Algoritma Brute Force untuk Penyelesian *Word Puzzle*

Laporan Tugas Kecil

Oleh:

Muhammad Risqi Firdaus 13520043

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

A. Algoritma *Brute Force*

Pertama program akan membuat sebuah array dinamis menggunakan vector of vector of char untuk menyimpan matrix puzzle. Program dibaca perline, untuk tiap line yang dibaca program akan membaca panjang barisnya kemudian menyiapkan ukuran matrix yang sesuai. Setelah itu, program akan men-*split* string yang dibaca dari file ke char-char huruf untuk dimasukkan ke dalam matrix. Ketika membaca baris kosong program akan switch ke pembacaan kata. Kata akan disimpan di array dinamis (vector) dalam string.

Dalam pencarian kata, hal yang pertama dilakukan melakukan perulangan per kata yang ingin dicari. Perulangan dilakukan hingga menemukan kata atau telah men-scan semua huruf dari matrix. Dilakukan pengecekan perhuruf, jika huruf pertama pada kata ditemukan dalam matrix, maka akan dicek posisinya, apakah mencukupi untuk terdapat kata di sana. Jika posisi huruf pada matrix lebih panjang sama dengan panjang kata (ke salah satu arah mata angin) maka akan dilakukan penscanan satu persatu sesuai dengan arah mata angin yang ada, jika ditemukan kata yang sesuai pada penscanan satu per satu huruf dari salah satu arah mata angin, maka variabel found akan diubah menjadi true, sehingga akan keluar dari perulangan pencarian per huruf, dan kembali ke sequential kata, menuju ke kata berikutnya. Seterusnya dilakukan hingga semua kata ditemukan.

Dalam fungsi pencarian, akan di scan 1 per satu, jika semua huruf cocok hingga mencapai huruf terakhir dari kata yang dicapai, maka akan dicetak di layar. Sebelumnya untuk tiap fungsi, akan membuat sebuah array matrix yang berisi character ‘-’. Dalam menscan, untuk tiap huruf yang benar akan diubah dari ‘-’ menjadi karakter hurufnya. Jika ada perbedaan dari huruf matrix dan hurf pada kata, maka pencarian akan diberhentikan, dan tanda check akan diubah menjadi false, sehingga prosedur pencetakan akan dilompati dan berpindah ke pencarian pada arah mata angin lain. Jika hingga semua huruf katra dibandingkan dan hasilnya seusai, maka matrix ‘-’ dan huruf yang sesuai tadi akan diprint dan mengubah nilai found menjadi true, yang artinya akan keluar dari perulangan pencarian kaata.

