LAPORAN TUGAS KECIL 1

Penyelesaian Word Search Puzzle dengan Algoritma Brute Force

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma (Stima)
pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

Saul Sayers (K1) **13520094**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

A. Algoritma *brute force*

Algoritma *brute force* adalah penyelesaian suatu masalah dengan pendekatan yang sederhana, langsung, jelas, dan mudah dipahami. Dengan algoritma *brute force*, penyelesaian suatu masalah cenderung dilakukan dengan mencari dan meninjau semua kasus yang ada. Dengan demikian, algoritma *brute force* dapat dikatakan sebagai algoritma yang lempang (*straight-forward)*. Kelebihan dari algoritma ini adalah cukup mudah untuk dipahami dan dapat diterapkan untuk hampir semua permasalahan komputasi. Akan tetapi, kekurangan dari algoritma ini adalah tidak efisien karena membutuhkan langkah yang banyak dalam penyelesaiannya sehingga membutuhkan waktu yang cukup lama.

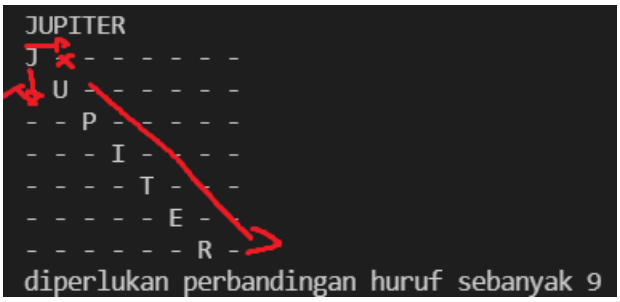
Dalam Tugas Kecil ini, digunakan algoritma *brute force* dalam implementasi penyelesaian *word search puzzle*. Sebelum menerapkan algoritma *brute force*, program pertama – tama akan membaca input dari sebuah file txt pada folder test. Isi dari file tersebut adalah sebuah *word search puzzle* berbentuk segiempat kemudian diikuti oleh sebuah baris kosong dan dilanjutkan oleh beberapa keywords yang akan dicari dalam puzzle tersebut. Program akan membaca tiap karakter dari puzzle lalu menyimpannya dalam bentuk matriks, sementara untuk daftar keywords akan dibaca per-line dan disimpan dalam sebuah array of string (dengan mengabaikan newline pada akhir tiap keyword). Berikut adalah langkah algoritma *brute force* yang digunakan dalam program yang telah saya buat.

Pertama, program akan mengiterasikan dan mencari tiap keyword secara satu per satu, dimulai dengan keyword teratas yang terdapat pada file input hingga keyword terbawah yang terdapat pada file input. Untuk tiap keyword yang dicari, program akan mencari terlebih dahulu huruf pertama keyword tersebut dalam puzzle. Pencarian huruf pertama dilakukan secara iterasi while untuk tiap kolom di tiap baris pada matriks, dimulai dari kolom dan baris pertama sehingga menggunakan nested loop (j untuk indeks kolom dan i untuk indeks baris).

Kedua, Apabila sudah ditemukan huruf pertama dari keyword tersebut, maka akan diperiksa 8 arah dari huruf tersebut yakni atas, bawah, kiri, kanan, tenggara, timur laut, barat laut, dan barat daya (dengan urutan tersebut) untuk sisa hurufnya, huruf pertama tidak perlu dicek ulang agar efisien dan tidak redundant. Apabila ditemukan sebuah arah yang dapat membentuk keywords, maka arah lain juga tidak perlu dicek. Pengecekan arah tersebut bermaksud untuk memeriksa apakah dari huruf pertama tersebut terdapat suatu arah yang membentuk keyword yang dicari. Pengecekan dilakukan dengan looping while dan berhenti apabila telah melakukan iterasi sebanyak panjang keyword yang ingin dicari, atau berhenti saat ditemukan satu huruf yang salah (seperti algoritma searching). Apabila terdapat satu huruf pun yang tidak sesuai dengan keyword, maka program akan berhenti mencari di arah tersebut kemudian mencoba pada arah lain. Perlu dicatat bahwa pengecekan arah memiliki constraint tersendiri, yakni apabila panjang kata melebihi sisa huruf pada arah yang dicari maka tidak perlu diperiksa. Hal tersebut dilakukan agar pencarian lebih efektif dan tidak sampai out of bounds.

Ketiga, program mengecek apakah terdapat suatu arah yang dapat membentuk keyword tersebut dari huruf pertama yang sudah didapat sebelumnya. Apabila keyword terbentuk, maka program akan langsung mencetak matriks yang berisi jawaban untuk keyword tersebut, dengan elemen lain yang tidak bersangkutan dicetak dengan karakter '-' beserta banyaknya perbandingan huruf yang dilakukan untuk tiap keywords dan langsung berada di langkah keempat (pencarian keyword lain). Apabila keyword tidak terbentuk, maka program akan lanjut mencari huruf pertama dari keyword tersebut pada puzzle selain dari huruf pertama yang telah ditemukan tadi. Langkah kedua dan ketiga akan dilakukan lagi pada huruf pertama yang baru hingga ditemukan sebuah solusi atau pencarian huruf pada matriks sudah dilakukan hingga akhir. Perlu dicatat juga perbandingan huruf untuk tiap keywords, baik saat keyword ditemukan ataupun tidak ditemukan, akan dijumlahkan di sebuah variabel total agar dapat dilihat pula di akhir banyaknya perbandingan huruf selama eksekusi program.

Keempat, program akan berada di tahap selesai pencarian untuk sebuah keyword, baik karena sudah ditemukan solusinya ataupun tidak ditemukan tetapi sudah berada di akhir matriks puzzle (semua kasus sudah diperiksa). Program akan kembali lagi ke langkah pertama untuk keyword lain yang akan dicari solusinya. Berikut adalah ilustrasi proses pencarian sebuah keyword dalam program :



Gambar 1.1 Ilustrasi proses pencarian solusi keyword dalam program

Karena dicari keyword JUPITER, maka program akan mencari huruf j pertama yang muncul kemudian memeriksa arahnya. Kebetulan huruf J berada di elemen pertama matriks sehingga hanya membutuhkan 1 step. Program mengecek ke arah kanan terlebih dahulu, dan ternyata salah sehingga tidak perlu dilanjutkan dan butuh hanya 1 step. Demikian juga berlaku pada arah bawah. Untuk arah kiri,atas, atas kiri, bawah kiri,atas kanan tidak perlu dicek karena proses pencarian out of bounds (sebagaimana constraint sudah dijelaskan di paragraf sebelumnya). Saat pengecekan arah bawah kanan, ditemukan sisa huruf UPITER sehingga membutuhkan 6 step. Total step yang didapat adalah 1 untuk pencarian huruf pertama, 1 dari arah kanan, 1 dari arah bawah, dan 6 dari arah bawahkanan sehingga totalnya membutuhkan 9 step.

