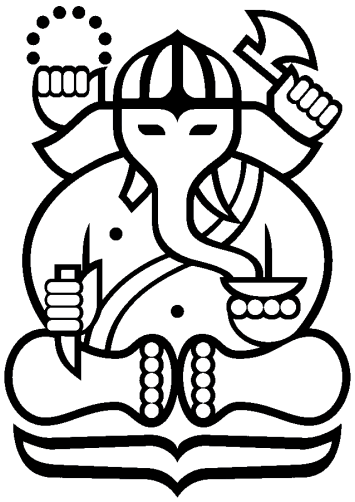
Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun Akademik 2023/2024

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun oleh:

Muhammad Gilang Ramadhan

13520137

K01

Program Studi S1 Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

BAB 1. Algoritma *Brute force* untuk menyelesaikan *Cyberpunk 2077 Breach Protocol*

Mind Idea to Solve the Problem:

Untuk memecahkan masalah ini dengan pendekatan brute force, tentunya kita terpikir untuk mengambil seluruh kemungkinan state yang dapat dipenuhi dari node asal (node-node yang berada pada baris pertama pada grid kumpulan token) sampai dengan maximum panjang node pada suatu state tersebut adalah buffer_size, karena bila kita mengambil node-node sampai melebihi kapasitas buffer_size, tentunya kita tidak bisa memasukkan semua node tersebut ke dalam buffer. Maka cukup ambil node sebanyak ≤ buffer_size saja.

Sekarang apakah mungkin kita bisa meng-construct hal yang demikian ? bisa, untuk cara awamnya ialah dengan melakukan permutasi terhadap susunan kemungkinan perubahan posisi relatif pada suatu posisi ke posisi selanjutnya. Misalkan kita punya grid yang terdiri dari *h* balok yang memuat *w* buah kotak *1x1*. Jika kita telah berpindah sembarang secara vertikal maka, selanjutnya kita dapat memilih sebanyak w-1 posisi grid lain untuk berpindah, untuk selanjutnya lagi, kita dapat memilih sebanyak h-1 posisi grid lain untuk berpindah ke posisi selanjutnya lagi, demikian seterusnya. Dengan demikian, diperoleh banyaknya kemungkinan solusi tersebut sebagai banyaknya kemungkinan perubahan posisi yang dilakukan untuk setiap posisi starting node pada setiap kolomnya yaitu:

$$w \prod_{i=1}^{buffer\ size\ -1} f_i \quad \ldots.(1)$$

dengan,

$$f_i = w - 1, \text{ jika i ganjil}$$

$$f_i = h - 1, \text{ jika i genap}$$

Namun, perhatikan juga bahwa pada setiap posisi state tersebut berpindah, tidak boleh mengunjungi state posisi sebelumnya. Akibatnya, kita diharuskan untuk melakukan pengecekan terhadap setiap posisi tersebut untuk memastikan bahwa posisi telah dikunjungi atau belum.

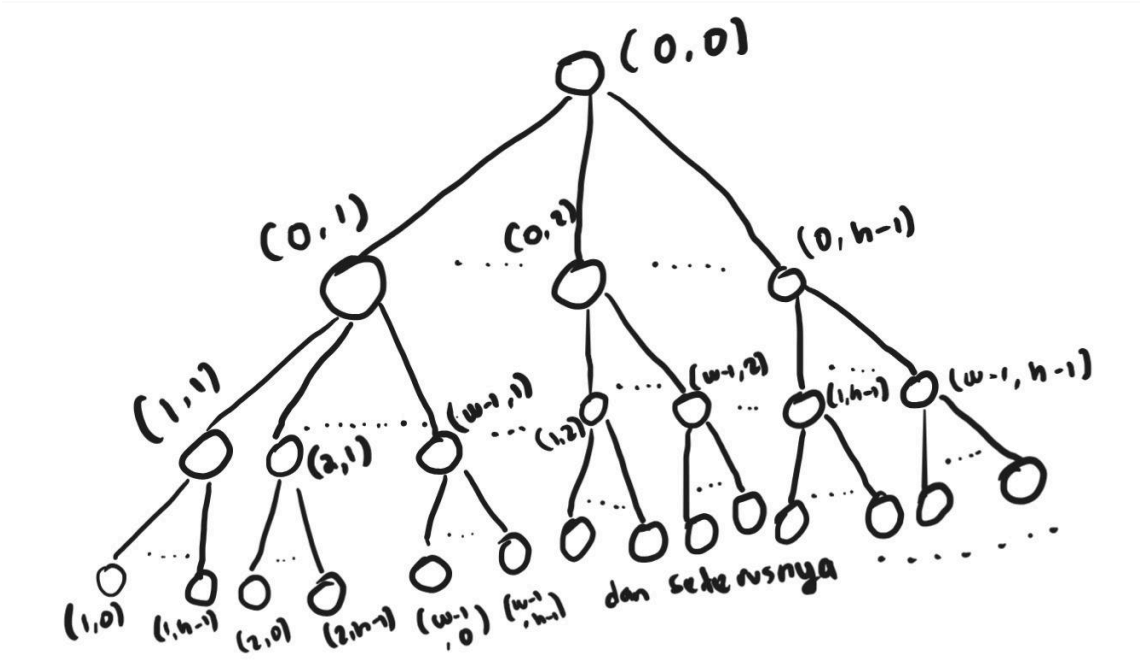
Setelah berhasil melakukan konstruksi terhadap algoritma pencariannya, saat sembari mendapatkan candidate solusi yang valid, maka kita bisa sambil mengecek untuk suatu perubahan posisi yang merupakan candidate solusi (memiliki panjang node = buffer_size) untuk diperiksa secara brute force melalui string matching apakah ada pada sequences atau tidak. Jika lebih besar maka, simpan state dan max_reward diganti dgn reward tersebut.

