

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

BAB I

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*.

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <fstream>
5  #include <algorithm>
6  #include <ostream>
7  #include <cstdlib>
8  #include <filesystem>
9  #include <stdlib.h>
10 #include <unordered_set>
11 #include <set>
12 #include <cstdio>
13 #include <chrono>
```

Gambar 2.1 Source Program 1

```
15 using namespace std;
16 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.

```
18 int N = 0; // jumlah baris
19 int M = 0; // jumlah kolom
20 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 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*.

```
22 vector<string> wordlist;  
23 vector<vector<char>> answers;  
24 vector<vector<char>> temp;  
25 vector<vector<char>> puzzle;
```

Gambar 2.4 Source Program 4

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

```
27 void bacaDaftarKata(string filename);  
28 void bacaPuzzle(string filename);  
29 void cariKata(vector<vector<char>> puzzle, vector<string> word);  
30 void printKata(string filename);  
31 void printPuzzles();  
32 void printHasil();  
33 void printDaftarKata();  
34 void searchAnswers();
```

Gambar 2.5 Source Program 5

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

```

111 void bacaPuzzle(string filename){
112     ifstream pzlReader(filename);
113
114     string line;
115
116     if (pzlReader.is_open()){
117         while (getline(pzlReader, line)){
118
119             if(line.size() == 1){
120                 break;
121             }
122
123             if (N == 0){
124                 M = (line.size()+1)/2;
125             }
126             N++;
127
128             vector <char> row;
129             vector <char> row1;
130             for (int i=0; i < line.size(); i+=2){
131                 row.push_back(line[i]);
132                 row1.push_back('-');
133             }
134             answers.push_back(row1);
135             puzzle.push_back(row);
136         }
137         pzlReader.close();
138     }
139 }
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(){
145
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;
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 }
```

Gambar 2.8 Source Program 8

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

```
167 // FUNGSI UNTUK MEMBACA DAFTAR KATA YANG HARUS DICARI DARI INPUT FILE
168
169 void bacaDaftarKata(string filename){
170     ifstream ktReader(filename);
171
172     string kata;
173
174     if (ktReader.is_open()){
175
176         while (getline(ktReader, kata)){
177             if (kata.size() == 1){
178                 while(getline(ktReader, kata)){
179                     wordlist.push_back(kata);
180                 }
181             }
182         }
183         ktReader.close();
184     }
185 }
```

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.

```

187 // FUNGSI UNTUK MENAMPILKAN DAFTAR KATA YANG HARUS DICARI KE LAYAR
188
189 void printDaftarKata(){
190     for (int i=0; i< wordlist.size(); i++){
191         cout<<wordlist[i]<<endl;
192     }
193 }

```

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

```

201 bool search2D(vector<vector<char>> pzl, int row, int col, string word, int z){
202
203     temp = answers;
204
205     if(pzl[row][col] != word[0]){
206         answers = temp;
207         totCompare++;
208         return false;
209     }
210     int len;
211     if(z != wordlist.size()-1){
212         len = word.length()-1;
213     }else{
214         len = word.length();
215     }
216
217
218     for (int dir = 0; dir < 8; dir++){
219         int k, rd = row + x[dir], cd = col + y[dir];
220         for(k = 1; k < len; k++){
221             if (rd >= N || rd < 0 || cd >= M || cd < 0){
222                 answers = temp;
223                 totCompare++;
224                 break;
225             }
226             if (pzl[rd][cd] != word[k]){
227                 answers = temp;
228                 totCompare++;
229                 break;

```

Gambar 2.12 Source Program 12


```

231         answers[row][col] = pz1[row][col];
232         answers[rd][cd] = pz1[rd][cd];
233         rd += x[dir], cd += y[dir];
234         totCompare++;
235     }
236     totCompare++;
237     if(k == len){
238         k = 1;
239         dir = 0;
240         return true;
241     }
242 }
243 return false;
244 }

```

Gambar 2.13 Source Program 13