B. *Source Program*

Source code program ditulis dalam bahasa pemrograman C dan dibungkus hanya dalam 1 file main.c. Berikut adalah source code yang terdapat dalam file main.c

```c
//Nama : Saul Sayers
//NIM  : 13520094
//Kelas: K01 Stima
//Tucil 1 STIMA - Word Search Puzzle Solver

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include "boolean.h"

typedef struct {
    char contents[50][50];
    int rowEff;
    int colEff;
} Matrix;

typedef struct {
    char contents[50][50];
    int Neff;
} Keywords;

/* *** Selektor *** */
#define ROWS(M) (M).rowEff
#define COLS(M) (M).colEff
#define ELMT(M, i, j) (M).contents[(i)][(j)]
#define NEFF(K) (K).Neff
#define ELMTK(K,i) (K).contents[(i)]

void displayMatrix(Matrix m){
    int i, j;
    for (i = 0; i < ROWS(m); i++){
        for (j = 0; j < COLS(m); j++){
            if (j==0) {
                printf("%c", ELMT(m,i,j));
            }
            else {
                printf(" %c", ELMT(m,i,j));
            }
        }
        if (i != ROWS(m) -1) {
            printf("\n");
        }
    }
}
```

```
void readinput(Matrix *crossword, Matrix *hasil, Keywords *keys){
    //Reset crossword dan daftar Keywords
    ROWS(*crossword) = 0;
    COLS(*crossword) = 0;
    NEFF(*keys) = 0;
    ROWS(*hasil) = 0;
    COLS(*hasil) = 0;
    FILE *tape;

    //bagian input nama file hingga benar
    printf("Silahkan input nama file: ");
    char namafile[70] ;
    fflush(stdin); //agar tidak ada error bug di input sebelumnya
    gets(namafile);
    char direktori[80];
    strcpy(direktori, "../test/");
    strcat(direktori, namafile);
    tape = fopen(direktori, "r");
    while (tape == NULL)
    {
        printf("Nama file tidak ditemukan\nSilahkan input ulang nama file: ");
        strcpy(direktori, "../test/");
        gets(namafile);
        strcat(direktori, namafile);
        tape = fopen(direktori, "r");
    }

    //bagian konversi isi file ke matrix crossword
    printf("Word Search Puzzle-nya adalah: \n");
    char c = fgetc(tape);
    char csebelum = c;
    int baris = 0, kolom = 0;
    while ((c != '\n') || (csebelum != '\n')){
        putchar(c);
        if (c == '\n') {
            ROWS(*crossword) ++;
            baris++;
            COLS(*crossword) = kolom;
            kolom = 0;
        }
        if (c != ' ' && c != '\n') {
            ELMT(*crossword,baris,kolom) = c;
            kolom++;
        }
```

```
        csebelum = c;
        c = fgetc(tape);
    }
    //bagian konversi isi file ke keywords
    char line[100];
    while (fgets(line, sizeof(line), tape)){
        strcpy(ELMTK(*keys,NEFF(*keys)), line);
        NEFF(*keys) ++;
    }


    //bagian menghapus newline dari tiap keywords
    int i,j;
    for (i=0; i<NEFF(*keys)-1; i++){
        ELMTK(*keys,i)[strlen(ELMTK(*keys,i))-1] = '\0';
    }

    //INISIALISASI MATRIKS HASIL BUAT PERCETAKAN
    ROWS(*hasil) = ROWS(*crossword);
    COLS(*hasil) = COLS(*crossword);
    for (i=0; i<ROWS(*hasil);i++){
        for (j=0; j<COLS(*hasil); j++) {
            ELMT(*hasil, i,j) = '-' ;
        }
    }

    //bagian menutup tape
    fclose(tape);
}

void checkright(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (j<COLS(crossword) - panjang + 1)){
        boolean betul = true;
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i,j+1) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            j++;
            (*perbandingan) ++ ;
        }
        if (betul) {
```

```
            j = j - panjang + 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i,j+bebas) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i,j+bebas) = '-';
            }
        }
    }
}

void checkdown(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (i < ROWS(crossword) - panjang + 1)){
        boolean betul = true;
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i+1,j) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            i++;
            (*perbandingan) ++ ;
        }
        if (betul) {
            i = i - panjang + 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i+bebas,j) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i+bebas,j) = '-';
            }
        }
    }
```

```
}

void checkleft(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (j>panjang -2)){
        boolean betul = true;
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i,j-1) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            j--;
            (*perbandingan) ++ ;
        }
        if (betul) {
            j = j + panjang - 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i,j-bebas) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i,j-bebas) = '-';
            }
        }
    }
}

void checkup(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (i>panjang -2)){
        boolean betul = true;
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i-1,j) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            i--;
```

```
            (*perbandingan) ++ ;
        }
        if (betul) {
            i = i + panjang - 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i-bebas,j) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i-bebas,j) = '-';
            }
        }
    }
}

void checkdownright(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (j<COLS(crossword) - panjang + 1) && (i < ROWS(crossword) - panjang + 1)){
        boolean betul = true;
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i+1,j+1) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            j++;
            i++;
            (*perbandingan) ++ ;
        }
        if (betul) {
            j = j - panjang + 1;
            i = i - panjang + 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i+bebas,j+bebas) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
```

```
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i+bebas,j+bebas) = '-';
            }
        }
    }
}

void checkdownleft(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (j>panjang -2) && (i < ROWS(crossword) - panjang + 1)){
        boolean betul = true;
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i+1,j-1) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            j--;
            i++;
            (*perbandingan) ++ ;
        }
        if (betul) {
            j = j + panjang - 1;
            i = i - panjang + 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i+bebas,j-bebas) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i+bebas,j-bebas) = '-';
            }
        }
    }
}

void checkupright(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (j<COLS(crossword) - panjang + 1) && (i>panjang -2)){
        boolean betul = true;
```

```
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i-1,j+1) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            j++;
            i--;
            (*perbandingan) ++ ;
        }
        if (betul) {
            j = j - panjang + 1;
            i = i + panjang - 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i-bebas,j+bebas) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i-bebas,j+bebas) = '-';
            }
        }
    }
}

void checkupleft(int a, int i, int j, int *perbandingan, Matrix crossword, Matrix hasil, Keywords keys,
boolean *ketemu){
    int panjang = strlen(ELMTK(keys,a));
    if (!(*ketemu) && (j>panjang -2) && (i>panjang -2)){
        boolean betul = true;
        int k = 1;
        while (betul && k < panjang ){
            if (ELMT(crossword,i-1,j-1) != ELMT(keys,a,k)){
                betul = false;
            }
            k++;
            j--;
            i--;
            (*perbandingan) ++ ;
        }
        if (betul) {
            j = j + panjang - 1;
```

```
            i = i + panjang - 1;
            int bebas ;
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i-bebas,j-bebas) = ELMT(keys,a,bebas);
            }
            *ketemu = true;
            printf("%s\n", ELMTK(keys,a));
            displayMatrix(hasil);
            printf("\n");
            for (bebas=0; bebas< panjang; bebas ++) {
                ELMT(hasil,i-bebas,j-bebas) = '-';
            }
        }
    }
}

void solvePuzzle(Matrix *crossword, Matrix *hasil, Keywords *keys){
    printf("\nSolusi dari puzzle adalah: \n\n");
    int a,i,j;
    int perbandingan=0, perbandingansebelum=0;
    int selisih = 0;
    boolean ketemu;
    for (a=0;a<NEFF(*keys);a++) {
        i = 0;
        j = 0;
        ketemu = false;
        while(i < ROWS(*crossword) && !ketemu){
            while (j < COLS(*crossword) && !ketemu){
                perbandingan ++ ;
                if (ELMT(*crossword,i,j) == ELMT(*keys,a,0)) {
                    checkright(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                    checkdown(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                    checkleft(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                    checkup(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                    checkdownright(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                    checkdownleft(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                    checkupright(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                    checkupleft(a,i,j,&perbandingan,*crossword, *hasil, *keys,&ketemu);
                }
                j++;
            }
            i++;
            j=0;
        }
        i=0;
```

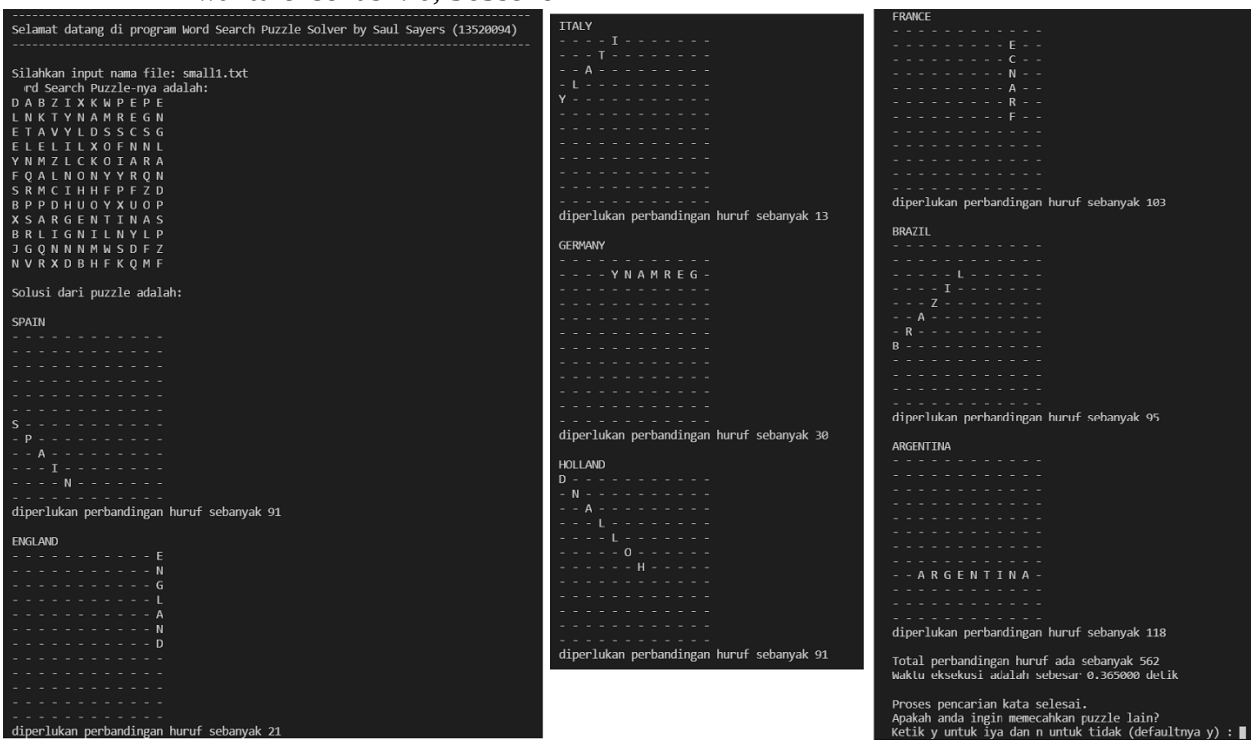
```
        if (!ketemu) {
            printf("%s tidak ditemukan dalam puzzle\n\n", ELMTK(*keys,a));
        }
        else {
            printf("diperlukan perbandingan huruf sebanyak %d\n\n", perbandingan - perbandingansebelum);
        }
        perbandingansebelum = perbandingan;
    }
    printf("Total perbandingan huruf ada sebanyak %d\n", perbandingan);

}

int main(){
    boolean kelar = false;
    char opsi;
    clock_t waktu;
    double waktueksekusi;
    Matrix crossword;
    Matrix hasil;
    Keywords keys;
    printf("--------------------------------------------------------------------------------\n");
    printf("Selamat datang di program Word Search Puzzle Solver by Saul Sayers (13520094)\n");
    printf("--------------------------------------------------------------------------------\n\n");
    while (!kelar){
        readinput(&crossword, &hasil, &keys);
        waktu = clock();
        solvePuzzle(&crossword, &hasil, &keys);
        waktu = clock() - waktu;
        waktueksekusi = ((double)waktu)/CLOCKS_PER_SEC;
        printf("Waktu eksekusi adalah sebesar %f detik\n\n", waktueksekusi);
        printf("Proses pencarian kata selesai.\nApakah anda ingin memecahkan puzzle lain?\nKetik y untuk
iya dan n untuk tidak (defaultnya y) : ");
        scanf(" %c",&opsi);
        printf("--------------------------------------------------------------------------------\n");
        if (opsi == 'n' ){
            kelar = true;
            printf("Terimakasih sudah menggunakan program Word Search Puzzle Solver by Saul Sayers
(13520094)\n");
            printf("--------------------------------------------------------------------------------\n\n");
        }
    }
    return 0;
}
```