Begitu seterusnya, ulangi langkah-langkah di atas, sampai dengan semua candidate posisi selesai diperoleh dan diperiksa.

Implementasi Algoritma

1. Algoritma Penelusuran Token Pada Grid

   Untuk implementasi algoritma, kita bisa menggunakan algoritma traversal Depth First Search untuk membangkitkan seluruh permutasi perubahan posisi node sepanjang buffer_size yang mungkin dari masing-masing posisi starting node pada row paling atas dengan posisi kolom tertentu. Untuk ilustrasi gambarnya disajikan melalui gambar 1 di bawah ini.



Gambar 1. Ilustrasi Pembangkitan Semua Kandidat Solusi

(Dokumentasi Pribadi Penulis)

Pada pembangkitan tersebut, misalkan kita mulai pada state ganjil (mulai dari 1,2, sampai seterusnya), misalkan state mulai pada posisi (x,0), maka kita bisa membangkitkan node-node (x,1), (x,2), …., (x,h-1). Untuk setiap node-node yang dibangkitkan tersebut, cek terlebih dahulu apakah node tersebut sudah dikunjungi sebelumnya atau belum, jika sudah dikunjungi maka kita cari node yang lain yang belum dikunjungi pada node-node yang telah dibangkitkan sebelumnya. Kita pilih salah satu secara bergantian untuk state selanjutnya. Kemudian pada state selanjutnya (state genap), misalnya kita bisa memilih node (x,1), maka kita bisa membangkitkan nodes: (0,1), (1,1), …., (x-1,1), (x+1,1), …., (w-1,1), dengan $x \neq 0$.

Setelah itu, pada setiap pembangkitan satu graph yang terdiri dari buffer_size node yang merupakan kandidat solusi yang akan diperiksa, untuk mendapatkan kandidat solusi pertama kita cukup mengikuti traversal tersebut dari level 0 sampai dengan level buffer size - 1, kemudian untuk peralihan ke level yang selanjut, maka kita cukup mengecek apakah level + 1 tersebut ≤ banyaknya kandidat solusi yang sudah dikumpulkan sebelumnya, kalau iya berarti kita harus meng-assign kandidat solusi tersebut dengan node-node sekarang sedemikian sehingga kandidat solusi yang lama dibuang dan diganti dengan node baru pada level sekarang dan banyaknya node pada kandidat solusi sama dengan level sekarang + 1.

Lakukan proses pembangkitan node-node tersebut dan proses *maintenance* tersebut sampai dengan tidak ada lagi node yang bisa dibangkitkan (Semua node yang membentuk buffer) sudah habis.

Oleh karena itu, banyaknya operasi pembangkitan node dan pengecekkannya adalah sesuai dengan persamaan (1) pada *mind idea*. Akibatnya kompleksitas waktu pada algoritma ini adalah $O(N^{buffer\ size})$.

2. Algoritma *Comparison String Matching Sequences*

   Pada setiap *buffer* yang dihasilkan kita perlu melakukan *comparison* apakah *buffer* tersebut memiliki *reward* terbaik atau tidak. Caranya adalah dengan melakukan perbandingan terhadap variabel yang disimpan yang mulanya di-*assign* 0 pada *initial*

condition, kemudian untuk setiap *candidate state* yang diperoleh kita bandingkan *reward*-nya dengan nilai pada variabel tersebut, jika *reward* tersebut > nilai variabel tersebut, maka jadikan nilai *reward* tersebut sebagai nilai variabel tersebut dan simpan state kandidat tersebut untuk menyimpan posisi dari kandidat *state* pada *grid*.

Pertama-tama kita inisialisasikan nilai count_reward dengan 0 dan kemudian, kita periksa pada setiap *sequence* ke-1, *sequence* ke-2, …, *sequence* ke-*num_sequence* apakah terdapat pada pada *buffer* tersebut atau tidak. Caranya adalah seperti menjadikan isi *buffer* tersebut sebagai *string*, kemudian untuk setiap substring pada *buffer* cukup dicek apakah memuat string *sequence* ke-i atau tidak. Jika ya, maka tambahkan nilai *reward* pada *sequence* tersebut ke variabel *count_reward*.

Setelah setiap selesai melakukan proses pengecekan semua sequences pada satu sub-node atau full buffer itu sendiri, lakukan perbandingan terhadap nilai count_reward dan max_reward. Jika count_reward > max_reward, maka assign max_reward dengan count_reward dan simpan state buffer tersebut. Pada kasus ini, max_reward diubah hanya ketika ada value yang lebih besar daripada dirinya karena, untuk setiap reward yang memiliki value yang sama tetapi panjang *buffer*-nya lebih besar tentunya kita lebih baik menyimpan yang buffer yang panjangnya lebih kecil saja sehingga akan mengefisienkan penggunaan buffer.

Lakukan langkah comparison tersebut secara terus menerus setelah kandidat solusi ditemukan dan pada akhirnya ditemukannya solusi max_reward yang terbesar. Karena secara brute force kita menjamin selalu bisa menghasilkan solusi optimum dengan cara membandingkan seluruh kemungkinan state pada kasus ini.

Pada operasi ini, dilakukan operasi pengecekan candidate sebanyak:

Num_buffer_checking * Num_seq_checking

Dengan:

Num_buffer_checking = length(candidate)

Num_seq_checking = length(seq1) + length(seq2) + … + length(seq$seq_{num\ sequence}$).

Sehingga diperoleh *approximate time complexity*-nya adalah O(N^3).

