Laporan Tugas Besar IF2211 Strategi Algoritma  
**Implementasi Algoritma Brute Force untuk Penyelesaian Word-Search Puzzle   
dengan Program Word Search Puzzle Solver berbasis Command-Line Interface (CLI) dalam Bahasa C++**



Dibuat oleh:

Alifia Rahmah

NIM 13520122

Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung

# 1. ALGORITMA BRUTE FORCE

## 1.1. Penjelasan Singkat

Algoritma brute force adalah algoritma dengan pendekatan paling lempang (straightforward) untuk menyelesaikan suatu permasalahan. Algoritma brute force memecahkan permasalahan dengan sangat sederhana, langsung, dan jelas caranya, dengan mencoba-coba semua kemungkinan dan langkah untuk menyelesaikan persoalan. Salah satu implementasi algoritma brute force yaitu untuk melakukan pencocokan string secara brute force.

## 2.1. Pendekatan Program

Algoritma brute force pada penyelesaian word search puzzle diterapkan pada pencarian tiap kata dalam matriks huruf.

# 2. SOURCE PROGRAM

## 2.1. Program Utama (main.cpp)

/\* main.cpp \*/

/\* Program utama word-search puzzle solver \*/

#include <iostream>

#include <string.h>

#include <vector>

#include "lib/lib.h"

using namespace std;

int main(){

  string filename;

  vector<vector<char>> m;

  vector<string> wl;

  // isi vector matrix & wordlist

  cout << "Insert filename (from test folder): ";

  cin >> filename;

  readFile(&filename, &m, &wl);

  wordSearch(&m, &wl);

  return 0;

}

## 2.2. Pembacaan file (readfile.h, readfile.cpp)

readfile.h

/\* readfile.h \*/

/\* Header file untuk readfile.cpp \*/

#include <iostream>

#include <string.h>

#include <fstream>

#include <vector>

#include "readfile.cpp"

using namespace std;

void readFile(string \*filename, vector<vector<char>> \*matrix, vector<string> \*wordlist);

readfile.cpp

/\* readfile.cpp \*/

/\* Mengambil matriks dan daftar kata dari file .txt dalam folder test \*/

#include <iostream>

#include <string.h>

#include <fstream>

#include <vector>

using namespace std;

vector<char> vectorizeLine(string s){

vector<char> x;

for(int i = 0; i < s.length(); i+=2){

x.push\_back(s[i]);

}

return x;

}

void readFile(string \*filename, vector<vector<char>> \*matrix, vector<string> \*wordlist){

string line, fullpath;

bool inwordlist;

ifstream file("../test/" + \*filename);

if (file.is\_open()){

while(getline(file, line)){

if(line == ""){

inwordlist = true;

} else {

if (inwordlist){

(\*wordlist).push\_back(line);

} else {

(\*matrix).push\_back(vectorizeLine(line));

}

}

}

file.close();

} else {

cout << "Unable to open file.";

}

}

## 2.3. Pencarian kata (wordsearch.h, wordsearch.cpp)

wordsearch.h

/\* wordsearch.h \*/

/\* Header file untuk wordsearch.cpp \*/

#include <iostream>

#include <vector>

#include <string>

#include <chrono>

#include "wordsearch.cpp"

using namespace std;

void wordSearch(vector<vector<char>> m, vector<string> l);

wordsearch.cpp

/\* wordsearch.cpp \*/

/\* Mencari semua kata dari daftar kata dalam matriks dengan algoritma brute-force \*/

#include <iostream>

#include <vector>

#include <string>

#include <chrono>

using namespace std;

// ke atas

void strMatch1(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int it, int j, int \*lettercount, string \*l,

bool \*found){

int i = it;

while((!(\*found)) && (m->at(i).at(j) == l->at(\*lettercount)) && (i >= 0)){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

i--;

if (i < 0) break;

}

}

// ke kanan atas

void strMatch2(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int it, int jt, int \*lettercount, string \*l,

bool \*found){

int i = it, j = jt;

while((!(\*found)) && (m->at(i).at(j) == l->at(\*lettercount)) && (i >= 0) && (j < m->at(i).size())){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

i--;

j++;

if (i < 0) break;

if (j == m->at(i).size()) break;

}

}

// ke kanan

void strMatch3(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int i, int jt, int \*lettercount, string \*l,

bool \*found){

int j = jt;

while((!(\*found)) && (m->at(i).at(j) == l->at(\*lettercount)) && (j < m->at(i).size())){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

j++;

if (j == m->at(i).size()) break;