C. Screenshots *input* dan *output*

a. Testcase ukuran small

1. **small1.txt**

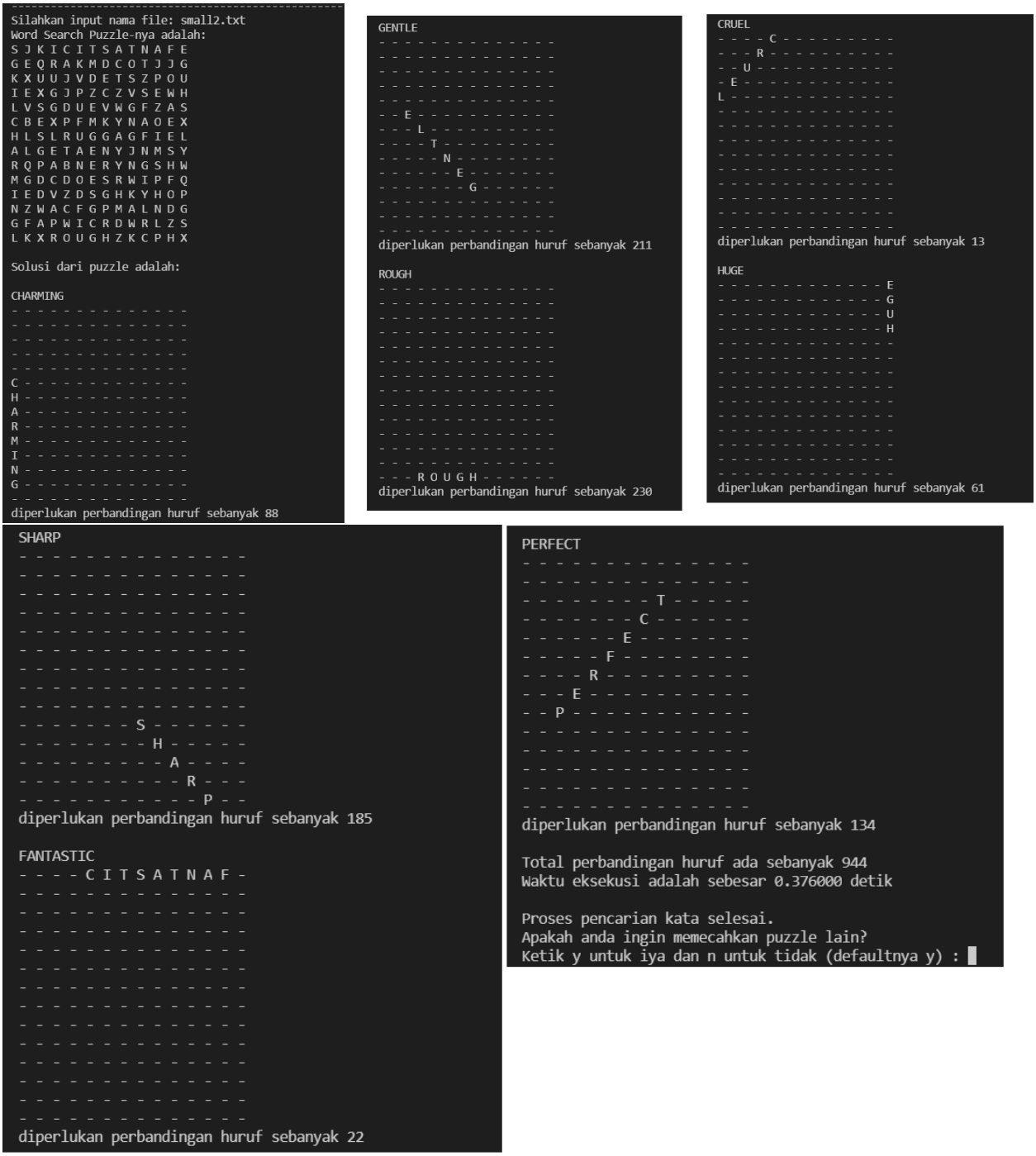
   ukuran puzzle : 14 baris x 12 kolom
   banyaknya keywords : 8 kata
   total perbandingan huruf : 562 karakter
   waktu eksekusi : 0, 365sekon



Gambar 3.1 hasil output untuk testcase small1.txt

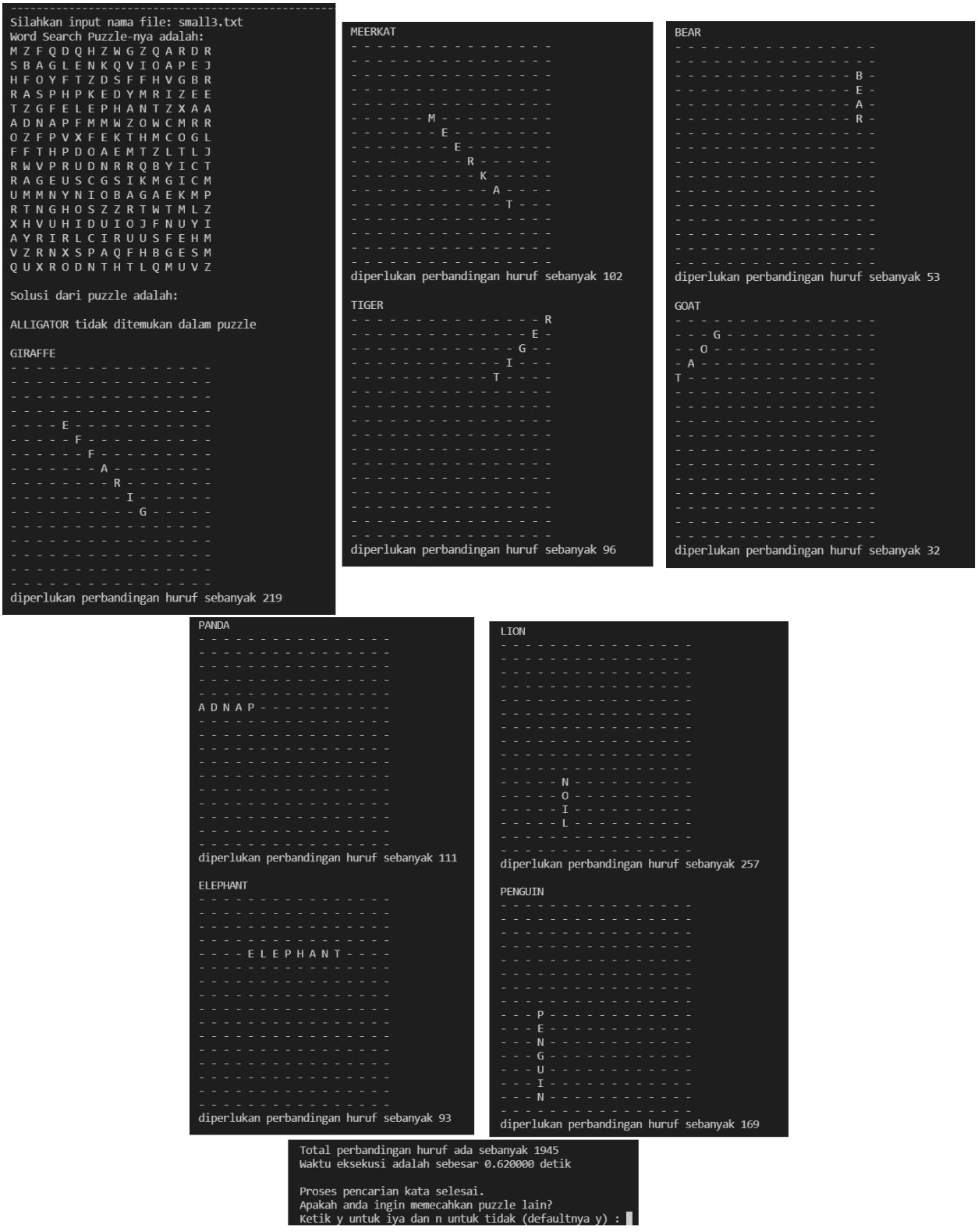
2. **small2.txt**

   ukuran puzzle : 16 baris x 14 kolom
   banyaknya keywords : 8 kata
   total perbandingan huruf : 944 karakter
   waktu eksekusi : 0,374 sekon



Gambar 3.2 hasil output untuk testcase small2.txt

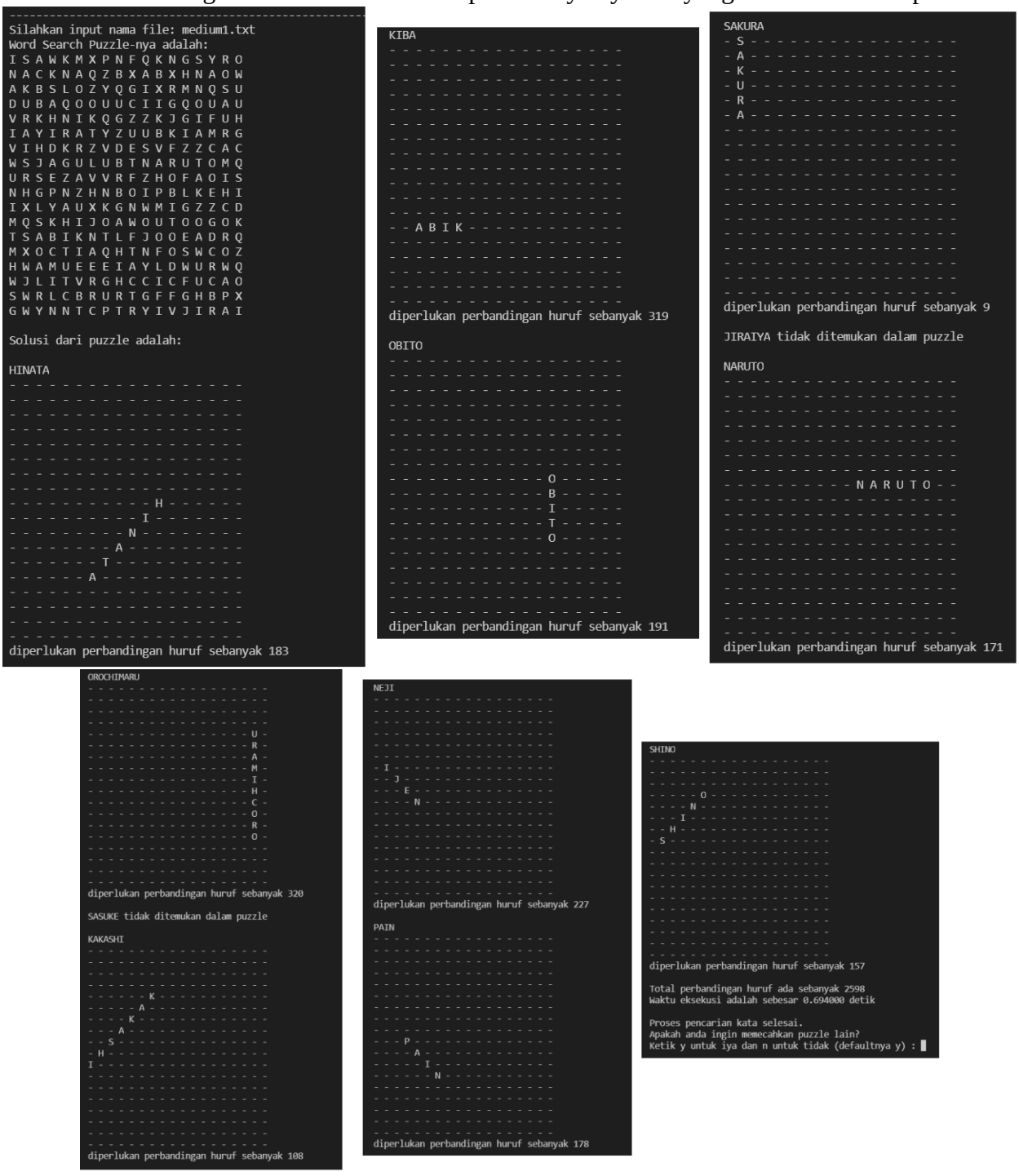
3. **small3.txt**
   ukuran puzzle : 18 baris x 16 kolom
   banyaknya keywords : 10 kata
   total perbandingan huruf : 1945 karakter
   waktu eksekusi : 0,62 sekon