Dari bagian *Mind Idea* perhatikan bahwa kita perlu melakukan pengecekan seluruh *buffer* yang memiliki panjang $\leq$ buffer_size. Maka dengan demikian diperlukan buffer_size operasi untuk melakukan keseluruhan pembangkitan dan *comparison* operasi. Dengan demikian *time complexity* total pada algoritma ini ialah O(N x N^3 x $N^{buffer\ size}$) = O($N^{buffer\ size + 4}$).

3. Alur *Logic* Pembangkitan *token random* pada fitur *input model* 2

   1. Pembangkitan *random input matrix*

   Untuk pembangkitan *random input matrix*, akan dilakukan randomisasi pada posisi *unique token* pada *array*. Sehingga akan dilakukan randomisasi pada bilangan integer dari 0 sampai panjang *unique token*. Kemudian setelah bilangan *random* tersebut berhasil dirandomisasi, maka lakukan *mapping* antara posisi *unique token* dengan *value char*-nya. Untuk *mapping*-nya harusnya berjalan secara konstan O(1) karena menggunakan *array vector* biasa. Lakukan terus menerus sampai dengan seluruh *cell* pada *matrix* terisi.

   2. Pembangkitan random input *sequence* dan *reward*-nya

   Untuk proses *random input sequence* menggunakan pola *sequence* yang dibentuk pada *matrix input*. Untuk setiap *starting point* pada *matrix input*, kita randomisasi perubahan posisi yang mengikuti selang seling (aturan pemilihan *node* yang valid ke *buffer*). Akibatnya *sequence* yang *generate* dengan cara ini pasti selalu valid sebagai *buffer*. Oleh karena itu, untuk setiap pembangkitan kandidat solusi pada *buffer*, maka kita selalu bisa mengecek bahwa pada *buffer* tersebut selalu memuat salah satu *sequence* dari semua *sequences* yang telah di-*generate*. Dengan demikian *sequence* yang di-*generate* dengan *logic* ini menjadi tidak *useless*.

BAB 2. Source Code dalam Bahasa C++

```
#include <bits/stdc++.h>
using namespace std;

#define sz(v) (ll)v.size()

struct Token {
char first;
char second;
};

typedef int ll;
typedef vector<ll> vl;
typedef vector<vl> vvl;
typedef pair<ll,ll> pl;
typedef vector<Token> vt;
typedef vector<vt> vvt;
typedef vector<bool> vb;

struct Coordinate{
ll x;
ll y;
};

struct Node {
Coordinate pos;
ll level;
};

ll buffer_size,matrix_width,matrix_height,num_sequences,max_reward;

vvt matrix,sequence;
vl sequence_reward;
vt tempVec;
Token tempToken;
ll tempVal;
Coordinate tempCoord;
ll numInput;

vector<Coordinate> candidateSolution;
```

```
vector<Coordinate> realSolution;
vector<vb> isVisited;
double time_taken;
char isRecursive;
ll currBufferSz,counTokenUnique;
vt tempTokenUnique;
ll maxSequenceLength;
```

Gambar 2. Header dan Inisialisasi Variabel Global

```
void randomizeInputMat(ll row, ll col){
for(ll i = 0; i<row; i++){
for(ll j = 0; j<col; j++){
ll tempRanUniquePos = rand()%sz(tempTokenUnique);
tempToken.first = tempTokenUnique[tempRanUniquePos].first;
tempToken.second = tempTokenUnique[tempRanUniquePos].second;
tempVec.emplace_back(tempToken);
}
matrix.emplace_back(tempVec);
tempVec.clear();
}
}
```

Gambar 3. Procedure randomisasi terhadap inputan matrix

```
void randomizeSequence(ll num){
for(ll i = 0; i<num; i++){
ll length = rand()%(buffer_size) + 1;
ll currX = rand()%(matrix_width);
ll currY = rand()%(matrix_height);
tempToken.first = matrix[currY][currX].first;
tempToken.second = matrix[currY][currX].second;
isVisited[currY][currX] = true;
tempVec.emplace_back(tempToken);
ll choseFirstDir = rand()%2; // 1 for vertical, 0 for horizontal
for(ll j = 0; j<length-1; j++){
if(choseFirstDir){
ll tempRandY = rand()%(matrix_height);
while(currY==tempRandY||isVisited[tempRandY][currX]){
tempRandY = rand()%(matrix_height);
}
currY=tempRandY;
```

```
}else{
ll tempRandX = rand()%(matrix_width);
while(currX==tempRandX||isVisited[currY][tempRandX]){
tempRandX = rand()%(matrix_width);
}
currX=tempRandX;
}
tempToken.first = matrix[currY][currX].first;
tempToken.second = matrix[currY][currX].second;
isVisited[currY][currX] = true;
tempVec.emplace_back(tempToken);
choseFirstDir++;
choseFirstDir %= 2;
}
sequence.emplace_back(tempVec);
sequence_reward.emplace_back((rand()%203) - 101);
tempVec.clear();
for(ll k = 0; k<matrix_height; k++){
for(ll l = 0; l<matrix_width; l++){
isVisited[k][l] = false;
}
}
}
}
```

Gambar 4. Procedure randomisasi terhadap inputan sequences dan sequence_rewards

```
void initVisitChecker(){
for(ll i = 0; i<matrix_height; i++){
vb tempVb;
for(ll j = 0; j<matrix_width; j++){
tempVb.push_back(false);
}
isVisited.emplace_back(tempVb);
}
}
```

Gambar 5. Procedure Inisialisasi terhadap isVisited checker matrix

```
void printInput(){
cout << buffer_size << "\n";
cout << matrix_width << " " << matrix_height << "\n";
```

```
for(ll i=0; i<matrix_height; i++){
for(ll j=0; j<matrix_width; j++){
cout << matrix[i][j].first << matrix[i][j].second << " ";
}
cout << "\n";
}

cout << num_sequences << "\n";

for(ll i = 0; i<num_sequences; i++){
for(ll j = 0; j<sz(sequence[i]); j++){
cout << sequence[i][j].first << sequence[i][j].second << " ";
}
cout << "\n";
cout << sequence_reward[i] << "\n";
}
}
```

