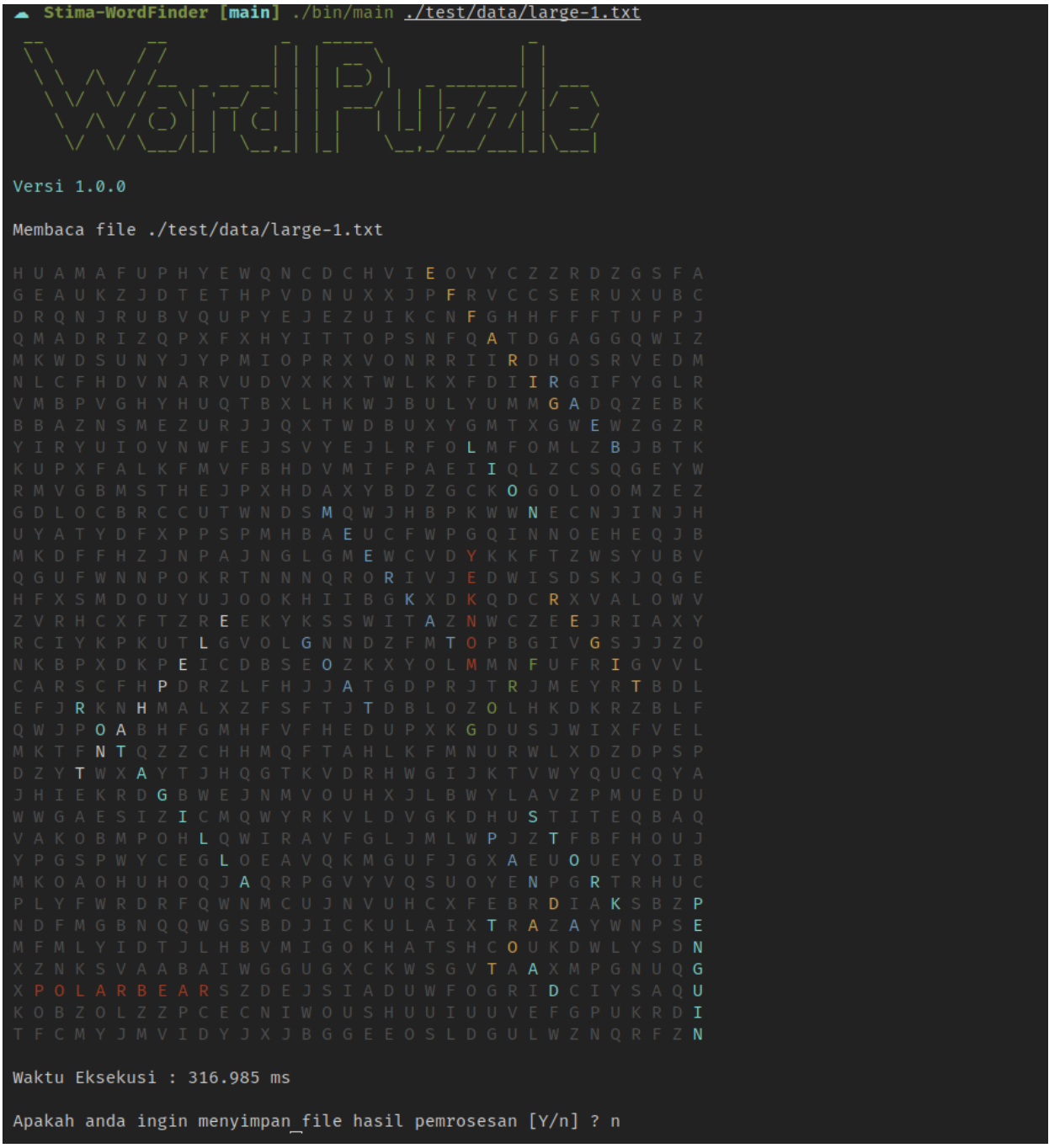
Masukan

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

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

ALLIGATOR
BEAR
ELEPHANT
FROG
GIRAFFE
GOAT
LION
MEERKAT
MONKEY
PANDA
PENGUIN
POLARBEAR
STORK
TIGER
TOAD
```

Tangkapan Layar Keluaran



- Large 2

Masukan

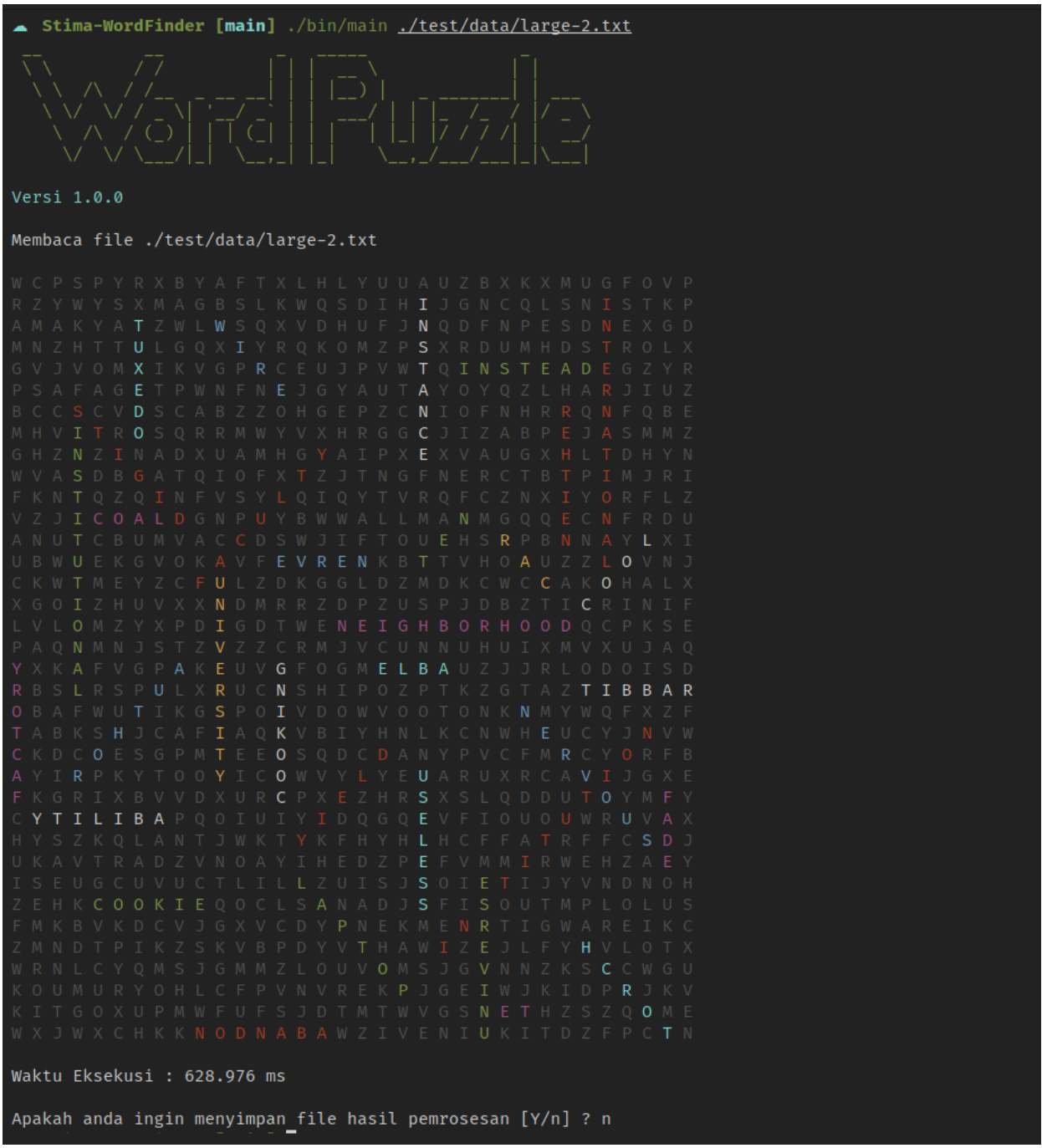
```
W C P S P Y R X B Y A F T X L H L Y U U A U Z B X K X M U G F O V P
R Z Y W Y S X M A G B S L K W Q S D I H I J G N C Q L S N I S T K P
A M A K Y A T Z W L W S Q X V D H U F J N Q D F N P E S D N E X G D
M N Z H T T U L G Q X I Y R Q K O M Z P S X R D U M H D S T R O L X
G V J V O M X I K V G P R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

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

ABANDON
ABILITY
ABLE
AUTHOR
COOKIE
COOKING
COOL
FACTORY
FACULTY
FADE
INSTANCE
INSTEAD
INSTITUTION
INSTITUTIONAL
NEIGHBORHOOD
NEITHER
NERVE
NERVOUS
NET
YIELD
WIRE
DIGITS
UNIVERSE
UNIVERSITY
USELESS

```
INTERNATIONAL
LAPTOP
CAR
TUXEDO
RABBIT
COAL
TORCH
```