Gambar 3.3 hasil output untuk testcase small3.txt

b. Testcase ukuran medium

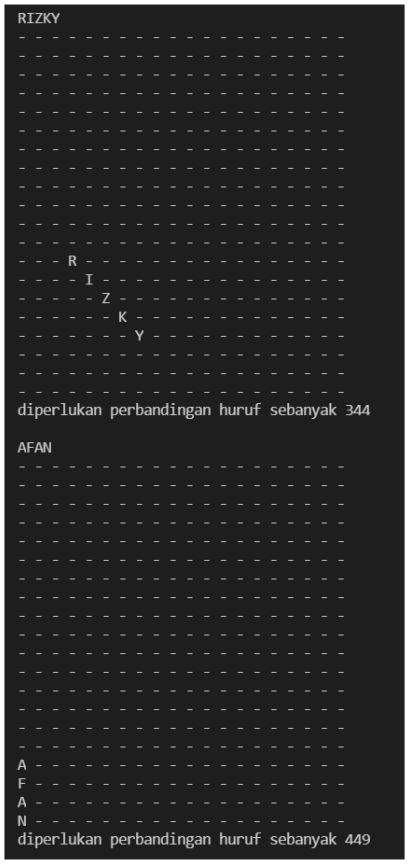
1. **<u>medium1.txt</u>**
   ukuran puzzle : 20 baris x 18 kolom
   banyaknya keywords : 12 kata
   total perbandingan huruf : 2598 karakter
   waktu eksekusi : 0.634 sekon
   Dengan catatan dibuat beberapa dummy keyword yang tidak ada dalam puzzle

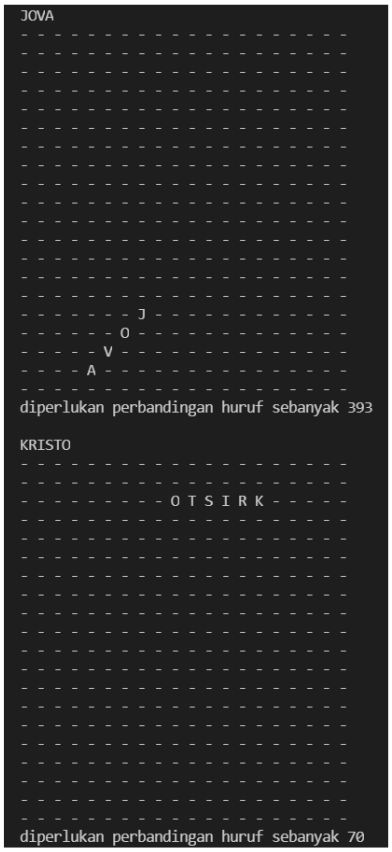


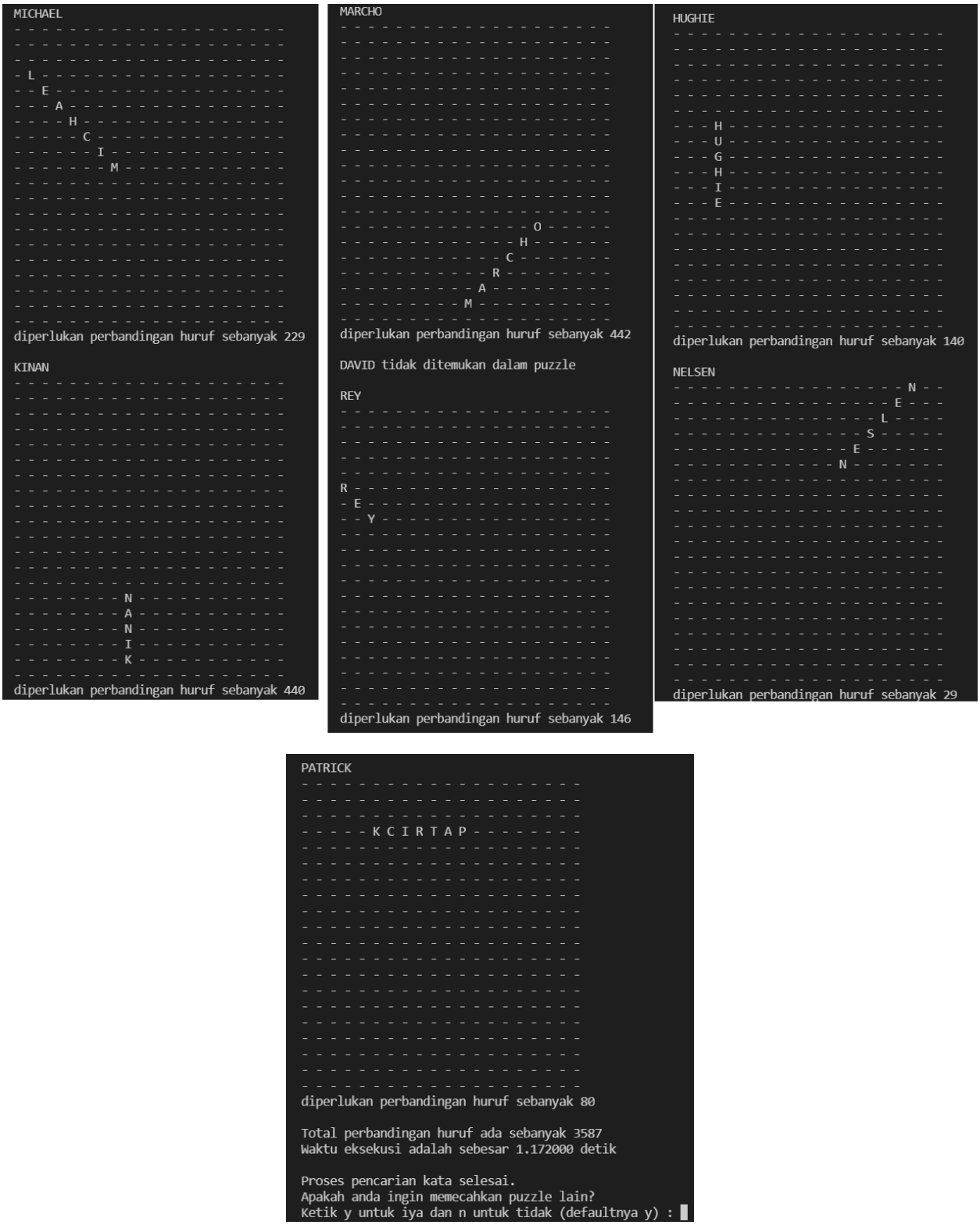
Gambar 3.4 hasil output untuk testcase medium1.txt

2. **medium2.txt**

ukuran puzzle : 22 baris x 20 kolom
banyaknya keywords : 13 kata
total perbandingan huruf : 3587 karakter
waktu eksekusi : 1,172 sekon
Dengan catatan dibuat beberapa dummy keyword yang tidak ada dalam puzzle