B. Source Program

Program ditulis dengan bahasa C++ dengan bantuan modul iostream, fstream, string, vector, serta chrono. Pada program terdapat 8 fungsi pengecekan untuk tiap arah mata angin, 2 prosedur matrix (pencetakan dan pembuat matrix char), sertaz sebuah fungsi utama.
Berikut source code-nya

```
#include<iostream>
#include<fstream>
#include<string>
#include<vector>
#include <chrono>

using namespace std;
using namespace std::chrono;

void printVec(vector<vector <char>> vect){
   for(int i=0;i<vect.size();i++){
```

```
        for(int j=0;j<vect[i].size();j++){
            cout<<vect[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<"\n";
}

void makeAlp(vector<vector <char>> *vect){
    for(int i=0;i<(*vect).size();i++){
        for(int j=0;j<(*vect)[i].size();j++){
            (*vect)[i][j]='-';
        }
    }
}
bool cekKiri(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);
    int i=row;
    int j=col;

    while(k<word.length() and check){
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            j--;
            k++;
        }else{
            check=false;
        }
    }

    if(check){
        printVec(checker);
    }

    return check;
}
```

```
bool cekKiriAtas(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);

    int i=row;
    int j=col;

    while(k<word.length() and check){
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            j--;
            i--;
            k++;
        }else{
            check=false;
        }
    }

    if(check){
        printVec(checker);
    }

    return check;
}

bool cekKiriBawah(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);

    int i=row;
    int j=col;

    while(k<word.length() and check){
```

```
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            j--;
            i++;
            k++;
        }else{
            check=false;
        }
    }

    if(check){
        printVec(checker);
    }

    return check;
}

bool cekKanan(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);

    int i=row;
    int j=col;
    while(k<word.length() and check){
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            j++;
            k++;
        }else{
            check=false;
        }
    }

    if(check){
        printVec(checker);
    }
```

```
    return check;
}

bool cekKananAtas(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);

    int i=row;
    int j=col;

    while(k<word.length() and check){
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            j++;
            i--;
            k++;
        }else{
            check=false;
        }
    }

    if(check){
        printVec(checker);
    }

    return check;
}

bool cekKananBawah(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);

    int i=row;
    int j=col;
```

```
    while(k<word.length() and check){
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            j++;
            i++;
            k++;
        }else{
            check=false;
        }
    }

    if(check){
        printVec(checker);
    }

    return check;
}


bool cekBawah(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);

    int i=row;
    int j=col;

    while(k<word.length() and check){
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            i++;
            k++;
        }else{
            check=false;
        }
    }
```

```
    if(check){
        printVec(checker);
    }

    return check;
}

bool cekAtas(int row, int col, string word, vector<vector<char>> arrOfAlp){
    bool check = true;

    int k = 0;
    vector<vector <char>> checker (arrOfAlp.size(), vector<char> (arrOfAlp[0].size()));
    makeAlp(&checker);

    int i=row;
    int j=col;

    while(k<word.length() and check){
        if(arrOfAlp[i][j]==word[k]){
            checker[i][j]=arrOfAlp[i][j];
            i--;
            k++;
        }else{
            check=false;
        }
    }

    if(check){
        printVec(checker);
    }

    return check;
}

int main(){
    string input;
    cout<<"Masukkan namafile: ";
    cin>>input;
    ifstream f(input);
    string str;
```

```
vector<string> arrOfWords;
vector<vector<char>> arrOfAlp;
int i=0;
while(getline(f,str)&&str!=""){
    int col=0;
    vector<char> row;
    row=vector<char> ((str.length()+1)/2);
    for(int j=0; j<str.length();j++){
        if(str[j]!=' '){
            row[col]=str[j];
            col++;
        }
    }
    arrOfAlp.push_back(row);
}

//printVec(arrOfAlp);
while(getline(f,str)){
    arrOfWords.push_back(str);
}

f.close();
auto start = high_resolution_clock::now();

for(int i=0;i<arrOfWords.size();i++){
    int itc=0;
    int itr =0;
    bool found = false;
    while(itr<arrOfAlp.size() and !found){
        if(arrOfAlp[itr][itc]==arrOfWords[i][0]){
            if(itc+1>=arrOfWords[i].size()){
                found = cekKiri(itr,itc,arrOfWords[i],arrOfAlp);
                if(itr+1>=arrOfWords[i].length() && !found){
                    found=cekKiriAtas(itr,itc,arrOfWords[i],arrOfAlp);
                }

                if(arrOfAlp.size()-itr>arrOfWords[i].length() && !found){
                    found=cekKiriBawah(itr,itc,arrOfWords[i],arrOfAlp);
                }
            }
```

```
            if(arrOfAlp[itr].size()-itc>=arrOfWords[i].length() && !found){
                found = cekKanan(itr,itc,arrOfWords[i],arrOfAlp);
                if(itr+1>=arrOfWords[i].length() && !found){
                    found=cekKananAtas(itr,itc,arrOfWords[i],arrOfAlp);
                }

                if(arrOfAlp.size()-itr>=arrOfWords[i].length() && !found){
                    found=cekKananBawah(itr,itc,arrOfWords[i],arrOfAlp);
                }
            }

            if(itr>=arrOfWords[i].length()-1 && !found){
                found = cekAtas(itr,itc,arrOfWords[i],arrOfAlp);
            }

            if(arrOfAlp.size()-itr>arrOfWords[i].length() && !found){
                found = cekBawah(itr,itc,arrOfWords[i],arrOfAlp);
            }

            if(itc==arrOfAlp[itr].size()-1){
                itc=0;
                itr++;
            }else{
                itc++;
            }

        }else{
            if(itc==arrOfAlp[itr].size()-1){
                itc=0;
                itr++;
            }else{
                itc++;
            }
        }
    }
}
auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);

cout << "Time taken by function: "
```

```
        << duration.count() << " microseconds" << endl;
    return 0;
}
```

