# LAPORAN
# TUGAS KECIL 1
# "Penyelesaian Word Search Puzzle dengan Algoritma *Brute Force*"
# IF221 LOGIKA KOMPUTASIONAL


Oleh:

Ng Kyle / 13520040

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


2021

# Bagian I
# Deskripsi Algoritma dan Kompleksitas Algoritma

1. **Algoritma *Brute Force* dalam Word Search Puzzle**
   Algoritma *Brute Force* sebagaimana menjadi dasar algoritma yang digunakan dalam pengerjaan tugas dan program ini adalah algoritma yang *straight-forward* dan dirancang sesuai dengan deskripsi persoalan yang dimaksud. Dengan dasar tersebut, Word Search Puzzle mengharuskan kita mencari kata yang sesuai dengan 8 arah baik horizontal, vertikal, maupun diagonal. Dengan pendekatan *Brute Force*, kita akan mengharuskan mencocokkan setiap huruf pada kata yang dicari dalam papan dalam ke-8 arah tersebut. Dalam hal ini, diberi batasan bahwa setiap kata hanya perlu dicari 1 buah kata.

2. **Perincian dan Penjelasan Algoritma**
   Dalam pengaplikasian algoritma *Brute Force* dalam Word Search Puzzle, dapat dibagi menjadi 2 tahap :
   a. Pre-processing Board dan Kata
      Tahap ini merupakan persiapan papan (Board) yang berisi huruf dan juga kata yang akan dicari. Secara Algoritma, tahapan ini di luar scope tugas ini sehingga tidak akan dijelaskan algoritmanya. Namun sebagai referensi, secara umum struktur program bagian ini menghasilkan :

| Komponen | Implementasi |
|---|---|
| Board Huruf | Array of struct : [Char and Colour Code (int)] |
| List Kata | Linked List of Node : [String and Length (int)] |

   b. Algoritma Pencarian
      Algoritma yang dibuat dibagi menjadi 2 jenis, yaitu algoritma pencarian *brute force* tanpa optimasi dan algoritma *brute force* dengan fungsi heuristik (optimized).

      Tahap ini merupakan fokus utama dari program Word Search. Secara implementasi menggunakan Algoritma *Brute Force* dengan fungsi heuristic (optimized) dengan rincian :
      i. Mengambil data (kata dan panjang) kata pertama dari List Kata.
      ii. Memeriksa huruf pertama pada kata dengan huruf pada board. Jika sesuai akan melakukan pencarian pada tahap (iii), jika tidak maka akan diambil kata selanjutnya pada List Kata hingga huruf pertama pada kata sesuai dengan huruf pada Board atau List Kata sudah diperiksa seluruhnya maka akan lanjut ke tahap (iv).
      iii. Dilakukan pemeriksaan ke 8 arah, secara berurutan diperiksa dengan arah pertama arah kanan, lalu kanan bawah, bawah, kiri bawah, kiri, kiri atas, atas, dan kanan atas. Dilakukan pemeriksaan setiap huruf pada kata yang diperiksa jika memenuhi batasan arah (panjang kata < banyak huruf pada arah tersebut) hingga :
         a. Setiap huruf pada kata telah diperiksa dan sesuai dengan huruf pada Board. Kata yang bersesuaian yang telah diperiksa pada Board akan diberi warna lalu kata akan dihapus dari List Kata. Tidak perlu memeriksa arah lainnya.

b. Terdapat huruf yang tidak sesuai (secara terurut) antara kata yang diperiksa dan Board, maka akan diperiksa arah selanjutnya.

Langkah (iii) dilakukan hingga semua arah diperiksa atau kata sudah ditemukan.

iv. Tahap (ii) dan (iii) dilakukan terus menerus untuk setiap posisi pada Board. Jika sudah selesai memeriksa semua kata dan arah pada posisi tersebut, maka akan melakukan pemeriksaan pada posisi selanjutnya secara row-major hingga mencapai akhir Board yaitu pojok kanan bawah.

Sedangkan untuk versi tanpa optimasi :

i. Mengambil kata dari listWord.

ii. Memeriksa satu per satu kata pada List Kata dengan huruf pada board. Pencarian untuk suatu kata berhenti jika :
   a. Sudah memeriksa seluruh huruf pada kata, maka kata sudah ditemukan dan diberi warna pada board, kata dihapus dari List Kata.
   b. Pencarian pada indeks di luar board atau ditemukan huruf yang tidak sesuai antara huruf pada kata dan pada Board. Dilakukan pencarian ke arah lainnya, secara berurutan kanan, kanan bawah, bawah, kiri bawah, kiri, kiri atas, atas, kanan atas.

iii. Langkah (i) dan (ii) dilakukan untuk semua kata dan posisi pada Board hingga mencapai akhir Board (posisi pojok kanan bawah).

Keterangan :

Optimasi yang dilakukan dengan Algoritma ini adalah :

1. Pemeriksaan kata hanya dilakukan ketika huruf pertama sudah sesuai. Mengurangi pengecekan redundan huruf pertama ke tujuh arah lainnya.
2. Kata yang sudah ditemukan akan dihapus dari List Kata, mengurangi jumlah pengecekan. Berdasarkan batasan program.
3. Pemeriksaan batas berdasarkan panjang kata.

## 3. Analisis Kompleksitas Algoritma

Kompleksitas Algoitma yang digunakan dengan keterangan notasi sebagai berikut:

- N : Panjang Baris pada Board.
- M : Panjang Kolom pada Board.
- L : Panjang rata-rata seluruh kata.
- K : Banyak kata yang dicari.

Dengan N dan M >> L >> K.

Berikut analisis setiap kasusnya (untuk Wordsearch teroptimasi):

a. Best Case :

i. Semua kata ditemukan pada awal Board. (Trivial)
   O(L * K)

ii. Semua huruf pada kata tidak sesuai dengan huruf pada Board kecuali pada posisi terakhir.
   O(N * M * K)
   Penjelasan :

Untuk setiap posisi, akan diperiksa huruf awal setiap kata, dilakukan sebanyak N * M * K perbandingan. Pada posisi terakhir akan diperiksa setiap kata seperti pada Best Case Trivial sebanyak L * K perbandingan. Secara keseluruhan dilakukan : T(N * M * K + L * K) = O(N * M * K)

b. Worst Case
   Setiap kata diperiksa hingga seluruh hurufnya dan gagal memenuhi kesamaan pada huruf terakhir. Semua kata ditemukan pada posisi terakhir.
   O (N * M * L * K)
   Penjelasan :
   Untuk setiap posisi yaitu N * M dilakukan perbandingan terhadap L * K huruf ke 8 arah. Secara keseluruhan dilakukan perbandingan sebanyak : T(8 * N * M * L * K) = O(N * M * L * K).
c. Average Case
   Secara rata-rata, akan diperiksa huruf tidak secara seluruhnya dan tersebar pada Board. Namun secara keseluruhan akan terevaluasi O(N * M * K).