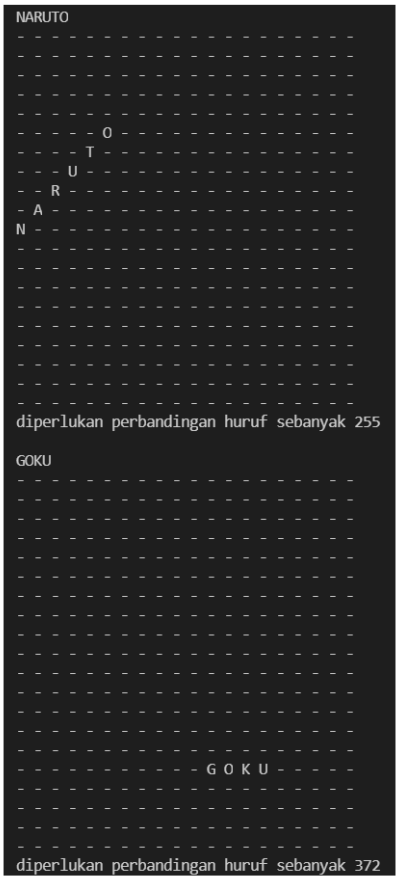


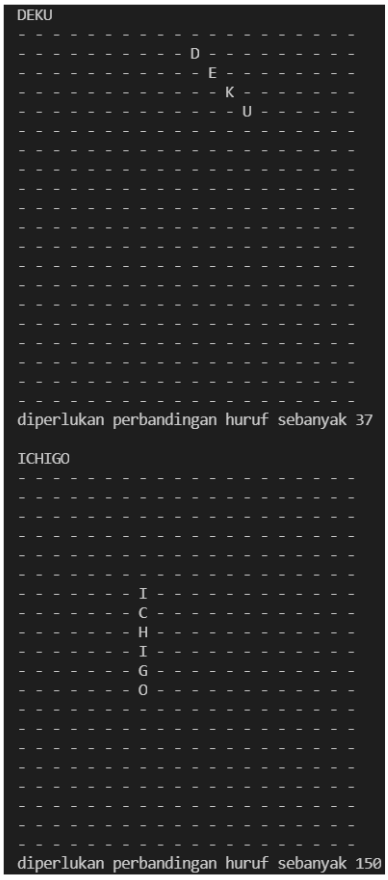


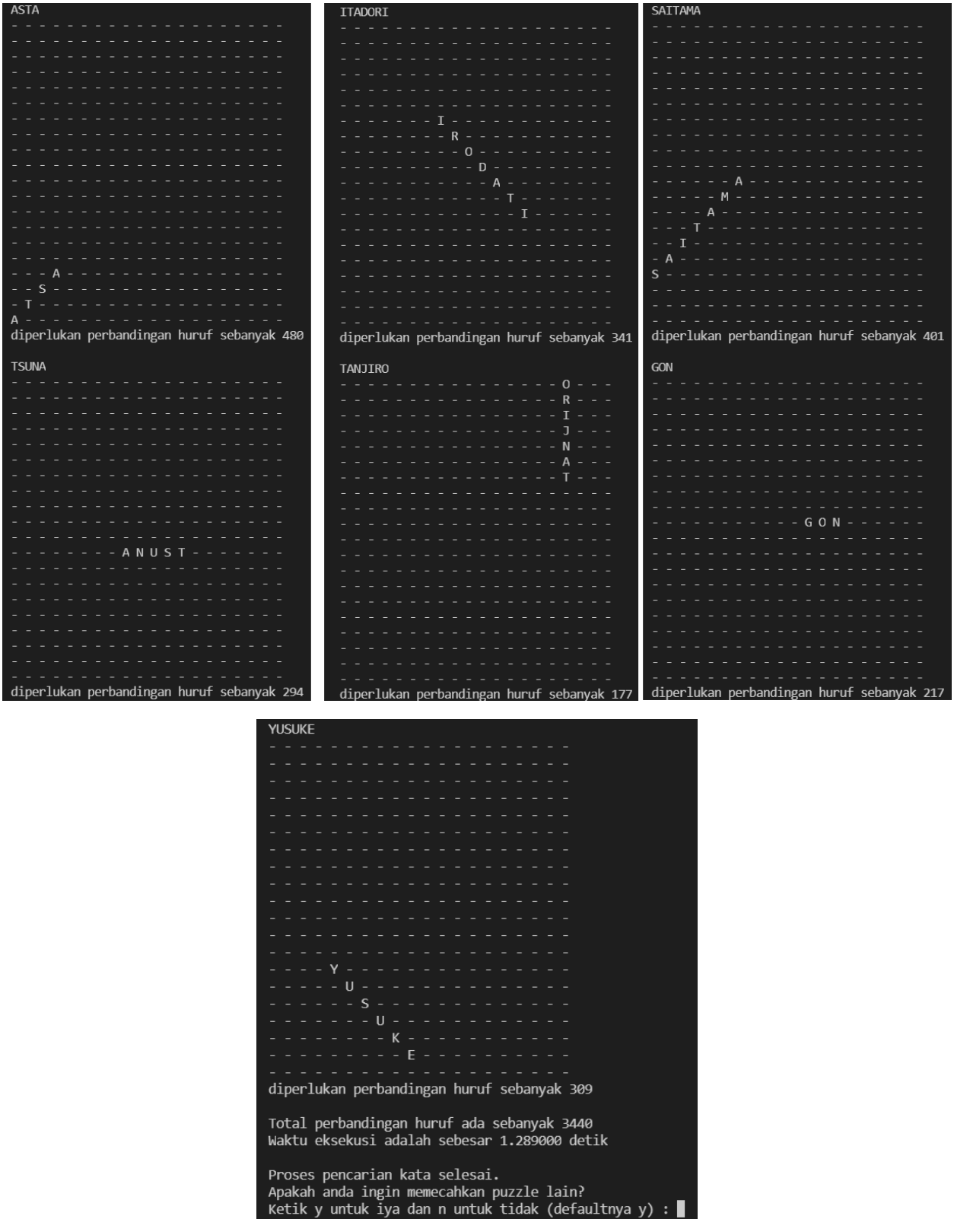


Gambar 3.5 hasil output untuk testcase medium2.txt

3. **medium3.txt**
   ukuran puzzle : 24 baris x 22 kolom
   banyaknya keywords : 12 kata
   total perbandingan huruf : 3440 karakter
   waktu eksekusi : 1,289 sekon





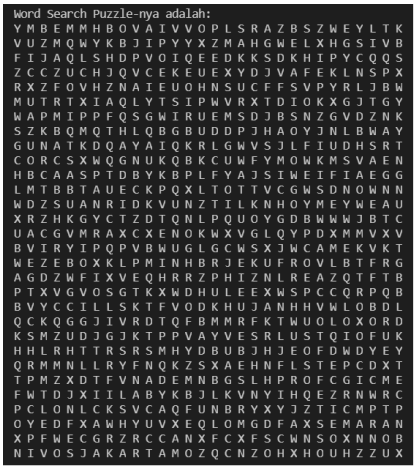


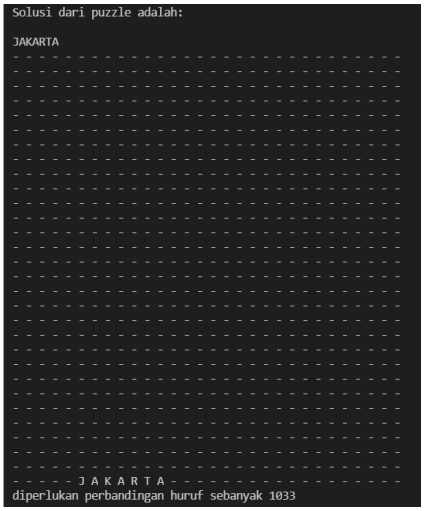
Gambar 3.6 hasil output untuk testcase medium3.txt

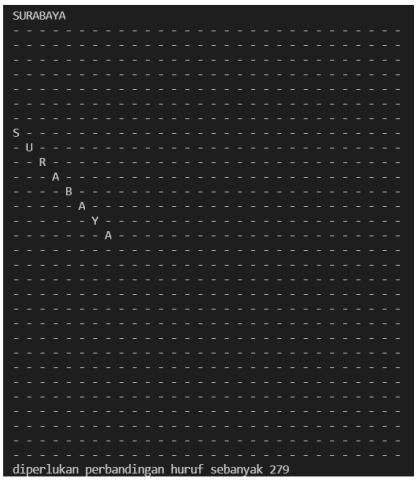
c. Testcase ukuran big

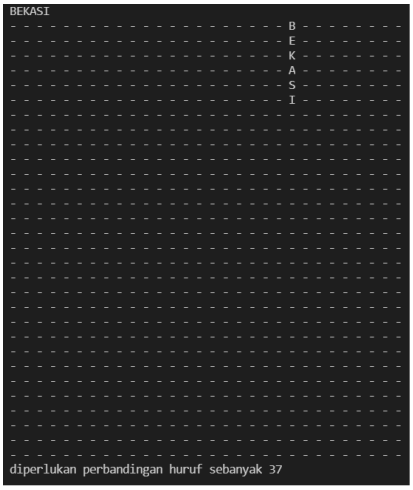
1. big1.txt

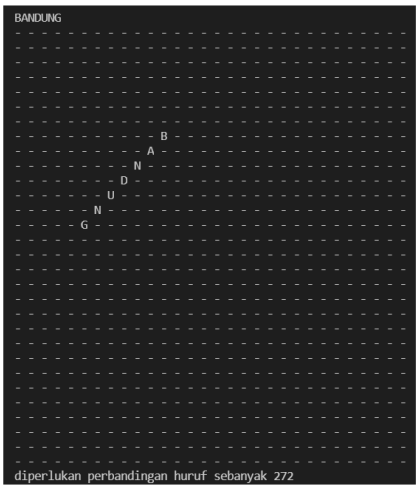
ukuran puzzle : 32 baris x 30 kolom
banyaknya keywords : 13 kata
total perbandingan huruf : 7878 karakter
waktu eksekusi : 2,3 sekon
Dengan catatan dibuat beberapa dummy keyword yang tidak ada dalam puzzle