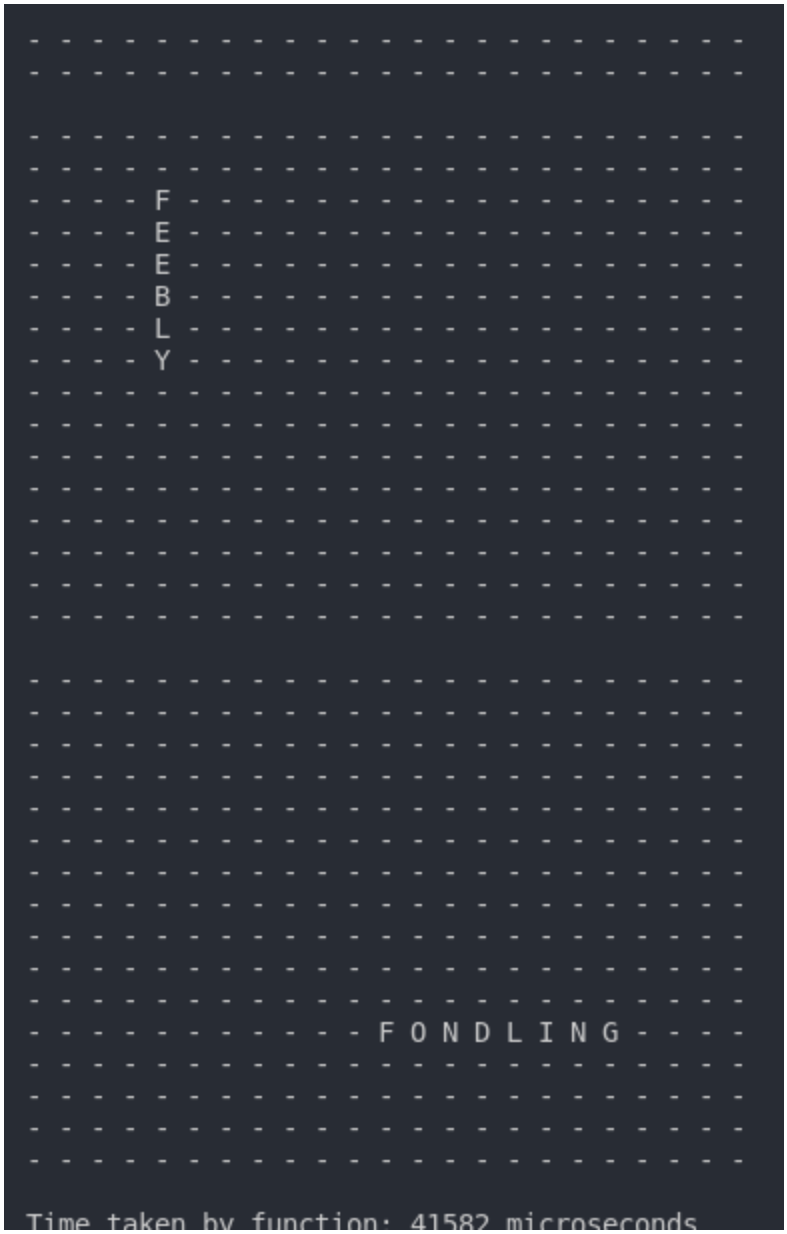
C. Source Program

