## Laporan Tugas Kecil 1 IF 2211 Strategi Algoritma

## Penyelesaian Word Search Puzzle dengan Algoritma Brute Force

Disusun oleh:

## **Ghazian Tsabit Alkamil**

13520165



PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG TAHUN AJARAN 2021/2022

#### **BABI**

### Algoritma Brute Force

Algoritma *Brute Force* adalah suatu pendekatan untuk memecahkan suatu persoalan secara straightforward. Biasanya algoritma *brute force* didasarkan pada pernyataan pada persoalan (*problem statement*) dan definisi atau konsep yang digunakan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung, dan jelas caranya (*obvious way*).

Langkah-langkah yang diterapkan pada implementasi program penyelesaian word search puzzle adalah sebagai berikut.

Program akan mencocokkan huruf yang terdapat pada baris dan kolom pertama pada puzzle dengan huruf pertama dari kata pertama yang terdapat pada *wordlist*. Apabila huruf tersebut tidak cocok maka program akan mencocokkan huruf yang terdapat pada baris pertama kolom kedua dan seterusnya sampai huruf pertama pada kata pertama wordlist ditemukan atau sampai semua huruf pada puzzle telah ditelusuri.

Apabila huruf pertama pada kata pertama yang terdapat pada wordlist ditemukan pada puzzle, maka langkah selanjutnya adalah mencari huruf kedua pada kata tersebut dengan cara mencari huruf tersebut pada puzzle, tentunya huruf kedua harus berada pada posisi yang berdekatan dengan huruf pertama. Pencarian huruf kedua dilakukan dengan pencarian dengan delapan arah, yaitu arah timur, tenggara, selatan, barat daya, barat, barat laut, utara, dan timur laut. Arah yang pertama kali di cari adalah arah barat laut, kemudian ke arah selanjutnya sesuai dengan perputaran arah jarum jam. Apabila huruf kedua tersebut cocok, maka akan dicari lagi huruf selanjutnya dengan arah yang sama. Misal huruf kedua ditemukan pada arah selatan maka huruf ketiga akan diperiksa di arah selatan juga dan langkah tersebut dilakukan sejumlah panjang huruf tersebut dikurang satu. Apabila perulangan berhasil dilakukan sebanyak panjang huruf dikurang satu maka kata tersebut ada pada puzzle.

Apabila huruf tidak cocok atau pada arah tersebut tidak terdapat huruf atau melebihi ukuran dari puzzle maka program akan mencari pada arah selanjutnya, apabila dari delapan arah tersebut tidak ditemukan huruf yang cocok maka program akan mencari ulang huruf pertama pada kata ke posisi selanjutnya, kemudian mencari huruf kedua lagi ke delapan arah dan seterusnya. Apabila sampai akhir dari puzzle kata tersebut tidak ditemukan, maka kata tersebut tidak terdapat pada puzzle.

Langkah-langkah yang telah dijelaskan diatas dilakukan sebanyak jumlah kata yang harus dicari.

# BAB II Source Program

Program penyelesaian *word search puzzle* ini diimplementasikan dengan menggunakan bahasa pemrograman c++.

Pada *Gambar 2.1* terlihat beberapa file header yang digunakan pada implementasi program penyelesaian *word search puzzle*.

```
#include <iostream>
    #include <vector>
    #include <string>
    #include <fstream>
    #include <algorithm>
    #include <ostream>
    #include <cstdlib>
    #include <filesystem>
    #include <stdlib.h>
    #include <unordered set>
10
    #include <set>
11
    #include <cstdio>
12
    #include <chrono>
```

Gambar 2.1 Source Program 1

```
using namespace std;using namespace std::chrono;
```

Gambar 2.2 Source Program 2

Gambar 2.3 menunjukkan inisialisasi variabel N yang menunjukkan jumlah baris dari matriks puzzle, variabel M menunjukkan jumlah kolom dari matriks puzzle, dan variabel totCompare yang menunjukkan jumlah perbandingan kata yang dilakukan oleh program.

```
int N = 0; // jumlah baris
int M = 0; // jumlah kolom
int totCompare = 0;
```