Gambar 6. Procedure untuk melakukan print terhadap input

```
void saveInput(string namaFile){
ofstream outputFile("../test/input/"+namaFile);
if (outputFile.is_open()) {
outputFile << buffer_size << "\n";
outputFile << matrix_width << " " << matrix_height << "\n";
for(ll i=0; i<matrix_height; i++){
for(ll j=0; j<matrix_width; j++){
outputFile << matrix[i][j].first << matrix[i][j].second << " ";
}
outputFile << "\n";
}

outputFile << num_sequences << "\n";

for(ll i = 0; i<num_sequences; i++){
for(ll j = 0; j<sz(sequence[i]); j++){
outputFile << sequence[i][j].first << sequence[i][j].second << " ";
}
outputFile << "\n";
outputFile << sequence_reward[i] << "\n";
}
outputFile.close();
```

```
cout << "Data has been saved successfully.\n";
} else {
cout << "Unable to open file.\n";
}
}
```

Gambar 7. Procedure save input into file .txt

```
void manualInputDataTxtFile(string namaFile){
ifstream inputFile("../test/input/"+namaFile);

while(!inputFile.is_open()){
cout << "File yang dicari tidak ada!\n";
cout << "Masukkan nama file yang ingin diinput dengan format .txt: ";
cin >> namaFile;
inputFile.open("../test/input/"+namaFile);
cout << "\n";
}

inputFile >> buffer_size >> matrix_width >> matrix_height;
for(ll i=0; i<matrix_height; i++){
for(ll j=0; j<matrix_width; j++){
inputFile >> tempToken.first >> tempToken.second;
tempVec.emplace_back(tempToken);
}
matrix.emplace_back(tempVec);
tempVec.clear();
}
inputFile >> num_sequences;
inputFile.ignore();
for(ll i = 0; i<num_sequences; i++){
string tempSeq; getline(inputFile,tempSeq);
ll szTempSeq = sz(tempSeq);
for(ll j = 0; j<szTempSeq; j++){
int tempMod = j%3;
if(tempMod==0){
tempToken.first = tempSeq[j];
}else if(tempMod==1){
tempToken.second = tempSeq[j];
tempVec.emplace_back(tempToken);
}
}
```

```
sequence.emplace_back(tempVec);
tempVec.clear();
inputFile >> tempVal;
sequence_reward.emplace_back(tempVal);
inputFile.ignore();
}
initVisitChecker();
inputFile.close();
}
```

Gambar 8. Procedure untuk melakukan input manual dari file .txt

```
void generateRandInput(){
cout << "Enter buffer size: ";
cin >> buffer_size;
cout << "\n";
cout << "Enter matrix width: ";
cin >> matrix_width;
cout << "\n";
cout << "Emter matrix height: ";
cin >> matrix_height;
cout << "\n";
cout << "Enter number of sequences: ";
cin >> num_sequences;
cout << "\n";
cout << "Enter maximum sequence length: ";
cin >> maxSequenceLength;
cout << "\n";
cout << "Enter number of unique token: ";
cin >> counTokenUnique;
cout << "\n";
cout << "Enter unique token [Asumsi token berupa alpha numeric dengan huruf kapital]
\n";
cout << "Command: ";
for(ll i = 0; i<counTokenUnique; i++){
cin >> tempToken.first >> tempToken.second;
tempTokenUnique.emplace_back(tempToken);
}
cout << "\n";
srand(time(NULL));
randomizeInputMat(matrix_height, matrix_width);
initVisitChecker();
```

```
randomizeSequence(num_sequences);
tempTokenUnique.clear();

cout << "Random Input Generated: \n";
printInput();

cout << "\n";
cout << "Do you want to save the input? (Y/N): ";
char isSimpanInput;
cin >> isSimpanInput;
cout << "\n";
if (isSimpanInput == 'Y' || isSimpanInput == 'y') {
string namaFile;
cout << "Enter File Name with .txt extension: ";
cin >> namaFile;
saveInput(namaFile);
}
}
```

Gambar 9. Procedure untuk melakukan generate terhadap random input

```
void inputData(){
cout << "Silahkan pilih model input data anda: \n";
cout << "1. Input Data Manual\n";
cout << "2. Input Data dengan Random Generator\n";
cout << "Command [1,2]: ";
cin >> numInput;
cout << "\n";
if(numInput==1){
cout << "Masukkan nama file yang ingin diinput dengan format .txt: ";
string namaFile; cin >> namaFile;
manualInputDataTxtFile(namaFile);
}else{
generateRandInput();
}
}
```

Gambar 10. Procedure untuk memilih dan melakukan command input

```
void checkSequences(){
ll szCandidate = sz(candidateSolution);
ll countReward = 0;
```

```
for(ll i = 0; i<num_sequences; i++){
ll szSeq = sz(sequence[i]);
if(szCandidate>=szSeq){
for(ll j = 0; j<szCandidate; j++){
if(szCandidate-j>=szSeq){
bool flag = true;
for(ll k = 0; k<szSeq; k++){
if(sequence[i][k].first!=matrix[candidateSolution[j+k].y][candidateSolution[j+k].x].fi
rst
||
sequence[i][k].second!=matrix[candidateSolution[j+k].y][candidateSolution[j+k].x].seco
nd
){
flag = false;
break;
}
}
if(flag){
countReward += sequence_reward[i];
break;
}
}
}
}
}
if(countReward>max_reward){
max_reward = countReward;
realSolution.clear();
for(ll i = 0; i<szCandidate; i++){
realSolution.emplace_back(candidateSolution[i]);
}
}
}
```