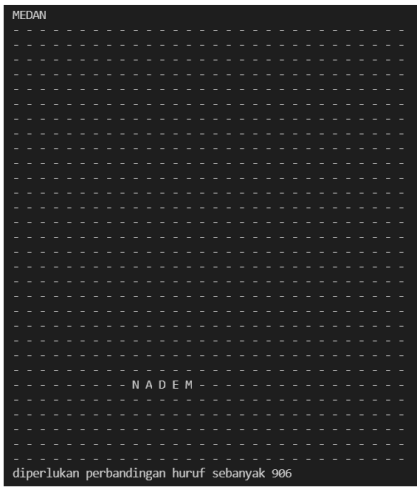


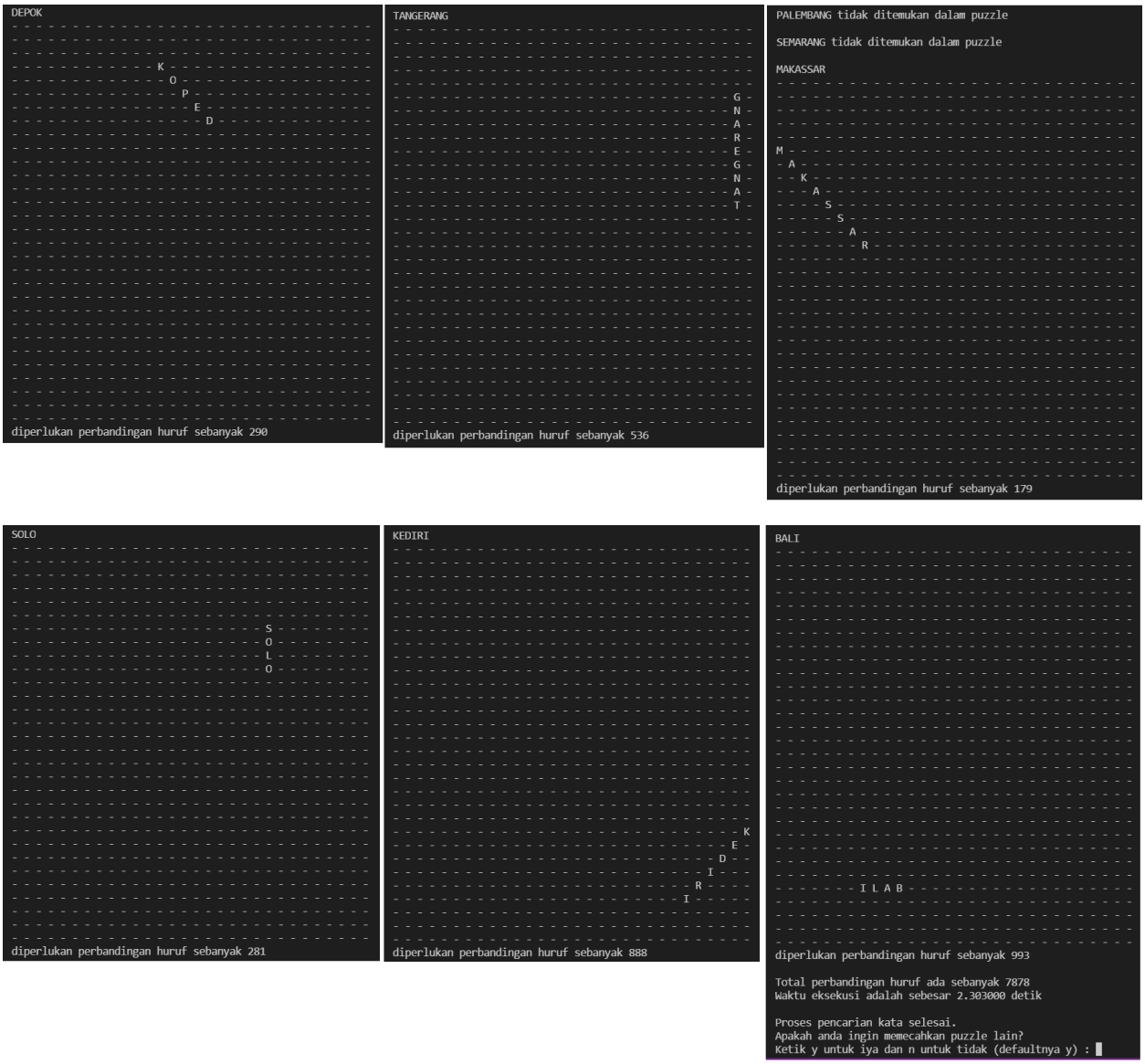






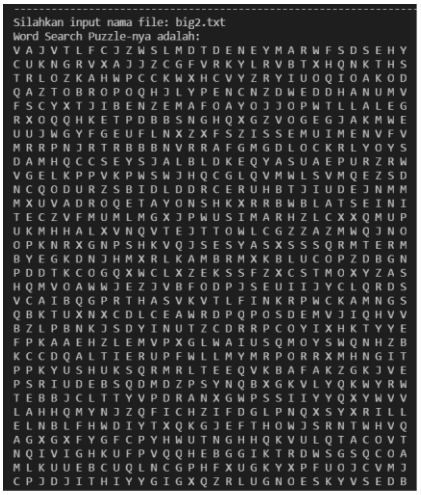


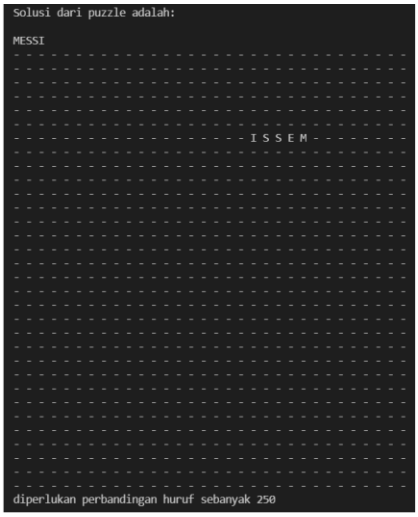


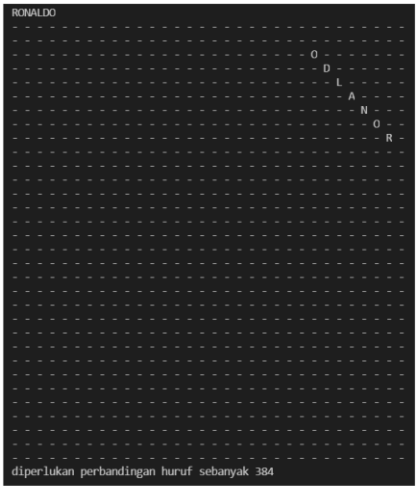


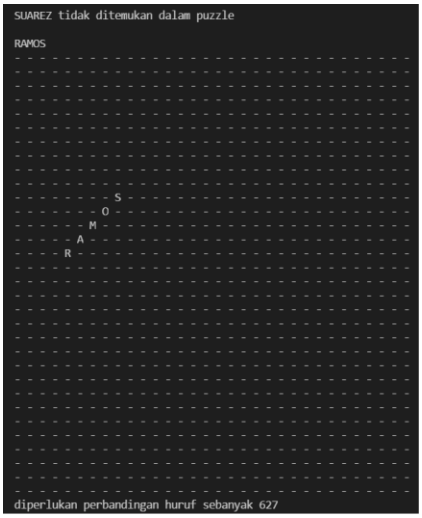
Gambar 3.7 hasil output untuk testcase big1.txt

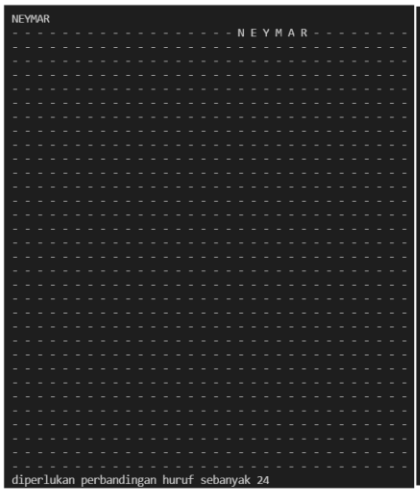
2. **big2.txt**
   ukuran puzzle : 34 baris x 32 kolom
   banyaknya keywords : 13 kata
   total perbandingan huruf : 8511 karakter
   waktu eksekusi : 3,3 sekon
   Dengan catatan dibuat beberapa dummy keyword yang tidak ada dalam puzzle

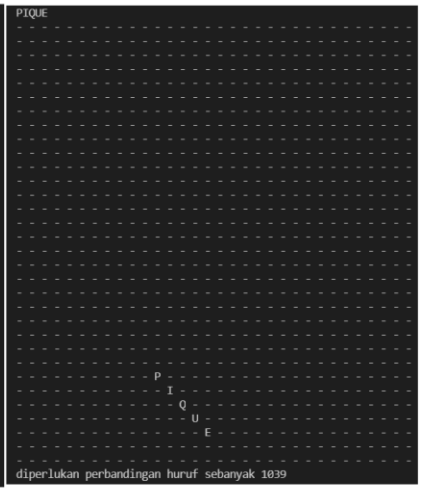

Silahkan input nama file: big2.txt
Word Search Puzzle-nya adalah:
V A J V T L F C J Z W S L M D T D E N E Y M A R W F S D S E H Y
C U K N G R V X A J J Z C G F V R K Y L R V B T X H Q N K T H S
T R L O Z K A H W P C C K W X H C V Y Z R Y I U O Q I O A K O D
Q A Z T O B R O P O Q H J L Y P E N C N Z D W E D D H A N U M V
F S C Y X T J I B E N Z E M A F O A Y O J J O P W T L L A L E G
R X O Q Q H K E T P D B B S N G H Q X G Z V O G E G J A K M W E
U U J W G Y F G E U F L N X Z X F S Z I S S E M U I M E N V F V
M R R P N J R T R B B B N V R R A F G M G D L O C K R L Y O Y S
D A M H Q C C S E Y S J A L B L D K E Q Y A S U A E P U R Z R W
V G E L K P P V K P W S W J H Q C G L Q V M W L S V M Q E Z S D
N C Q O D U R Z S B I D L D D R C E R U H B T J I U D E J N M M
M X U V A D R O Q E T A Y O N S H K X R R B W B L A T S E I N I
T E C Z V F M U M L M G X J P W U S I M A R H Z L C X X Q M U P
U K M H H A L X V N Q V T E J T T O W L C G Z Z A Z M W Q J N O
O P K N R X G N P S H K V Q J S E S Y A S X S S S Q R M T E R M
B Y E G K D N J H M X R L K A M B R M X K B L U C O P Z D B G N
P D D T K C O G Q X W C L X Z E K S S F Z X C S T M O X Y Z A S
H Q M V O A W W J E Z J V B F O D P J S E U I I J Y C L Q R D S
V C A I B Q G P R T H A S V K V T L F I N K R P W C K A M N G S
Q B K T U X N X C D L C E A W R D P Q P O S D E M V J I Q H V V
B Z L P B N K J S D Y I N U T Z C D R R P C O Y I X H K T Y Y E
F P K A A E H Z L E M V P X G L W A I U S Q M Q Y S W Q N H Z B
K C C D Q A L T I E R U P F W L L M Y M R P O R R X M H N G I T
P P K Y U S H U K S Q R M R L T E E Q V K B A F A K Z G K J V E
P S R I U D E B S Q D M D Z P S Y N Q B X G K V L Y Q K W Y R W
T E B B J C L T T Y V P D R A N X G W P S S I I Y Y Q X Y W V V
L A H H Q M Y N J Z Q F I C H Z I F D G L P N Q X S Y X R I L L
E L N B L F H W D I Y T X Q K G J E F T H O W J S R N T W H V Q
A G X G X F Y G F C P Y H W U T N G H H Q K V U L Q T A C O V T
N Q I V I G H K U F P V Q Q H E B G G I K T R D W S G S Q C O A
M L K U U E B C U Q L N C G P H F X U G K Y X P F U O J C V M J
C P J D J I T H I Y Y G I G X Q Z R L U G N O E S K Y V S E D B