Keluaran



- Large 3

Input

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

AOEDE
ARISTOTLE
ATLAS
CALYPSO
CARPO
CENTAURUS
CERES
COMET
DIONE
EARTH
ERIS
EUROPA
EXOPLANETS
GANYMEDE
```

```
GONGGONG
HAUMEA
HEGEMONE
HELENE
IJRAQ
IO
JANUS
JUPITER
KALE
MAKEMAKE
MARS
MERCURY
METHONE
MEYER
MIRANDA
MOON
NEBULA
NEPTUNE
ORCUS
ORLANDO
PAALIAQ
PALLENE
PANDORA
PLANET
PLUTO
PROMETHEUS
QUAOAR
SALACIA
SATURN
SEDNA
SOLAR
STAR
SUN
TITAN
URANUS
VENUS
```

Output

- **Medium 1**

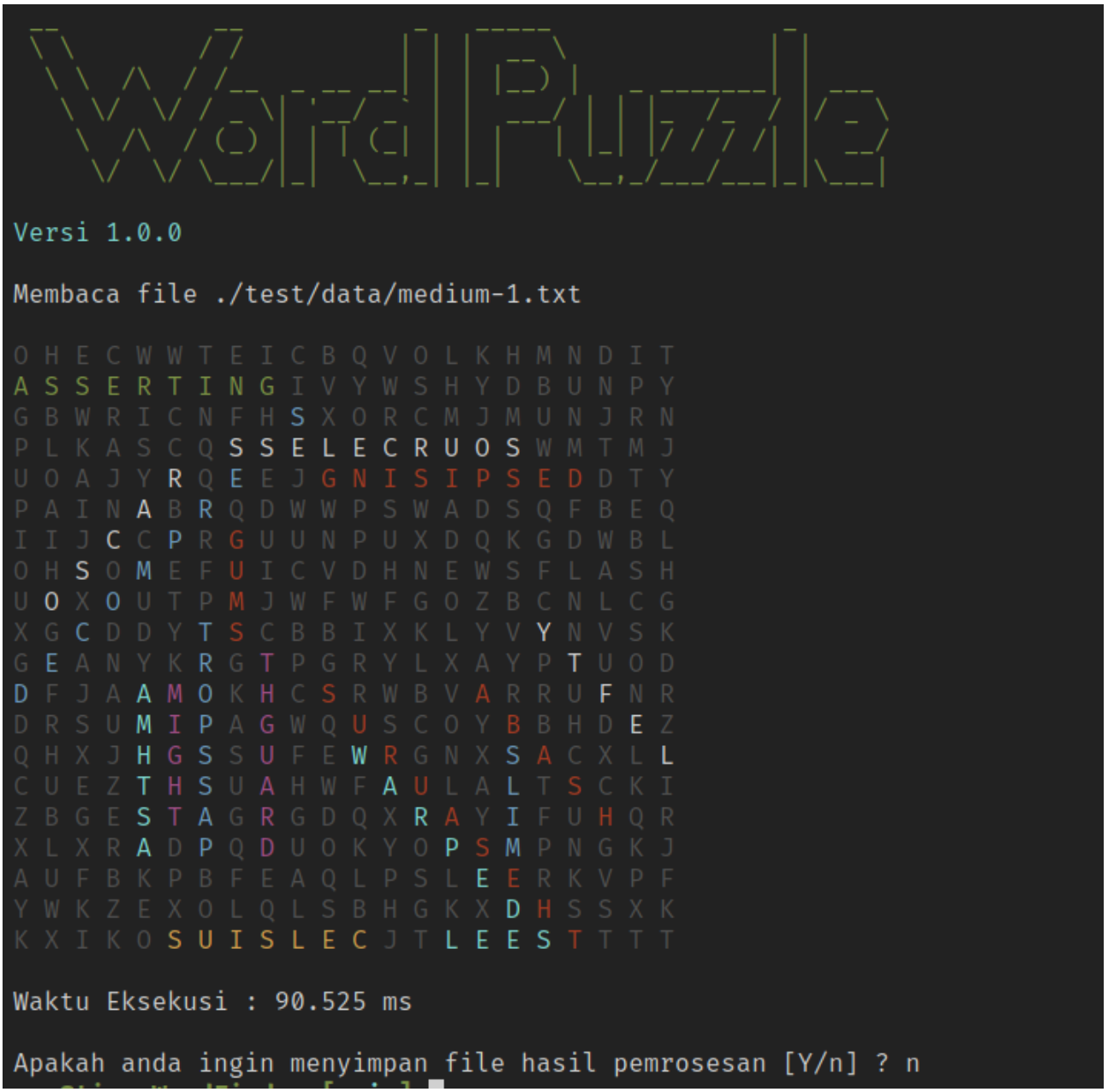
  Input

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

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

ABASH
ASSERTING
ASTHMA
CELSIUS
DECOMPRESS
DESPISING
DRAUGHT
GLANCELAUGH
LEES
LEFTY
MIGHT
NEWSDLASH
OSCAR
PASSPORT
SLIME
SMUG
SOURCELESS
THESAURUS
WARPED
```