Gambar 11. Procedure untuk comparison buffer with sequences

```
void searchBruteForce1(ll row,ll col){
stack<Node> tempStack;
Node tempPl1,tempPl2;
tempPl1.pos.x = col; tempPl1.pos.y = row; tempPl1.level = 0;
tempStack.push(tempPl1);
ll currLevel = 0;
```

```
while (!tempStack.empty())
{
tempPl1 = tempStack.top();
tempStack.pop();
ll tempCandSz = sz(candidateSolution);
if(tempPl1.level+1<=tempCandSz){
for(ll i = 0; i<=(tempCandSz-tempPl1.level-1); i++){
isVisited[candidateSolution.back().y][candidateSolution.back().x] = false;
candidateSolution.pop_back();
}
}
currLevel = tempPl1.level;
tempCoord.x = tempPl1.pos.x;
tempCoord.y = tempPl1.pos.y;
isVisited[tempPl1.pos.y][tempPl1.pos.x] = true;
candidateSolution.emplace_back(tempCoord);

if(currLevel==currBufferSz-1){
checkSequences();
continue;
}

ll tempMod = currLevel%2;
ll length = (tempMod == 0) ? matrix_height : matrix_width;
ll indexMove = (tempMod == 0) ? tempPl1.pos.y : tempPl1.pos.x;

// Maju!
for (ll i = length-1; i>=indexMove+1; i--){
if(tempMod==0){
if(isVisited[i][tempPl1.pos.x]) continue;
tempPl2.pos.y = i; tempPl2.pos.x = tempPl1.pos.x;
}else{
if(isVisited[tempPl1.pos.y][i]) continue;
tempPl2.pos.y = tempPl1.pos.y; tempPl2.pos.x = i;
}
tempPl2.level = currLevel+1;
tempStack.push(tempPl2);
}

// Mundur
for (ll i=0; i <=indexMove-1; i++){
if(tempMod==0){
```

```
if(isVisited[i][tempPl1.pos.x]) continue;
tempPl2.pos.y = i; tempPl2.pos.x = tempPl1.pos.x;
}else{
if(isVisited[tempPl1.pos.y][i]) continue;
tempPl2.pos.y = tempPl1.pos.y; tempPl2.pos.x = i;
}
tempPl2.level = currLevel+1;
tempStack.push(tempPl2);
}
}
}
```

Gambar 12. Procedure Algoritma pembangkitan node non-recursive dengan stack

```
void searchBruteForce2Rec(ll level, ll row, ll col) {
tempCoord.x = col;
tempCoord.y = row;
isVisited[row][col] = true;
candidateSolution.emplace_back(tempCoord);

if(level==currBufferSz-1){
checkSequences();
return;
}

ll tempMod = level%2;
ll length = (tempMod == 0) ? matrix_height : matrix_width;
ll indexMove = (tempMod == 0) ? row : col;

// Next, Mundur!
for(ll i = indexMove-1; i>=0; i--){
if(tempMod == 0){
if(isVisited[i][col]) continue;
searchBruteForce2Rec(level+1,i,col);
}else {
if(isVisited[row][i]) continue;
searchBruteForce2Rec(level+1,row,i);
}
isVisited[candidateSolution.back().y][candidateSolution.back().x] = false;
candidateSolution.pop_back();
}
```

```
// Next, Maju!
for(ll i = indexMove+1; i<length; i++){
if(tempMod == 0){
if(isVisited[i][col]) continue;
searchBruteForce2Rec(level+1,i,col);
}else {
if(isVisited[row][i]) continue;
searchBruteForce2Rec(level+1,row,i);
}
isVisited[candidateSolution.back().y][candidateSolution.back().x] = false;
candidateSolution.pop_back();
}
}
```

Gambar 13. Procedure Algoritma pembangkitan node dengan recursive

```
void printSolution(){
cout << "\n";
cout << "Output:\n";
cout << "Possible Solution: " << "\n";
cout << max_reward << "\n";

ll szSolution = sz(realSolution);
if(szSolution>=1){
for(ll i = 0; i<szSolution; i++){
cout          <<          matrix[realSolution[i].y][realSolution[i].x].first          <<
matrix[realSolution[i].y][realSolution[i].x].second << " ";
}
cout << "\n";

for(ll i = 0; i<szSolution; i++){
cout << realSolution[i].x + 1 << ", " << realSolution[i].y + 1 << "\n";
}
cout << "\n";
}else{
cout << "No Solution\n";
}
cout << "Time Execution: " << fixed << time_taken * 1000 << setprecision(5);
cout << " ms\n";
}
```

Gambar 14. Procedure menampilkan solusi ke terminal

```
void printSolutionToTXT(string namaFile){
ofstream outputFile("../test/output/"+namaFile);
if (outputFile.is_open()) {
outputFile << max_reward << "\n";
ll szSolution = sz(realSolution);
if(szSolution>=1){
for(ll i = 0; i<szSolution; i++){
outputFile        <<        matrix[realSolution[i].y][realSolution[i].x].first        <<
matrix[realSolution[i].y][realSolution[i].x].second << " ";
}
outputFile << "\n";

for(ll i = 0; i<szSolution; i++){
outputFile << realSolution[i].x + 1 << ", " << realSolution[i].y + 1 << "\n";
}
outputFile << "\n";
}else{
outputFile << "No Solution\n";
}
outputFile << "Time Execution: " << fixed << time_taken * 1000 << setprecision(5);
outputFile << " ms\n";
outputFile.close();
cout << "Data has been saved successfully.\n";
} else {
cout << "Unable to open file.\n";
}
}
```