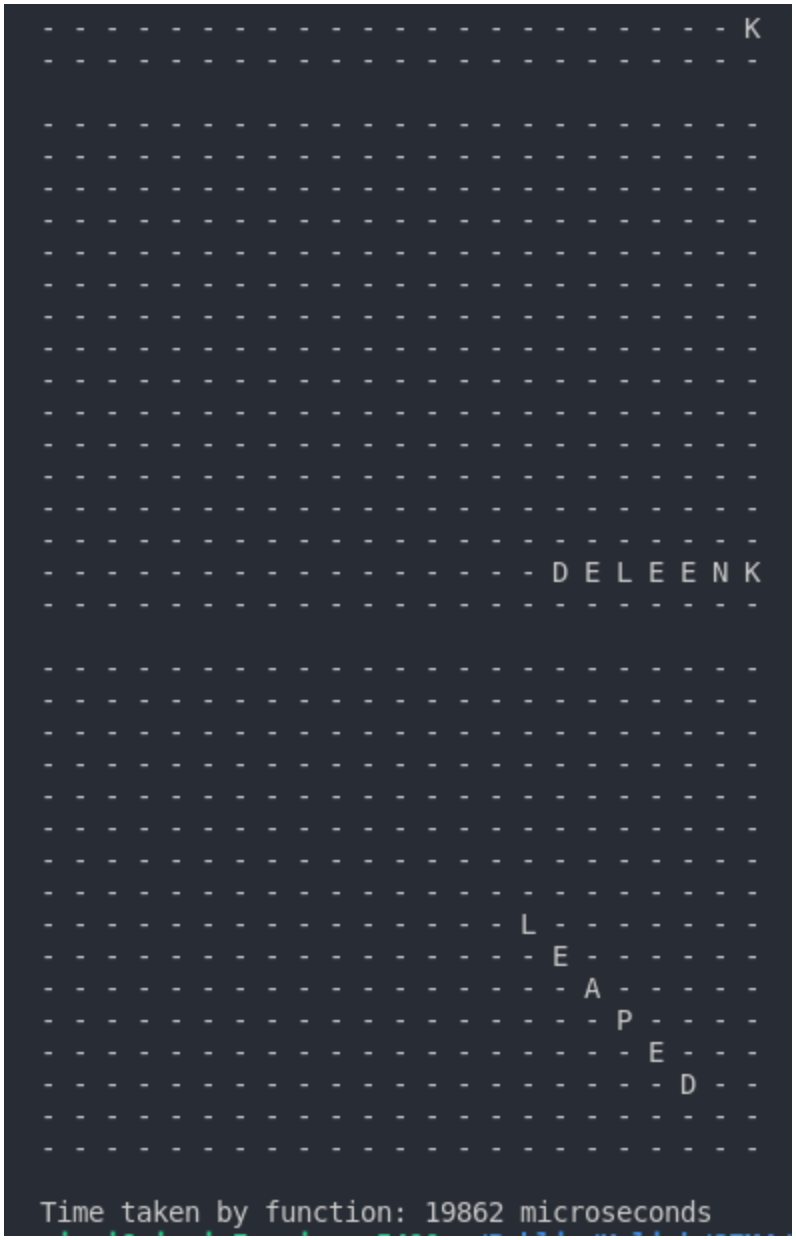
1. Small

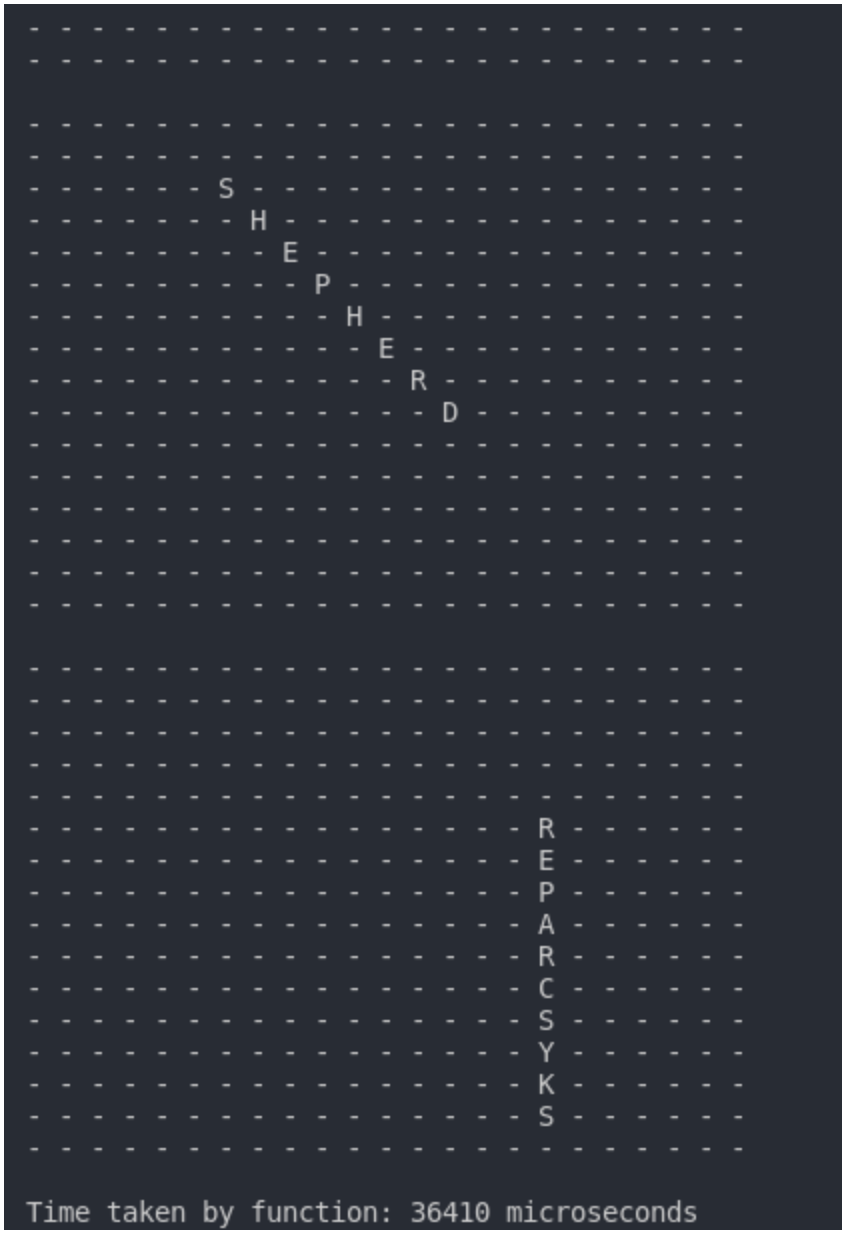

N
G
G
R
A
N
D
F
A
T
H
E
R
H I D E O U T
Time taken by function: 5222 microseconds