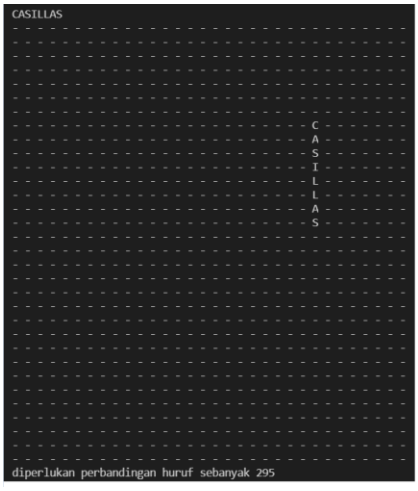

Solusi dari puzzle adalah:
MESSI
I S S E M
diperlukan perbandingan huruf sebanyak 250

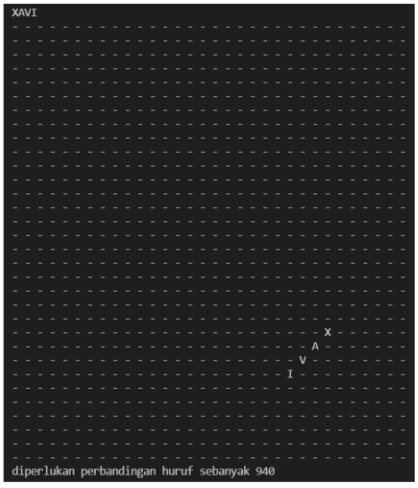

RONALDO
O
D
L
A
N
O
R
diperlukan perbandingan huruf sebanyak 384

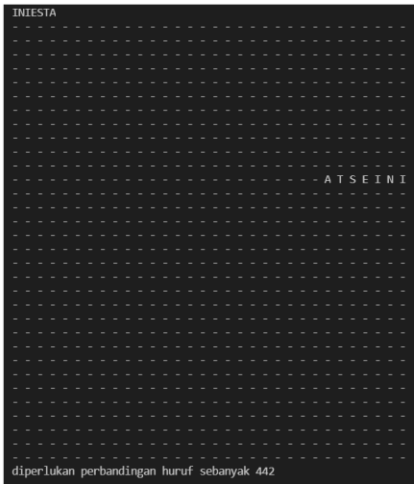

SUAREZ tidak ditemukan dalam puzzle
RAMOS
S
O
M
A
R
diperlukan perbandingan huruf sebanyak 627

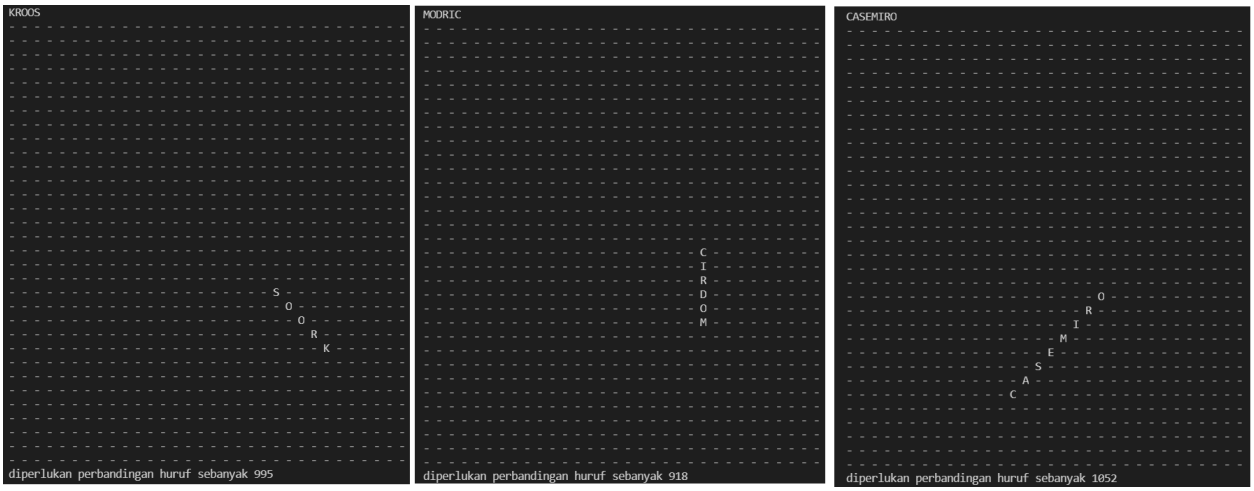

NEYMAR
N E Y M A R
diperlukan perbandingan huruf sebanyak 24


PIQUE
P
I
Q
U
E
diperlukan perbandingan huruf sebanyak 1039


CASILLAS
C
A
S
I
L
L
A
S
diperlukan perbandingan huruf sebanyak 295

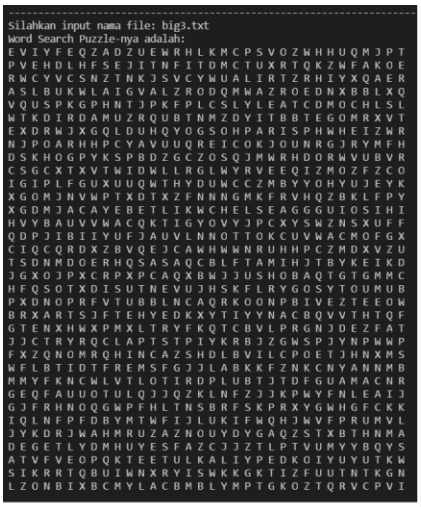

XAVI
X
A
V
I
diperlukan perbandingan huruf sebanyak 940

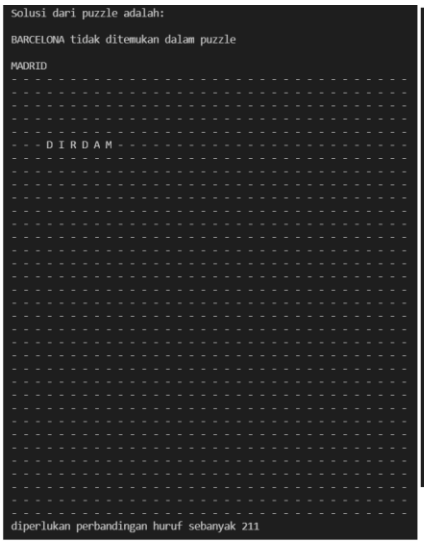

INIESTA
A T S E I N I
diperlukan perbandingan huruf sebanyak 442

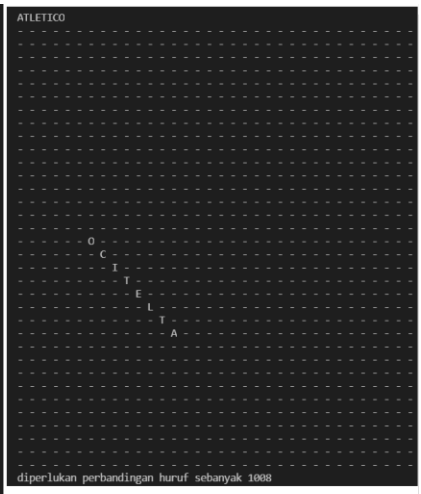


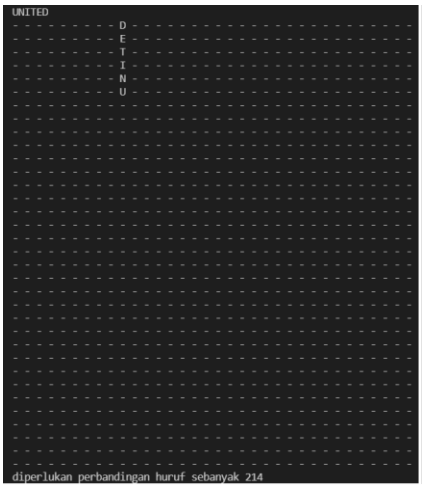
Gambar 3.8 hasil output untuk testcase big2.txt

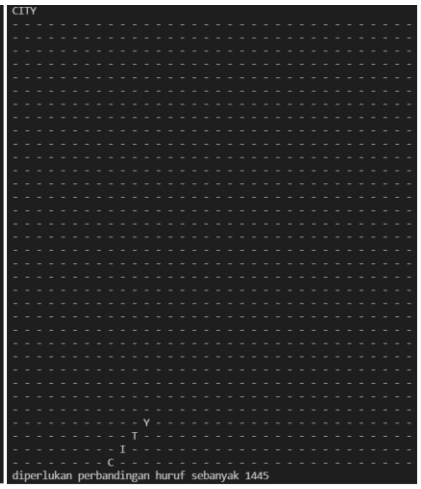
3. **big3.txt**
   ukuran puzzle : 36 baris x 34 kolom
   banyaknya keywords : 15 kata
   total perbandingan huruf : 13202 karakter
   waktu eksekusi : 5,05 sekon
   Dengan catatan dibuat beberapa dummy keyword yang tidak ada dalam puzzle