Gambar 15. Procedure penulisan solusi ke file .txt untuk di-save

```
// Solve with brute force
void solve(){
for(ll i = 0; i<matrix_width; i++){
if(isRecursive=='Y'){
for(ll j = 1; j<=buffer_size; j++){
currBufferSz = j;
searchBruteForce2Rec(0,0,i);
while(!candidateSolution.empty()){
isVisited[candidateSolution.back().y][candidateSolution.back().x] = false;
candidateSolution.pop_back();
}
```

```
}
}
else{
for(ll j = 1; j<=buffer_size; j++){
currBufferSz = j;
searchBruteForce1(0,i);
while(!candidateSolution.empty()){
isVisited[candidateSolution.back().y][candidateSolution.back().x] = false;
candidateSolution.pop_back();
}
}
}
}
}
```

Gambar 16. Procedure utama untuk solve dengan brute force

```
int main() {
ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);
cout << "Apakah ingin menggunakan algo recursive (Y\\N)? ";
cin >> isRecursive;
cout << "\n";
inputData();
clock_t start, end;
start = clock();
solve();
end = clock();
time_taken = double(end - start) / double(CLOCKS_PER_SEC);
printSolution();
cout << "\n";
cout << "Apakah ingin menyimpan solusi? [Y/N]: ";
char isSimpanSolusi; cin >> isSimpanSolusi;
cout << "\n";
if(isSimpanSolusi=='Y'||isSimpanSolusi=='y'){
cout << "Masukkan Nama File dengan format .txt: ";
string namaFile; cin >> namaFile;
printSolutionToTXT(namaFile);
}
cout << "Terima kasih telah bermain :D\n";
return 0;
}
```

Gambar 17. Fungsi utama untuk mengeksekusi program

BAB 3. Pengujian (Input/Output)

3.1. Test Case 1

test > input > input1.txt

```
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
```

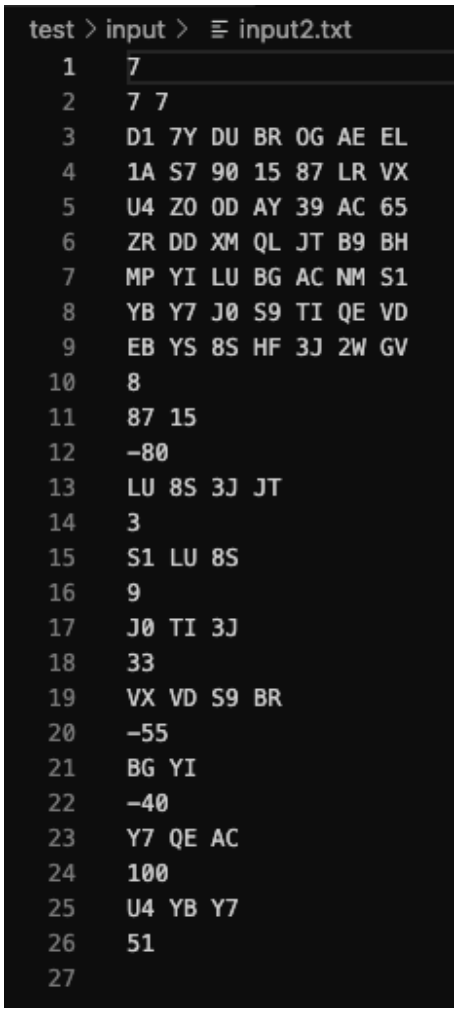
Gambar 18. Input Test Case 1

test > output > output1.txt

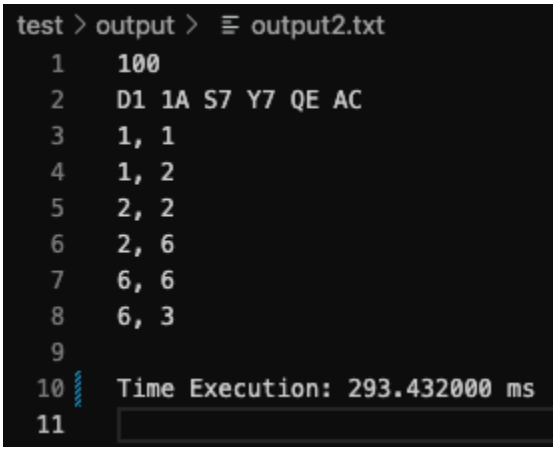
```
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 4
5, 4

Time Execution: 55.141000 ms
```

