Laporan Tugas Kecil 1 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Semester II Tahun 2023/2024

oleh

Andhika Tantyo Anugrah 13522094

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

Daftar Isi	2
BAB 1 Algoritma Bruteforce	3
BAB 2 Source Code	4
BAB 3 Eksperimen	8
BAB 4 Pranala ke Repository	11
BAB 5 Refleksi	12
Daftar Pustaka	13
Lampiran	14

BAB 1
Algoritma *Bruteforce*

Algoritma *brute force* adalah algoritma yang menyelesaikan suatu persoalan dengan cara mencari semua kemungkinan solusi yang mungkin hanya dengan bergantung pada kekuatan komputasi yang besar. Oleh karena itu, algoritma *brute force* juga sering disebut *complete search* atau *exhaustive search*. Walaupun cara untuk mencarinya terlihat tidak efisien, karakteristik solusi yang dihasilkan dari algoritma ini pasti optimal. Kompleksitas waktu dari algoritma dalam konteks pencarian larik adalah O(mn), linear jika salah satu variabelnya konstan dan kuadratik jika kedua variabelnya bertambah.

Langkah-langkah algoritma *brute force* yang digunakan untuk menyelesaikan permasalahan *mini-game Cyberpunk 2077 Breach Protocol*:

1. Iterasi secara horizontal pertama dilakukan pada baris pertama, selanjutnya cari semua kombinasi sekuens secara vertikal dan dilanjutkan secara selang-seling.
2. Pada proses pencarian, sekuens dengan skor terbesar (lama) akan dibandingkan dengan sekuens yang baru ditemukan, jika sekuens dengan skor terbesar lebih kecil dibandingkan yang baru, maka akan dilakukan penggantian. Sedangkan jika sekuens lama dan baru memiliki skor yang sama, pilih sekuens yang memiliki ukuran yang lebih pendek agar solusi menjadi lebih optimal.
3. Ulangi terus sampai semua kemungkinan sekuens ditemukan. Pada akhirnya, sekuens dengan skor terbesar dan ukuran yang lebih pendek akan menjadi sekuens yang paling optimal.