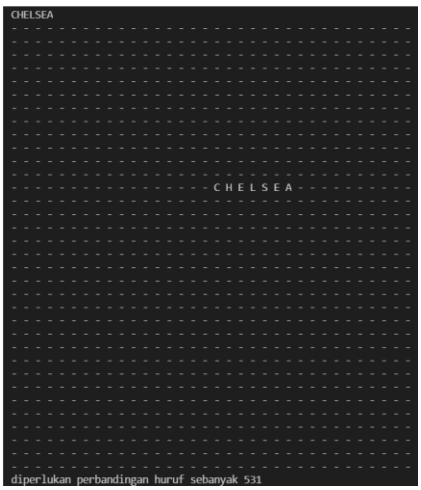

Silahkan input nama file: big3.txt
Word Search Puzzle-nya adalah:
E V I Y F E Q Z A D Z U E W R H L K M C P S V O Z W H H U Q M J P T
P V E H D L H F S E J I T N F I T D M C T U X R T Q K Z W F A K O E
R W C Y V C S N Z T N K J S V C Y W U A L I R T Z R H I Y X Q A E R
A S L B U K W L A I G V A L Z R O D Q M W A Z R O E D N X B B L X Q
V Q U S P K G P H N T J P K F P L C S L Y L E A T C D M O C H L S L
W T K D I R D A M U Z R Q U B T N M Z D Y I T B B T E G O M R X V T
E X D R W J X G Q L D U H Q Y O G S O H P A R I S P H W H E I Z W R
N J P O A R H H P C Y A V U U Q R E I C O K J O U N R G J R Y M F H
D S K H O G P Y K S P B D Z G C Z O S Q J M W R H D O R W V U B V R
C S G C X T X V T W I D W L L R G L W Y R V E E Q I Z M O Z F Z C O
I G I P L F G U X U U Q W T H Y D U W C C Z M B Y Y O H Y U J E Y K
X G O M J N V W P T X D T X Z F N N N G M K F R V H Q Z B K L F P Y
X G D M J A C A Y E B E T L I K W C H E L S E A G G G U I O S I H I
H V Y B A U V V W A C Q K T I G Y O V Y J P C X Y S W Z N S X U F F
Q D P J I B I I Y U F J A U V L N N O T T O K C U V W A C M O F G X
C I Q C Q R D X Z B V Q E J C A W H W W N R U H H P C Z M D X V Z U
T S D N M D O E R H Q S A S A Q C B L F T A M I H J T B Y K E I K D
J G X O J P X C R P X P C A Q X B W J J U S H O B A Q T G T G M M C
H F Q S O T X D I S U T N E V U J H S K F I R Y G O S V T O U M U B
P X D N O P R F V T U B B L N C A Q R K O O N P B I V E Z T E E O W
B R X A R T S J F T E H Y E D K X Y T I Y Y N A C B Q V V T H T Q F
G T E N X H W X P M X L T R Y F K Q T C B V L P R G N J D E Z F A T
J J C T R Y R Q C L A P T S T P I Y K R B J Z G W S P J Y N P W W P
F X Z Q N O M R Q H I N C A Z S H D L B V I L C P O E T J H N X M S
W F L B T I D T F R E M S F G J J L A B K K F Z N K C N Y A N N M B
M M Y F K N C W L V T L O T I R D P L U B T J T D F G U A M A C N R
G E Q F A U U O T U L Q J J Q Z K L N F Z J J K P W Y F N L E A I J
G J F R H N O Q G W P F H L T N S B R F S K P R X Y G W H G F C K K
I Q L N F P F D B Y M T W F I J L U K I F W Q H J W V F P R U M V L
J Y K D R J W A H M R U Z A Z N O U Y D Y G A Q Z S T X B T H N M A
D E G E T L Y D M H U Y E S F A Z C J J Z T L P T V U M Y Y B Q Y S
A T V F V E O P Q K T E E T U L K A L I Y P E D K O I Y U Y U T K W
S I K R R T Q B U I W N X R Y I S W K K G K T I Z F U U T N T K G N
L Z O N B I X B C M Y L A C B M B L Y M P T G K O Z T Q R V C P V I

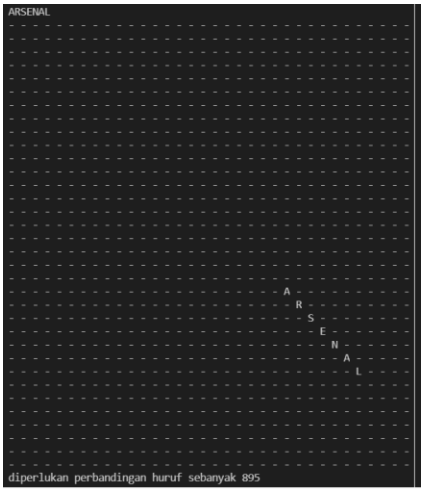

Solusi dari puzzle adalah:
BARCELONA tidak ditemukan dalam puzzle
MADRID
D I R D A M
diperlukan perbandingan huruf sebanyak 211

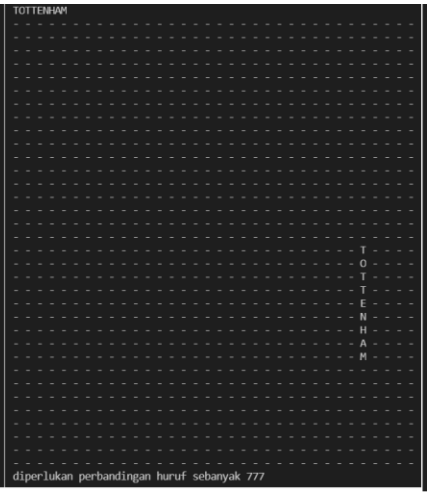

ATLETICO
O
C
I
T
E
L
T
A
diperlukan perbandingan huruf sebanyak 1008

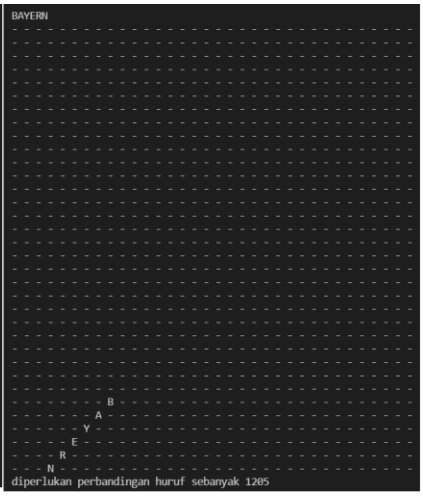

UNITED
D
E
T
I
N
U
diperlukan perbandingan huruf sebanyak 214

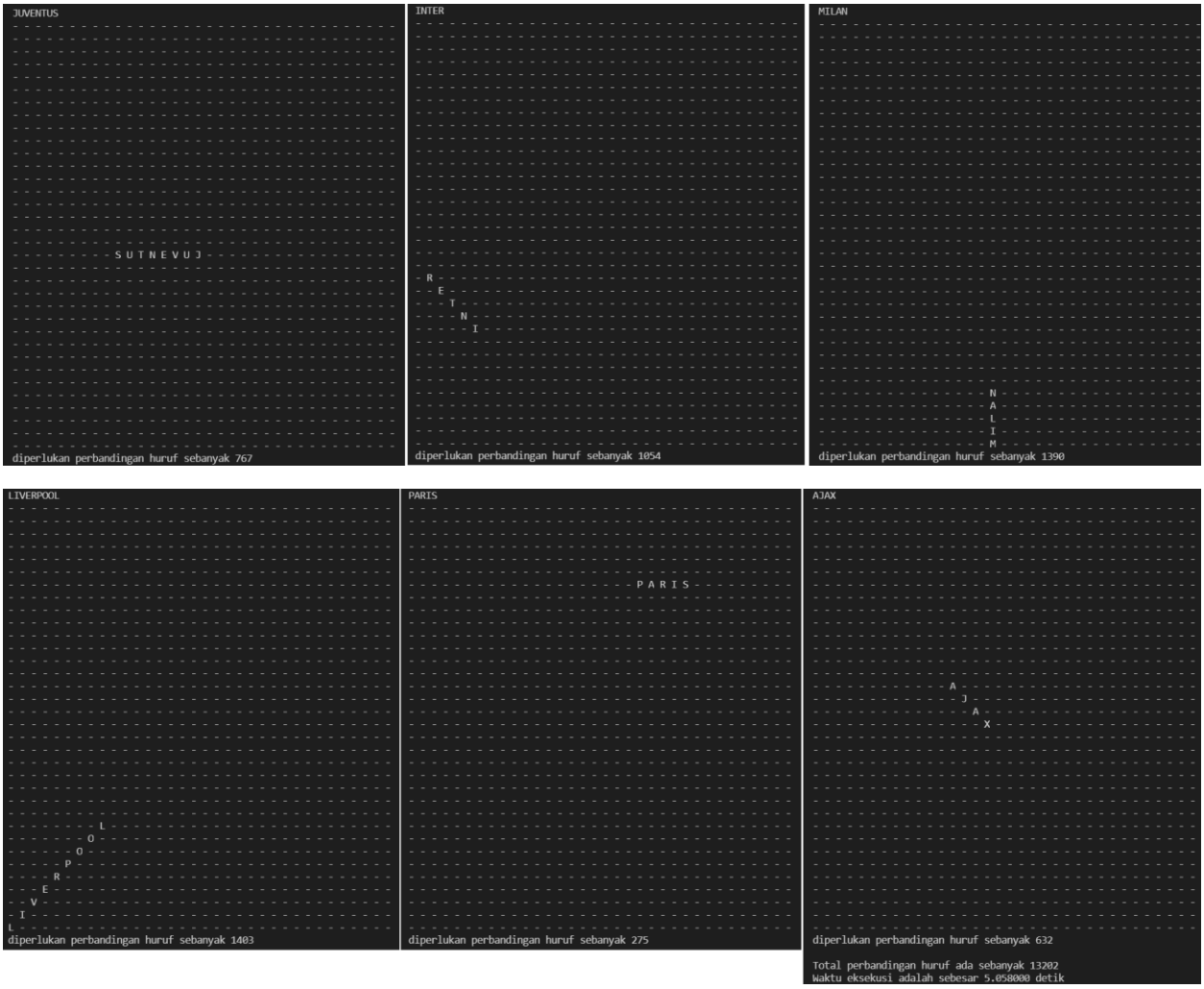

CITY
Y
T
I
C
diperlukan perbandingan huruf sebanyak 1445


CHELSEA
C H E L S E A
diperlukan perbandingan huruf sebanyak 531


ARSENAL
A
R
S
E
N
A
L
diperlukan perbandingan huruf sebanyak 895


TOTTENHAM
T
O
T
T
E
N
H
A
M
diperlukan perbandingan huruf sebanyak 777


BAYERN
B
A
Y
E
R
N
diperlukan perbandingan huruf sebanyak 1205



Gambar 3.9 hasil output untuk testcase big3.txt

D. Alamat Drive

https://github.com/saulsayerz/Tucil1_13520094

E. Tabel Checklist

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.	✓	