Output



- **Medium 2**

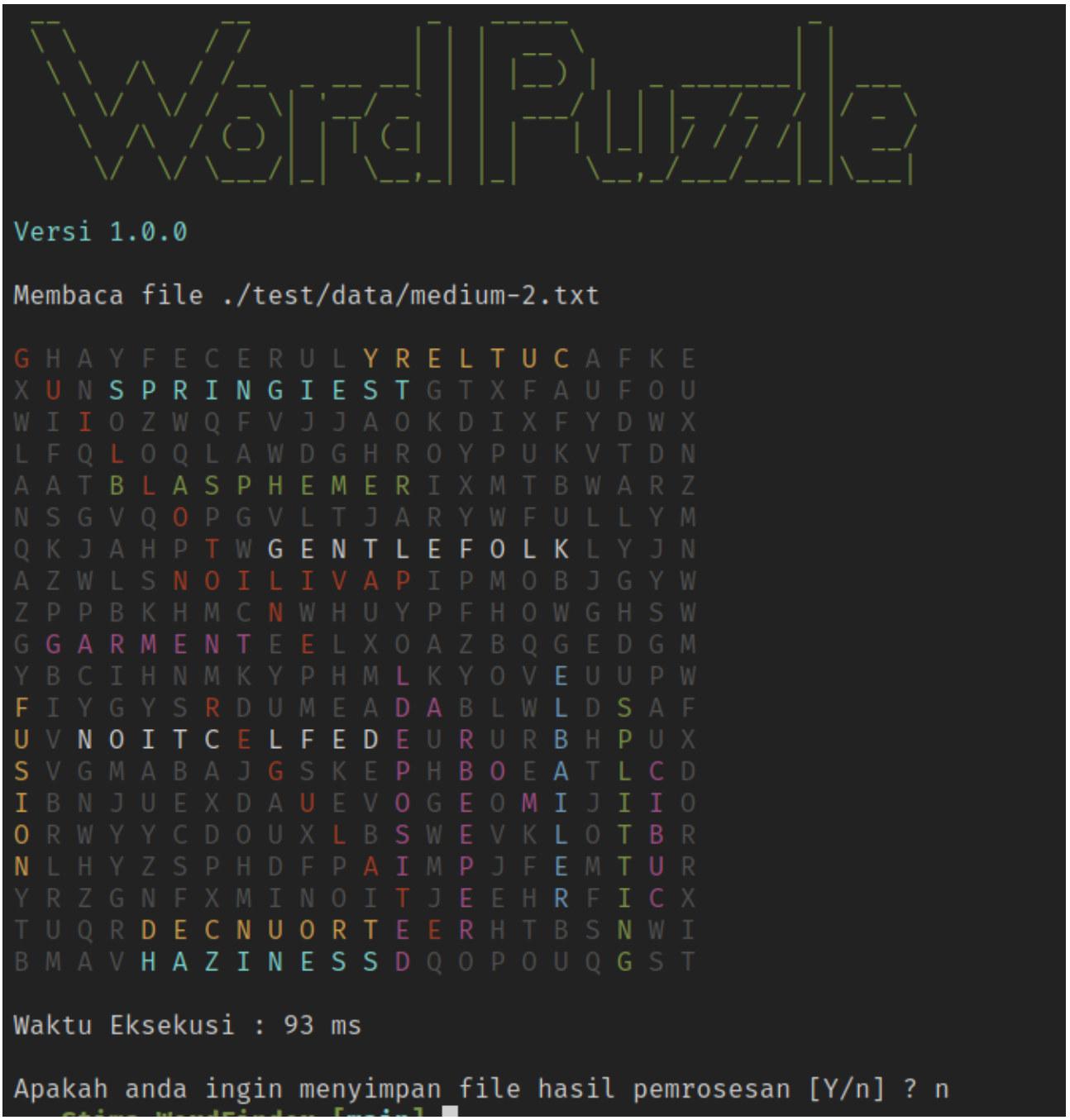
  Input

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

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

BEEPER
BLASPHEMER
CUBIC
CUTLERY
DEFLECTION
DEPOSITED
FUSION
GARMENT
GENTLEFOLK
GUILLOTINE
HAYFEVER
HAZINESS
MORAL
PAVILION
REGULATE
RELIABLE
RUEFULLY
SPLITTING
SPRINGIEST