## Bagian II
## Source Program

Program dibagi menjadi bebarapa bagian untuk menjaga modularitas. Berikut merupakan snip source code dan penjelasan kegunannya.

1. boolean.h
   Define header untuk Boolean true-1 dan false-0

```c
/* Definisi type boolean */

#ifndef _BOOLEAN_h
#define _BOOLEAN_h

#define boolean unsigned char
#define true 1
#define false 0

#endif
```

2. color.c / color.h
   File untuk print karakter berwarna dengan ANSII Code

```c
/* Header File untuk Print Color */

void print_color(int ccode, char str);
/* ccode merupakan color code 0-11
(0-6 merupakan warna dengan ansi FG code 31-36)
(7-13 merupakan warna dengan ansi FG code 91-96)*/
```

```c
#include <stdio.h>
#include "color.h"
void print_color(int ccode, char str){
    switch(ccode){
        case -1:
            printf("%c", str);
            break;
        case 0:
            printf("\033[31m%c\033[0m", str);
            break;
        case 1:
            printf("\033[32m%c\033[0m", str);
            break;
        case 2:
            printf("\033[33m%c\033[0m", str);
            break;
        case 3:
            printf("\033[34m%c\033[0m", str);
            break;
        case 4:
            printf("\033[35m%c\033[0m", str);
            break;
        case 5:
            printf("\033[36m%c\033[0m", str);
            break;
        case 6:
            printf("\033[91m%c\033[0m", str);
            break;
        case 7:
            printf("\033[92m%c\033[0m", str);
            break;
        case 8:
            printf("\033[93m%c\033[0m", str);
            break;
        case 9:
            printf("\033[94m%c\033[0m", str);
            break;
        case 10:
            printf("\033[95m%c\033[0m", str);
            break;
        case 11:
            printf("\033[96m%c\033[0m", str);
            break;
    }
}
```

3. words.h / words.c
   ADT LinkedList berfungsi untuk menyimpan data kata dan panjangnya.

```c
#include "words.h"
#include <stdlib.h>

Address newNode(int len){
    Address pnew;
    pnew = (Address) malloc(sizeof(Node));
    LEN(pnew) = len;
    NEXT(pnew) = NULL;
    WORD(pnew) = (char *) malloc(sizeof(char) * (
len+1));
    return pnew;
}

void delNode(Address *p, Address *pbef,
LinkedList *listWords){
    Address pout;
    pout = *p;
    if (*pbef != NULL){
        NEXT(*pbef) = NEXT(*p);
        *p = NEXT(*p);
    }
    else{
        *listWords = NEXT(*p);
        *p = NULL;
    }
    free(pout);
}
```

```c
#ifndef WORDS_H
#define WORDS_H

#include <stdlib.h>
/* Node Address for Word Linked List */
typedef struct node * Address;
typedef struct node {
    char* word;
    int len;
    Address next;
} Node;

typedef Address LinkedList;

Address newNode(int len);
void delNode(Address *p, Address *pbef,
LinkedList *listWords);
#define WORD(p) (p)->word
#define LEN(p) (p)->len
#define NEXT(p) (p)->next

#endif
```

4. filereads.h / filereads.h
   Membaca input file dan membuat Board dan LinkedList sebagai isi query kata.

```c
/* File: fileread.h */
/* Header Pembacaan Konfigurasi File */
#ifndef FILEREAD_H
#define FILEREAD_H

#include "words.h"
typedef struct {
    int ccode;
    char ch;
} board;

board * allocBoardMat(int rows, int cols);
void readDim(LinkedList *listWords, int *rows, int
 *cols);
board * readMat (LinkedList *listWords, int *rows
, int *cols);

#endif
```

```c
/* File: tileread.c */
/* Implementasi Pembacaan Konfigurasi Tile */

#include <stdio.h>
#include <stdlib.h>
#include "boolean.h"
#include "words.h"
#include "filereads.h"

char currentChar;
static FILE * ftape; //Pita karakter untuk file
static int retval;

board * allocBoardMat(int rows, int cols){
    board *mat = (board *)malloc(rows * cols *
sizeof(board));
    return mat;
}

void readDim(LinkedList *listWords, int *rows,
int *cols){
   boolean bef = false;
// Boolean periksa char sebelumnya enter
   boolean isBoard = true;
   boolean isWord = false;
   Address p;
   int len;
   *rows = 1;
   *cols = 0;

   /* Periksa dimensi Board */
   while (isBoard) {
      fscanf(ftape,"%c",&currentChar);
      if (currentChar == '\n'){
         if (bef == true){
            isBoard = false;
         }
         else{
            bef = true;
         }
      }
      else if (currentChar != ' '){
         if (bef == true) {
            *rows += 1;
            *cols = 0;
         }
         bef = false;
         *cols += 1;
      }
   }
   p = *listWords;
   Address pin = NULL;
   Address ptrav = NULL;
```

```
/* Set buffer linked list length */
   len = 0;
   while (fscanf(ftape,"%c",&currentChar) != EOF
){
      if (currentChar == '\n') {
         isWord = false;
         pin = newNode(len);
         if (*listWords == NULL){
            *listWords = pin;
            ptrav = *listWords;
         }
         else{
            NEXT(ptrav) = pin;
            ptrav = NEXT(ptrav);
         }
         len = 0;
      }
      else {
         isWord = true;
         len += 1;
      }
   }
   pin = newNode(len);
   NEXT(ptrav) = pin;
}

board * readMat (LinkedList *listWords, int *rows
, int *cols){
   int i, j;
   boolean bef = false;
   boolean isBoard = true;
   board * mat;
   int pos;
   Address p;
   char fname[50];
   *listWords = NULL;

   /*File read*/
   do {
      printf("Masukkan nama file: ");
      scanf("%s", &fname);
      ftape = fopen(fname, "r");
      if (ftape == NULL){
         printf(
"File Not Found. Please Specify folder directory
 or check file name. Ex : ../test/in.txt
\n");
      }
   } while(ftape == NULL);

   readDim(listWords, rows, cols);
   mat = allocBoardMat(*rows, *cols);
   p = *listWords;

   ftape = fopen(fname, "r");
   i = 0;
   j = 0;
```

```
// Loop Set Matrix
   while (isBoard) {
      fscanf(ftape,"%c",&currentChar);
      if (currentChar == '\n'){
         i+= 1;
         j = 0;
         if (bef == true) {
            isBoard = false;
         }
         else{
            bef = true;
         }
      }
      else if (currentChar != ' '){
         bef = false;
         mat[i * (*cols) + j].ch = currentChar;
         mat[i * (*cols) + j].ccode = -1;
         j+= 1;

      }
      else {
         bef = false;
      }
   }
   //Set LinkedList Words
   p = *listWords;
   pos = 0;
   while (fscanf(ftape,"%c",&currentChar) != EOF
){
      if (currentChar == '\n') {
         WORD(p)[pos] = '\0';
         pos = 0;
         p = NEXT(p);
      }
      else {
         WORD(p)[pos] = currentChar;
         pos += 1;
      }
   }
   WORD(p)[pos] = '\0';

   for (int i = 0 ; i < (*rows); i++){
      for (int j = 0; j < (*cols); j++){
         printf("%c ", mat[i*(*cols) +j].ch);
      }
      printf("\n");
   }

   printf("\n\n");

   p = *listWords;
   while(p != NULL){
      printf("%s\n", WORD(p));
      p = NEXT(p);
   }
   return mat;
}
```