Gambar 2.3 Source Program 3

Gambar 2.4 menunjukkan variabel global yang digunakan pada program penyelesaian word search puzzle. Variabel wordlist yang memiliki tipe vektor yang bertipe string berfungsi untuk menyimpan daftar kata yang harus dicari pada puzzle. Variabel answers yang bertipe vektor dua dimensi berfungsi untuk menyimpan jawaban yang telah dihasilkan dari fungsi search2D. Variabel temp bertipe vektor dua dimensi yang berfungsi untuk menyimpan vektor yang yang memiliki ukuran sama dengan puzzle tetapi berisi karakter "-" untuk setiap kolom dan barisnya. Variabel puzzle bertipe vektor dua dimensi yang berisi puzzle yang berasal dari input file.

```
vector<string> wordlist;
vector<vector<char>> answers;
vector<vector<char>> temp;
vector<vector<char>> puzzle;
```

Gambar 2.4 Source Program 4

Gambar 2.5 berisi inisialisasi dari fungsi-fungsi yang akan digunakan pada program implementasi penyelesaian *word search puzzle*.

```
void bacaDaftarKata(string filename);
void bacaPuzzle(string filename);
void cariKata(vector<vector<char>> puzzle, vector<string> word);
void printKata(string filename);
void printPuzzles();
void printHasil();
void printDaftarKata();
void searchAnswers();
```

Gambar 2.5 Source Program 5

Gambar 2.6 menunjukkan fungsi *bacaPuzzle* yang memiliki fungsi untuk membaca file input dan mengisikan nilainya ke variabel *puzzle* dan *answers*.

```
void bacaPuzzle(string filename){
         ifstream pzlReader(filename);
         string line;
         if (pzlReader.is_open()){
             while (getline(pzlReader, line)){
                 if(line.size() == 1){
                     break;
                 if (N == 0){
                     M = (line.size()+1)/2;
                 N++;
                 vector <char> row;
                 vector <char> row1;
                 for (int i=0; i < line.size(); i+=2){
                     row.push_back(line[i]);
                     row1.push_back('-');
                 answers.push_back(row1);
                 puzzle.push_back(row);
             pzlReader.close();
140
```

Gambar 2.6 Source Program 6

Gambar 2.7 menunjukkan fungsi *printPuzzles* yang berfungsi untuk menampilkan *puzzle* yang sebelumnya nilainya sudah diisi ketika fungsi *bacaPuzzle* dipanggil.

```
144
      void printPuzzles(){
146
          for (int i=0; i< N; i++){
147
              vector<char> row = puzzle[i];
148
              for (int j = 0; j < M; j++){
149
                   cout<< row[j] << " ";
150
151
              cout << endl;</pre>
152
153
          }
154
```

Gambar 2.7 Source Program 7

Gambar 2.8 menunjukkan fungsi *printHasil* yang berfungsi untuk menampilkan *answers* ke layar yang sebelumnya nilainya telah diisi ketika fungsi search 2D telah dipanggil.

```
156  void printHasil(){
157
158     for(int i = 0; i < N; i++){
159         vector <char> row1 = answers[i];
160         for(int j = 0; j < M; j++){
161             cout<< row1[j]<<" ";
162         }
163             cout<<endl;
164         }
165    }</pre>
```

Gambar 2.8 Source Program 8

Gambar 2.9 menunjukkan fungsi *bacaDaftarKata* yang memiliki fungsi untuk mengisikan nilai dari variabel *wordlist* dari input file.

```
// FUNGSI UNTUK MEMBACA DAFTAR KATA YANG HARUS DICARI DARI INPUT FILE
     void bacaDaftarKata(string filename){
          ifstream ktReader(filename);
170
171
         string kata;
173
          if (ktReader.is_open()){
175
              while (getline(ktReader, kata)){
176
                  if (kata.size() == 1){
                      while(getline(ktReader, kata)){
179
                          wordlist.push back(kata);
                      }
                  }
              ktReader.close();
184
          }
```

Gambar 2.9 Source Program 9

Gambar 2.10 menunjukkan fungsi *printDaftarKata* yang memiliki fungsi untuk menampilkan nilai dari *wordlist* ke layar yang mana nilai nya telah diisi ketika fungsi *bacaDaftarKata* telah dipanggil.

```
// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

// FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR

/
```