TROUNCED
```

Output



- **Medium 3**

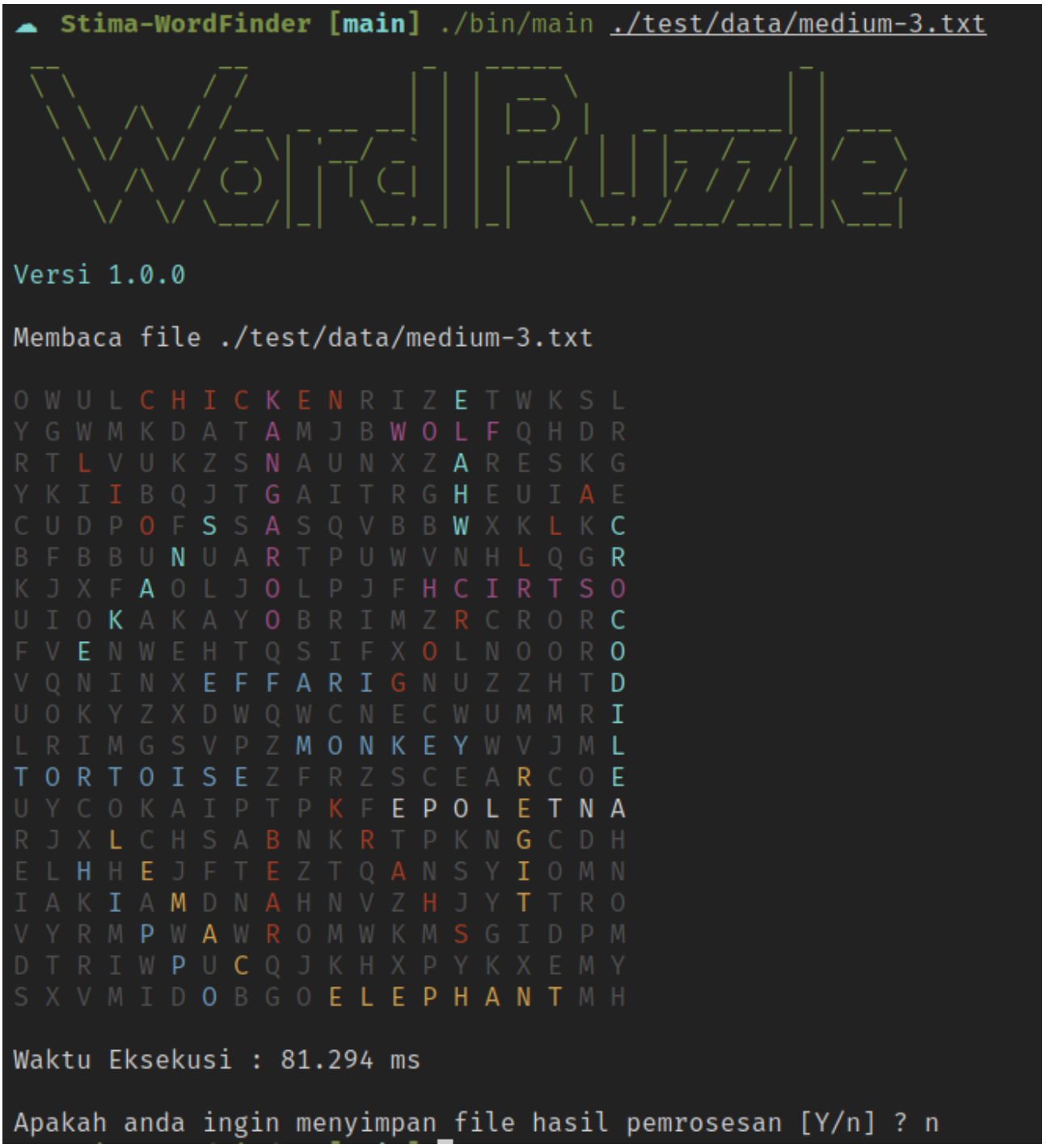
  Input

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

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

ANTELOPE
BEAR
CAMEL
CHICKEN
CROCODILE
ELEPHANT
GIRAFFE
GORILLA
HIPPO
KANGAROO
LION
MONKEY
OSTRICH
SHARK
SNAKE
TIGER
TORTOISE
CULTURE
WHALE
WOLF
```