5. solver.c
   File berisi algoritma utama dalam menyelesaikan word search. Fokus utama algoritma untuk versi optimized.

```c
#include <stdio.h>
#include "solver.h"
#include "words.h"
#include "boolean.h"
#include <stdint.h>
#include <time.h>
void solve(board * mat, LinkedList * listWords, int rows, int cols){
    Address p, pbef;
    char * cur_word;
    int cur_len;
    int tot_perb;
    int n_checked;
    int i,j,k;
    int codeColor;
    boolean isFound, isEq;
    tot_perb = 0;
    n_checked = 0;
    i =0;
    j = 0;
    codeColor = -1;
    struct timespec begin;
    struct timespec end;
    struct timespec tprint;
    clock_gettime(CLOCK_MONOTONIC, &begin);

    while (i < rows && *listWords != NULL){
        j = 0;
        while (j < cols && *listWords != NULL){
            p = *listWords;
            pbef = NULL;
            while (p != NULL){
                isFound = false;
                cur_word = WORD(p);
                cur_len = LEN(p);
                if (cur_word[0] == mat[i * cols + j].ch){
                    // cek kanan
                    if (j + cur_len <= cols){
                        k = 0;
                        isEq = true;
                        while (k < cur_len && isEq){
                            n_checked++;
                            if (mat[i * cols + j+k].ch != cur_word[k]){
                                isEq = false;
                            }
                        else k += 1;
                        }
                        if(isEq){

                            // Set Warna
                            codeColor = (codeColor+1)%12;
                            isFound = true;
                            for(k = 0; k < cur_len; k++){
                                mat[i * cols + j+k].ccode = codeColor;
                            }
                            // Hapus
                            delNode(&p,&pbef,listWords);
                        }
                    }
```

```c
// cek kanan-bawah
if (j + cur_len <= cols && i + cur_len <= rows && !(
isFound)){
    k = 0;
    isEq = true;
    while (k < cur_len && isEq){
        n_checked++;
        if (mat[(i+k) * cols + j+k].ch != cur_word[k]){
            isEq = false;
        }
        else k += 1;
    }
    if(isEq){
        // Set Warna
        codeColor = (codeColor+1)%12;
        isFound = true;
        for(k = 0; k < cur_len; k++){
            mat[(i+k) * cols + j+k].ccode = codeColor;
        }
        // Hapus
        delNode(&p,&pbef,listWords);
    }
}

// cek bawah
if (i + cur_len <= rows && !(isFound)){
    k = 0;
    isEq = true;
    while (k < cur_len && isEq){
        n_checked++;
        if (mat[(i+k) * cols + j].ch != cur_word[k]){
            isEq = false;
        }
        else k += 1;
    }
    if(isEq){
        // Set Warna
        codeColor = (codeColor+1)%12;
        isFound = true;
        for(k = 0; k < cur_len; k++){
            mat[(i+k) * cols + j].ccode = codeColor;
        }
        // Hapus
        delNode(&p,&pbef,listWords);
    }
}
```

```
// cek bawah-kiri
if (i + cur_len <= rows && j - cur_len >= -1 && !(isFound)){
    k = 0;
    isEq = true;
    while (k < cur_len && isEq){
        n_checked++;
        if (mat[(i+k) * cols + j-k].ch != cur_word[k]){
            isEq = false;
        }
        else k += 1;
    }
    if(isEq){
        // Set Warna
        codeColor = (codeColor+1)%12;
        isFound = true;
        for(k = 0; k < cur_len; k++){
            mat[(i+k) * cols + j-k].ccode = codeColor;
        }
        // Hapus
        delNode(&p,&pbef,listWords);
    }
}

// cek kiri
if (j - cur_len >= -1 && !(isFound)){
    k = 0;
    isEq = true;
    while (k < cur_len && isEq){
        n_checked++;
        if (mat[i * cols + j-k].ch != cur_word[k]){
            isEq = false;
        }
        else k += 1;
    }
    if(isEq){
        // Set Warna
        codeColor = (codeColor+1)%12;
        isFound = true;
        for(k = 0; k < cur_len; k++){
            mat[i * cols + j-k].ccode = codeColor;
        }
        // Hapus
        delNode(&p,&pbef,listWords);
    }
}
```

```
// cek kiri-atas
if (j - cur_len >= -1 && i-cur_len >= -1 && !(isFound)){
    k = 0;
    isEq = true;
    while (k < cur_len && isEq){
        n_checked++;
        if (mat[(i-k) * cols + j-k].ch != cur_word[k]){
            isEq = false;
        }
        else k += 1;
    }
    if(isEq){
        // Set Warna
        codeColor = (codeColor+1)%12;
        isFound = true;
        for(k = 0; k < cur_len; k++){
            mat[(i-k) * cols + j-k].ccode = codeColor;
        }
        delNode(&p,&pbef,listWords);
    }
}

// atas
if (i-cur_len >= -1 && !(isFound)){
    k = 0;
    isEq = true;
    while (k < cur_len && isEq){
        n_checked++;
        if (mat[(i-k) * cols + j].ch != cur_word[k]){
            isEq = false;
        }
        else k += 1;
    }
    if(isEq){
        // Set Warna
        codeColor = (codeColor+1)%12;
        isFound = true;
        for(k = 0; k < cur_len; k++){
            mat[(i-k) * cols + j].ccode = codeColor;
        }
        // Hapus
        delNode(&p,&pbef,listWords);
    }
}
```

```
                    // atas-kanan
                    if (i-cur_len >= -1 && j+cur_len <= rows && !(isFound)){
                        k = 0;
                        isEq = true;
                        while (k < cur_len && isEq){
                            n_checked++;
                            if (mat[(i-k) * cols + j+k].ch != cur_word[k]){
                                isEq = false;
                            }
                            else k += 1;
                        }
                        if(isEq){
                            // Set Warna
                            codeColor = (codeColor+1)%12;
                            isFound = true;
                            for(k = 0; k < cur_len; k++){
                                mat[(i-k) * cols + j + k].ccode = codeColor;
                            }
                            // Hapus
                            delNode(&p,&pbef,listWords);
                        }
                    }

                }
                if (!isFound) {
                    pbef = p;
                    p = NEXT(p);
                }

            }
            j += 1;
        }
        i += 1;
    }
```

```
clock_gettime(CLOCK_MONOTONIC, &end);

    for (i = 0 ; i < rows; i++){
        for(j = 0 ; j < cols ; j++){
            print_color(mat[i*cols + j].ccode, mat[i*cols + j].ch);
            printf(" ");
        }
        printf("\n");
    }
    clock_gettime(CLOCK_MONOTONIC, &tprint);

    uint64_t end_ns = (end.tv_sec * 1000000000) + end.tv_nsec;
    uint64_t begin_ns = (begin.tv_sec * 1000000000) + begin.tv_nsec;
    uint64_t tprint_ns = (tprint.tv_sec * 1000000000) + tprint.tv_nsec;
    int64_t elapsed = end_ns - begin_ns;
    int64_t elapsprint = tprint_ns - begin_ns;
    printf("\nCharacter Comparisons: %d\n", n_checked);
    printf("Algorithm Time: %ld nanoseconds (%lf seconds)\n",elapsed, (double)elapsed/1000000000);
    printf("Algorithm Time + Print: %ld nanoseconds (%lf seconds)\n",elapsprint, (double)elapsprint/1000000000);
}
```