Gambar 19. Output Test Case 1

3.2. Test Case 2

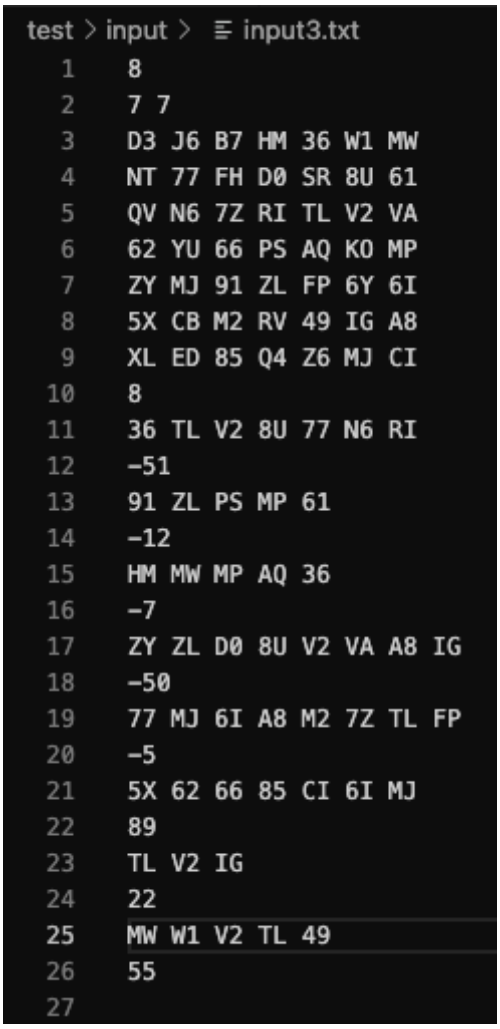


Gambar 20. Input Test Case 2

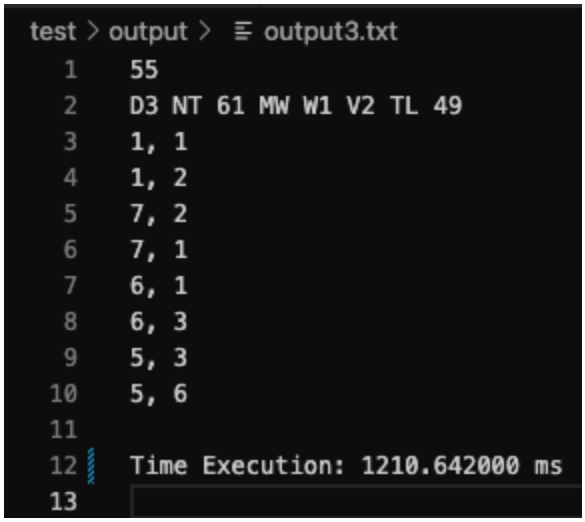


Gambar 21. Output Test Case 2

3.3. Test Case 3

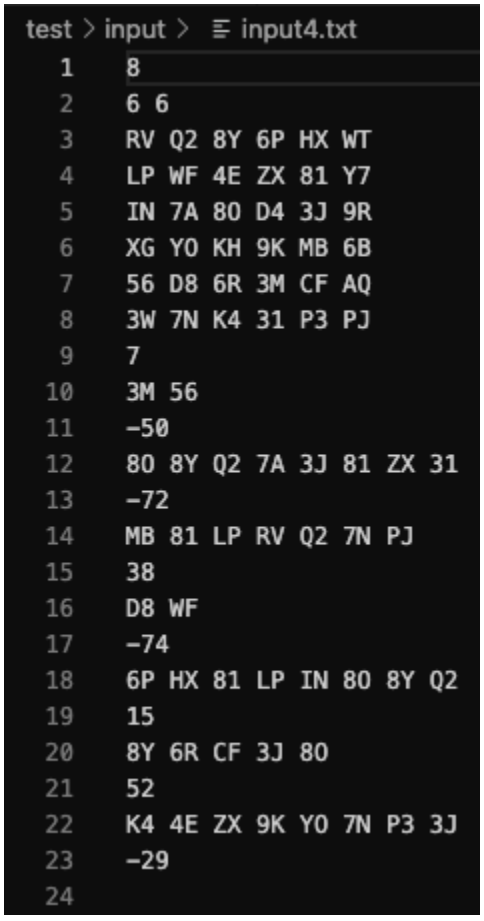


Gambar 22. Input Test Case 3



Gambar 23. Output Test Case 3

3.4. Test Case 4



Gambar 24. Input Test Case 4

test > output > output4.txt

```
52
8Y 6R CF 3J 8O
3, 1
3, 5
5, 5
5, 3
3, 3

Time Execution: 276.517000 ms

```

Gambar 25. Output Test Case 4

3.5. Test Case 5

test > input > input5.txt

```
8
8 8
K1 I9 MM UF C6 AM L8 28
2G WL D8 QT Q2 W7 UM R5
VC TQ 36 RH 10 KH 36 1W
S3 V2 KK 6G EC 9L SA SK
CL K6 4X B4 PI O4 BC OF
0N 31 ZL ND 6R RI QV PO
T4 5Q P0 I6 XC 95 UN 22
WH 50 B2 DZ TI 07 MF 5N
9
B2 50
-90
UM R5 28 L8 BC PI XC
-63
SA S3 T4 I6 ND RI KH
-11
9L S3
7
07 50 31
39
AM KH
4
QV 0N VC
17
50 07 AM 28 OF
30
UN SA 9L KH 1W R5 QT B4
-66

```

Gambar 26. Input Test Case 5

test > output > output5.txt

```
56
L8 QV 0N VC KH 07 50 31
7, 1
7, 6
1, 6
1, 3
6, 3
6, 8
2, 8
2, 6

Time Execution: 5778.179000 ms

```

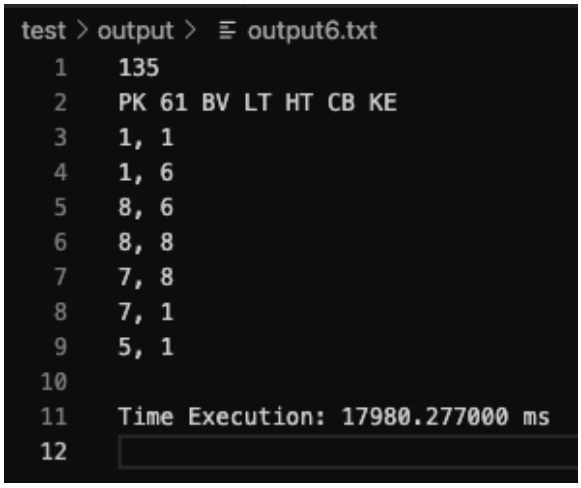
Gambar 27. Output Test Case 5

3.6. Test Case 6

test > input > input6.txt

```
8
9 9
PK G4 FR M4 KE 7I CB IV 7D
U0 WA HJ XW A7 2B 2F YC 04
1F RD 60 M5 25 TQ 4A FR 9D
HW WR 4P 22 U9 B9 DT F2 JV
E8 GC KX EX RD PH ZW 53 XR
61 1K ZQ XN TB U4 RC BV RU
AO YO B1 AS 57 99 E1 GE B3
BV L7 6E I3 6Q 79 HT LT Q3
ST U6 6F AG 4H JW QW B1 W7
10
PK 61
59
7I JW 4H A7 U0 1F
38
RD 25 57 YO U6 W7 7D M4
9
M5
-32
F2 U9 RD PH U4 XN AS
-98
HT E1 57 RD
-23
RD U6 4H U9 WR
-45
LT HT
38
HW 61 XN XW WA YO
52
CB KE
38