}

}

// ke kanan bawah

void strMatch4(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int it, int jt, int \*lettercount, string \*l,

bool \*found){

int i = it, j = jt;

while((!(\*found)) && (m->at(i).at(j) == l->at(\*lettercount)) && (i < m->size()) && (j < m->at(i).size())){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

i++;

j++;

if (i == m->size()) break;

if (j == m->at(i).size()) break;

}

}

// ke bawah

void strMatch5(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int it, int j, int \*lettercount, string \*l,

bool \*found){

int i = it;

while((!(\*found)) && (m->at(i).at(j) == l->at(\*lettercount)) && (i < m->size())){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

i++;

if (i == m->size()) break;

}

}

// ke kiri bawah

void strMatch6(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int it, int jt, int \*lettercount, string \*l,

bool \*found){

int i = it, j = jt;

while(!(\*found) && (i >= 0) && (j >= 0)){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

i++;

j--;

if (i == m->size()) break;

if (j < 0) break;

}

}

// ke kiri

void strMatch7(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int i, int jt, int \*lettercount, string \*l,

bool \*found){

int j = jt;

while(!(\*found) && (j >= 0)){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

j--;

if (j < 0) break;

}

}

// ke kiri atas

void strMatch8(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int it, int jt, int \*lettercount, string \*l,

bool \*found){

int i = it, j = jt;

while(!(\*found) && (i >= 0) && (j >= 0)){

(\*lettercount)++;

coordlist->push\_back({i, j});

if(\*lettercount == l->length()){

\*found = true;

}

i--;

j--;

if (i < 0) break;

if (j < 0) break;

}

}

/\* strMatch : Search kelanjutan sesuai direction, berhenti kalo k == l.length ato i/j mentok \*/

void strMatch(vector<vector<char>> \*m,

vector<vector<int>> \*coordlist,

int i, int j, int \*lettercount, string \*l, short int direction,

bool \*found){

// buat ngebungkus, fungsi cuma dijalanin kalo belom ketemu aja

if(!(\*found)){

switch(direction){

case 1:

strMatch1(m, coordlist, i, j, lettercount, l, found);

break;

case 2:

strMatch2(m, coordlist, i, j, lettercount, l, found);

break;

case 3:

strMatch3(m, coordlist, i, j, lettercount, l, found);

break;

case 4:

strMatch4(m, coordlist, i, j, lettercount, l, found);

break;

case 5:

strMatch5(m, coordlist, i, j, lettercount, l, found);

break;

case 6:

strMatch6(m, coordlist, i, j, lettercount, l, found);

break;

case 7:

strMatch7(m, coordlist, i, j, lettercount, l, found);

break;

case 8:

strMatch8(m, coordlist, i, j, lettercount, l, found);

break;

}

if(!(\*found)){ // kalo dari 8 arah belom ketemu full kata

coordlist->clear();

direction = -1;

}

}

}

/\* strSearch : Search per kata, kalo found diwarnain \*/

void strSearch(vector<vector<char>> \*m, vector<vector<short int>> \*mcolor, string \*l, int idx){

int i = 0, j, lettercount = 0, comparison = 0;

short int direction = -1;

bool found = false;

vector<vector<int>> coordlist; // list koordinat masing2 huruf dalam 1 kata yg lagi dicari

auto start = chrono::high\_resolution\_clock::now();

// cari huruf pertama (sequential search)

while(!found && (i < m->size())){

j = 0;

while(!found && (j < m->at(i).size())){

if (m->at(i).at(j) == l->at(0)){ // kalo ketemu huruf pertama

lettercount=1;

coordlist.push\_back({i, j});

if(lettercount == l->length()){

found = true; // kalo cuma 1 huruf

} else {

// cari huruf kedua di 8 arah

if (i > 0){

if (m->at(i-1).at(j) == l->at(1)){

direction = 1; // atas

strMatch(m, &coordlist, i-1, j, &lettercount, l, direction, &found);

}

comparison++;

if (j < m->at(i).size()-1){

if (m->at(i-1).at(j+1) == l->at(1)){

direction = 2; // kanan atas

strMatch(m, &coordlist, i-1, j+1, &lettercount, l, direction, &found);

}

comparison++;

}

if (j > 0){

if (m->at(i-1).at(j-1) == l->at(1)){

direction = 8; // kiri atas

strMatch(m, &coordlist, i-1, j-1, &lettercount, l, direction, &found);

} else {

comparison++;

}

}

}

if (i < m->size()-1){

if (m->at(i+1).at(j) == l->at(1)){

direction = 5; // bawah

strMatch(m, &coordlist, i+1, j, &lettercount, l, direction, &found);