INDIFFERENCE
K
E
E
P
I
N
G
LOOSENED
Time taken by function: 8427 microseconds

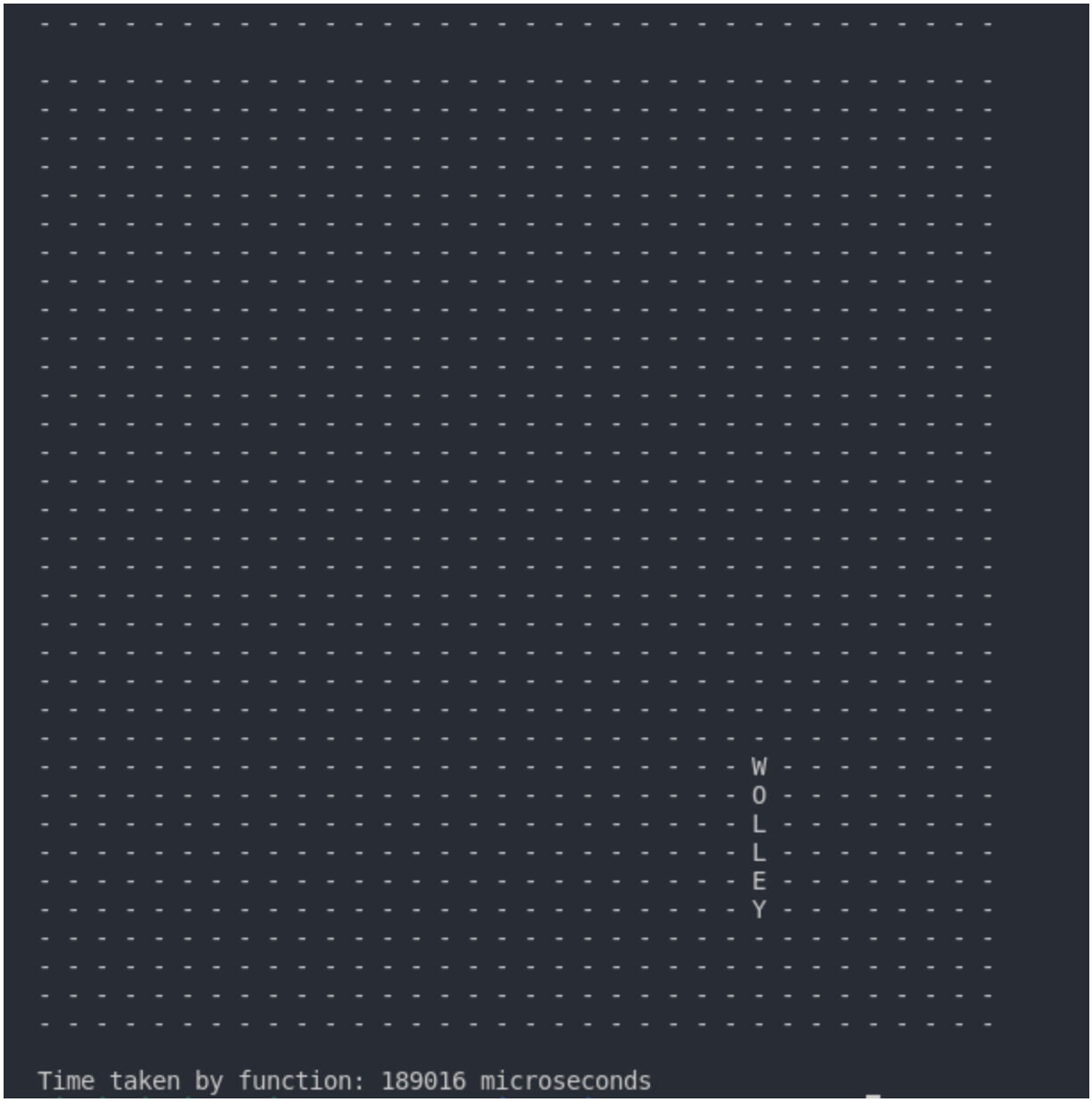
2. **Medium**

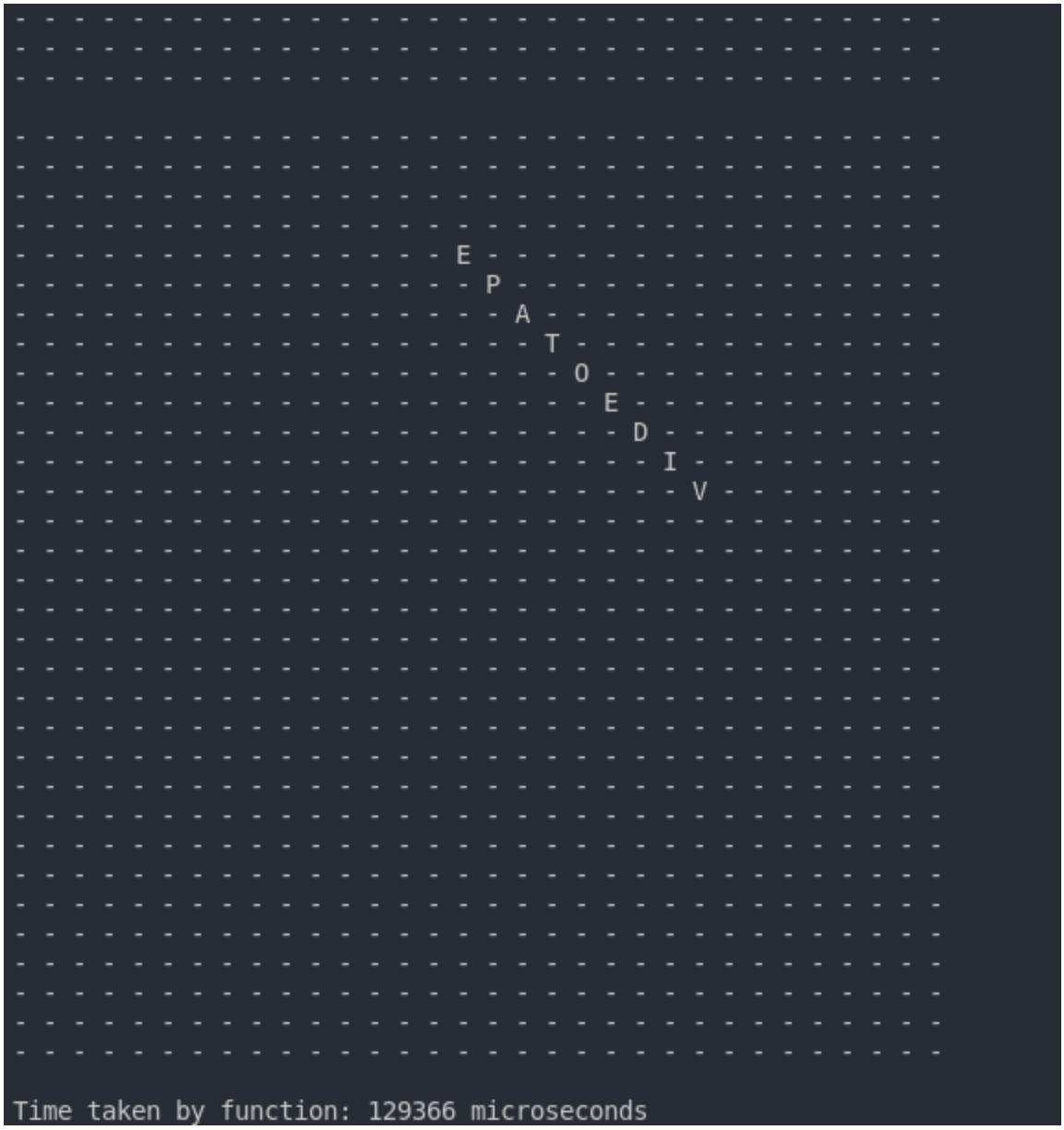

F
E
E
B
L
Y
F O N D L I N G
Time taken by function: 41582 microseconds


K
D E L E E N K
L
E
A
P
E
D
Time taken by function: 19862 microseconds