```

246 void cariKata(vector<vector<char>> pz1, vector<string> wordlist){
247
248     bool status = false;
249
250     for(int i = 0; i < wordlist.size(); i++){
251         auto start = high_resolution_clock::now();
252         for (int row = 0; row < N; row++){
253             for (int col = 0; col < M; col++){
254                 if (search2D(puzzle, row, col, wordlist[i], i)){
255                     status = true;
256                     cout<<wordlist[i]<<endl;
257                     cout<<endl;
258                     printHasil();
259                     cout<<endl;
260                     auto stop = high_resolution_clock::now();
261                     auto duration = duration_cast<microseconds>(stop - start);
262                     cout << "Waktu eksekusi: " << duration.count()/1000 << " milisekon" << endl;
263                     cout << "Total perbandingan huruf: " << totCompare << endl;
264                     totCompare = 0;
265                     cout<<endl;
266                     answers = temp;
267                 }
268             }
269         }
270     }
271 }

```

Go to Line/Column

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.

```
65     string filepath;
66     string finalpath;
67     cout<<"input path file: ";
68     cin>>filepath;
69     finalpath = "../test/"+filepath;
70
71     if (std::filesystem::exists(finalpath)){
72         bacaPuzzle(finalpath);
73
74         cout<<endl;
75         cout<<"=====PUZZLE===== "<<endl;
76         cout<<endl;
77         printPuzzles();
78
79
80         cout<<endl;
81         cout<<"=====WORD LIST===== " <<endl;
82
83         bacaDaftarKata(finalpath);
84         cout<<endl;
85         printDaftarKata();
86         cout<<endl;
87         cout<<"=====RESULT===== " <<endl;
88         cout<<endl;
89         cout<<"Loading..."<<endl;
90         cout<<endl;
91
92         cariKata(puzzle, wordlist);
```

Gambar 2.15 Souce Code 15

```
95     }else{
96         cout<<"File tidak ditemukan, pastikan nama file sudah tepat !"<<endl;
97     }
98
99     //system("pause");
100    char ch;
101    while (cin.readsome(&ch, 1) != 0)
102    ;
103    cout << "Press any key to continue..."<<endl;
104    getchar();
105    getchar();
106    return 0;
107 }
```

Gambar 2.16 Source Code 16

BAB III

Hasil Program

1. Input



Gambar 3.1 Input Nama File

1	J S O L U T I S
2	S U N A R U U A
3	N E P T U N E T
4	S O N I E I S U
5	R C E V T R E R
6	A H T R A E S N
7	M M E R C U R Y
8	
9	EARTH
10	JUPITER
11	MARS
12	MERCURY
13	NEPTUNE
14	SATURN
15	URANUS
16	VENUS

Gambar 3.2 Contoh Susunan File yang Diinput

2. Output

```
=====PUZZLE=====
J S O L U T I S
S U N A R U U A
N E P T U N E T
S O N I E I S U
R C E V T R E R
A H T R A E S N
M M E R C U R Y

=====WORD LIST=====
EARTH
JUPITER
MARS
MERCURY
NEPTUNE
SATURN
URANUS
VENUS
```

Gambar 3.3 Output 1 test.txt

```
=====RESULT=====
Loading...
EARTH
- - - - -
- - - - -
- - - - -
- - - - -
- H T R A E - -
- - - - -

Waktu eksekusi: 2 milisekon
Total perbandingan huruf: 132
```

Gambar 3.4 Salah Satu Output test.txt (Small)

```

LANDAK
- - - - -
- L - - - - -
- - A - - - - -
- - - N - - - - -
- - - - D - - - -
- - - - - A - - -
- - - - - - K -

Waktu eksekusi: 1 milisekon
Total perbandingan huruf: 124

```

Gambar 3.5 Salah Satu Output test4.txt (Small)

```

BECAK
- - K A C E B -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -

Waktu eksekusi: 2 milisekon
Total perbandingan huruf: 81

```

Gambar 3.6 Salah Satu Output test5.txt (Small)

```

KOL
- - - - -
- - - - -
- - - - -
- - - - -
- L - - - - -
- O - - - - -
- K - - - - -

Waktu eksekusi: 2 milisekon
Total perbandingan huruf: 54

```

Gambar 3.7 Salah Satu Output test6.txt (Small)

Waktu eksekusi: 17 milisekon
Total perbandingan huruf: 2395

Gambar 3.12 Salah Satu Output test8.txt (Large)

UTAH

Waktu eksekusi: 11 milisekon
Total perbandingan huruf: 727

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
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.	✓	

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