```

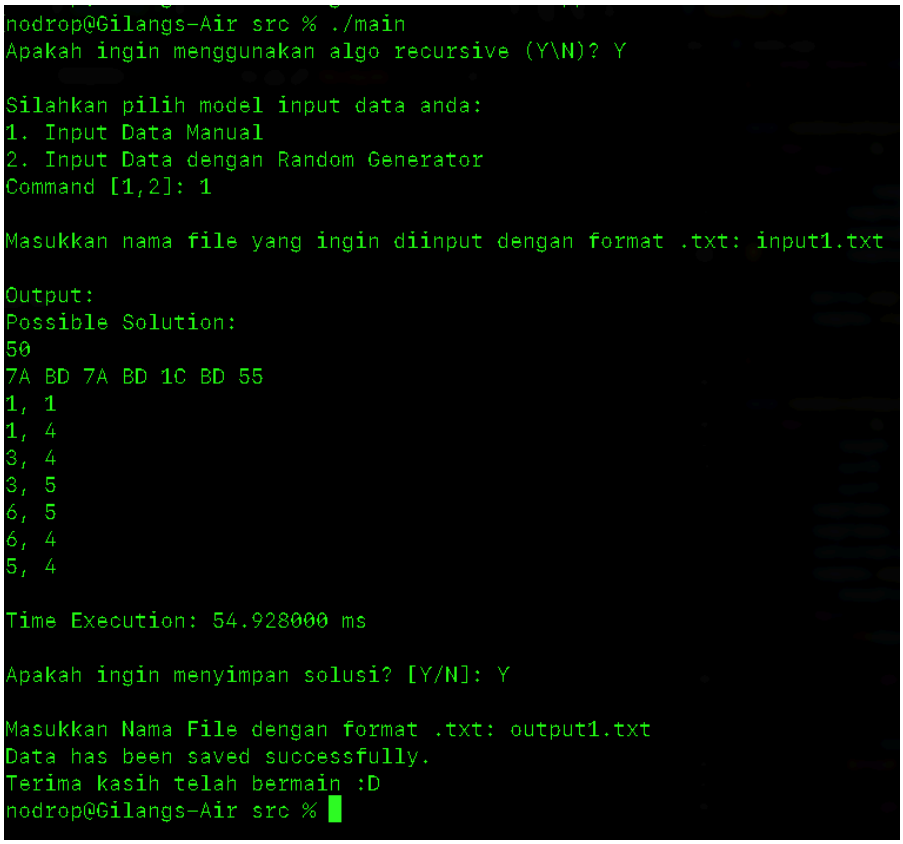
Gambar 28. Input Test Case 6



Gambar 29. Output Test Case 6

3.7 Tampilan Antar Muka CLI Program

1. Inputan Manual



Gambar 30. Interaksi Input dan hasil output pada inputan model 1

2. Inputan Matrix & Sequences Random Generator

```
nodrop@Gilangs-Air src % ./main
Apakah ingin menggunakan algo recursive (Y\N)? Y

Silahkan pilih model input data anda:
1. Input Data Manual
2. Input Data dengan Random Generator
Command [1,2]: 2

Enter buffer size: 8

Enter matrix width: 8

Emter matrix height: 8

Enter number of sequences: 9

Enter maximum sequence length: 9

Enter number of unique token: 8

Enter unique token [Asumsi token berupa alpha numeric dengan huruf kapital]
Command: A2 3C B8 I9 4N 8Y T0 5U
```

```
Random Input Generated:
8
8 8
3C T0 I9 B8 4N A2 8Y 8Y
T0 3C T0 4N 4N I9 B8 T0
I9 5U 5U I9 B8 4N I9 8Y
4N 4N I9 4N A2 T0 5U 3C
A2 4N I9 8Y B8 B8 4N 4N
4N T0 3C 3C T0 I9 4N A2
5U 8Y 3C B8 A2 3C 4N I9
A2 5U B8 A2 4N B8 I9 8Y
9
B8 3C 4N 4N I9 T0
78
I9
-8
T0 4N I9 I9 T0
-93
3C 8Y 5U 3C A2 4N
1
5U B8 4N I9 4N
-4
3C I9 4N I9 5U I9 4N B8 8Y
87
T0 T0 8Y
1
A2 8Y I9 4N 4N
-28
3C
-10

Do you want to save the input? (Y/N): Y
```

```
Output:
Possible Solution:
61
B8 3C 4N 4N I9 T0 T0 8Y
4, 1
4, 6
1, 6
1, 4
3, 4
3, 2
8, 2
8, 1

Time Execution: 5325.583000 ms

Apakah ingin menyimpan solusi? [Y/N]: Y

Masukkan Nama File dengan format .txt: output7.txt
```

Gambar 31. Interaksi input dan hasil output pada inputan model 2

Lampiran

Lampiran 1

Checklist Penilaian:

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		✓

Lampiran 2

Link Repository Github: https://github.com/NoDrop3011/Tucil1_13520137