BAB 2
Source Code

```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include <vector>
#include <chrono>
#include <algorithm>
using namespace std;
using namespace std::chrono;

#define buff info[0]
#define width info[1]
#define height info[2]
#define nseq info[3]

typedef long long ll;
typedef struct{
    vector<string> seqLine;
    ll reward;
} Seq;
typedef struct{
    string str;
    bool seen;
} Elmt;

class Sequence{
    public:
        vector<ll> steps;
        ll reward;
        Sequence(string t, vector<ll> s, ll r){
            steps = s;
            reward = r;
        }
};

// Global
vector<ll> info;
vector<vector<string>> mat;
vector<Seq> sequences;
vector<string> uniqToken;
vector<ll> steps; // Empty arr
Sequence optimum("", steps, 0);

void getPrompt(bool save, char *input, ll *normInput){
    do{
        printf((save) ? ("\n\nApakah ingin menyimpan solusi? (y/n) ") : ("Input dengan file? (y/n) "));
        cin >> *input;
        *normInput = tolower(*input);
    }while(*normInput != 121 && *normInput != 110);
    // Note: 'y' is 121 and 'n' is 110
}

void getFileName(string *path){
    // assumes the file's location is always in the test folder
    *path = "../test/";

    string temp;
    printf("Masukkan nama file: ");
    cin >> temp;
    *path = *path + temp + ".txt";
    // assumes the file's extension is always .txt
}

void writeCoordinate(bool file, ofstream *MyFile, vector<ll> steps){
    if(!file){
        for(size_t i = 0; i < steps.size(); i++){
            printf("%lld,%lld\n", steps[i % width] + 1, steps[i / width] + 1);
        }
    }
    else{
        for(size_t i = 0; i < steps.size(); i++){
            *MyFile << steps[i % width] + 1 << "," << steps[i / width] + 1 << "\n";
        }
    }
}

void writeRes(bool file, ofstream *MyFile, vector<ll> steps, duration<long long int, std::ratio<1, 1000>> duration){
    if(file){
        *MyFile << optimum.reward << "\n";
            if(optimum.reward > 0){
                for(size_t i = 0; i < optimum.steps.size(); i++){
                    *MyFile << mat[optimum.steps[i] / width][optimum.steps[i] % width] <<   " ";
                }
                *MyFile << "\n";
                writeCoordinate(file, MyFile, optimum.steps);
            }
            *MyFile << "\n" << duration.count() << " ms";
    }
    else{
        cout << optimum.reward << "\n";
        if(optimum.reward > 0){
            for(size_t i = 0; i < optimum.steps.size(); i++){
                cout << mat[optimum.steps[i] / width][optimum.steps[i] %width] <<   " ";
```

```
            }
            cout << "\n";
            writeCoordinate(file, MyFile, optimum.steps);
        }
        cout << "\n" << duration.count() << " ms";
    }
}

void tallyScore(vector<ll> path){
    ll score = 0;
    string str = "";
    for(size_t i = 0; i < path.size(); i++){
        str.append(mat[path[i] / width][path[i] % width]);
    }
    for(size_t i = 0; i < sequences.size(); i++){
        string temp = "";
        for(size_t j = 0; j < sequences[i].seqLine.size(); j++){
            temp.append(sequences[i].seqLine[j]);
        }
        if(str.find(temp) != string::npos){
            score += sequences[i].reward;
        }
    }
    if(optimum.reward < score){
        optimum.reward = score;
        optimum.steps = path;
    }
    if(optimum.reward == score && path.size() < optimum.steps.size()){
        optimum.reward = score;
        optimum.steps = path;
    }
}

void solver(vector<ll> path){
    ll mod;
    bool isVertical = true;
    if(!path.size()){
        for(ll i = 0;i < width; i++){
            path = {};
            path.push_back(i);
            solver(path);
        }
    }
    else if(path.size() == (size_t) buff){
        tallyScore(path);
    }
    else if(isVertical){
        mod = path.back() % width;
        for(ll i = 0; i < height; i++){
            vector<ll> newPath;
            if(find(path.begin(), path.end(), (width * i) + mod) == path.end()){
                newPath = path;
                newPath.push_back((width * i) + mod);
                solver(newPath);
            }
        }
        isVertical = false;
        tallyScore(path);
    }
    else if(!isVertical){
        mod = (path.back() / width) * width;
        for(ll i = mod; i < mod + width; i++){
            vector<ll> newPath;
            if(find(path.begin(), path.end(), i) == path.end()){
                newPath = path;
                newPath.push_back(i);
                solver(newPath);
            }
        }
        isVertical = true;
        tallyScore(path);
    }
}

int main(){
    system("CLS");
    char input;
    ll normInput;
    string path, line;
    repeat:
        getPrompt(0, &input, &normInput);
    if(normInput == 121){ // Input by file
        getFileName(&path);
        ifstream inputFile(path);
        if(inputFile.is_open()){
            ll i = 0;
            // Read game's metadata
            while(i < 3 && getline(inputFile, line)){
                stringstream ss(line);
                ll temp;
                while(ss >> temp){
                    info.push_back(temp);