Output



- **Small 1**

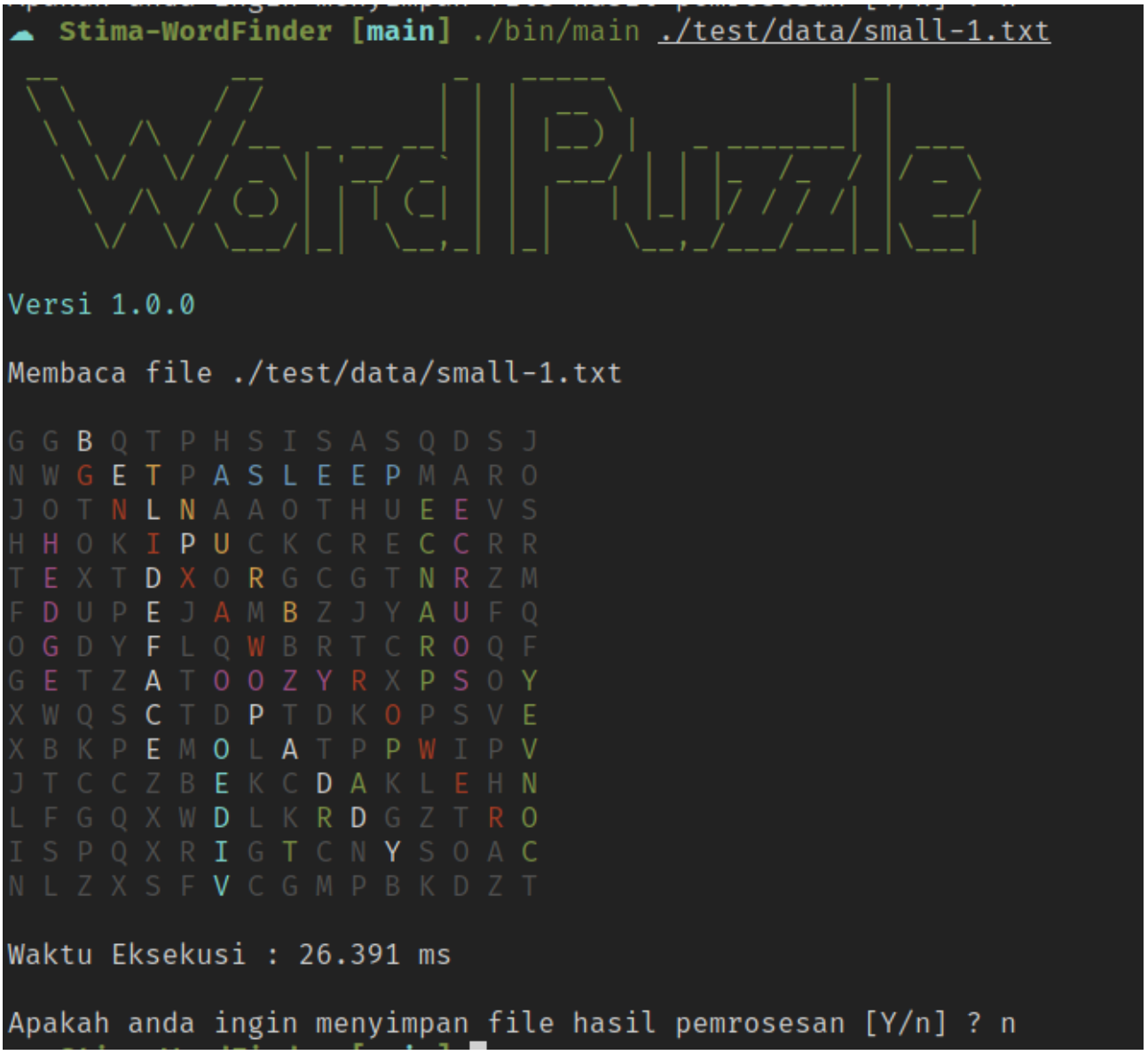
  Input

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

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

ASLEEP
PART
BRUNT