Gambar 2.10 Source Program 10

Gambar 2.11 sampai Gambar 2.14 menunjukkan fungsi-fungsi utama yang digunakan untuk mengimplementasikan program penyelesaian *word search puzzle*, yang mana fungsi fungsi tersebut khususnya fungsi *search2D* dan fungsi *cariKata* menerapkan algoritma *brute force*.

```
197 int x[] = { -1, -1, -1, 0, 0, 1, 1, 1 };
198 int y[] = { -1, 0, 1, -1, 1, -1, 0, 1 };
```

Gambar 2.11 Source Program 11

```
bool search2D(vector<vector<char>> pzl, int row, int col, string word, int z){
    temp = answers;
    if(pzl[row][col] != word[0]){
        answers = temp;
        totCompare++;
    int len;
   if(z != wordlist.size()-1){
        len = word.length()-1;
    }else{
        len = word.length();
    for (int dir = 0; dir < 8; dir++){
        int k, rd = row + x[dir], cd = col + y[dir];
        for(k = 1; k < len; k++){}
            if (rd >= N || rd < 0 || cd >= M || cd < 0){
                answers = temp;
                totCompare++;
                break;
            if (pzl[rd][cd] != word[k]){
                answers = temp;
                totCompare++;
```

Gambar 2.12 Source Program 12

```
answers[row][col] = pzl[row][col];
answers[rd][cd] = pzl[rd][cd];
answers[rd][cd] = pzl[rd][cd];
answers[rd][cd] = pzl[rd][cd];
answers[rd][cd] = pzl[row][col];
answers[rd][cd] = pzl[row][col];
answers[row][col] = pzl[row][col] = pzl[row][col
```

Gambar 2.13 Source Program 13

```
void cariKata(vector<vector<char>> pzl, vector<string> wordlist){
    bool status = false;
    for(int i = 0; i < wordlist.size(); i++){</pre>
        auto start = high_resolution_clock::now();
        for (int row = 0; row < N; row++){</pre>
             for (int col = 0; col < M; col++){
                 if (search2D(puzzle, row, col, wordlist[i], i)){
                     status = true;
                     cout<<wordlist[i]<<endl;</pre>
                     cout<<endl;</pre>
                     printHasil();
                     cout<<endl;</pre>
                     auto stop = high_resolution_clock::now();
                     auto duration = duration_cast<microseconds>(stop - start);
                     cout << "Waktu eksekusi: "<< duration.count()/1000 << " milisekon" << endl;</pre>
                     cout << "Total perbandingan huruf: "<< totCompare <<endl;</pre>
                     totCompare = 0;
                     cout<<endl;</pre>
                     answers = temp;
```

Gambar 2.14 Source Program 14

Gambar 2.15 dan Gambar 2.16 menunjukkan fungsi *main* yang berfungsi untuk memanggil semua fungsi yang sebelumnya telah dibuat.

```
string filepath;
string finalpath;
cout<<"input path file: ";</pre>
cin>>filepath;
finalpath = "../test/"+filepath;
if (std::filesystem::exists(finalpath)){
   bacaPuzzle(finalpath);
   cout<<endl;</pre>
   cout<<endl;</pre>
   printPuzzles();
   cout<<endl;</pre>
   cout<<"========" <<endl;</pre>
   bacaDaftarKata(finalpath);
   cout<<endl;</pre>
   printDaftarKata();
   cout<<endl;</pre>
   cout<<"======== <<endl;
   cout<<endl;</pre>
   cout<<"Loading..."<<endl;</pre>
   cout<<endl;</pre>
   cariKata(puzzle, wordlist);
```

Gambar 2.15 Souce Code 15

Gambar 2.16 Source Code 16

## BAB III Hasil Program

### 1. Input



Gambar 3.1 Input Nama File

```
SUNARUUA
   NEPTUNET
   SONIEISU
   RCEVTRER
   AHTRAESN
   MMERCURY
9
   EARTH
   JUPITER
11
   MARS
12
   MERCURY
13
   NEPTUNE
14
   SATURN
   URANUS
  VENUS
```

Gambar 3.2 Contoh Susunan File yang Diinput

### 2. Output

```
===========PUZZLE==========
JSOLUTIS
SUNARUUA
NEPTUNET
SONIEISU
RCEVTRER
AHTRAESN
MMERCURY
EARTH
JUPITER
MARS
MERCURY
NEPTUNE
SATURN
URANUS
VENUS
```