} else {

comparison++;

}

if (j < m->at(i).size()-1){

if (m->at(i+1).at(j+1) == l->at(1)){

direction = 4; // kanan bawah

strMatch(m, &coordlist, i+1, j+1, &lettercount, l, direction, &found);

} else {

comparison++;

}

}

if (j > 0){

if (m->at(i+1).at(j-1) == l->at(1)){

direction = 6; // kiri bawah

strMatch(m, &coordlist, i+1, j-1, &lettercount, l, direction, &found);

} else {

comparison++;

}

}

}

if (j > 0){

if (m->at(i).at(j-1) == l->at(1)){

direction = 7; // kiri

strMatch(m, &coordlist, i, j-1, &lettercount, l, direction, &found);

} else {

comparison++;

}

}

if (j < m->at(i).size()-1){

if (m->at(i).at(j+1) == l->at(1)){

direction = 3; // kanan

strMatch(m, &coordlist, i, j+1, &lettercount, l, direction, &found);

} else {

comparison++;

}

}

}

} else {

comparison++;

}

if (!found){

coordlist.clear();

}

j++;

}

i++;

}

auto end = chrono::high\_resolution\_clock::now();

long long diff = chrono::duration\_cast<chrono::nanoseconds>(end - start).count();

// pake coordlist buat isi mcolor

// kode warna ulang2 aja

if (found) {

int color = 31 + (idx % 6);

for(i = 0; i < coordlist.size(); i++){

mcolor->at(coordlist[i][0]).at(coordlist[i][1]) = color;

}

cout << \*l << " found\n";

cout << "Total time: " << diff << " nanoseconds\n";

cout << "Total comparison: " << comparison << endl;

}

}

void print\_color(char s, short int x){

cout << "\033[" << x << "m" << s << "\033[0m";

}

// fungsi utama yang dipanggil dari main

// search masing2 kata dalam l, print ulang pake mcolor, tulis keseluruhan waktu

void wordSearch(vector<vector<char>> \*m, vector<string> \*l){

int i, j;

vector<vector<short int>> mcolor;

vector<short int> x;

// inisialisasi mcolor

for(i = 0; i < m->size(); i++){

for(j = 0; j < m->at(i).size(); j++){

x.push\_back(0);

}

mcolor.push\_back(x);

x.clear();

}

x.erase(x.begin(), x.end());

auto startall = chrono::high\_resolution\_clock::now();

// cari semua kata + taroh kode2 warna di mcolor

for(i = 0; i < l->size(); i++){

cout << "\nSearching for " << l->at(i) << "..." << endl;

strSearch(m, &mcolor, &(l->at(i)), i); // search tiap string di matriks

}

cout << "\nAll search completed.\n";

auto endall = chrono::high\_resolution\_clock::now();

long long duration = chrono::duration\_cast<chrono::nanoseconds>(endall - startall).count();

cout << "Total time: " << duration << " nanoseconds\n\n";

// print semua warna

for(i = 0; i < m->size(); i++){

for(j = 0; j < m->at(i).size(); j++){

print\_color(m->at(i).at(j), mcolor.at(i).at(j));

if(j != m->at(i).size()-1){

cout << " ";

}

}

cout << "\n";

}

}

# 3. SCREENSHOT PROGRAM

## 3.1. Test Case 1 (14 x 14, kecil)

N C W T D E S E B E I S B K

S G X O A P E N L K P P K N

V X O V D B R N A S R K S O

S R X Z A N L T C O E G L F

R V E O B I I E K V N R E H

C B V F C H S W B F K X G S

G H J N V E M A O K K Y N L

J F E Q L I D M A X E R Y V

L P S D T V H L R F L G H I

A X J R T T G W D G T J P H

T G B E A Y J C P D T K N E

K O O B N L Q S R N O A D H

A O O F X C H A I R B V C L

A M X B D A K B V J D O H F

BLACKBOARD

CHAIR

PENCIL

BOOK

DOOR

TABLE

BOTTLE

PEN

WINDOW

Text

Description automatically generated

## 3.2. Test Case 2 ()

## 3.3. Test Case 3 ()

## 3.4. Test Case 4 ()

## 3.5. Test Case 5 ()

## 3.6. Test Case 6 ()

## 3.7. Test Case 7 ()

## 3.8. Test Case 8 ()

## 3.9. Test Case 9 ()

# 4. ALAMAT DRIVE PROGRAM

GitHub Repository: <https://github.com/alifiarahmah/WordSearchPuzzle-Solver.git>

# 5. LAIN-LAIN

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