                    // info[1] shall be the width and info[2] shall be the height
                    i++;
                }
            }
```

```
            printf("Buffer size: %lld\n", buff);
            printf("Width: %lld\n", width);
            printf("Height: %lld\n", height);

            // Read the matrix
            mat.resize(height, vector<string>(width));
            for(i = 0; i < height; i++){
                getline(inputFile, line);
                for(ll j = 0; j < width; j++){
                    mat[i][j] = line.substr(3*j, 2);
                    /*  "7A 55 E9 E9"
                         ^  ^  ^  ^
                        pos: 0123456789
                        this way of input has a weakness: cannot read false input -> program crashes immediately
                        */
                    cout << mat[i][j] << " ";
                }
                printf("\n");
            }
            // Read the seq's count
            getline(inputFile, line);
            stringstream ss(line);
            ss >> info[3];
            // info[3] shall be the nseq
            printf("Number of sequences: %lld\n", nseq);

            // Read the sequences
            sequences.resize(nseq);
            for(i = 0; i < nseq; i++){
                getline(inputFile, line);
                for(size_t j = 0; j < (line.length() + 1)/3; j++){
                    string temp = line.substr(3*j, 2);
                    sequences[i].seqLine.push_back(temp);
                    cout << sequences[i].seqLine[j] << " ";
                }
                printf("\n");
                getline(inputFile, line);
                stringstream ss(line);
                ss >> sequences[i].reward;
                printf("Reward: %lld\n", sequences[i].reward);
            }
            printf("\n");
            inputFile.close();
        }
        else{
            system("CLS");
            printf("An attempt to read the file failed!\n");
            goto repeat;
        }
    }
    else{ // Random input, need generator
        ll nToken, buffTemp, widthTemp, heightTemp, nSeqTemp, maxSeq;
        printf("Jumlah token unik: ");
        scanf("%lld", &nToken);
        cin.ignore();
        getline(cin, line);
        for(ll i = 0; i < nToken; i++){
            uniqToken.push_back(line.substr(3*i, 2));
        }
        printf("Ukuran buffer: ");
        scanf("%lld", &buffTemp);
        printf("Lebar dan tinggi matrix: ");
        scanf("%lld %lld", &widthTemp, &heightTemp);
        // cin >> widthTemp >> heightTemp;
        printf("Jumlah sekuens: ");
        scanf("%lld", &nSeqTemp);
        printf("Ukuran sekuens maksimal: ");
        scanf("%lld", &maxSeq);
        info.push_back(buffTemp); info.push_back(widthTemp);
        info.push_back(heightTemp); info.push_back(nSeqTemp);

        srand(time(nullptr));
        string token[nToken];
        sequences.resize(nseq);
        for(ll i = 0; i < height; i++){
            vector<string> temp = {};
            for(ll j= 0;j < width; j++){
                temp.push_back(token[rand() % nToken]); // 0 to nToken-1
            }
            mat.push_back(temp);
        }

        for(ll i = 0; i < nseq; i++){
            string temp = "";
            ll randTokenLength = rand() % maxSeq + 1; // 1 to maxSeq
            for(ll j = 0; j < randTokenLength; j++){
                temp += token[rand() % nToken];       // 0 to nToken-1
            }
            sequences[i].seqLine.push_back(temp);
            sequences[i].reward = rand() % 100 + 1;   // 1 to 100
        }


        // Testing
        // getFileName(&path);
        // ofstream MyFile(path);
```

```
            }
            mat.push_back(temp);
        }

        for(ll i = 0; i < nseq; i++){
            string temp = "";
            ll randTokenLength = rand() % maxSeq + 1; // 1 to maxSeq
            for(ll j = 0; j < randTokenLength; j++){
                temp += token[rand() % nToken];       // 0 to nToken-1
            }
            sequences[i].seqLine.push_back(temp);
            sequences[i].reward = rand() % 100 + 1;   // 1 to 100
        }


        // Testing
        // getFileName(&path);
        // ofstream MyFile(path);
        // MyFile << nToken << "\n" << buffTemp << "\n";

        // for(size_t i = 0; i < uniqToken.size(); i++){
        //     MyFile << uniqToken[i] << " ";
        // }

        // MyFile << "\n" << widthTemp << " " << heightTemp << "\n" << nSeqTemp << "\n" << maxSeq << "\n";
        // MyFile.close();
    }
    // Timer start
    auto start = high_resolution_clock::now();

    // Bruteforce
    for(ll i = 0; i < width; i++){
        solver({});
    }

    // Timer stop
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<milliseconds>(stop - start);

    printf("Score maksimum: %lld\n", optimum.reward);
    writeRes(false, NULL, steps, duration);

    // Write to file?
    getPrompt(1, &input, &normInput);
    if(normInput == 121){
        getFileName(&path);
        ofstream MyFile(path);
        writeRes(true, &MyFile, steps, duration);
        MyFile.close();
    }
}
```