6. solver_basic.c
   File berisi algoritma utama dalam menyelesaikan word search. Fokus utama algoritma untuk versi unoptimized.

```c
#include <stdio.h>
#include "solver.h"
#include "words.h"
#include "boolean.h"
#include <stdint.h>
#include <time.h>

boolean checkWord(board * mat, Address *p, int
rows, int cols, int currow, int curcol, int
horinc, int vertinc, int * n_checked){
    int i = currow;
    int j = curcol;
    int k = 0;
    while (i < rows && j < cols && i >= 0 && j
 >= 0 &&  k < LEN(*p)){
        *n_checked = *n_checked+1;
        if (mat[i*cols + j].ch != WORD(*p)[k]){
            return false;
        }
        else {
            i = i + vertinc;
            j = j + horinc;
            k = k +1;
        }
    }
    if (k != LEN(*p)){
        return false;
    }
    else {
        return true;
    }
}

void setColour( board * mat, int currow, int
curcol, int horinc, int vertinc, int colorcode,
int k, int cols){
    int r = currow;
    int c = curcol;
    for(int i = 0; i < k; i++){
        mat[r * cols + c].ccode = colorcode;
        r = r + vertinc;
        c = c + horinc;
    }
}
```

```c
void solve(board * mat, LinkedList * listWords,
int rows, int cols){
    Address p, pbef;
    char * cur_word;
    int cur_len;
    int tot_perb;
    int n_checked;
    int i,j,k;
    int codeColor;
    boolean isFound, isEq;
    tot_perb = 0;
    n_checked = 0;
    i =0;
    j = 0;
    codeColor = -1;
    struct timespec begin;
    struct timespec end;
    struct timespec tprint;
    clock_gettime(CLOCK_MONOTONIC, &begin);

    while (i < rows && *listWords != NULL){
        j = 0;
        while (j < cols && *listWords != NULL){
            p = *listWords;
            pbef = NULL;
            while (p != NULL){
                isFound = false;
                cur_word = WORD(p);
                cur_len = LEN(p);

                // Kanan
                isFound = checkWord(mat, &p, rows
, cols,i, j, 1,0, &n_checked);
                if (isFound){
                    codeColor = (codeColor+1)%12;
                    setColour(mat, i, j, 1,0,
codeColor, LEN(p), cols);
                    delNode(&p,&pbef,listWords);
                }
```

```
// Kanan Bawah
 if (!isFound){
     isFound = checkWord(mat, &p, rows, cols,i, j
, 1,1, &n_checked);
     if (isFound){
         codeColor = (codeColor+1)%12;
         setColour(mat, i, j, 1,1,codeColor, LEN(
p), cols);
         delNode(&p,&pbef,listWords);
     }
 }

 //bawah
 if (!isFound){
     isFound = checkWord(mat, &p, rows, cols,i, j
,0,1, &n_checked);
     if (isFound){
         codeColor = (codeColor+1)%12;
         setColour(mat, i, j, 0,1,codeColor, LEN(
p), cols);
         delNode(&p,&pbef,listWords);
     }
 }

 //kiri bawah
 if (!isFound){
     isFound = checkWord(mat, &p, rows, cols,i, j
,-1,1, &n_checked);
     if (isFound){
         codeColor = (codeColor+1)%12;
         setColour(mat, i, j, -1,1,codeColor, LEN
(p), cols);
         delNode(&p,&pbef,listWords);
     }
 }

 //kiri
 if (!isFound){
     isFound = checkWord(mat, &p, rows, cols,i, j
,-1,0, &n_checked);
     if (isFound){
         codeColor = (codeColor+1)%12;
         setColour(mat, i, j, -1,0,codeColor, LEN
(p), cols);
         delNode(&p,&pbef,listWords);
     }
 }

 // kiri atas
 if (!isFound){
     isFound = checkWord(mat, &p, rows, cols,i, j
,-1,-1, &n_checked);
     if (isFound){
         codeColor = (codeColor+1)%12;
         setColour(mat, i, j, -1,-1,codeColor,
LEN(p), cols);
         delNode(&p,&pbef,listWords);
     }
 }
```

```
            // atas
            if (!isFound){
                isFound = checkWord(mat, &p, rows
, cols,i, j,0,-1, &n_checked);
                if (isFound){
                    codeColor = (codeColor+1)%12;
                    setColour(mat, i, j, 0,-1,
codeColor, LEN(p), cols);
                    delNode(&p,&pbef,listWords);
                }
            }

            // kanan atas
            if (!isFound){
                isFound = checkWord(mat, &p, rows
, cols,i, j,1,-1, &n_checked);
                if (isFound){
                    codeColor = (codeColor+1)%12;
                    setColour(mat, i, j, 1,-1,
codeColor, LEN(p), cols);
                    delNode(&p,&pbef,listWords);
                }
            }

            // Kasus tidak ditemukan
            if (!isFound) {
                pbef = p;
                p = NEXT(p);
            }

        }
        j += 1;
    }
    i += 1;
}
```

```
    clock_gettime(CLOCK_MONOTONIC, &end);

    for (i = 0 ; i < rows; i++){
        for(j = 0 ; j < cols ; j++){
            print_color(mat[i*cols + j].ccode, mat[i
*cols + j].ch);
            printf(" ");
        }
        printf("\n");
    }
    clock_gettime(CLOCK_MONOTONIC, &tprint);

    uint64_t end_ns = (end.tv_sec * 1000000000) +
end.tv_nsec;
    uint64_t begin_ns = (begin.tv_sec * 1000000000
) + begin.tv_nsec;
    uint64_t tprint_ns = (tprint.tv_sec * 1000000000
) + tprint.tv_nsec;
    int64_t elapsed = end_ns - begin_ns;
    int64_t elapsprint = tprint_ns - begin_ns;
    printf("\nCharacter Comparisons: %d\n",
n_checked);
    printf("Algorithm Time: %ld nanoseconds (%lf
 seconds)\n",elapsed, (double)elapsed/1000000000);
    printf("Algorithm Time + Print: %ld
 nanoseconds (%lf seconds)\n",elapsprint, (double)
elapsprint/1000000000);
}
```