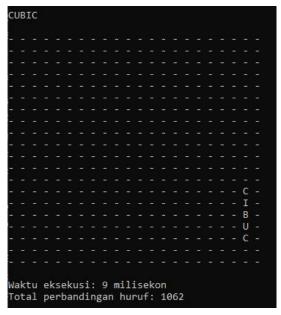
Gambar 3.3 Output 1 test.txt

Gambar 3.4 Salah Satu Output test.txt (Small)

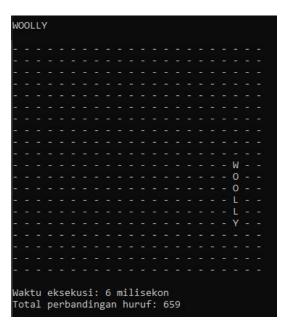
Gambar 3.5 Salah Satu Output test4.txt (Small)

Gambar 3.6 Salah Satu Output test5.txt (Small)

Gambar 3.7 Salah Satu Output test6.txt (Small)



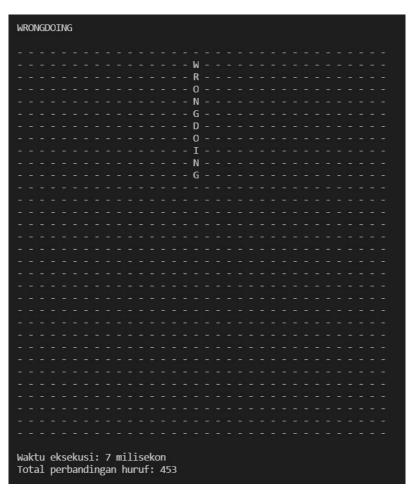
Gambar 3.8 Salah Satu Output test1.txt (Medium)



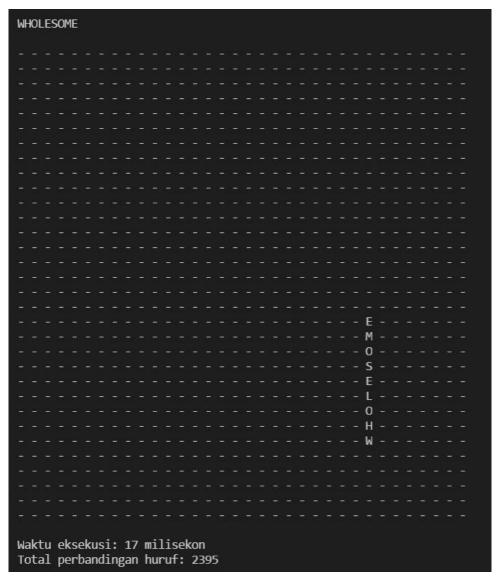
Gambar 3.9 Salah Satu Output test2.txt (Medium)



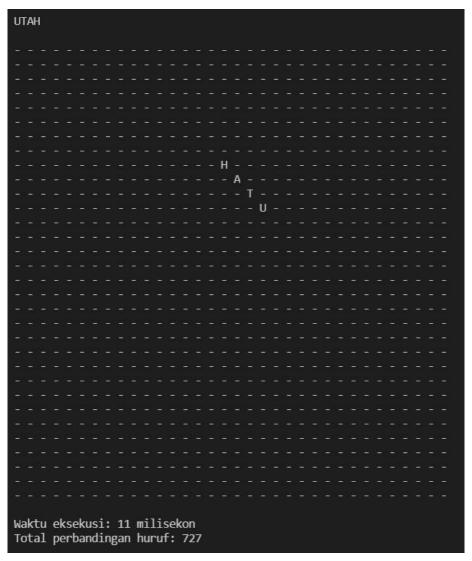
Gambar 3.10 Salah Satu Output test3.txt (Medium)



Gambar 3.11 Salah Satu Output test7.txt (Large)



Gambar 3.12 Salah Satu Output test8.txt (Large)



Gambar 3.13 Salah Satu Output test9.txt (Large)

## BAB IV Kesimpulan

Kesimpulan yang dapat diambil dari implementasi program penyelesaian word search puzzle adalah algoritma brute force dapat digunakan dengan baik pada program penyelesaian word search puzzle.

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

Source Code dari program ini dapat diakses pada link berikut : <a href="https://github.com/ZianTsabit/Tucil-Stima-Puzzle-Word">https://github.com/ZianTsabit/Tucil-Stima-Puzzle-Word</a>