PLEB
CONVEY
PRANCE
DEFACE
ROWER
HEDGE
SOURCE
OOZY
VIDEO
PADDY
WAXING
```

Output



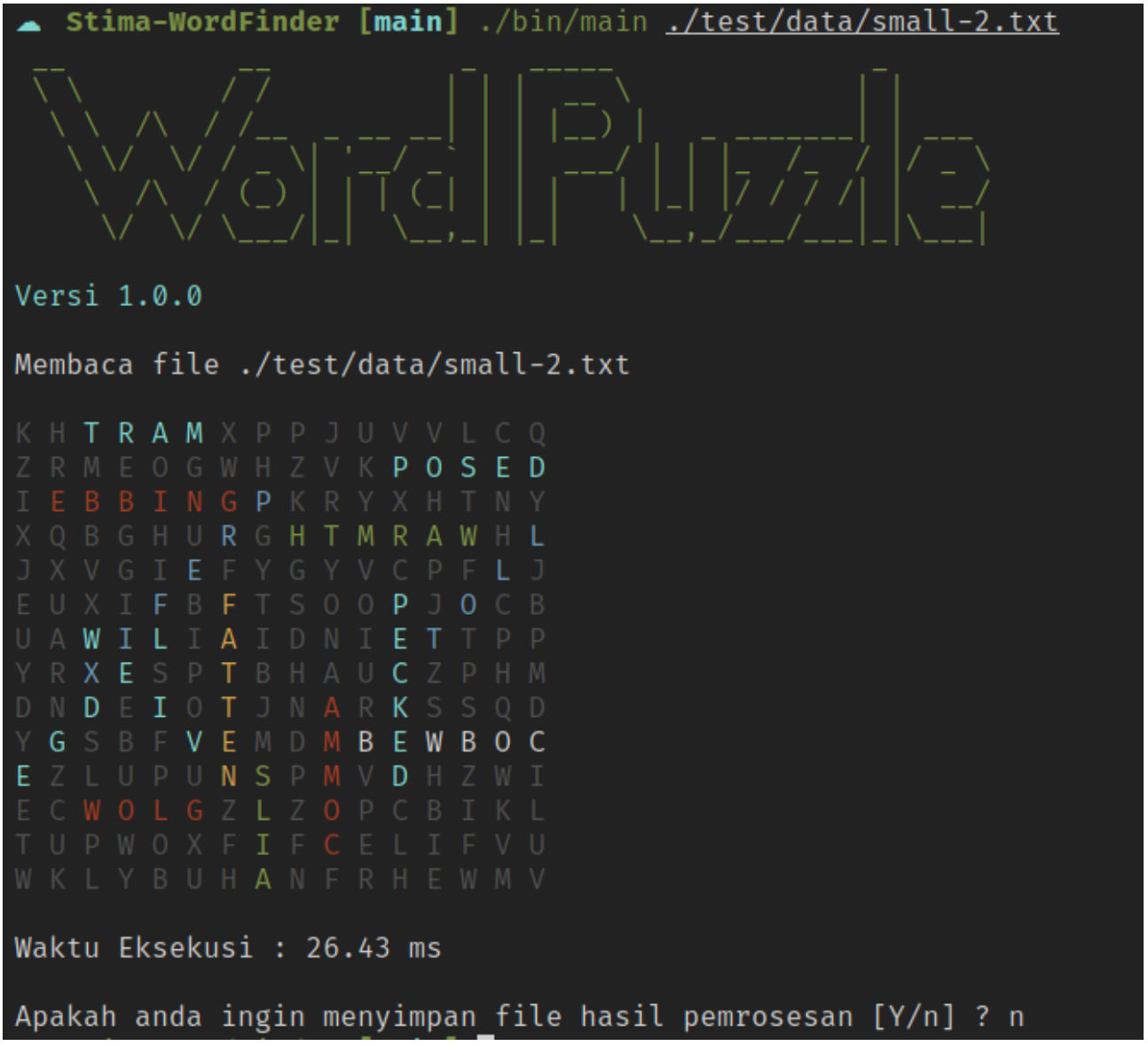
- **Small 2**

  Input

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