BAB 3
Eksperimen

No.	Gambar Input	Gambar Output
1.	7 6 6 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BD 7A 1C 1C 1C 55 55 7A 55 7A 3 BD E9 1C 15 BD 7A BD 20 BD 1C BD 55 30	30 7A BD 1C BD 55 1,1 19,1 31,1 25,1 7,1 25 ms
2.	7 6 6 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BD 7A 1C 1C 1C 55 55 7A 55 7A 3 1C E9 15 55 1C 1C 20 E9 1C 55 BD 30	30 E9 1C 55 BD 3,3 9,3 33,3 27,3 24 ms

3.	7 6 6 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BD 7A 1C 1C 1C 55 55 7A 55 7A 3 7A 15 7A BD 20 7A BD 1C 30	65 7A BD 1C 1,1 19,1 31,1 22 ms
4.	10 6 6 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 3 7A 15 7A BD 20 7A 7A 1C 30	15 7A 1,1 26 ms

5.	10 6 6 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 7A 3 1C BD 7A 7A 15 7A 7A 7A 7A 1C 20 7A 7A 7A 7A 7A 7A 100	100 7A 7A 7A 7A 7A 7A 1,1 7,1 13,1 19,1 25,1 31,1 24 ms
6.	7 6 6 7A 55 E9 E9 1C 55 55 7A 1C 7A E9 55 55 1C 1C 55 E9 BD BD 1C 7A 1C 55 BD BD 55 BD 7A 1C 1C 1C 55 55 7A 55 7A 3 7A -15 7A BD -20 7A BD 1C -100	0 29 ms

BAB 4
Pranala ke *Repository*

https://github.com/CrystalNoob/Tucil1_13522094

BAB 5

Refleksi

Dari tugas kecil ini, saya mendapat kesempatan untuk mengeksplor bahasa C++ dengan lebih dalam dari sebelumnya. Saya dapat menemukan kekurangan-kekurangan saya dan akan berusaha untuk menjadi lebih baik untuk kedepannya. Awalnya, tugas kecil ini membingungkan saya. Saya tidak tahu harus mulai dari mana. Tetapi, setelah beberapa hari membaca spesifikasi dan akhirnya mencoba mengerjakannya, pada akhirnya, saya dapat menyelesaikan tugas kecil ini, walaupun tidak maksimal. Saat mengerjakan tugas besar ini, saya mendapati pembagian waktu untuk eksplorasi bahasa dan mengerjakan tugas menjadi vital. Kode yang awalnya saya pikir bisa berjalan dengan baik ternyata tidak bisa dikompilasi karena satu dan berbagai hal. Menjelang *deadline* tugas kecil ini, saya juga belajar untuk tetap berusaha dan berjuang sampai akhir. Sekian dari refleksi diri saya, terima kasih.

Daftar Pustaka

1. Spesifikasi Tugas Kecil 1 Stima 2023/2024.docx - Google Docs
2. cplusplus.com/reference/
3. cpp/ - cppreference.com
4. C++ Tutorial (w3schools.com)
5. Explicit initialization with constructors (C++ only) - IBM Documentation
6. Stack Overflow - Where Developers Learn, Share, & Build Careers

Lampiran

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca masukan berkas .txt	✓	
4. Program dapat menghasilkan masukan secara acak	✓	
5. Solusi yang diberikan program optimal		✓
6. Program dapat menyimpan solusi dalam berkas .txt	✓	
7. Program memiliki GUI		✓