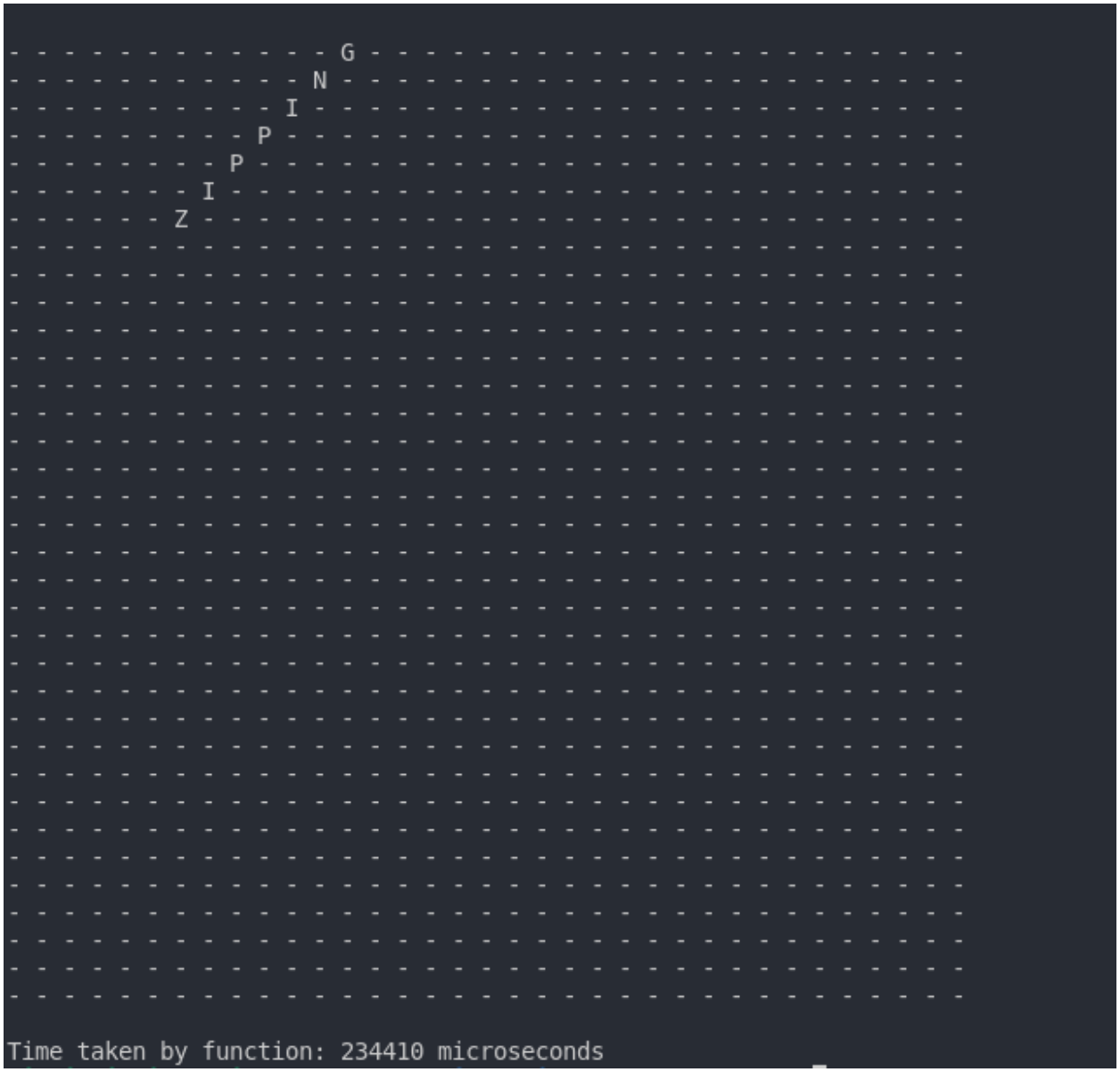

S
H
E
P
H
E
R
D
R
E
P
A
R
C
S
Y
K
S
Time taken by function: 36410 microseconds

3. **Large**


W
O
L
L
E
Y
Time taken by function: 189016 microseconds



Time taken by function: 129366 microseconds



Link G-Drive:
https://drive.google.com/drive/folders/1vpCdk3ggYHLFHEFDJWMva8oBYOP1oFNm?usp=sharing

Checklist

a. Pembagian tugas masing-masing anggota kelompok

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa	V	

kesalahan (no syntax error)		
2. Program Berhasil *running*	V	
3. Program dapat membaca file masukan dan menulis luaran	V	
4. Program berhasil menemukan semua kata dalam puzzle	V	