```
AILS
MART
COBWEB
PECKED
COMMA
POSED
EBBING
PREFIX
FATTEN
TOLL
GLOW
VIEW
LEDGE
WARMTH
```

Output



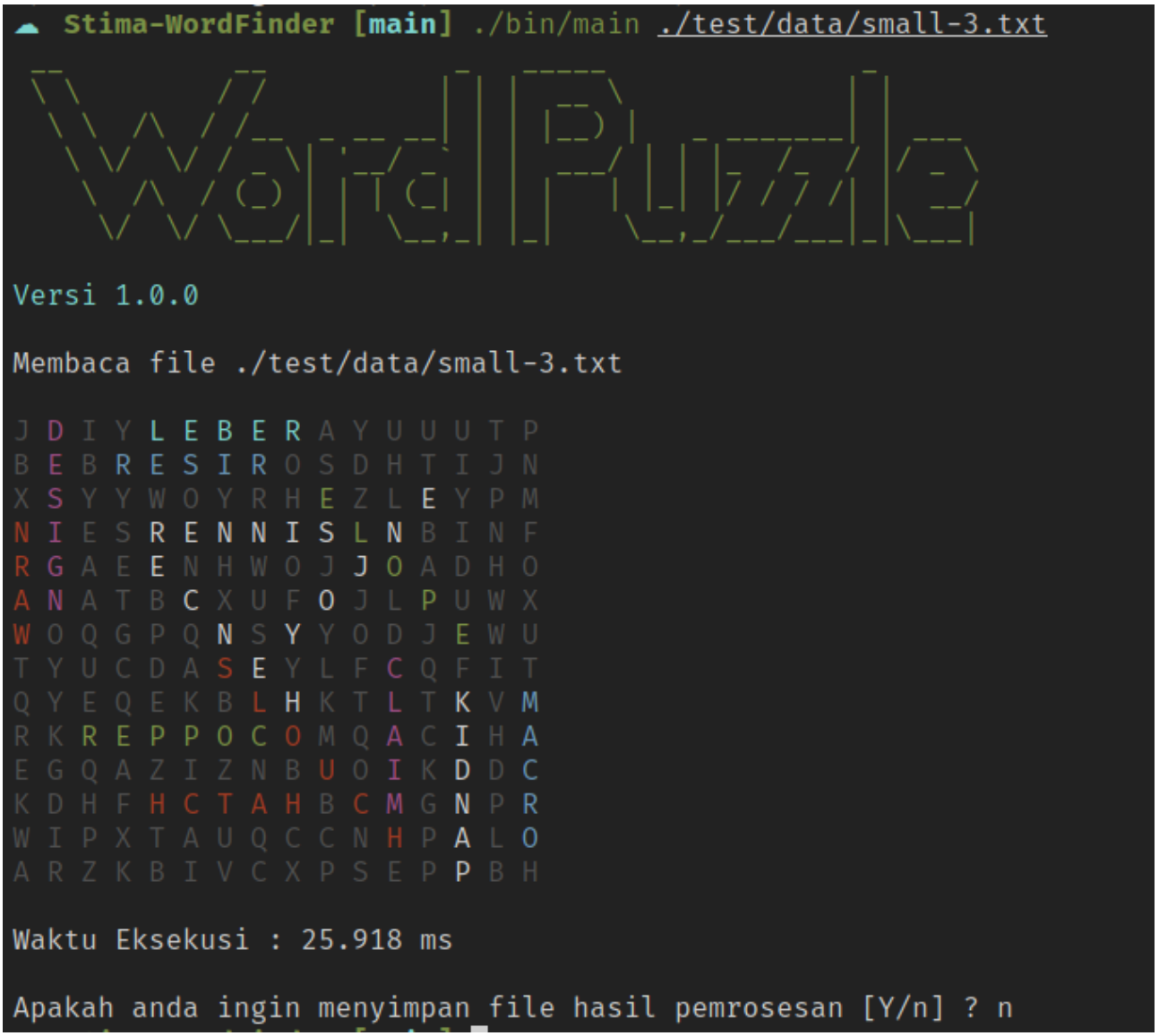
- **Small 3**

Input

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

CLAIM
KIDNAP
COPPER
MACRO
DESIGN
REBEL
ELOPE
RISER
ENJOY
SINNER
HATCH
SLOUCH
HENCE
WARN
```

Output



F. Tautan Penting

Berikut ini adalah tautan penting yang berisi kode sumber program

https://github.com/bayusamudra5502/Stima-WordFinder