7. solver.h
   Header file untuk solver.c dan solver_basic.c.

```c
/* Header File solver.h */
#include "filereads.h"
#include "color.h"
#include "boolean.h"
void solve(board * mat,LinkedList * listWords, int rows, int cols);
boolean checkWord(board * mat, Address *p, int rows, int cols, int currow, int curcol, int horinc, int vertinc, int * checked);
void setColour( board * mat, int currow, int curcol, int horinc, int vertinc, int colorcode, int k, int cols);
```

8. main.c
   Main file, memanggil fungsi-fungsi lain.

```c
#include "filereads.h"
#include "words.h"
#include "color.h"
#include "solver.h"
#include <time.h>
int main() {
    board * mat= 0;
    int rows;
    int cols;
    int boxes;
    LinkedList listWords;
    mat = readMat(&listWords, &rows, &cols);
    solve(mat, &listWords, rows, cols);
    return 0;
}
```

## Bagian III
## Screenshoot Program Testing

Berikut merupakan kumpulan hasil penjalanan program

<table>
<tr><th colspan="3">Small 1</th></tr>
<tr><td colspan="3">File name: small1.txt</td></tr>
<tr><td>Words</td><td colspan="2">Board Size</td></tr>
<tr><td>15</td><td colspan="2">14 x 12</td></tr>
<tr><td colspan="2">Board</td><td>Query</td></tr>
<tr><td colspan="2">K F B Q K S U P E R S U S B<br>V N A P T O H C R H I M J C<br>I G A C E W T P K W N E Y A<br>Q L H B E L I K C Q S O W P<br>Y A X Y O B A T I Q T P H C<br>X Z N T D E O Z T T A E A U<br>Z Y N A W F N O A E G N T T<br>K C S I D L Y R K D R S S Y<br>T C A T N O C T E G A E A W<br>Q I L B I L B B V R M A P C<br>U Q T I E R A H S L I B P F<br>M A R G E L E T C V O O A A</td><td>CAPCUT<br>TIKTOK<br>GETCONTACT<br>LAZADA<br>TELEGRAM<br>SHAREIT<br>NEOBANK<br>WHATSAPP<br>OPENSEA<br>INSTAGRAM<br>DANA<br>SUPERSUS<br>FACEBOOK<br>TWITTER<br>BLIBLI</td></tr>
<tr><td colspan="3">Result (Optimized)</td></tr>
<tr><td colspan="3">K F B Q K S U P E R S U S B<br>V N A P T O H C R H I M J C<br>I G A C E W T P K W N E Y A<br>Q L H B E L I K C Q S O W P<br>Y A X Y O B A T I Q T P H C<br>X Z N T D E O Z T T A E A U<br>Z Y N A W F N O A E G N T T<br>K C S I D L Y R K D R S S Y<br>T C A T N O C T E G A E A W<br>Q I L B I L B B V R M A P C<br>U Q T I E R A H S L I B P F<br>M A R G E L E T C V O O A A<br><br>Character Comparisons: 415<br>Algorithm Time: 14600 nanoseconds (0.000015 seconds)<br>Algorithm Time + Print: 44999500 nanoseconds (0.044999 seconds)</td></tr>
<tr><td colspan="3">Result (Unoptimized)</td></tr>
<tr><td colspan="3">Character Comparisons: 8866<br>Algorithm Time: 55400 nanoseconds (0.000055 seconds)<br>Algorithm Time + Print: 29248900 nanoseconds (0.029249 seconds)</td></tr>
<tr><td colspan="3">Small 2</td></tr>
<tr><td colspan="3">File name: small2.txt</td></tr>
<tr><td>Words</td><td colspan="2">Board Size</td></tr>
</table>

| 18 | 16 x 16 |
|---|---|

| Board | Query |
|---|---|

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

```
MCDONALDS
STARBUCKS
CHIKFILA
TACOBELL
WENDYS
BURGERKING
DUNKIN
SUBWAY
DOMINOS
CHIPOTLE
SONICDRIVEIN
PIZZAHUT
PANERABREAD
KFC
POPEYES
ARBYS
DAIRYQUEEN
LITTLECAESARS
```

Result (Optimized)

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

Character Comparisons: 730
Algorithm Time: 17900 nanoseconds (0.000018 seconds)
Algorithm Time + Print: 41148900 nanoseconds (0.041149 seconds)
```

Result (Unoptimized)

```
Character Comparisons: 17837
Algorithm Time: 105000 nanoseconds (0.000105 seconds)
Algorithm Time + Print: 40750400 nanoseconds (0.040750 seconds)
```

Small 3

File name: small3.txt

| Words | Board Size |
|---|---|
| 20 | 16 X 16 |

| Board | Query |
|---|---|


```
BITCOIN
ETHEREUM
TETHER
BNB
USDCOIN
CARDANO
SOLANA
XRP
TERRALUNA
POLKADOT
DOGECOIN
AVALANCHE
BINANCEUSD
SHIBAINU
POLYGONMATIC
TERRAUSD
COSMOS
WRAPPED
UNISWAP
CHAINLINK
```

Result (Optimized)

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

Character Comparisons: 947
Algorithm Time: 18800 nanoseconds (0.000019 seconds)
Algorithm Time + Print: 41792900 nanoseconds (0.041793 seconds)
```

Result (Unoptimized)

```
Character Comparisons: 18407
Algorithm Time: 109900 nanoseconds (0.000110 seconds)
Algorithm Time + Print: 40376100 nanoseconds (0.040376 seconds)
```

Medium 1

File name: medium1.txt

| Words | Board Size |
|---|---|
| 25 | 22 x 22 |

| Board | Query |
|---|---|

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

```
JURONGEAST
BUKITBATOK
BUKITGOMBAK
BRICKLAND
CHOACHUKANG
YEWTEE
SUNGEIKADUT
KRANJI
MARSILLING
WOODLANDS
ADMIRALTY
BOONLAY
SEMBAWANG
CANBERRA
YISHUN
YIOCHUKANG
ANGMOKIO
BRADDEL
DHOBBYGHAUT
RAFFLESPLACE
MARINASOUTHPIER
PASIRRIS
TANAHMERAH
PAYALEBAR
COMMONWEALTH
```

Result (Optimized)

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

Character Comparisons: 2003
Algorithm Time: 42800 nanoseconds (0.000043 seconds)
Algorithm Time + Print: 84417200 nanoseconds (0.084417 seconds)
```

Result (Unoptimized)

```
Character Comparisons: 46088
Algorithm Time: 266700 nanoseconds (0.000267 seconds)
Algorithm Time + Print: 67251200 nanoseconds (0.067251 seconds)
```

| Medium 2 | |
|---|---|
| File name: medium2.txt | |
| Words | Board Size |
| 26 | 24 x 22 |

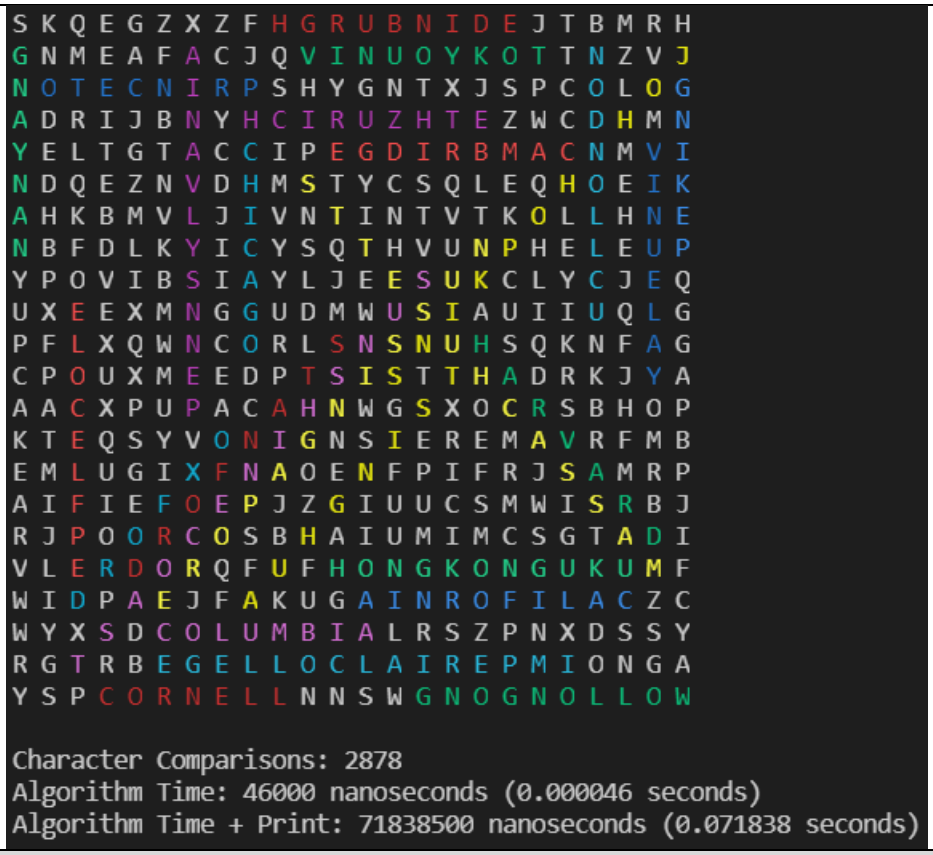
Board

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

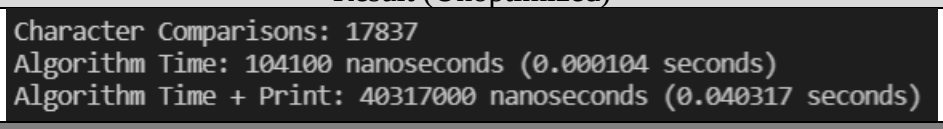
Query

```
MASSACHUSETTS
OXFORD
STANFORD
CAMBRIDGE
HARVARD
CALIFORNIA
IMPERIALCOLLEGE
ETHZURICH
UCLLONDON
CHICAGO
NUSSINGAPORE
NANYANG
PENNSYLVANIA
EPFLECOLE
YALEUNIV
EDINBURGH
TSINGHUA
PEKING
COLUMBIA
PRINCETON
CORNELL
HONGKONGUKU
TOKYOUNIV
JOHNHOPKINS
WOLLONGONG
SUNSHINECOAST
```

Result (Optimized)




Result (Unoptimized)




Medium 3

File name: medium3.txt

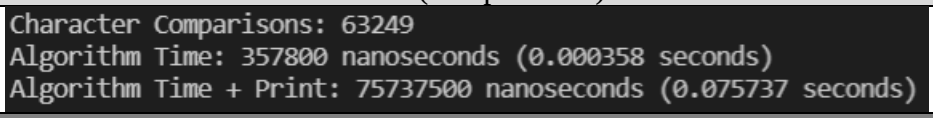
| Words | Board Size |
|---|---|
| 30 | 22 x 22 |

| Board | Query |
|---|---|


INSTAGRAM
CHRISTIANORONALDO
KYLIE
LEOMESSI
THEROCK
SELENAGOMEZ
ARIANAGRANDE
KIMKARDASHIANWEST
BEYONCE
JUSTINBIEBER
KHLOEKARDASHIAN
KENDALL
NATIONALGEOGRAPHIC
TAYLORSWIFT
NIKE
JENNIFERLOPEZ
VIRATKOHLI
BARBIE
NJ
KOURTNEY
MILEYCYRUS
KATYPERRY
KEVINHART
DEMILOVATO
ZENDAYA
CARDIB
BADGALRIRI
ELLENDEGENERES
KINGJAMES
REALMADRIDCF

Result (Optimized)


Result (Unoptimized)

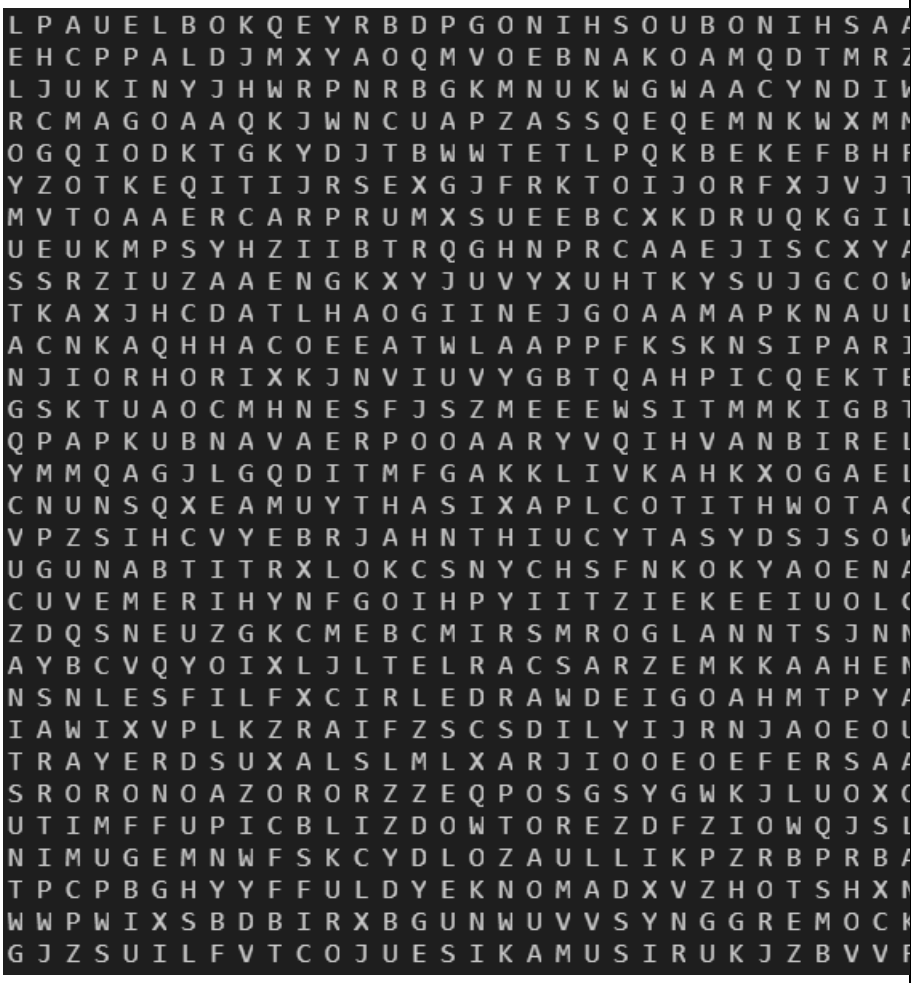



Large 1

File name: large1.txt

| Words | Board Size |
|---|---|
| 38 | 32 x 30 |

| Board | Query |
|---|---|



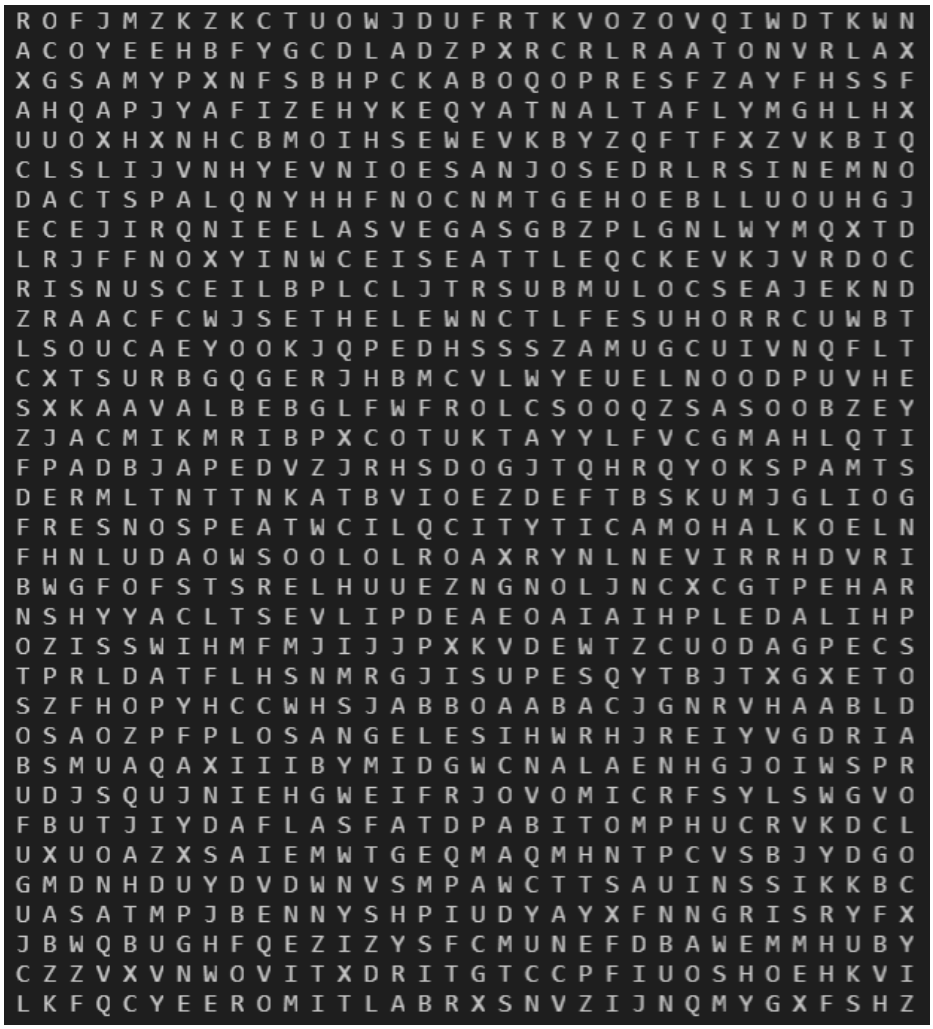
EZRASCARLET
LEVI
LAWLIETL
MONKEYDLUFFY
RORONOAZORO
OKABERINTAROU
LIGHTYAGAMI
KILLUAZOLDYCK
EDWARDELRIC
UZUMAKINARUTO
GINTOKISAKATA
GUTS
KURISUMAKISE
UCHIHAITACHI
ERENYEAGER
MIKASAACKERMANN
GILDARTSCLIVE
KENKANEKI
REM
SAITAMA
KAKASHIHATAKE
SPIKESPIEGEL
GOJOUSATORU
KAZUTOKIRIGAYA
LAXUSDREYAR
YATO
MEGUMIN
JOSEPHJOESTAR
ROYMUSTANG
HISOKAMOROW
ZEROTWO
MAISAKURAJIMA
SHINOBUOSHINO
KAMINA
TAIGAAISAKA
ICHIGO
TANJIRO
ZANITSU

Result (Optimized)


Result (Unoptimized)

```
Character Comparisons: 152240
Algorithm Time: 906700 nanoseconds (0.000907 seconds)
Algorithm Time + Print: 118427800 nanoseconds (0.118428 seconds)
```

| Large 2 | |
|---|---|
| File name: large2.txt | |
| Words | Board Size |
| 40 | 34 x 34 |
| Board | Query |

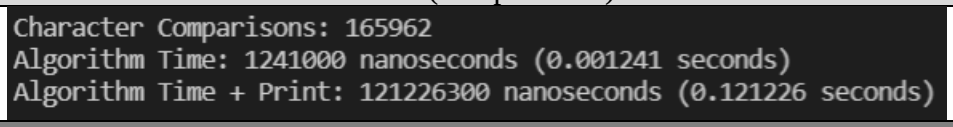


R O F J M Z K Z K C T U O W J D U F R T K V O Z O V Q I W D T K W N
A C O Y E E H B F Y G C D L A D Z P X R C R L R A A T O N V R L A X
X G S A M Y P X N F S B H P C K A B O Q O P R E S F Z A Y F H S S F
A H Q A P J Y A F I Z E H Y K E Q Y A T N A L T A F L Y M G H L H X
U U O X H X N H C B M O I H S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

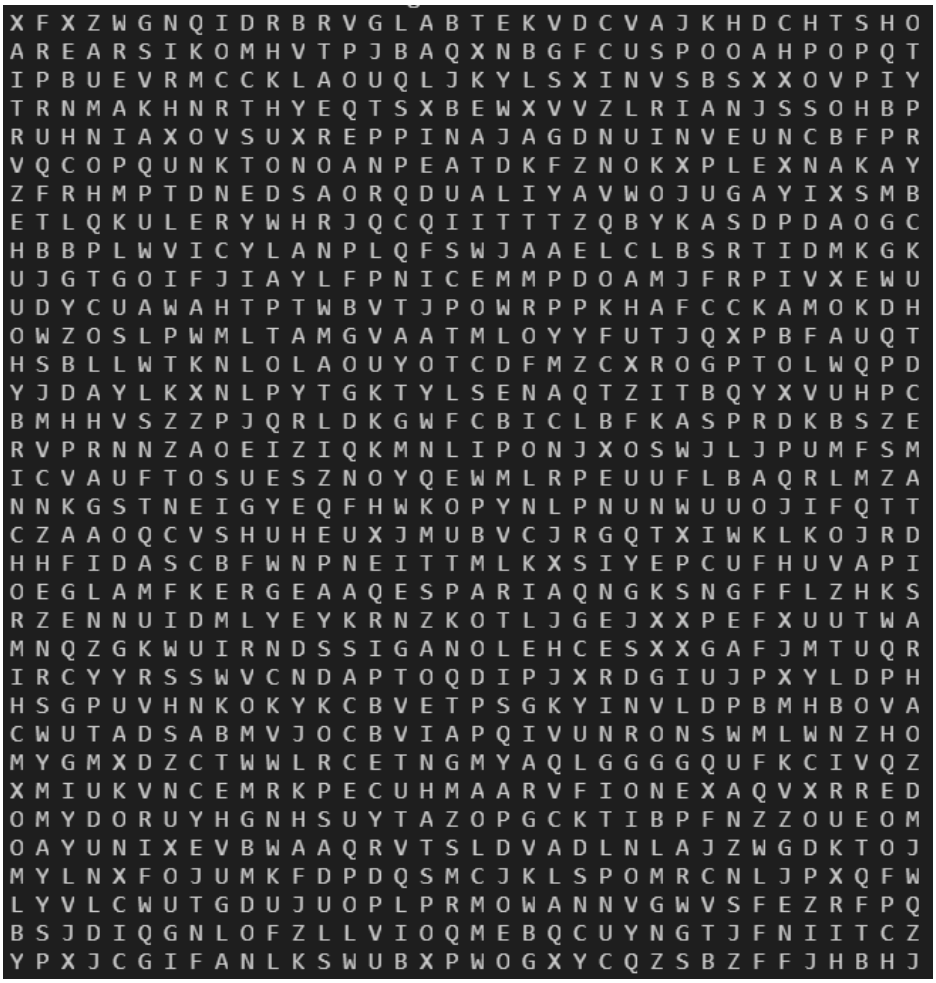
NEWYORK
LOSANGELES
CHICAGO
HOUSTON
PHOENIX
PHILADELPHIA
SANANTONIO
SANDIEGO
DALLAS
SANJOSE
AUSTIN
JACKSONVILLE
FORTWORTH
COLUMBUS
INDIANAPOLIS
CHARLOTTE
SANFRANCISCO
SEATTLE
DENVER
WASHINGTON
NASHVILLE
OKLAHOMACITY
ELPASO
BOSTON
PORTLAND
LASVEGAS
DETROIT
MEMPHIS
LOUSVILLE
BALTIMORE
MILWAUKEE
ALBUQUERQUE
TUCSON
FRESNO
SACRAMENTO
KANSASCITY
MESA
ATLANTA
OMAHA
COLORADOSPRINGS

Result


Result (Unoptimized)




| Large 3 | |
|---|---|
| File name: large3.txt | |
| Words | Board Size |
| 32 | 34x36 |
| Board | Query |



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

```
AGAMAETIKA
ALGEO
ALSTRUKDAT
ARSIKOM
ASLING
BASDAT
COMPUTERS
CRYPTOGRAPHY
DASPRO
ECHELON
ENGLISH
FIDAS
KALKULUS
KIDAS
KURIKULUM
LOGIKAKOMP
MATDIS
MATRIKS
OLAHRAGA
OOP
OTOMATA
PAR
PENGKOM
PRD
PROBSTAT
RPL
SISOP
INFORMATIKA
STIMA
ENGINEERING
LAPTOP
LABTEKV
```

Result (Optimized)


Result (Unoptimized)




Note : Simulasi dilakukan pada keadaan yang berbeda (aplikasi yang sedang berjalan), sehingga berpengaruh ke waktu proses print unoptimized lebih cepat, namun secara waktu algoritmis tetap jauh lebih cepat algoritma teroptimasi walaupun keadaan yang lebih kurang memadai.

## Bagian IV
## Link Source Code / Github

Berikut terlampir link drive source code serta isinya sesuai dengan spesifikasi :
https://drive.google.com/drive/folders/18E6JIM-H7Qm8Hm9hen7Qd8B1k019gEzo?usp=sharing

Berikut Repository Github:
https://github.com/Nk-Kyle/WordSearch.git

| 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 | ✓ | |